On-Device Continual Learning with STT-Assisted-SOT MRAM based In-Memory Computing

Fan Zhang, *Graduate Student Member, IEEE*, Amitesh Sridharan, William Hwang, Fen Xue, Wilman Tsai, Shan Xiang Wang, *Fellow, IEEE*, Deliang Fan, *Member, IEEE*

***Abstract*—Due to the separate memory and computation units in traditional Von-Neumann architecture, massive data transfer dominates the overall computing system's power and latency, known as the 'Memory-Wall' issue. Especially with ever-increasing deep learning-based AI model size and computing complexity, it becomes the bottleneck for state-of-the-art AI computing systems. To address this challenge, In-Memory Computing (IMC) based Neural Network accelerators have been widely investigated to support AI computing within memory. However, most of those works focus only on inference. The on-device training and continual learning have not been well explored yet. In this work, for the first time, we introduce *on-device continual learning* with STT-assisted-SOT (SAS) Magnetic Random Access Memory (MRAM) based IMC system. On the hardware side, we have fabricated a SAS-MRAM device prototype with 4 Magnetic Tunnel Junctions (MTJ, each at 100nm $\times$ 50nm) sharing a common heavy metal layer, achieving significantly improved memory writing and area efficiency compared to traditional SOT-MRAM. Next, we designed fully digital IMC circuits with our SAS-MRAM to support both neural network inference and on-device learning. To enable efficient on-device continual learning for new task data, we present an 8-bit integer (INT8) based continual learning algorithm that utilizes our SAS-MRAM IMC-supported bit-serial digital in-memory convolution operations to train a small parallel reprogramming Network (Rep-Net) while freezing the major backbone model. Extensive studies have been presented based on our fabricated SAS-MRAM device prototype, cross-layer device-circuit benchmarking and simulation, as well as the on-device continual learning system evaluation.**

***Index Terms*—Continual Learning, In-Memory Computing, MRAM, Neural Network**

I. Introduction

Nowadays, deep neural networks (DNNs) have demonstrated great performance improvement in many cognitive applications, leading to a wide applications of edge/IoT-AI applications. Such smart devices collect new data across various domains/tasks in the real world, which will be used to continuously adapt the background DNN model for improved performance or even to adapt to different new tasks. This process is generally considered as *continual learning*. Constrained by the hardware resources (e.g., power, memory, size, etc.) of edge or IoT devices, it is common practice to send the newly collected data to the cloud for learning. Then, the updated new model will be deployed back to the edge/IoT devices for local AI inference processing [1]. Nevertheless, this approach suffers from large communication overhead between cloud and edge, as well as the user data privacy concern. Thus, *on-device continual learning* is essential to be investigated.

From the computing hardware side, DNNs demand numerous multiply and accumulate (MAC) operations and data movement. In conventional architectures (e.g., CPUs, GPUs), the energy of massive off-chip data communication could be almost two orders of magnitude higher than data processing itself, known as 'memory wall' [2]. To address this issue, In-Memory Computing (IMC) has attracted tremendous attention as a promising solution due to its capability to perform computation directly within memory [3], [4]. Different types of IMC designs based on either CMOS or post-CMOS non-volatile memory (NVM) technologies have been demonstrated, such as SRAM [3], [5], DRAM [6], ReRAM [2], [7], MRAM [4], [8]–[14], PCM [6], and etc.

Among various NVM technologies, Magnetic RAM (MRAM) holds great promise due to its zero standby leakage, high write/read speed & efficiency, compatibility with CMOS fabrication process, scalability, superior endurance, excellent retention time, and high integration density [8]. Correspondingly, many MRAM-based IMC designs [9]–[12], [15], [16] have been proposed to support DNN computation, especially for inference acceleration. SOT (Spin-Orbit Torque) MRAM and STT (Spin-Transfer Torque) MRAM are widely adopted devices in MRAM-based IMC accelerator designs. However, SOT-MRAM introduces additional access transistors to establish the SOT path, resulting in significant area overhead. On the other hand, STT-MRAM necessitates large currents, leading to reduced energy efficiency. To address these limitations while leveraging the superior properties of SOT-MRAM and STT-MRAM, a hybrid solution known as STT-assisted-SOT (SAS) MRAM has been proposed [13], [17]–[19].

SAS-MRAM offers a compelling alternative by requiring only a single access transistor to form the SOT path for multiple Magnetic Tunnel Junctions (MTJs) within the same heavy metal layer. Consequently, it occupies considerably less area compared to SOT-MRAM. Compared with STT-MRAM, the assistance of the SOT path significantly reduces the current required for writing data, further enhancing its

Manuscript created September 2023;

Fan Zhang, Amitesh Sridharan, and Deliang Fan are with the Department of Electrical and Computer Engineering, Johns Hopkins University, Baltimore, MD 21218 USA (e-mail: fzhang62, dfan10@jh.edu).
William Hwang, Fen Xue, and Shan Xiang Wang are with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA.
Wilman Tsai and Shan Xiang Wang are with the Department of Materials Science & Engineering, Stanford University, Stanford, CA 94305, USA.

energy efficiency. The aforementioned advantages position SAS-MRAM as a highly favorable choice over STT-/SOT-MRAM for IMC accelerator designs. However, despite its promising characteristics, the utilization of SAS-MRAM in IMC design remains largely unexplored. To the best of our knowledge, this paper represents the first attempt to investigate the design space of SAS-MRAM, particularly in the context of on-device continual learning accelerator design. By embarking on this research endeavor, we aim to shed light on the potential of SAS-MRAM as a novel and efficient solution in the field of IMC for on-device AI learning.

In the traditional continual learning approach, to deploy the re-trained or fine-tuned DNN model from the cloud to the NVM-based IMC accelerator, the general practice is to re-program the NVM cell (e.g., MRAM cell's conductance) to update the corresponding weight parameters. However, this procedure has to update nearly all non-volatile memory cells to represent the newly learned weight parameters, which is certainly not an efficient approach. Thus, an efficient and NVM IMC-friendly on-device continual learning framework is eagerly desired. There are mainly two roadblocks. First, the training memory cost (including activation, gradients, weights, etc.) is significantly higher than the inference-only process. To address such, *memory-efficient continual learning* has been recently investigated for on-device continual learning [20]–[22]. The second caveat is the complex floating-point MAC operation. Unlike the forward pass, in which the integer-only and quantization method have been well studied, DNN training usually needs to maintain the fidelity of the gradient during backpropagation to meet the high accuracy demand [23], which typically requires high-precision floating-point number operations. However, edge or IoT devices usually have restricted computation memory and power to support massive floating-point operations, which is one of the main reasons most edge/IoT-AI focuses on inference only. To address these constraints, recent studies have explored the *Integer-only training method* to enable efficient learning capabilities for edge devices, bridging the gap between high-precision demands and the computational realities of edge computing [24]–[27].

Considering the above-discussed challenges in hardware memory-wall and on-device continual learning algorithm. In this work, we present a new *on-device continual learning* framework with STT-assisted-SOT (SAS) MRAM-based IMC design. Our detailed technical contributions are:

I. For the memory device, we fabricated a 4-MTJ SAS-MRAM device prototype, where 4 MTJs (each with 100nm $\times$ 50nm) share the same heavy metal line to form one memory cell macro, which improves the writing and area efficiency. This 4-MTJ SAS-MRAM not only inherits the benefits of MRAM but also shows superior properties to normal STT-/SOT-MRAM. Our experiment results show that the SAS-MRAM has $\sim 3.4\times$ smaller write energy and similar area as STT-MRAM, but greatly improved density compared with traditional SOT-MRAM.

II. To the best of our knowledge, we introduce the first IMC circuits to utilize the SAS-MRAM to implement *digital bit-serial in-memory convolution* operations supporting both neural networks forward and backward passes computations. In our design, all the operations are performed in a purely digital way. Thus the power-hungry ADCs/DACs are not necessary. Moreover, our design only needs regular read/write operations at the sub-array level. It allows us to reuse the mature memory array design, but to add digital peripheral circuits to implement in-memory process element (PE).

III. To demonstrate the potential learning capability and capitalize the proposed SAS-MRAM IMC design, we present the *INT8 Reprogramming network (Rep-Net)*, which leverages the digital bit-serial in-memory convolution supported by our SAS-MRAM IMC hardware. This approach involves training a compact reprogramming network with a fully 8-bit integer training method for new task data while keeping the backbone model fixed. This strategy effectively reduces both computation and activation memory costs during the on-device continual learning process, demonstrating the practicality and efficiency of our design.

IV. Finally, using our experimentally benchmarked SAS-MRAM device model, we conduct comprehensive cross-layer device-circuit IMC system evaluation, as well as continual learning system performance benchmarking.

The rest of the paper is organized as follows: Section II covers the background. Section III introduces the SAS-MRAM structure and related works. Section IV demonstrates the SAS-MRAM-based digital IMC hardware design. The methodology of the proposed INT8-Rep-Net learning method is given in Section V. Section VI shows the performance of learning different tasks and system evaluation based on experimentally benchmarked cross-layer simulation framework. In the end, Section VII concludes the paper.

II. Background

A. Magnetoresistive random-access memory (MRAM)

Magnetoresistive random-access memory (MRAM) is an emerging non-volatile memory (NVM) where the data is stored in terms of electron spin. The Magnetic Tunnel Junction (MTJ), as the basic memory cell device, typically consists of multiple layers where two ferromagnetic layers sandwich a thin insulating layer. One of the two ferromagnetic layers of a MTJ is a permanent magnet with a particular polarity called 'fixed layer'. Another ferromagnetic layer's magnetization can be changed by external stimulus. Thus, it is called the 'free layer'. As shown in Fig. 1(a), when the free layer's magnetization is opposite that of the fixed layer, this MTJ is in Anti-Parallel State (AP), making it have high electrical resistance. On the contrary, in Fig. 1(b), the free layer's magnetization has the same polarity as the fixed layer. This MTJ is in Parallel State (P), with a relatively low electrical resistance. By measuring the resistance of the MTJ cell, the resistance-encoded memory data can be read out.

There are multiple switching mechanisms available to change the free layer's magnetization. For example, Spin Transfer Torque (STT) and Spin-Orbit Torque (SOT) were introduced decades ago and have been widely used in recent works [8], [28]–[30]. The STT-MRAM has the simple 1T1M structure, but it requires a high write current and leads to high programming energy [28]. Also, the STT-MRAM only has

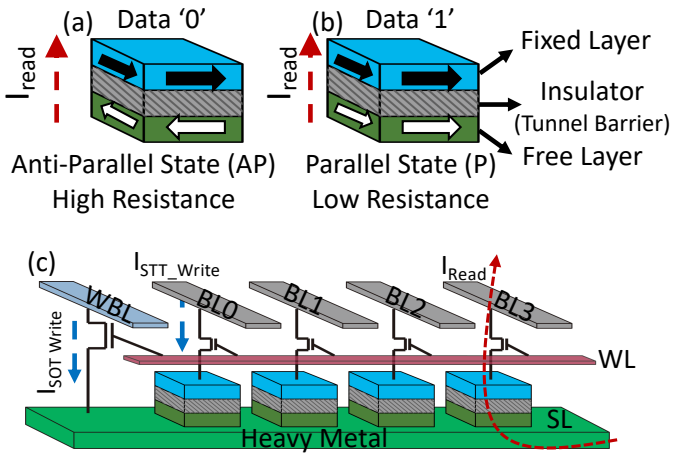

Fig. 1. (a) Anti-Parallel state with high resistance. (b) Parallel state with low resistance. (c) 4-MTJ SAS MRAM's structure with read and write paths.

two terminals, causing the read disturbance failure due to the same read and write path [29]. The SOT-MRAM separates the read and write current paths due to the extra heavy metal layer for writing. With the help of spin-orbit coupling and separated read/write path, SOT-MRAM requires much lower write energy and offers much more reliable operation. However, the extra write transistor seriously affects the high area density properties of MRAM.

B. Continual learning

Continual learning aims to continually learn multiple tasks that arrive in a sequential manner, meanwhile without forgetting prior learned knowledge. Plentiful continual learning methods have been developed and can be generally divided into three categories: 1) *Regularization-based methods* (e.g., [31]–[33]) preserve knowledge from old tasks by incorporating an additional regularization term in the loss function. This regularization constrains weight updates when learning new tasks. For instance, Elastic Weight Consolidation (EWC) [33] evaluates the importance of weights using the Fisher Information matrix and regularizes updates on important weights. 2) *Structure-based methods* (e.g., [20], [34]–[37]) adapt model parameters or architectures sequentially as new tasks are introduced. 3) *Memory-based methods* can be further divided into memory-replay methods and orthogonal-projection-based methods. Memory-replay methods (e.g., [38]–[40]) store and replay data from old tasks when learning new tasks. On the other hand, orthogonal-projection-based methods (e.g., [41]–[45]) update the model for each new task in a direction orthogonal to the subspace spanned by the inputs of old tasks.

When considering NVM-based accelerator designs, it's essential to keep in mind that NVMs typically exhibit limited endurance for data rewriting and consume substantially more energy for writing or updating data compared to reading data. Consequently, retraining or updating the entire model using memory replay or regularization terms may not be well-suited to these constraints. Additionally, due to the chain rule involved in backpropagation, saving all intermediate activations and gradients becomes necessary for calculating subsequent or preceding layers. Depending on the batch size, this intermediate data can potentially become much larger than the model's weights.

To address this challenge, [20] introduced an innovative structure known as Rep-Net. Rep-Net incorporates a parallel re-programmable tiny branch alongside the main backbone model, and these two branches are combined through element-wise addition. By avoiding multiple activations and keeping the backbone model fixed, the training of the small parallel re-programmable branch does not require the storage of activations and gradients on the larger backbone model. Consequently, training Rep-Net demands significantly fewer resources, making on-device learning a feasible prospect.

However, it's worth noting that the original Rep-Net still relies on floating-point-based operations, which can be inefficient in NVM-based accelerator designs. To overcome this inefficiency, we have integrated a fully integer-based training approach with Rep-Net, named INT8-Rep-Net, to enable NVM-friendly on-device learning.

C. Inference and Training with Weight Quantization

A common approach to compress DNN for the resource-limited device is quantizing 32-bit floating-point numbers (FP32) to low-bit discrete representations, i.e., 8-bit integers (INT8).

Two widely used DNN quantization approaches are Post-training quantization (PTQ) and Quantization-aware training (QAT) [46], [47]. Both are supported by popular DNN frameworks such as Pytorch and TensorFlow. PTQ takes a pre-trained FP32 network and converts it directly into a fixed-point/integer network without re-training. Moreover, PTQ has little hyperparameter tuning, making it easy to use. The fundamental step in the PTQ process is finding suitable quantization ranges for each layer/channel. Then it uses the pre-defined low-bit fixed-point/integer to represent the FP32 weight & input to the closest level.

However, when aiming for low-bit quantization of activations, such as 4-bit and below, it is difficult for PTQ to mitigate the quantization error incurred by low-bit quantization. Quantization-aware training (QAT) has been proposed to solve this, considering the quantization error/noise during training, which achieves much better accuracy. However, the higher accuracy comes with the expected higher cost in hyperparameter search and training time.

Although PTQ and QAT can successfully quantize DNN models while keeping high accuracy, they mainly target to generate weight quantized models for inference but still use high computation cost floating-point numbers during training. It thus makes on-device learning to be extremely difficult in resource-limited devices.

III. STT-Assisted-SOT MRAM (SAS-MRAM)

A. SAS-MRAM

Recent advancements in MTJ devices, highlighted by the experimental work of Garello et al [48]. and Hu et al. [49], have significantly propelled the field of memory technology. [48] demonstrates the first full-scale integration of

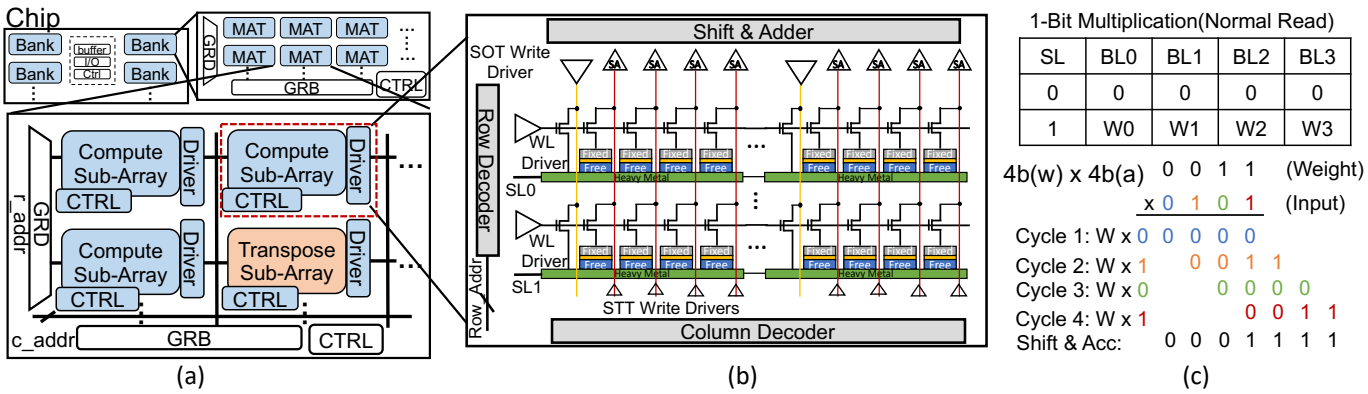

Fig. 2. (a) Proposed SAS-MRAM based IMC architecture. (b) compute sub-array for MAC operation. (c) example bit-serial matrix MAC flow.

top-pinned perpendicular MTJs on a 300mm wafer using CMOS-compatible processes with a notable 210ps SOT-driven switching. This development underscores the potential of SOT-MRAM to replace SRAM cache due to its large endurance, sub-nanosecond switching times, and low power operation. On the other hand, [49] introduces Double Spin-Torque Magnetic Tunnel Junctions (DS-MTJs) in STT-MRAM showcases a reliable 300ps STT-driven switching, along with improved activation energy and magnetoresistance, indicating its potential for last-level cache applications. To get the benefit from both STT-MRAM and SOT-MRAM and avoid/mitigate their deficiency, the STT-Assisted-SOT MRAM (SAS-MRAM) has been proposed [13], [17]–[19]. The SAS-MRAM shares the SOT write path (i.e., heavy metal layer) with multiple MTJs to separate the read path and write path, leading to several times memory density improvement compared to SOT-MRAM, as shown in Fig. 1(c). In SAS-MRAM device, writing can be performed in a single step and does not require a bidirectional SOT current as in traditional SOT-MRAM. First, a strong current pulse, sufficient to overcome the device's anisotropy, is applied to the heavy metal layer. The strong SOT torque effectively neutralizes the device's state such that the free layer of the MTJ is suspended midway between the P and the AP states. Subsequently, it applies a small STT current to each bitline and releases the SOT pulse. Here, a small STT torque is sufficient to deterministically break the symmetry between the P and AP states, causing each bit to relax to the desired magnetic state. In this process, both the SOT and STT current are much smaller than the SOT-/STT-only switching mechanism to save writing energy, while lowering average transistor counts per bit compared to STT-MRAM. To facilitate the read operation in SAS-MRAM, the process begins by applying a small SL read voltage to the shared heavy metal layer. The gate terminals of the MTJs' access transistors are connected to a common wordline (WL) for the simultaneously read operation. Upon activation of this WL, the P/AP states of the MTJs are manifested as low/high resistance levels, respectively. Consequently, the data stored in the SAS-MRAM is easily retrieved by sensing and interpreting the current in each BL, which directly correspond to the MTJ's resistance states. Our fabricated 4-MTJ SAS-MRAM device prototype performance is shown in the later experiment section.

B. Related works

Recently, researchers are increasingly drawn to the remarkable efficiency of SAS-MRAM, leading to the emergence of SAS-MRAM-based DNN accelerator designs. In reference to [19] and [13], these studies suggest the stacking of 4 MTJs on the same heavy metal layer to create a multi-level MRAM device. This innovative design leverages 4 MTJs per cell, enabling the representation of 5 distinct states within each cell: 4AP, 3AP/1P, 2AP/2P, 1AP/3P, and 4P. During both read and MAC operations, a small read voltage is simultaneously applied to all 4 MTJs. Subsequently, the read currents are accumulated on the BL and detected by an ADC to determine the outcome.

However, this analog computing paradigm doesn't fully exploit the potential of SAS-MRAM for several reasons. First, in a fully digital design, the 4 MTJs could potentially represent up to $2^4 = 16$ states, resulting in significantly higher storage density compared to the analog computing system. Second, the use of power-hungry and area-consuming ADCs diminishes the efficiency gains offered by SAS-MRAM. Finally, both [19] and [13] exclusively focus on using SAS-MRAM as the inference accelerator, while overlooking its potential for on-device learning. These factors serve as strong motivations for us to introduce a comprehensive fully digital framework based on SAS-MRAM. This framework not only facilitates inference but also enables on-device continual learning.

IV. IMC with STT-Assisted-SOT MRAM

A. Overview of proposed IMC architecture and circuits

Fig. 2 shows the overview of the proposed IMC architecture and circuits with SAS-MRAM, where each memory cell consists of 4 MTJs sharing the same heavy metal line representing 4 bits every memory cell macro. The overall system consists of I/O interface for data exchange, buffers to latch input or temporarily store the intermediate result, and interface controller to decode instruction. Multiple *compute* and *transposable* memory sub-arrays are grouped as MAT, and multiple MATs are grouped to construct the bank, in H-tree structure. As the base component, compute and transposable sub-arrays consist of SAS-MRAM array (with each cell macro of 4 MTJs), row/col decoder, WL drivers, SOT/STT write drivers, sense

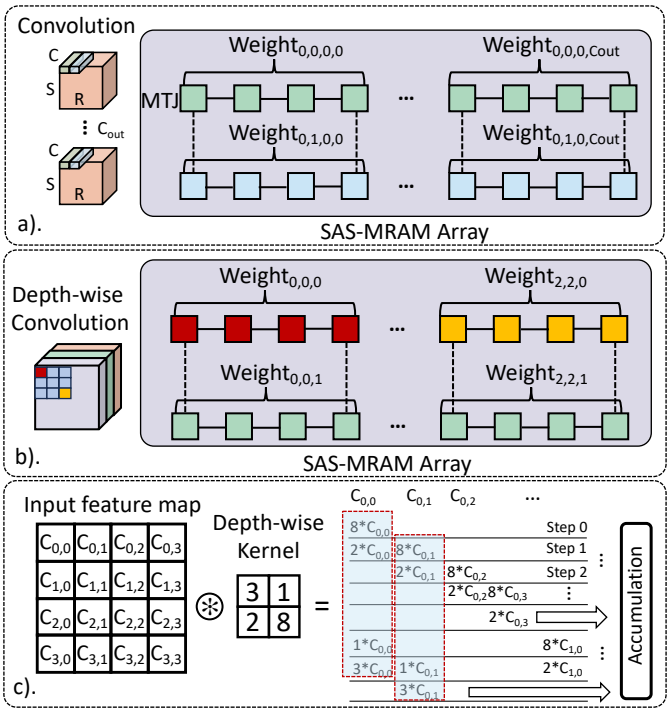

Fig. 3. (a) Mapping conventional convolution on SAS-MRAM array. (b) Mapping depth-wise convolution on SAS-MRAM array. (c) Dataflow of depth-wise convolution.

amplifiers, shift & adders, and local/global buffers. Each compute and transposable sub-array is designed to perform digital bit-serial Vector-Matrix Multiplication (VMM) to support in-memory convolution operation.

B. Continual Learning Model Mapping

As illustrated in Fig. 3 the convolution weight kernel is mapped to the SAS-MRAM array. Since each SAS-MRAM cell has 4 MTJs, i.e., 4 bits, we use two adjacent memory cells in the same world line to represent one 8-bit weight parameter. The convolution input (i.e., activation) is stored in the local buffer, which is either from the previous layer or from I/O interface. It is fed into the SAS-MRAM array in a bit-serial manner.

Similar to previous studies [10], [12], [50], in order to maximize parallelism and data reuse, the conventional convolution kernels are unrolled based on the output feature dimension. Kernels that correspond to the same output feature are strategically mapped onto identical columns, and those mapped onto the same rows are designed to share the same input data as shown in Fig. 3 (a). By implementing this data mapping and activating the SAS-MRAM array in a sequential row-by-row order, convolutions across various kernels (represented as columns in the SAS-MRAM) can be executed simultaneously, thereby achieving high parallelism.

Regarding depth-wise convolutions, where each input feature map is associated with its own 2D convolution kernel, the result retains the same dimension as the input feature map. This particular character makes it impractical to unroll the convolution kernel along the output feature dimension, thus

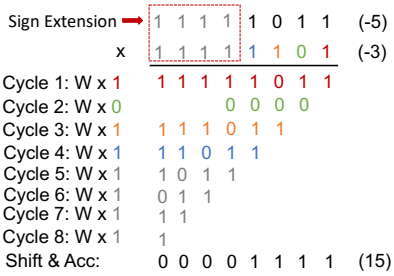

Fig. 4. Sign extension to handle the negative operands in shift-and-accumulation based multiplication.

mandating an alternate mapping strategy specifically for depth-wise convolution. Fig.3 (c) illustrates the data flow for depth-wise convolution. It is evident that while there is no input sharing between different kernels, every element in the input feature map engages in multiplication with every element in the 2D convolution kernels. The results of these multiplications are temporarily stored in a local buffer and then incrementally combined with operands from subsequent cycles. Given SAS-MRAM's exclusive support for multiplication, each 2D depth-wise convolution kernel is linearly unrolled into a 1D vector and mapped on the same row in SAS-MRAM array, as depicted in Fig.3 (b). Upon row activation, every element within the convolution kernel concurrently multiplies with the same input through the shift-and-accumulate process. These products are then preserved in the local buffer and methodically summed in a predefined order to generate the result of depth-wise convolution.

The continual learning model contains two different modules: fixed module and learnable module. Given that the fixed modules within the continual learning model are solely engaged in the forward pass, whereas the learnable modules participate in both the forward and backward passes, a clear distinction in their involvement is evident. In order to efficiently support the VMM, a compute sub-array is employed to facilitate the mapping of the fixed modules, mainly supporting the DNN forward operation. The learnable modules in the continual learning model not only necessitate VMM but also require VMM with a transposed weight matrix during back-propagation. Consequently, a specially designed transposable sub-array is implemented to accommodate the mapping of the learnable modules to support both forward and backward operations. The size of the learnable module is much smaller than the fixed module (e.g., 5%), which determines the ratio of compute sub-arrays over transposable sub-arrays.

C. Compute Sub-Array for Forward Pass

To better explain the digital bit-serial in-memory convolution process, as shown in Fig. 2(c), for simplicity, we use a 4-bit input (4b(a)) and 4-bit weight (4b(w)) as an example. It could be easily extended to 8-bit convolution. For bit-serial convolution, two steps are needed: **1)** one bit partial product (i.e., bitwise 'AND' operation implemented using our memory cell design), and **2)** shift & accumulation, implemented using the digital peripheral circuits associated with each memory array.

For the one bit partial product (i.e., bitwise AND operations), the input bit is broadcasted to the gate terminals of

the read access transistors connected with the fixed layer of each MTJ on the same row. Then, based on the stored weight value (i.e., the high/low resistance level connected with the drain terminal of the read access transistor), the sense amplifier (SA) associated with each bit-line could detect the resistance level (i.e. AND output). For example, if the input bit is '1', then only the low resistance of MTJ (i.e., representing weight value as '1'), will make the SA outputs '1'. It can be seen that the in-memory-AND operation could be naturally implemented through the existing read access transistor and SA by reprogramming the wordline driver to the convolution input bits. If the input bit is '0' (i.e. gate terminal voltage is 0), no matter what weight value is stored in the MTJ cell, the output from the SA should be always '0'. Thus, in this case, all the read access transistors are off, and the row decoder skips the enable signal for SA making the SA outputs keep on '0'. It's worth highlighting that, in contrast to the analog-based IMC design, only a single row is activated within the same compute sub-array. This distinction arises from the implementation of the in-memory-AND operation through the standard read operation. To streamline the design and minimize overhead, we can leverage the well-established WL driver and SA design from mature MRAM-based storage designs. Furthermore, this compute sub-array can also serve as storage, adding versatility to its functionality. When the next bit is fed into the array, the previous cycle's result is collected at the shift&adder peripheral digital circuits to be shifted and added with the following cycle result. Through such a way, a fully digital multi-bit multiplication could be implemented in a pipeline style as shown in Fig. 2(c).

Given that both the weight and input of the convolution can be either positive or negative, our design employs a 2's complement format to effectively manage negative operands. This approach aligns closely with the primary steps involved in the aforementioned unsigned integer scenario. Initially, the Sense Amplifier (SA) fetches the weight based on the input bit-serials through the in-memory-AND operation facilitated by SAS-MRAM. Subsequently, the result undergoes an extension process to avoid data overflow during the shift&accumulation phase. Contrary to the unsigned scenario where extended bits are filled with '0', in the 2's complement format, these bits are filled with the sign bit. Specifically, negative values are extended with '1's, and positive values with '0's to ensure computational correctness. This process is exemplified in Fig. 4.

D. Transposable Sub-Array for Forward and Backward Passes

The forward inference computation mainly requires the above-discussed in-memory convolution. In order to support on-device learning, we also designed the transposable sub-array to support both forward and backward passes. The forward pass is the same as the compute sub-array. In backward pass, the main computations required are *error propagation*, *gradient calculation*, and *weight update*, as discussed in the next section, equations 1-3. Those major convolution-related computations still leverage our existing forward circuits. While, as shown in equation-1 and -2, the error e and learnable

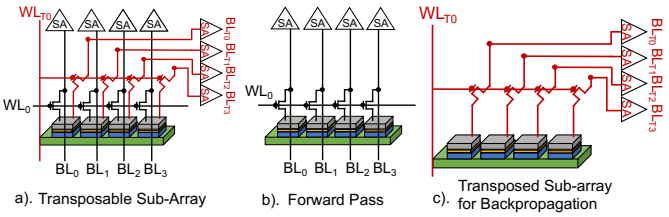

Fig. 5. Transposable sub-array design for on-device learning support.

weight w need extra transpose operations. For error e, since it propagates layer by layer, it will be stored in the local buffer as the intermediate data. The local controller could effortlessly implement the transpose of e through the reading buffer in a transposed order. In contrast, the transpose of learnable w needs an extra read access transistor per cell to support transposed read, i.e., transposed in-memory-AND operation, while the follow-up shift & add circuits could be shared. However, as we will introduce our INT8-Rep-Net algorithm in the next section, it's important to note that only a small portion (approximately $\sim 5\%$) of the total weights needs to be updated for on-device new task learning. Thus, only a small portion of weight memory needs the transposed design to support weight updates.

Figure 5 illustrates the transposable sub-array circuit design, which builds upon the compute sub-array, incorporating additional components, such as the transpose transistor, WL, BL, and peripheral circuits to enable both horizontal and vertical read access. In the transposable array figure, we only show the circuit of one 4-MTJ memory macro to illustrate how to support both forward and backward computations. For simplicity, we omit the transistor used for the SOT path since it is solely used for writing data to the SAS-MRAM device, not in the compute stage leveraging read operations. As depicted in Figure 5(b), in the forward pass, the sense amplifier, access transistor, and BL located atop the SAS-MRAM are activated, while the transpose WL/BL/access transistor/peripheral circuit remain deactivated. Conversely, during the backpropagation pass, the activated components are opposite to the forward pass. Specifically, the transpose access transistor establishes a connection between SAS-MRAM cells within the same row, forming the transpose BL, BL_T, and transpose WL, WL_T. Consequently, the sense amplifier on the right side of the array can retrieve the computation result with the transposed weight matrix w^T, as illustrated in Figure 5(c).

V. INT8-Rep-Net for On-Device Continual Learning

In this section, we introduce the proposed INT8-Rep-Net for on-device continual training framework. As discussed in our prior work [20], the Rep-Net structure is a memory-efficient continual learning method. As shown in Fig. 6, it has a weight-frozen backbone model as the main path (trained offline on large-scale background dataset, e.g., imagenet), and a small learnable path (i.e., Rep-Net) to learn new task-specific feature and accommodate the intermediate activation, for new downstream task data. This scheme has demonstrated state-of-the-art new task adaption in both performance and training

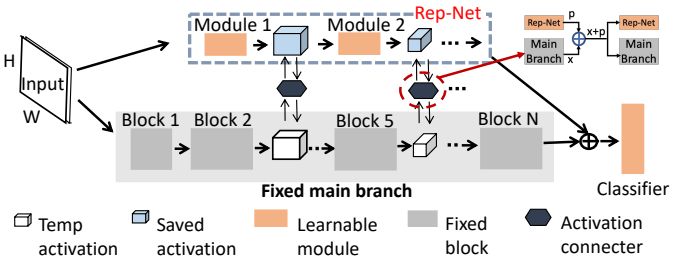

Fig. 6. Reprogramming Network (Rep-Net) for new task learning.

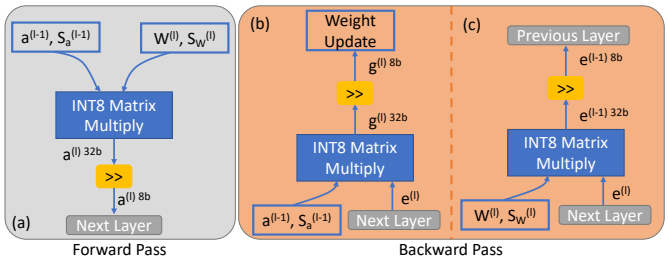

Fig. 7. Main process graph for both forward pass and backward pass. The fixed main branch only needs to support the forward pass. The learnable module needs to support both forward and backward passes for training.

memory/computation cost reduction. While the original Rep-Net is learned in floating-numbers (FP32), we draw inspiration from recent studies on integer-only training methods [24]–[27]. In this work, we propose and redesign the continual learning process with only INT8 operations to make it IMC hardware friendly and further reduce computation/memory cost.

A. Reprogramming Network (Rep-Net)

In the Rep-Net architecture (Fig. 6), the input is fed to the backbone and Rep-Net paths in parallel. The working principle is to interchange its intermediate features with the backbone model through activation connector. In this way, backbone and Rep-Net can mutually benefit and improve the overall performance of new task learning. The objective is to adjust the intermediate activation in a new domain of knowledge for continual learning. Note that, it is designed for task-specific continual learning, i.e., one new Rep-Net for one new task to alleviate the *catastrophic forgetting*. To minimize the memory overhead during the training phase and enable the on-device training feature, the backbone model is fixed (i.e., both weights and architecture) as highlighted in grey in Fig. 6. Only the Rep-Net path and the shared last classification layer (highlighted in orange) are learned on the new task data. Compared with the backbone model, the Rep-Net path is a lightweight neural network with only a few convolution modules and minor memory overhead (i.e., activation storage and additional parameters). Although Rep-Net significantly saves the memory cost during continual learning, it does not change the high computation demand of training since the computation during backward propagation is still performed with floating-point (FP32) operations. To reduce the high computation cost during training, in this work, we propose a new hardware-friendly learning method that uses INT8-only operations during the entire training process, instead of floating-point (FP32) number operations, to save both computation and memory cost (i.e., from 32-bit to 8-bit).

B. INT8 Training Scheme

As defined in IEEE 754, a floating-point number consists of a sign bit, exponential bits, and fraction bits representing an extensive range of decimal numbers. In contrast, the 8-bit binary number can only represent up to 256 levels. The weight/activation usually has a narrow distribution within the same layer. However, the weight/activation across different layers may be distributed over an extensive range. Simply using the 8-bit number for all layers would cause large quantization errors and lead to an unacceptable performance drop. Thus, traditional PTQ and QAT use the 8-bit or even less-bit quantization only in the forward pass. But PTQ and QAT methods still use the floating-point numbers during backpropagation to keep high precision, as gradient and error propagation are usually more sensitive to quantization error, especially in deeper models. To summarize, the following are the fundamental computations required during backpropagation:

$$\text{Error propagation}: \quad e^{l-1} = (W^l)^T \times e^l \tag{1}$$

$$\text{Gradient}: \quad g^l = a^l \times (e^l)^T \tag{2}$$

$$\text{Weight Update}: \quad W^l_{new} = W^l_{old} - g^l \tag{3}$$

Fig. 7 shows our key steps and method of INT8-Rep-Net training. Inspired by the observations of weight/activation distribution, we use a *shifting factor* (s_a), which is shared within the same layer, to improve the representation range. The shifting factor is deployed only in the trainable Rep-Net path and the last classifier layer during training, while the backbone main path model is frozen with pre-trained 8-bit quantization. As shown in Fig. 7, weight and activation have their own shifting factor and are shared within the same layer. The 'a' represents activation, 'w' represents weight, 'e' represents error, and 'g' represents gradient. The upper script 'l' represents the $l-th$ layer. The lower script 32 means it is a 32-bit number. Without explicit notation, the default number is 8-bit (INT8).

The key component in both forward and backward passes is the INT8 Matrix Multiply which takes 8-bit number operands and accumulates the result in 32-bit numbers. For example, given the activation a and its shifting factor s_a, the actual value is calculated as $a \times 2^{s_a}$. Since the shifting factors are shared by the entire layer, we can extract the shifting operation and perform it after the INT8 computation, namely, the convolution between $w \times 2^{s_w}$ and $a \times 2^{s_a}$ can be transferred as two steps: 1) calculate the INT8 multiplication $w \times a$; 2) shift it by $2^{s_w+s_a}$. The computation result is temporarily stored as 32-bit number in the buffer. To interface with the INT8 layers before and after, we need to shift it to 8-bit and update the shifting factor accordingly. Note that, during the backward pass, the weight and error matrix need to be transposed, which is supported by our hardware design. Then, the INT8 matrix

TABLE I
INT8-REP-NET ON-DEVICE CONTINUAL LEARNING ACCURACY

Configuration	Flowers	Cars	CUB	Food	Pets	Aircraft
Classifier-Only(FP32)	91	48.4	67.2	64.3	87.8	43.4
MobileNet-V2 FP32 + Rep-Net FP32 [20]	95	88	71.9	78.22	89.3	78.8
MobileNet-V2 INT8 + Rep-Net FP32	94.7	86.8	71.6	78.2	89.1	78.5
MobileNet-V2 INT8 + Rep-Net INT8	94.7	84.9	70.2	78.1	88.3	77.8

multiply operation is the same as the forward pass. Due to the limited range of weight in the integer training scheme, it isn't easy to adopt the learning rate concept. Therefore, we use the gradient to determine the magnitude of the update. Empirically, we observed it works well for the shallow INT8-Rep-Net path.

VI. EXPERIMENT RESULTS

A. INT8-Rep-Net Performance

In this section, we first evaluate the INT8-Rep-Net for learning new tasks. For a fair comparison, we choose the popular MobileNet-V2 [51] as our backbone model, which is pre-trained off-line on ImageNet dataset. We use the most popular six datasets for continual learning performance evaluation, including CUBS [52], Stanford Cars [53], Flowers [54], Food [55], Pets [56], and Aircraft [57]. For all the experiments, we report the accuracy of finetuning classification layer (i.e., baseline) and Rep-Net with different precision configurations, as in Table I.

In all experiments, we assign 6 learnable modules for INT8-Rep-Net, each module consisting of 1 pooling layer and 2 convolution layers. MobileNet-V2 (FP32) + Rep-Net (FP32) in [20] demonstrated the best state-of-the-art performance in algorithm with floating-point weights for both backbone and Rep-Net paths, serving as the SOTA baseline. To deploy in IMC, we first quantize the backbone model to 8-bit with QAT and leave Rep-Net in FP32, which shows slightly lower accuracy. In our method, we leverage our proposed INT8-training algorithm to only use INT8 operations during Rep-Net continual learning. As can be seen in Table I, our method with both backbone and Rep-net in 8-bit achieves close-to full precision accuracy with only ∼1% drop on average. While, the memory cost could be reduced by 4× from 32 bit to 8 bit, as well as great reduction in computing complexity for on-device learning.

B. SAS-MRAM Device Prototype and System Evaluation

TABLE II
SAS-MRAM DEVICE PROTOTYPE DATA

Symbol	Description	Value
α	damping constant	0.008
θ_{SHA}	spin Hall angle	0.6
M_s	saturation magnetization	1.3 MA/m
AR	MTJ aspect ratio	1.0(z-type) or 3.0(x-type)
t_E	free layer thickness	1 nm(z-type)
K_u	first order uniaxial anisotropy constant	0.31 MJ/m3(x-type)
R	Resistance (Low / High)	4408Ω / 8759Ω
E_{write}	Single bit Set/Reset Energy	0.048pJ

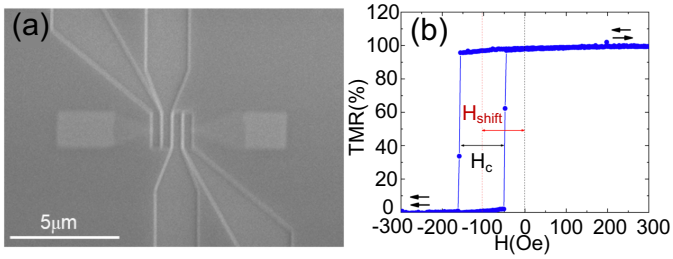

Fig. 8. (a)SEM micrograph of the top electrodes and SOT ohmic contacts in a 4-MTJ SAS device. (b)TMR (∼ 100%) curve of a 100nm × 50nm MTJ cell.

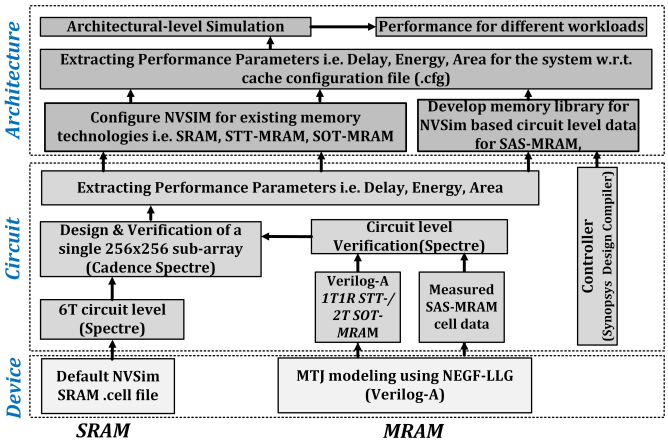

Fig. 9. Cross-layer device-architecture evaluation framework.

Fig. 8 displays the SEM micrograph of our fabricated 4-MTJ SAS-MRAM device prototype along with the measured TMR curve. The device parameters used in hardware evaluation are summarized in Table II, where our measured TMR is ∼98.7%.

To establish a comprehensive cross-layer device-architecture framework for system evaluation and comparison, we developed an in-house evaluation framework utilizing array-level circuit modeling techniques based on PIMA-SIM [60], NVSIM [61], TSMC 28nm Product Development Kit (PDK), and our experimentally benchmarked Verilog-A SAS MTJ model, as illustrated in Fig. 9. This framework contains the development and extraction of SPICE-compatible MRAM device model, circuit-level performance for sub-array circuits, and architecture-level performance estimation for peripheral circuits. For the characterization of bit-cell performance of MRAM, we employ a combined approach employing the Non-Equilibrium Green's Function (NEGF) and Landau-Lifshitz-Gilbert (LLG) equations, as previously discussed in [16],

TABLE III
COMPARISON WITH STATE-OF-THE-ART IMC BASED ACCELERATORS

Reference	Ours	Nature'22 [10]	JETCAS'22 [12]	TC'22 [14]	JSSC'19 [58]	ESSCIRC'22 [5]	ISSCC'22 [59]	JSSC'22 [50]
Technology	SAS-MRAM	STT-MRAM	STT-MRAM	SOT + STT	SRAM	SRAM	SRAM	ReRAM
Tech Node	28nm	28nm	65nm	28nm	65nm	65nm	28nm	40nm
Capability	Inference & Learning	Inference	Inference	Inference	Inference	Inference	Inference	Inference
Precision	8-bit	8-bit	BNN	8-bit/16-bit	BNN	BNN	8-bit	8-bit
Bit Cell Density	1.25T	2T-2M	2T-2M	2T-1M	8T	8T	6T	1T-1R
Supply Voltage	0.8V	0.8-1.8V	-	0.8-1.8V	0.68-1.2V	0.8V-1.2V	0.8V	0.9V
Max Frequency	200MHz	11.1MHz	-	-	100MHz	1230MHz	333MHz	-
Macro Size	4KB	512B	2KB	14.75KB	4.8KB	2KB	4KB	8KB
Performance (GOPS)	~3277(1b)	-	3280	-	295	1259.52	-	-
Energy Effi.(TOPS/W)	~131	405	319(1b/1b)	1.1(16b/16b)	20.6	34.98	27.38(8b/8b)	7(8b)

TABLE IV
HARDWARE SYSTEM SPECIFICATION

Component	Area(um2)	Energy(pJ)
Memory Array(64 x 512)	429.62	-
Column Decoder + Driver	243.18	7.9
Row Decoder + Driver	37.3	3.4
Shift Adder	2857.93	334.17
Adder Tree(128 units)	44113	814.9
Global Buffer(64 × 112 × 112 × 4)	6500,182	0.002/bit/access
Global ReLU	719.7	0.6

[62], [63]. These equations are instrumental in capturing the intricate behaviors exhibited by the bit-cells, which are experimentally benchmarked in this work with our fabricated SAS MTJ device data in Table II. At the circuit level, we construct 256 × 256 memory sub-array with peripheral circuits, simulated in Cadence Spectre with the 28 nm TSMC PDK library. Furthermore, to evaluate the timing, energy, and area characteristics of the various memory technologies at the architecture level, we leverage the well-established memory architecture evaluation tools NVSIM and PIMA-SIM. This combination of tools provides flexibility in memory configuration, allowing for the organization of banks, mats, and subarrays, as well as peripheral circuitry design.

The memory configuration is facilitated through the utilization of two levels of configuration files, enabling precise specification. Specifically, at the cell level, the device and circuit-level specifications are defined using NVSIM's .cell file. Conversely, at the architecture level, the memory organization and optimization targets are configured through NVSIM's .cfg file. This comprehensive configuration scheme allows for thorough exploration of the memory system and optimization possibilities. These methodologies leverage the detailed device and circuit-level data to provide a comprehensive understanding of the application performance within the studied platforms.

To map the entire model, we construct the memory with the following hierarchy: 16 × 16 banks, each bank has 2 × 2 MATs, and each MAT has a 64 × 512 subarray. Each cycle only activates 16 × 2 subarrays, and each subarray activates only one row. Thus, the whole memory can store 4MB of data, whereas the MobileNet-V2 has ~3.4M parameters.

Fig. 10 demonstrates the area breakdown for the proposed system design. In order to support the execution of MobilNet-V2 as the backbone model, the buffer should be large enough to fit the largest activation, 32×112×112. Also, the SAS-MRAM array should be large enough to store all the weight

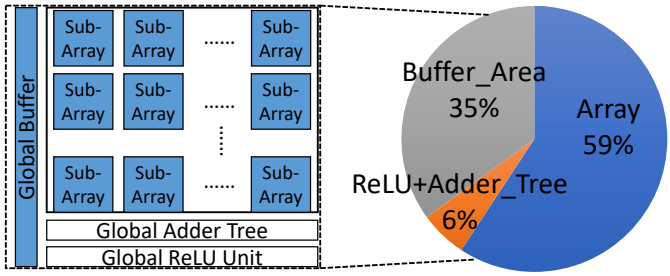

Fig. 10. Area breakdown of 8-bit MobileNet-V2 hardware deployment. The main peripheral circuit modules include ReLU module, adder tree, and buffers.

parameters. Note that, the array area here contains all circuits shown in Fig. 2(b), including memory array, row/col decoder, shift adder, etc.

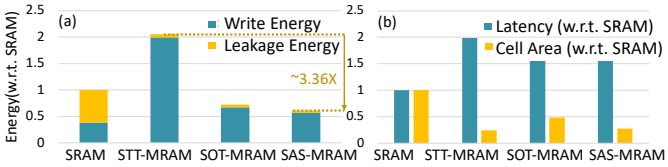

Fig. 11. (a) Energy consumption of updating MobileNet-V2 weights in one training epoch; (b) memory write latency and area comparison.

Fig. 11(a) reports the normalized energy consumption of updating the MobileNet-V2 weights in one training epoch. It could be seen that the memory write energy of SRAM is the smallest. However, if considering the leakage energy, both the SOT-MRAM and SAS-MRAM achieve much smaller weight updating energy. In particular, the SAS-MRAM achieves around 3.36× smaller writing energy than STT-MRAM counterparts. Fig. 11(b) reports the memory write latency and cell area comparison. It can been seen that SRAM has the smallest writing speed. However, due to 6 transistors per cell design, its area is the largest among all candidates. As expected, the 1T1M STT-MRAM area is the smallest. Due to the shared writing path between multiple MTJs in our SAS-MRAM design, the cell area is close to STT-MRAM and much smaller than the traditional 2T1M SOT-MRAM.

Table III compares our proposed SAS-MRAM based IMC accelerator design with other state-of-the-art IMC designs. While other designs in the field have primarily concentrated on enabling efficient inference processes, our proposed solution goes beyond that to incorporate the essential aspect of on-device learning.

Samsung [10] introduced an STT-MRAM based crossbar array that facilitates DNN model inference. This array incorporates a time-to-digital converter (TDC) readout circuit and employs a 2T-2M MRAM structure. In this design, the dot-product or Vector-Matrix Multiplication (VMM) operation is achieved by sensing the column resistance using TDC. To address the challenge of low resistance in individual Magnetic Tunnel Junctions (MTJs), it implemented a 2T-2M structure. This structure consists of two parallel 1T-1M STT-MRAM cells, resulting in a super-linear accumulation of column resistance, despite the relatively low resistance range of approximately 13K to 26K Ohms. However, it is worth noting that the 2T-2M structure is associated with certain drawbacks. It occupies a substantial area and necessitates a large amount of current and energy for data writing operations.

Pham et al. [12] utilized a similar 2T-2M structure to implement in-Memory XNOR operations, limiting its support to BNN models exclusively. Their approach demonstrates outstanding energy efficiency by focusing solely on the power consumption of the array itself, encompassing WL, BL, precharge circuits, and sense amplifiers. While this approach may be reasonable for small-scale macro-level designs, it becomes less applicable for large-scale designs. In larger designs, the energy consumption of interconnections, H-trees, and decoders significantly surpasses that of the MRAM array itself.

Kim et al. [14] introduced CRISP, a digital IMC architecture designed for parallel matrix multiplication. Their approach combines the use of STT and SOT-MRAM in their design, with STT-MRAM serving as the weight storage array and SOT-MRAM forming two computing arrays for multiplication and addition operations respectively. By adopting a digital IMC design, CRISP presents a solution that eliminates the necessity for power-intensive ADCs while enabling the execution of arbitrary precision DNN models with flexibility in cycle counts. However, it is important to note that this approach has a drawback in the form of a large number of required cycles, which affects the throughput and ultimately hampers energy efficiency as well.

The ReRAM-based design [50] performs analog-domain computing where the addition operation is implemented by accumulating current along the same bit-line. Then ADC converts such current back to the digital domain. Although ReRAM is another promising NVM with low leakage, due to the power-hungry ADCs, its overall energy efficiency ($\sim$ 7TOPS/W) is not as good as other designs. Other recent SRAM-based IMC designs [5], [58], [59] implement in-memory computing by either modifying the bit-cell design or adding extra digital logic peripheral circuits. Such modification leverages the SRAM's fast speed to achieve high throughput. However, these designs sacrifice area efficiency and inevitably lead to lower energy efficiency due to the high leakage power.

With consideration of interconnections, decoders, drivers, and other peripheral circuits, our SAS-MRAM-based fully digital design shows great performance in terms of ($\sim$ 3277 GOPS) and energy-efficiency ($\sim$ 131 TOPS/W). Its high energy efficiency comes from several aspects: 1) fully digital design without power-hungry ADCs; 2) NVM significantly reduces leakage power; 3) our SAS-MRAM cell requires only 5 transistors for every 4 bits; 4) our digital MAC operations could be efficiently implemented by our bit-serial and efficient in-memory-AND circuit design leveraging the existing memory read circuits.

VII. Conclusion

In this work, we are the first to demonstrate an efficient on-device continual learning system with SAS-MRAM based in-memory computing design. Different cross-layer hardware and algorithm co-designs have been proposed to improve the overall system performance. From the hardware side, we fabricated 4-MTJ SAS-MRAM device prototype and design fully digital in-memory computing circuits to support both the inference and on-device learning operations. From the algorithm side, we presented efficient 8-bit reporgamming network for IMC-friendly on-device continual learning algorithm. Comprehensive experiments have been conducted to prove the ultra-high energy efficiency of our developed on-device continual learning AI system.

VIII. Acknowledge

This research was supported in part by by the National Science Foundation (NSF) under Grant No. 2314591, No. 2349802, No. 2342726, and No. 2414603. Additionally, we acknowledge the support from Precourt Institute for Energy and the SystemX Alliance at Stanford, and NSF grants No. 2314591 (ACED Fab) and No. 2328804 (FuSe). Part of this work was performed at SNF and SNSF, supported by NSF award No. ECCS-2026822.

References

[1] S. Abdulrahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi, and M. Guizani, "A survey on federated learning: The journey from centralized to distributed on-site learning and beyond," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5476–5497, 2021.

[2] S. Mittal, "A survey of reram-based architectures for processing-in-memory and neural networks," *Machine Learning and Knowledge Extraction*, vol. 1, no. 1, pp. 75–114, 2019.

[3] C. Eckert, X. Wang, J. Wang, A. Subramaniyan, R. Iyer, D. Sylvester, D. Blaauw, and R. Das, "Neural cache: Bit-serial in-cache acceleration of deep neural networks," in *Proceedings of the 45th Annual International Symposium on Computer Architecture*, ser. ISCA '18, 2018, p. 383–396.

[4] D. Fan and S. Angizi, "Energy efficient in-memory binary deep neural network accelerator with dual-mode sot-mram," in *2017 IEEE International Conference on Computer Design (ICCD)*, 2017, pp. 609–612.

[5] A. Sridharan, S. Angizi, S. K. Cherupally, F. Zhang, J.-S. Seo, and D. Fan, "A 1.23-ghz 16-kb programmable and generic processing-in-sram accelerator in 65nm," in *ESSCIRC 2022- IEEE 48th European Solid State Circuits Conference (ESSCIRC)*, 2022, pp. 153–156.

[6] S. Li, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, and Y. Xie, "Drisa: A dram-based reconfigurable in-situ accelerator," in *2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2017, pp. 288–301.

[7] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *Proceedings of the 43rd International Symposium on Computer Architecture*, ser. ISCA '16. IEEE Press, 2016, p. 14–26. [Online]. Available: https://doi.org/10.1109/ISCA.2016.12

[8] Y. Luo, P. Kumar, Y.-C. Liao, W. Hwang, F. Xue, W. Tsai, S. X. Wang, A. Naeemi, and S. Yu, "Performance benchmarking of spin-orbit torque magnetic ram (sot-mram) for deep neural network (dnn) accelerators," in *2022 IEEE International Memory Workshop (IMW)*, 2022, pp. 1–4.

[9] Z. He, S. Angizi, and D. Fan, "Accelerating low bit-width deep convolution neural network in mram," in *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2018, pp. 533–538.

[10] S. Jung, H. Lee, S. Myung, H. Kim, S. K. Yoon, S.-W. Kwon, Y. Ju, M. Kim, W. Yi, S. Han, B. Kwon, B. Seo, K. Lee, G.-H. Koh, K. Lee, Y. Song, C. Choi, D. Ham, and S. J. Kim, "A crossbar array of magnetoresistive memory devices for in-memory computing," *Nature*, vol. 601, no. 7892, pp. 211–216, Jan 2022. [Online]. Available: https://doi.org/10.1038/s41586-021-04196-6

[11] J. Doevenspeck, K. Garello, B. Verhoef, R. Degraeve, S. Van Beek, D. Crotti, F. Yasin, S. Couet, G. Jayakumar, I. A. Papistas, P. Debacker, R. Lauwereins, W. Dehaene, G. S. Kar, S. Cosemans, A. Mallik, and D. Verkest, "Sot-mram based analog in-memory computing for dnn inference," in *2020 IEEE Symposium on VLSI Technology*, 2020, pp. 1–2.

[12] T.-N. Pham, Q.-K. Trinh, I.-J. Chang, and M. Alioto, "Stt-bnn: A novel stt-mram in-memory computing macro for binary neural networks," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 12, no. 2, pp. 569–579, 2022.

[13] J. Doevenspeck, K. Garello, S. Rao, F. Yasin, S. Couet, G. Jayakumar, A. Mallik, S. Cosemans, P. Debacker, D. Verkest, R. Lauwereins, W. Dehaene, and G. Kar, "Multi-pillar sot-mram for accurate analog in-memory dnn inference," in *2021 Symposium on VLSI Technology*, 2021, pp. 1–2.

[14] T. Kim, Y. Jang, M.-G. Kang, B.-G. Park, K.-J. Lee, and J. Park, "Sot-mram digital pim architecture with extended parallelism in matrix multiplication," *IEEE Transactions on Computers*, vol. 71, no. 11, pp. 2816–2828, 2022.

[15] N. Xu, Y. Lu, W. Qi, Z. Jiang, X. Peng, F. Chen, J. Wang, W. Choi, S. Yu, and D. S. Kim, "Stt-mram design technology co-optimization for hardware neural networks," in *2018 IEEE International Electron Devices Meeting (IEDM)*, 2018, pp. 15.3.1–15.3.4.

[16] S. Angizi, Z. He, A. Awad, and D. Fan, "Mrima: An mram-based in-memory accelerator," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 5, pp. 1123–1136, 2019.

[17] W. Hwang, F. Xue, F. Zhang, M.-Y. Song, C.-M. Lee, E. Turgut, T. C. Chen, X. Bao, W. Tsai, D. Fan, and S. X. Wang, "Energy efficient computing with high-density, field-free stt-assisted sot-mram (sas-mram)," *IEEE Transactions on Magnetics*, vol. 59, no. 3, pp. 1–6, 2023.

[18] Y. Takeuchi, C. Zhang, A. Okada, H. Sato, S. Fukami, and H. Ohno, "Spin-orbit torques in high-resistivity-w/cofeb/mgo," *Applied Physics Letters*, vol. 112, no. 19, p. 192408, 2018. [Online]. Available: https://doi.org/10.1063/1.5027855

[19] S. T. Ahmed, K. Danouchi, M. Hefenbrock, G. Prenat, L. Anghel, and M. B. Tahoori, "Scalable spintronics-based bayesian neural network for uncertainty estimation," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2023, pp. 1–6.

[20] L. Yang, A. S. Rakin, and D. Fan, "Rep-net: Efficient on-device learning via feature reprogramming," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 12 277–12 286.

[21] H. Cai, C. Gan, L. Zhu, and S. Han, "Tinytl: Reduce activations, not trainable parameters for efficient on-device learning," 2020. [Online]. Available: https://arxiv.org/abs/2007.11622

[22] T. Chen, B. Xu, C. Zhang, and C. Guestrin, "Training deep nets with sublinear memory cost," 2016. [Online]. Available: https://arxiv.org/abs/1604.06174

[23] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan, "Training deep neural networks with 8-bit floating point numbers," 2018. [Online]. Available: https://arxiv.org/abs/1812.08011

[24] M. Wang, S. Rasoulinezhad, P. H. W. Leong, and H. K.-H. So, "Niti: Training integer neural networks using integer-only arithmetic," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 11, pp. 3249–3261, 2022.

[25] S. Wu, G. Li, F. Chen, and L. Shi, "Training and inference with integers in deep neural networks," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=HJGXzmspb

[26] Y. Yang, S. Wu, L. Deng, T. Yan, Y. Xie, and G. Li, "Training high-performance and large-scale deep neural networks with full 8-bit integers," 2019.

[27] Y. Yang, X. Chi, L. Deng, T. Yan, F. Gao, and G. Li, "Towards efficient full 8-bit integer dnn online training on resource-limited devices without batch normalization," 2021.

[28] Y. Seo, K.-W. Kwon, X. Fong, and K. Roy, "High performance and energy-efficient on-chip cache using dual port (1r/1w) spin-orbit torque mram," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 6, no. 3, pp. 293–304, 2016.

[29] K.-W. Kwon, X. Fong, P. Wijesinghe, P. Panda, and K. Roy, "High-density and robust stt-mram array through device/circuit/architecture interactions," *IEEE Transactions on Nanotechnology*, vol. 14, no. 6, pp. 1024–1034, 2015.

[30] F. Zhang, S. Angizi, N. A. Fahmi, W. Zhang, and D. Fan, "Pim-quantifier: A processing-in-memory platform for mrna quantification," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 2021, pp. 43–48.

[31] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 139–154.

[32] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang, "Overcoming catastrophic forgetting by incremental moment matching," *Advances in neural information processing systems*, vol. 30, 2017.

[33] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," vol. 114, no. 13. National Acad Sciences, 2017, pp. 3521–3526.

[34] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," *arXiv preprint arXiv:1708.01547*, 2017.

[35] J. Yoon, S. Kim, E. Yang, and S. J. Hwang, "Scalable and order-robust continual learning with additive parameter decomposition," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=r1gdj2EKPB

[36] J. Serra, D. Suris, M. Miron, and A. Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4548–4557.

[37] L. Yang, S. Lin, J. Zhang, and D. Fan, "Grown: Grow only when necessary for continual learning," *arXiv preprint arXiv:2110.00908*, 2021.

[38] M. Riemer, I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, and G. Tesauro, "Learning to learn without forgetting by maximizing transfer and minimizing interference," *arXiv preprint arXiv:1810.11910*, 2018.

[39] Y. Guo, M. Liu, T. Yang, and T. Rosing, "Improved schemes for episodic memory-based lifelong learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1023–1035, 2020.

[40] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with a-gem," *arXiv preprint arXiv:1812.00420*, 2018.

[41] G. Zeng, Y. Chen, B. Cui, and S. Yu, "Continual learning of context-dependent processing in neural networks," *Nature Machine Intelligence*, vol. 1, no. 8, pp. 364–372, 2019.

[42] M. Farajtabar, N. Azizan, A. Mott, and A. Li, "Orthogonal gradient descent for continual learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 3762–3773.

[43] G. Saha, I. Garg, and K. Roy, "Gradient projection memory for continual learning," *arXiv preprint arXiv:2103.09762*, 2021.

[44] S. Lin, L. Yang, D. Fan, and J. Zhang, "Trgp: Trust region gradient projection for continual learning," *arXiv preprint arXiv:2202.02931*, 2022.

[45] ——, "Beyond not-forgetting: Continual learning with backward knowledge transfer," *arXiv preprint arXiv:2211.00789*, 2022.

[46] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan, "Pact: Parameterized clipping activation for quantized neural networks," *arXiv preprint arXiv:1805.06085*, 2018.

[47] J. Choi, S. Venkataramani, V. Srinivasan, K. Gopalakrishnan, Z. Wang, and P. Chuang, "Accurate and efficient 2-bit quantized neural networks." in *MLSys*, 2019.

[48] K. Garello, F. Yasin, S. Couet, L. Souriau, J. Swerts, S. Rao, S. Van Beek, W. Kim, E. Liu, S. Kundu, D. Tsvetanova, K. Croes, N. Jossart, E. Grimaldi, M. Baumgartner, D. Crotti, A. Fumémont, P. Gambardella, and G. Kar, "Sot-mram 300mm integration for low power and ultrafast embedded memories," in *2018 IEEE Symposium on VLSI Circuits*, 2018, pp. 81–82.

[49] G. Hu, C. Safranski, J. Z. Sun, P. Hashemi, S. L. Brown, J. Bruley, L. Buzi, C. P. D'Emic, E. Galligan, M. G. Gottwald, O. Gunawan, J. Lee, S. Karimeddiny, P. L. Trouilloud, and D. C. Worledge, "Double spin-torque magnetic tunnel junction devices for last-level cache applications," in *2022 International Electron Devices Meeting (IEDM)*, 2022, pp. 10.2.1–10.2.4.

[50] J.-H. Yoon, M. Chang, W.-S. Khwa, Y.-D. Chih, M.-F. Chang, and A. Raychowdhury, "A 40-nm, 64-kb, 56.67 tops/w voltage-sensing computing-in-memory/digital rram macro supporting iterative write with verification and online read-disturb detection," *JSSC*, 2022.

[51] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," 2019. [Online]. Available: https://arxiv.org/abs/1801.04381

[52] C. Wah *et al.*, "The Caltech-UCSD Birds-200-2011 Dataset," California Institute of Technology, Tech. Rep. CNS-TR-2011-001, 2011.

[53] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3d object representations for fine-grained categorization," in *2013 IEEE International Conference on Computer Vision Workshops*, 2013, pp. 554–561.

[54] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *2008 Sixth Indian Conference on Computer Vision, Graphics Image Processing*, 2008, pp. 722–729.

[55] L. Bossard, M. Guillaumin, and L. Van Gool, "Food-101 – mining discriminative components with random forests," in *European Conference on Computer Vision*, 2014.

[56] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar, "Cats and dogs," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3498–3505.

[57] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi, "Fine-grained visual classification of aircraft," 2013.

[58] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A 64-tile 2.4-mb in-memory-computing cnn accelerator employing charge-domain compute," *JSSC*, 2019.

[59] B. Yan, J.-L. Hsu, P.-C. Yu, C.-C. Lee, Y. Zhang, W. Yue, G. Mei, Y. Yang, Y. Yang, H. Li, Y. Chen, and R. Huang, "A 1.041-mb/mm2 27.38-tops/w signed-int8 dynamic-logic-based adc-less sram compute-in-memory macro in 28nm with reconfigurable bitwise operation for ai and embedded applications," in *ISSCC*, vol. 65, 2022, pp. 188–190.

[60] S. Angizi, S. Tabrizchi, and D. Fan, "Pima-sim 1.0: A simple & flexible processing-in-memory support for nvsim," https://github.com/ASU-ESIC-FAN-Lab/PIMA-SIM.

[61] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE TCAD*, vol. 31, no. 7, pp. 994–1007, 2012.

[62] S. Angizi, Z. He, D. Reis, X. S. Hu, W. Tsai, S. J. Lin, and D. Fan, "Accelerating deep neural networks in processing-in-memory platforms: Analog or digital approach?" in *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2019, pp. 197–202.

[63] X. Fong, Y. Kim, K. Yogendra, D. Fan, A. Sengupta, A. Raghunathan, and K. Roy, "Spin-transfer torque devices for logic and memory: Prospects and perspectives," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 1, pp. 1–22, 2016.