BitS-Net: Bit-Sparse Deep Neural Network for Energy-Efficient RRAM-Based Compute-In-Memory

Foroozan Karimzadeh, *Graduate Student Member, IEEE*, Jong-Hyeok Yoon, *Member, IEEE*, and Arijit Raychowdhury, *Fellow, IEEE*

***Abstract*— The rising popularity of intelligent mobile devices and the computational cost of deep learning-based models call for efficient and accurate on-device inference schemes. We propose a novel model compression scheme that allows inference to be carried out using bit-level sparsity, which can be efficiently implemented using in-memory computing macros. In this paper, we introduce a method called BitS-Net to leverage the benefits of bit-sparsity (where the number of zeros are more than number of ones in binary representation of weight/activation values) when applied to compute-in-memory (CIM) with resistive RAM (RRAM) to develop energy efficient DNN accelerators operating in the inference mode. We demonstrate that BitS-Net improves the energy efficiency by up to 5x for ResNet models on the ImageNet dataset.**

***Index Terms*— Deep neural network, quantization, in memory computing, DNN accelerator.**

I. Introduction

RISING popularity of intelligent edge devices requires accurate data analysis to be performed at the edge. Recently, deep neural network (DNN) methods have been shown to out-perform classical machine learning methods and has become the state-of-the-art in many applications such as natural language processing [1], computer vision [2], [3], healthcare data analysis [4] and autonomous vehicles [5]. However, these powerful methods are usually large, over-parameterized and computationally heavy [6] which prohibit DNNs to run on resource constrained (battery, memory) embedded mobile applications and edge devices.

DNN compression methods such as quantization [7], pruning [8], [9] and low-rank decomposition [10] have been proposed in the literature to compress DNNs and enable them to run on embedded mobile applications [11]. Low-precision DNNs decrease the required storage memory as well as computational complexity [12]. After quantization, most of the multiplication-and-accumulate (MAC) operations can be replaced by simple bit-wise operations which cause large reductions in memory requirement and hardware complexity, enables us to fit the models in on-chip memory, increase energy efficiency and enhance speed during inference. However, a large reduction in precision causes information loss which can incur significant accuracy drop [12] unless the models are trained under proper constraints. This issue is more significant in complex data sets such as ImageNet [13]. Ideally, we want to use lower precision while achieving equivalent accuracy as the original network with floating point. Quantization can be done during or after training which are called quantization-aware training and post-training quantization, respectively. Post-training quantization is faster while quantization during training achieves higher accuracy since the network learns the low-precision weights better during training [7].

Previously published results have reported specialized hardware for DNN accelerators [6], [14] to enable DNN models that run on embedded mobile applications during inference. However, they were more focused on accelerating uncompressed DNN models, restricting their usage to small models. Without model compression, only small DNN models such as LeNet-5 can be run in an on-chip SRAM. In addition, DNNs are trained using float32 format numbers on GPU, however, it is very hard to use this format in on-chip DNN accelerator. This requires quantization before utilizing the model during the inference. Prior work in the literature [11], [15] have proposed methods to sparsify the model at the network-level where weights are replaced with zero based on a criteria to shrink the model size. However, this adds irregularity in the distribution of weights, which makes it harder for DNN accelerators to process sparse matrices [6].

In this paper, we proposed a novel quantization method leveraging the bit sparsity to quantize DNNs when used in RRAM based CIM. This implements a novel DNN accelerator that can be used in many applications related to edge computing. In resistive 1T-1R bitcells, the weights are encoded as the resistive of the cell [20]. Most RRAM cells encode binary information where the high resistance represents a "0" and a low resistance represents a "1". When a word-line voltage

Manuscript received October 28, 2021; revised December 31, 2021; accepted January 19, 2022. Date of publication February 1, 2022; date of current version April 28, 2022. This work was supported by the Semiconductor Research Corporation under Grant JUMP CBRIC Task 2777.004, Grant JUMP CBRIC Task 2777.005, and Grant JUMP CBRIC Task 2777.006. This article was recommended by Associate Editor T. Hanyu. *(Corresponding author: Foroozan Karimzadeh.)*

Foroozan Karimzadeh and Arijit Raychowdhury are with the Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA.

Jong-Hyeok Yoon is with the Department of Information and Communication Engineering, Daegu Gyeongbuk Institute of Science and Technology (DGIST), Daegu 42988, Republic of Korea (e-mail: fkarimzadeh6@gatech.edu; arijit.raychowdhury@ece.gatech.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSI.2022.3145687.

Digital Object Identifier 10.1109/TCSI.2022.3145687

of "0" is applied, the bit-cell is not turned on and there is no energy dissipated in the process (except for peripheral circuits). On the other hand, when a word-line voltage of "1" is applied, a current is allowed to flow through the bit-cell. In CIM operation, multiple word-lines are simultaneously turned-on based on the input data pattern, and the corresponding cells are activated. The current through the cells is subsequently accumulated on the bit-line and eventually sensed by an analog to digital converter (ADC). When the cell stores a "0" there is very little current (I_{OFF}) flowing through the cell and when the cell stores a "1" there is significantly larger current (I_{ON}) flowing through the cell. Hence, having more "0"s in the network, as opposed to "1"s reduces the total energy dissipated during inference. Further, recent advances in RRAM technology has enabled multiple-states per cell. In one particular implementation that has been described in details in [19], 3-levels per cell have been demonstrated. Experimental work on a 40nm embedded RRAM array [19], [20], show that 4 levels (2-bits) per cell are possible with tight resistance distributions. This allows higher memory density with 2-bits/cell, where each cell represents one of the four states 00, 01, 10 or 11 corresponding to four resistance levels 00 is the highest resistance and 11 is the lowest resistance state. In such a 2-bit/cell encoding it is desirable to have more 00s and 01s than 10s and 11s. This paper shows such a scheme (BitS-Net) where we train the network to increase the energy efficiency by increasing the number of 00s and 01s and reducing the number of 10s and 11s in the INT8 representation of the weights. This is a novel bit-level sparsification technique that leverages the key characteristics of the RRAM based CIM architectures. In traditional bit-level sparsity, the goal is to increase the number of zeros in the bit representation of weights/activation of neural networks. To do this, the network is typically quantized to the desired fixed-point numbers using a distance based measurement during training. The advantage of quantization during training in comparison to the Post-training quantization methods is that quantization-aware training helps DNNs to be trained for lower precision such as, INT8, without compromising on accuracy [20]. This is achieved by modeling quantization errors during training which helps in maintaining accuracy comparable to a Float32 format. In the next, we will leverage the efficiency of RRAM based CIM (Compute in memory) and bit-level sparsity. A compute-in-memory architecture has gained importance in achieving high-throughput low-latency AI systems [21]. A traditional Von Neumann architecture suffers from the latency and power dissipation caused by intra-chip data communication. Therefore, CIM architecture has emerged to overcome the aforementioned problems by conducting the computations in the memory. By employing CIM architectures along with the proposed bit-level sparsity, low-latency energy efficient computing systems can be achieved.

This paper makes the following contributions:

- The proposed method reduces the computational complexity and energy required during inference for a multi-bit encoded RRAM array.
- On state-of-the-art networks such as ResNet, our method achieves comparable accuracy than the original network.
- To the best of our knowledge, this is the first time a bit-level sparsity method has been proposed to leverage the benefits of CIM architecture.

II. Related Works and Motivation

Memory access is the bottleneck in large DNNs which dominates the total energy consumption. For instance, the size of AlexNet and VGG-16 Caffe model are over 200MB and 500MB, respectively. This makes it hard to deploy DNNs on edge devices since they do not fit in on-chip memory and therefore require the more costly DRAM accesses [6]. The cost of accessing DRAM is 640pJ for 32-bit coefficients which is 3 order of magnitude larger than accessing on-chip SRAM [15]. A new class of embedded non-volatile memory (eNVM) which perform matrix multiplication (y = W x) in a dense memory structure offers a solution to minimize data transport by performing compute in-memory [22].

In addition, DNNs are trained using floating point (i.e. float-32) format on GPUs. However, efficient hardware DNN accelerators operate on the fixed-point format. Therefore, by compressing the neural network through quantization and sparsification, we can reduce the total number of operations in the fixed-point format and shrink the size of DNN. However, the irregular pattern caused by compression prevent the efficient acceleration and requires the new indexing scheme which increase the memory usage [6]. Moreover, network compression causes information loss. Several papers have explored methods to quantize parameters (e.g., weights) and optimize the weights parameters simultaneously during training to gain better accuracy and compensate the information loss due to low-precision quantization [7], [12], [27]. However, many of them do not quantize the activation and the first and last layer of weights. INQ (Incremental Network Quantization) [25] efficiently convert any pre-trained full-precision convolutional neural network (CNN) model into a low-precision version using a masking method during training. The weights are quantized to either powers of two or zero but activation is still in full-precision which make it harder to run the model during inference. DOREFA-NET [26] quantize the network using low bit-width parameter gradients. SYQ [12] is another quantization scheme where reduces the information loss by learning a symmetric codebook for particular weight subgroups. In [7], a quantization method that allows inference to be carried out using integer-only arithmetic is proposed by introducing fake nodes to the model where weights and activation are quantized in this step during training. [27] introduces an Additive Powers-of-Two (APOT) quantization scheme to nonuniformly quantize the bell-shaped distribution of weights and activation by constraining all quantization levels as the sum of Powers-of-Two terms.

On the other hand, there are several methods that have been developed to sparsify the DNNs at the network level to shrink the network and eliminate unimportant weights [24], [28]. However, the sparsification causes irregularity in the weight matrices which makes it even harder than using the original

network on GPU or CPU [6]. It requires to store additional indexing matrix to keep track of the zero weights. Recently, several DNN architecture have been designed to accelerate the sparse network and cope with the irregularity caused by sparsity and unstructured pruning [6], [14], [28], [29]. Despite of the efforts done in the area of DNN accelerators, new architectures based on traditional CMOS still face the fundamental technological limitations of CMOS.

In this paper we propose a novel and CIM-aware DNN compression techniques which take advantage of CIM architecture and sparse the network at bit-level instead of network-level.

The Cambricon-X architecture [14] proposed an indexing method to enable using sparse DNNs by skipping the weights that randomly changed to zero. SCNN [28] is designed to accelerate DNN by proposing a new dataflow by encoding and maintaining non-zero weights and activations and efficiently delivering them to a multiplier array. EIE architecture [6] proposed a DNN accelerator and an indexing framework to accelerate the mathematical calculation of a sparse network for the compression method called deep compression [15] where pruning, quantization and Huffman encoding are employed to compress the network. The Eyeriss architecture [29] is designed to run compact DNNs by introducing a hierarchical mesh which is a flexible on-chip network that can be adapted to different data types and improves the utilization of the memory resources.

III. Proposed CIM-Aware Quantization

In this paper we propose a novel and CIM-aware DNN compression techniques which take advantage of CIM architecture and sparsify the network at the bit-level instead of the network-level. This technique can, of course, be combined with network-level sparsification techniques. In this section we go over the fundamentals of the BitS-Net.

A. Necessity of Compute-In-Memory Architecture

The Von-Neumann architecture has prevailed while supporting various tasks with centralized processing elements (PEs), control units, and memory. Since the advent of AI systems, the importance of DNNs has been on the rise featuring massive matrix-vector multiplication (MVM). Von Neumann architecture has strived to accommodate DNNs by relying upon its versatility. However, this architecture suffers from prohibitive power dissipation incurred by massive data transfer between the PEs and memory, retaining the weight of DNNs. In addition, von Neumann architecture features instruction-driven operation. Thus, data processing is not initiated promptly even if the input vector is ready for the MVM. Fig. 1 shows the structure of a systolic array. Considering the ability to conduct massive parallel computation, a weight-stationary systolic array appears to be one of the candidates to support energy-efficient MVM owing to the distributed data processing unit (DPU) that includes a PE and memory. A systolic array conducts data processing immediately once the input vector is applied owing to the data-driven operation of the DPUs. However, a clock-cycle-based propagation of intermediate data across the DPUs eventually incurs excessive latency that is

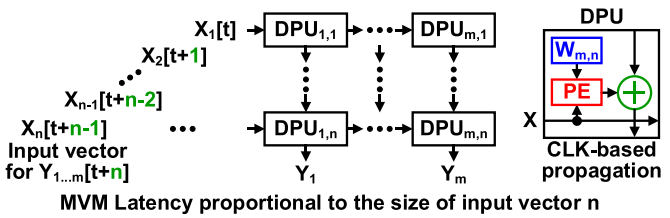


Fig. 1. Structure of a systolic array as an intermediate solution to supporting AI systems.

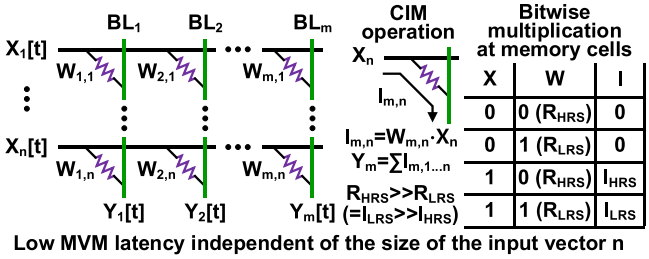


Fig. 2. Compute-in-memory architectures exploiting resistance of memory cells.

proportional to the size of the input vector. Thus, to achieve energy-efficient MVM with low latency independent of the size of the input vector, CIM architectures have come into the limelight. Fig. 2 shows the CIM architecture exploiting resistance of memory cells. Instead of the DPUs, a memory cell itself serves as a PE and memory simultaneously in CIM architectures. The memory cell retaining the weights generates the current that is the result of bit-wise multiplication. Owing to a current-summing BL structure in a memory array, the intermediate data are accumulated immediately, thereby achieving low latency in addition to energy-efficiency superior to the aforementioned architectures. In particular, the CIM architectures employing emerging memory such as PCRAM, MRAM, and RRAM have gained importance in achieving energy-efficient computing systems for AI owing to the inherent multiply-and-accumulate (MAC) functionality in BL structures, non-volatility, high bit density, and compatibility to CMOS process. Compared to other emerging memory, RRAM features energy-efficient read (RD) and the feasibility of multi-bit encoding owing to low latency and an appropriate ON/OFF ratio, respectively [30]–[33].

B. Preliminaries

Matrix-vector multiplication is a basic building block in many DNN models where dot products between weight (W) and its input values (X) is calculated in each layer. We denote the output of each hidden layer (a) as

$$Z = W^T X \tag{1}$$

$$a = \sigma(Z) \tag{2}$$

where T is a transpose function, σ is an element-wise non-linear activation function which is usually a Rectified Linear Unit (ReLU) [17].

In order to quantize the model, suppose $W \in R^{C_{out} \times C_{in} \times k \times k}$ is a $4D$ tensor representing kernels in a convolutional layer,

where C_{out} and C_{in} and k are the number of output channel, input channels and the kernel size, respectively. Therefore, the quantization scheme of the weights [27] is defined as

$$W_q = \alpha \Pi_{Q(1,b)} \lfloor \frac{W}{\alpha}, 1 \rceil \tag{3}$$

where b is the bit-width, α is the scaling factor and the scaling function $\lfloor ., 1 \rceil$ scales the weights into $[-1, 1]$. Then the scaled W is projected by $\Pi_{(.)}$ in an element-wise manner to the defined quantization levels. $Q(1, b)$ defines a set of quantization levels for example uniform quantization. For uniform and power of two (POT) quantizations, $Q(1, b)$ defines as Eqs. 4 and 5, respectively.

$$Q_U(1, b) = \{0, \frac{\pm 1}{2^{b-1} - 1}, \frac{\pm 2}{2^{b-1} - 1}, \ldots, \pm 1\} \tag{4}$$

$$Q_{POT}(1, b) = \{0, \pm 2^{-b+1}, \pm 2^{-b+2}, \ldots, \pm 1\} \tag{5}$$

Quantization maps each element in the weight matrix which is in floating-point format to a b-bit fixed-point representation. The result of convolution against quantization level is then re-scaled by multiplying to α. Generally, α is a floating-point number [27]. However, in order to have fully fixed-point computation, we round it to the closest integer in our proposed method. Therefore, arithmetical calculation in DNN models, for example in fully connected and convolutional layers, can be performed on an on-chip hardware using low-precision fixed-point operations, which are substantially cheaper in terms of memory and power than their floating-point equivalent [18].

During backpropagation, the Straight-Through Estimator (STE) [42] is adopted as proposed in [27] for the projection operation. The gradients of α are computes as follow

$$\frac{\partial W_q}{\partial \alpha} = \begin{cases} sign(W) & if\ |W| > \alpha \\ \Pi_{Q(1,b)} \frac{W}{\alpha} - \frac{W}{\alpha} & if\ |W| \leq \alpha \end{cases} \tag{6}$$

Nevertheless, the baseline accuracy cannot be achieved with uniform quantization since the results does not match the distribution of weights (typically normal distribution) [15]. Moreover, POT quantization is also not the efficient way of quantization for hardware since the negative values containing a lot of ones which requires a lot of energy for multiplication. In this paper, we proposed a highly efficient hardware-aware method for quantization and low-bit fixed-point computation that can be easily map to CIM hardware.

C. BitS-Net Training

In this paper we proposed a bit-level sparsity method that leverage the fact that there are a lot of zeros in the bit representation of weight values. The overall method is shown in figure 3. First, a coefficient set for quantization is defined. The criteria for calculating the coefficient set will be explained later. In the next step, the network is quantized during training for several epochs and finally the quantized network is achieved.

We will take advantage of bit-level sparsity to decrease the number of multiplication while exploiting CIM architecture. In resistive CIM architectures, the 0 and 1 states are represented by OFF and ON resistances on the RRAM cells

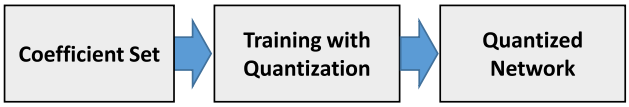


Fig. 3. BitS-Net algorithm. First, the coefficient set is defined. Next, the network is quantized during training step and finally, the quantized network is achieved.

respectively. To increase the energy efficiency of the CIM designs, it is beneficial to increase the number of 0s in the model representation that enables not only higher throughput (by under provisioning for the ADCs) but also increase the energy efficiency. Therefore, in the bit-level sparsity, we aim to increase the number of zeros in the bit representation of weights/activation of neural networks by quantizing the weights/activation to the desired fixed-point numbers using Eq. 3 during training. During our fixed-point training, for each layer, first the weights are scaled to be in the range of $W \in [-1, 1]$ using a scaling factor α. The CIM architecture consists of cells of two bits that can be 00, 01, 10 and 11. Multiplying to each of these two bits is done in different energy level ($E_{00} < E_{01} < E_{10} < E_{11}$). Therefore, the desired weight's values are the ones that have less bits with higher energy like 11 and more 00. During BitS-Net training, we quantize every floating point weight to its closest favorable coefficient C which is the set of possible coefficient and favorable fixed-point numbers that W can be quantized. C for INT8 numbers are shown in Table I. Since the coefficient sets are between $[-1, 1]$ and negative numbers have 11 in their two's complement representations which leads to a large energy requirement, during inference, we save $W_q + 1$ in memory in order to remove 11 in the coefficients. This is the reason we choose numbers represented at Table I where they do not have 11 in their binary representations during inference. As an example, the binary representations for $w1 = 0.375$ and $w2 = -0.375$ are 00.011000 and 11.101000. By adding one to the quantized weights ($W_q + 1$), the binary representation of $w1 + 1$ and $w2 + 1$ will be 01.01 10 00 and 10. 10 10 00. Therefore, there is no 11 in the binary format of coefficients at Table I. The numbers at set 1 and set 2 have no '11' and the maximum of four and three '10', respectively. By employing CIM architectures along with the proposed bit-level sparsity, low-latency energy efficient computing systems can be achieved.

The BitS-Net is explained in detail in algorithm 1. First the coefficient set is defined as explained above which will be an input for the quantization algorithm. In the forward path, first, the weights are quantized based on Eq. 3 by dividing the weights by a scaling factor (α_W) to make the weights in the range of [-1, 1]. Then, the scaled weights are quantized ($\Pi_{Q(1,b)}$) by calculating the minimum distance between the weights and the coefficient set as shown in Eq. 7.

$$\Pi_{Q(1,b)} = argmin_{c_i \in C} |c_i - |W_l|| \tag{7}$$

where $c_i \in C$ represents the possible numbers in the coefficient set. In the next step, the activation is quantized as well to have a fully quantized network. The activation values are first scaled to the range of [0, 1], since they are all positive numbers. Then they are quantized to the fixed-point numbers.

TABLE I
THE COEFFICIENT SET FOR QUANTIZATION

Set	Coefficient sets
Set 1	$\pm\{0, 0.3438, 0.3750, 0.4063, 0.5, 0.6250, 0.6563, 1.0\}$
Set 2	$\pm\{0, 0.3750, 0.5, 0.6250, 1.0\}$
Ternary	$\pm\{0, 1.0\}$

Algorithm 1 Bit-Level Sparsity and Network Quantization for Lossless CNNs With Low-Precision Weights

Input : x: the training data, Set of desired numbers to be quantized to: $C = \{c_1, c_2, \ldots, c_k\}$; $k \leq N$, with$\{W_l : 1 \leq l \leq L\}$: the pre-trained full-precision CNN model, loss function $L(Y, \hat{Y})$, learning rate, λ_t

Output: $\{\hat{W}_l : 1 \leq l \leq L\}$: the final low-precision model with the weights constrained to be in the desired set with higher bit-level sparsity. $\{\hat{x}_l : 1 \leq l \leq L\}$: quantized activation.

Forward propagation:
for $l = 1$ **to** L **do**
  $Q_{lw} = quantize(W_l)$ using $W_q = Round(\alpha_w)\Pi_{Q(1,b)}\lfloor\frac{W}{\alpha_w}, 1\rceil$
  $Q_{lx} = quantize(x_l)$ using $x_q = Round(\alpha_x)\Pi_{Q(1,b)}\lfloor\frac{x}{\alpha_x}, 1\rceil$
  Compute the output activations: $x_{out} = Conv(W_{lq}; x_{lq})$
  Compute the loss between the actual and predicted values: $L(Y, \hat{Y})$
end for
Backward propagation:
$\frac{\partial \hat{L}}{\partial Q_l} = WeightBackward(Q_l, \frac{\partial \hat{L}}{\partial \hat{Y}})$
$W_{t+1} = UpdateWeights(W_t, \frac{\partial \hat{L}}{\partial Q_l}, \lambda_t)$
$\lambda_{t+1} = UpdateLearningRate(\lambda_t, t)$

In this paper, we quantize the activations uniformly to the 8-bit fixed-point numbers (Eq. 4). Finally, the convolution value is computed between the quantized weight (W_{lq}) and the quantized activation (x_{lq}). The backpropagation is applied using the STE method as explained in Eq. 6 to update the weights.

IV. ALGORITHM-HARDWARE JOINT OPTIMIZATION IN CIM DURING INFERENCE

As an algorithm-hardware joint approach, the BitS-Net is proposed to attain superior system-level energy efficiency by addressing the bit-level sparsity of weights while considering energy per bit in CIM. In this paper, we demonstrate the superior energy efficiency of the proposed BitS-Net considering the measured energy per bit during CIM, thereby exhibiting the feasibility of the algorithm deployment to multi-bit resistive CIM (RCIM) architectures.

During inference, we use the following formula to classify the input images. As mentioned before, the goal is to eliminate

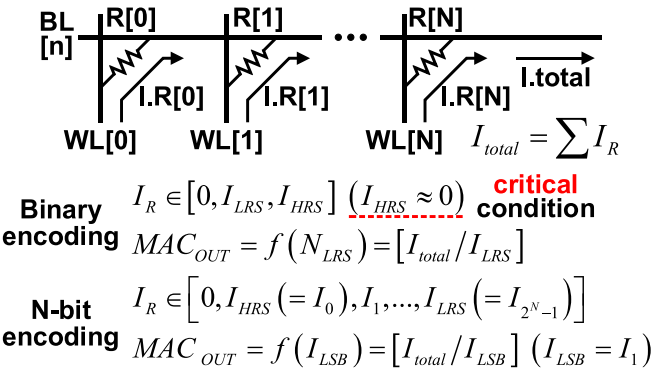


Fig. 4. Binary and multi-bit current-sensing RCIM at the bitline.

the number of 11 occurrence in the weight matrix during inference. Since we add one to W_q, $(W_q + 1)$, we need to modify Eq. 1 as shown in Eq. 8 in order to calculate the correct multiplication of weights and activation.

$$Z = ((W_q + 1)^T X) - \left(\sum_i x_i\right).\mathbf{1}^T \tag{8}$$

where x_i are elements of the input vector X and $\mathbf{1}$ is all-ones vector with dimension equal to the number of rows in the weight matrix (W).

A. Introduction to RCIM Architectures

RCIM architectures exploit a MAC-friendly BL structure [19], [20], [34]–[40]. To read the CIM result out, current- or voltage-sensing RD is employed at the BL. Fig. 4 shows the binary and multi-bit current-sensing RCIM at the BL. The current-sensing CIM is widespread in RCIM architectures. Each RRAM cell is programmed in a low resistance state (LRS) or a high resistance state (HRS) in binary encoding to represent the weights of AI systems. In multi-bit encoding, the RRAM cell is set to a certain resistance between the LRS and HRS resistance to program multi-level weights in a single RRAM cell. The output of the current-sensing RCIM is read out by the ratio of the total current at the BL to the LSB current which is the LRS current in binary encoding. Since the crucial premise of the current-sensing RCIM is that the HRS current is negligible, a sufficient margin between the LSB current and the HRS current is desirable. However, the resistance range of an RRAM cell is fixed such that multi-bit encoding exacerbates the narrow current margin. It eventually incurs logic ambiguity in the CIM output when the aggregate current from accessed HRS cells exceeds the LSB current. To surmount the aforementioned problems in current-sensing RCIM, voltage-sensing RD has been proposed in the prior arts [19], [20].

Fig. 5 depicts the current- and voltage-sensing RD in RCIM architectures. Compared to the current-sensing RCIM architectures where a fixed voltage at the BL is used to exploit the cell current relying on RRAM resistances, the voltage-sensing RCIM architectures use fixed or variable current to employ a wide range of readout BL voltages (V.RBLs) as the CIM output. The V.RBL reflects the parallel resistance of accessed RRAM cells. The voltage-sensing RD does not suffer from the aforementioned logic ambiguity since the parallel resistance has a unique value in a certain number

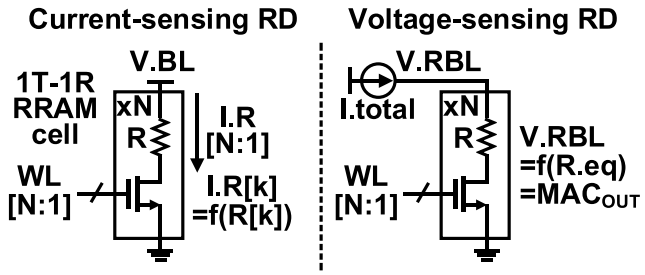


Fig. 5. Simplified structure of current- and voltage-sensing read in RCIM architectures.

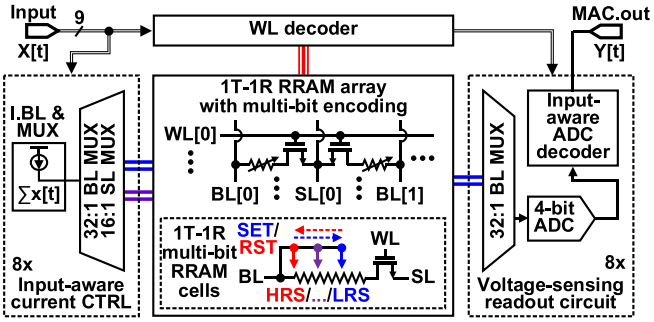
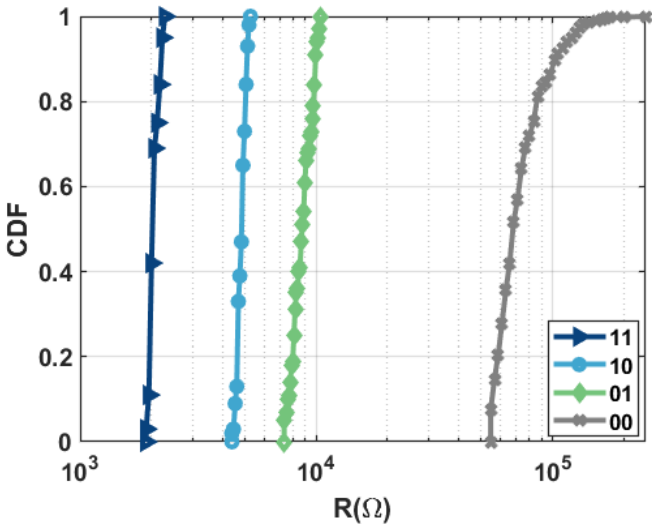


Fig. 6. Architecture of voltage-sensing multi-bit RCIM architecture.

of accessed RRAM cells (N.RRAM). However, the parallel resistance drastically decreases over N.RRAM such that a sufficient sampling margin in readout cannot be secured even employing diode-connected current sources [41]. Furthermore, it even worsens in multi-bit encoding. In this work, we employ the voltage-sensing multi-bit RCIM architectures overcoming the aforementioned problems to evaluate the system-level energy efficiency of the proposed BitS-Net.

B. Voltage-Sensing Multi-Bit RCIM Architecture in the Evaluation of the BitS-Net

Fig. 6 shows the simplified architecture of the voltage-sensing multi-bit RCIM architecture used in the evaluation of the BitS-Net [19]. As a solution to the drastic decrease of the V.RBL over the parallel resistances, the BL current control is employed to provide a unit current per accessed RRAM cell, thereby achieving a sufficient sampling margin in readout. The multi-bit encoding is conducted by iterative write (WR) with verification that reconfigures a WR pulse after monitoring the resistance of the programmed RRAM cell and initiates another WR process to achieve the target resistance [19], [20]. In readout, ADC-based readout circuit determines the MAC output considering the N.RRAM. The detailed description of the architecture is delineated in [19]. In this work, a 2-bit encoding is employed while securing sufficient resistance distance to avoid the invasion to an adjacent resistance state. Fig. 7 shows the measured resistance of the 2-bit weights in the BitS-Net from the test chip of the RCIM architecture [19]. Considering the 3σ-window of resistances, the resistance for the 2-bit weight (11, 10, 01, and 00) is determined. The nominal resistances of 2-bit encoding in an RRAM array are 2.08kΩ, 4.85kΩ, 8.77kΩ, and 76.31kΩ, respectively. With the 2-bit-encoded RRAM cells (11, 10, 01, and 00) and the ADC-based readout circuits, the measured energy per bit during CIM is 0.83, 0.47, 0.28, and 0.15 pJ/bit, respectively.



	11	10	01	00
μ	$2.08k\Omega$	$4.85k\Omega$	$8.77k\Omega$	$76.31k\Omega$
σ	97.65Ω	204.9Ω	$0.96k\Omega$	$25.55k\Omega$
E_{CIM}	$0.83\frac{pJ}{bit}$	$0.47\frac{pJ}{bit}$	$0.28\frac{pJ}{bit}$	$0.19\frac{pJ}{bit}$

Fig. 7. Measured resistance (R) distribution of the 2-bit weights in the BitS-Net.

TABLE II
THE ACCURACY AND TOTAL ENERGY OF CIFAR-10 ON BITS-NET USING COEFFICIENT SET 1, SET 2 AND TERNARY, IN COMPARISON TO THE BASELINE METHODS ON RESNET-20

Method	Bit Precision (W/A)	Accuracy (%)	Total Energy (J)
BitS-Net (Set 1)	8/8	91.63	0.37
BitS-Net (Set 2)	6/8	90.04	0.22
BitS-Net (Ternary)	3/3	79.84	0.09
POT	8/8	90.87	0.51
APOT	8/8	91.90	0.69
ADD-Net	8/8	91.60	0.88
8-bit precision	8/8	92.60	1.21
Full-Precision (Floating-point)	32/32	92.74	-

V. RESULTS

In this section, we present the experimental results of the algorithm and CIM hardware implementation of BitS-Net. To evaluate our method, BitS-Net is compared with several state-of-the-art baselines on ResNet architectures [23]. Baseline methods consist of APOT [27], POT [27], INQ [25], ADD-Net [43], 8-bit quantization and the original floating point networks. In our method both weights and activations of the networks are quantized in order to have fully quantized network that can be implemented in an on-chip hardware. It should be noted that that APOT and POT utilize 8-bit quantization for the first and last layers in the original papers, which incur more memory cost. INQ also used full precision (32 bits floating point) for activation which makes it impossible to run it in an on-chip system. In our implementation, we employ 8-bit quantization just for the last layer to balance the accuracy drop and the hardware overhead. In our proposed

TABLE III

THE FINAL CLASSIFICATION ACCURACY AND TOTAL ENERGY ACHIEVED BY BITS-NET METHOD AND THE BASELINES INCLUDING POT, APOT, INQ, UNIFORM QUANTIZATION AND THE FULL PRECISION NETWORK IN RESNET-18 AND RESNET-34 ARCHITECTURES

Network	Bit precision	ResNet-18			ResNet-34		
Accuracy (%) / Method	W/A	top-1	top-5	Total Energy (J)	top-1	top-5	Total Energy (J)
BitS-Net (Set 1)	8 / 8	67.73	87.64	139.54	70.08	89.28	221.46
BitS-Net (Set 2)	6 / 8	67.05	87.64	96.31	70.02	89.11	162.64
BitS-Net (Ternary)	3 / 3	63.69	85.48	41.84	64.51	86.21	53.94
POT	8 / 8	65.01	86.33	187.05	63.26	85.86	242.71
APOT	8 / 8	66.13	87.14	283.97	70.14	89.88	531.32
INQ	8 / 32	67.42	87.02	-	70.23	89.68	-
ADD-Net	8 / 8	66.01	87.50	337.23	69.11	89.21	633.32
POT	4 / 8	64.70	86.89	168.17	63.02	85.73	286.77
APOT	4 / 8	66.05	86.95	143.07	69.55	89.41	244.61
4-bit precision	4 / 8	67.91	88.59	140.14	69.12	89.29	240.56
8-bit precision	8 / 8	68.29	88.57	368.93	71.28	90.17	674.44
Full-Precision (Floating-point)	32 / 32	68.52	88.49	-	71.07	90.18	-

BitS-Net quantization algorithm, weights are quantized to coefficients in Table I to have an efficient CIM hardware for DNN accelerations. Moreover, we have used power of two values in order to quantize the activation as shown in Eq. 5.

A. Evaluation on CIFAR-10

The first dataset that we used is CIFAR-10 [44] with 50K training and 10K test images of size 32×32. The ResNet-20 network is utilized where its architectures for CIFAR-10 includes a convolutional layer followed by 3 residual blocks and a final FC layer. For CIFAR-10, we train the networks up to 200 epochs with a mini-batch size of 128 and learning rate of 0.04 at the beginning and scaling factor of 0.1 at epoch 80, 120. stochastic gradient descent (SGD) with momentum of 0.9 was implemented for optimization step. The results are demonstrated in Table II. The results show that BitS-Net (Set 1) can achieve 91.63% accuracy which is about 1% lower that the 8-bit precision model. BitS-Net (Set 2) achieved 90.04% accuracy while the original network before quantization achieved 92.74 % accuracy. Also, BitS-Net achieved lower required energy than the mentioned baseline methods.

B. Evaluation on ImageNet

To evaluate our method, we also used ImageNet datasets [13] with 1000 classes. The ImageNet dataset consists of 1.2M training and 50K validation images. For the pre-processing step [23], training images are randomly cropped and resized to 224×224 and the validation images are center-cropped to the same size. The ResNet models are implemented using PyTorch official implementation and initialized from the released pre-trained model. During training, batch size is set to 128 (based on the GPU memory size). We choose a learning rate of 0.1 with a decay factor of 0.1 at epoch 30, 60, 80, 100 during the 120 epochs for training the network. Moreover, SGD with the momentum of 0.9 is utilized to optimize the parameters. We quantize ResNet-18 and ResNet-34 on ImageNet dataset and the results are compared with the baseline methods in Table III. The results demonstrate that BitS-Net can achieve higher accuracy than the baseline quantization methods. The accuracy of BitS-Net set 1 and 2 are 67.73% and 67.05%, respectively while POT and APOT methods gained 65.01% and 66.13% on ResNet-18. BitS-Net keeps and quantizes the bigger weight values which have higher importance [24] and quantize the smaller values (less important) to zero which causes higher accuracy. PoT method suffers from the rigid resolution [27], and achieved the lowest accuracy compared to the other methods. The total energy (multiplications and ADC) of each methods are also estimated based on the actual measurement from the hardware. The result show that BitS-Net consume less energy that other baseline methods. As an example the energy required to run BitS-Net (Set 2) on ResNet-18 is $96.31J$ which is 4x less than 8-bit quantization ($368.93J$). Also, ADD-Net for 8 bit quantization requires 337.23 and $633.32J$ energy for ResNet-18 and ResNet-34 on ImageNet dataset, respectively, which are 2.5x and 3.5x higher energies than BitS-Net (Set 1) and BitS-Net (Set 2) on ResNet-18. BitS-Net set 1 consists of 15 numbers which are chosen from 8-bit numbers to eliminate all the '11' in the binary representations of both positive and negative numbers. On the other hand, the effective number of bits to handle the 15 values can be just 4 bits. Therefore, we also investigated and compared the BitS-Net results to the baseline methods with 4-bit weights. The results show the effectiveness of BitS-Net. All the baseline methods achieved higher energy than BitS-Net. For example, APOT quantization with 4-bit weights and 8-bit activation consumes 143.07 (J) which is higher than BitS-Net set 1 with 139.54 (J) required energy. In addition, the percentage of 00, 01, 10, 11 in the total weight matrices of ResNet-18 for BitS-Net3 (ternary), BitS-Net2 (set 2 coefficients), BitS-Net1 (set 1 coefficients), POT, APOT and 8-bit quantization methods is illustrated in Fig. 8. 2% '11' in BitS-Nets correspond to the last FC layer which is quantized to an 8-bit precision. 8-bit quantization has the most number

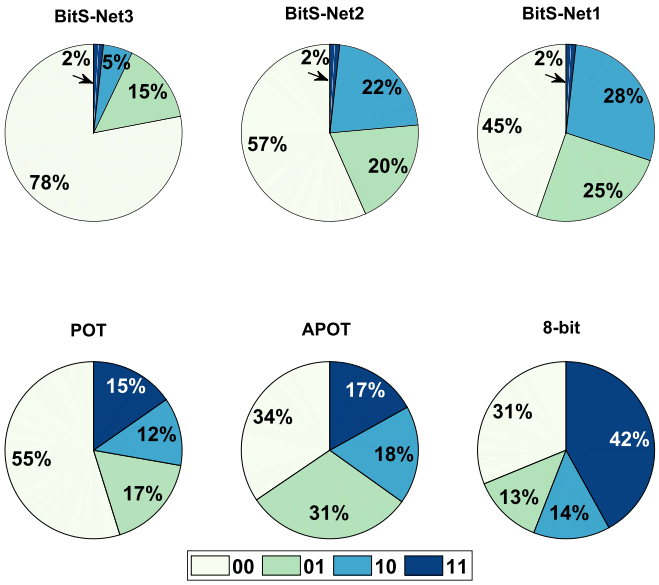


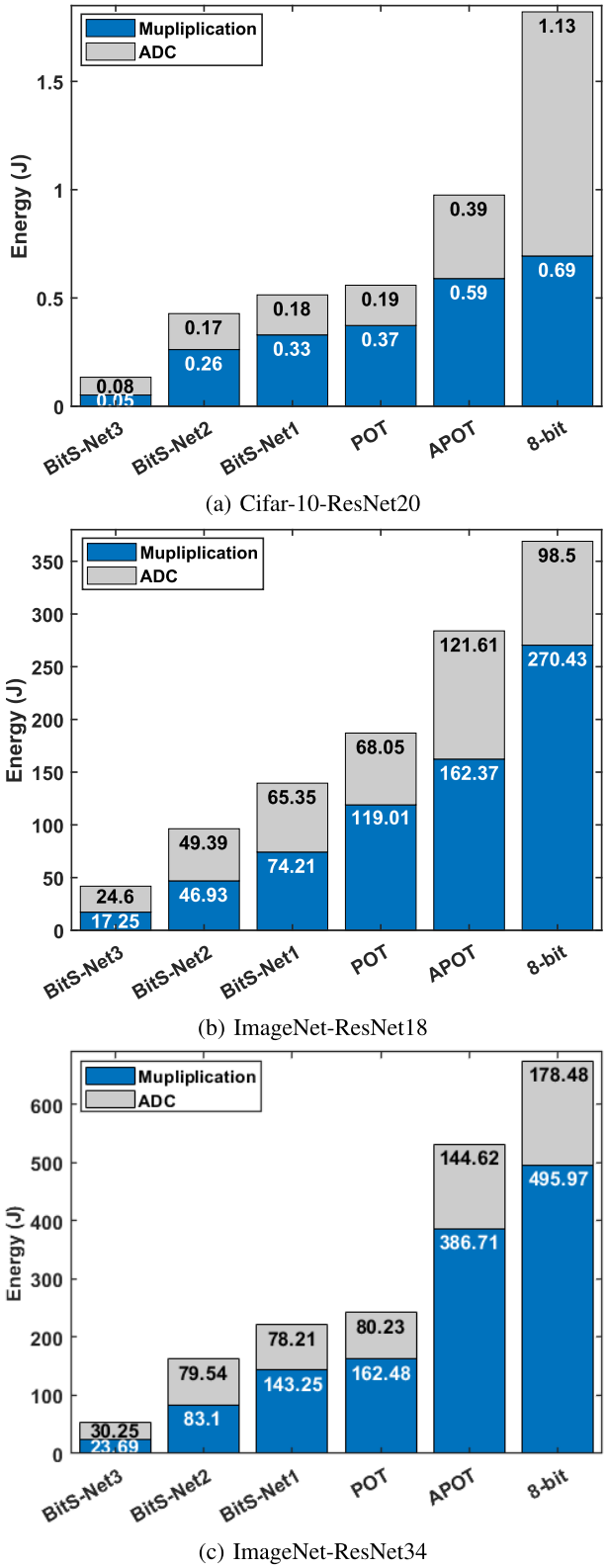
Fig. 8. Percentage of 00, 01, 10, 11 in the total weight values of ResNet-18 for BitS-Net3 (ternary), BitS-Net2 (set 2 coefficients), BitS-Net1 (set 1 coefficients), POT, APOT and 8-bit quantization methods.

of "11" values (42%) which consume the most energy during multiplication.

C. Hardware Evaluations

In this section, the results of hardware implementation for the BitS-Net and baseline methods are demonstrated to estimate key hardware metrics. The estimated energy includes the RRAM array, the ADC, the controller, and other peripheral circuits except for the voltage reference (VREF) generator. The VREF generator was excluded since a single set of the VREF generator is sufficient for the entire RRAM macro and the power consumption of the generator will be negligible while increasing the size of RRAM macro. We have estimated the energy through measurements from the RRAM hardware. However, the RRAM macro is not large enough to fit the entire model. Instead, we scan in the different model weights serial, write into the RRAM array and measure the array energy. In other words, since the RRAM macro is of limited size, we scan in the weights of the layers serially and conduct the energy estimation one layer at a time. As a prototype of the RRAM-based CIM architecture, the RRAM macro focuses on the CIM operation. Regarding array utilization, the system architecture that will use this macro will dictate what the system-level metrics would be. Similarly, hardware utilization is highly dependent on the system architecture, the compiler, and any extrapolation on hardware utilization, system-level costs, etc. will be too speculative for this work. We will need to define specific design architectures and compiler techniques that would be outside the scope.

With the 2-bit-encoded RRAM cells (00, 01, 10, and 11) [20], the measured energy per 2-bits during CIM is 1.46 pJ/2bits, 0.73 pJ/2bits, 0.36 pJ/2bits, 79 fJ/2bits, respectively. In addition the energy of ADC is 0.208 pJ/2bits. It should be noted that the cycle time is 20ns. The energy



(a) Cifar-10-ResNet20

(b) ImageNet-ResNet18

(c) ImageNet-ResNet34

Fig. 9. Energy of multiplication and ADC for BitS-Net3 (ternary), BitS-Net2 (set 2 coefficients), BitS-Net1 (set 1 coefficients), POT, APOT and 8-bit quantization methods on (a) ResNet-20 on CIFAR-10; (b) ResNet-18 on ImageNet and (c) ResNet-34 on ImageNet.

breakdown for multiplication and ADC are demonstrated in Fig. 9 which shows that BitS-Net requires less energy than the baseline method. BitS-Net (set2) is more than 5x

energy efficient compared to 8-bit quantization. For example, in ResNet-34, the total energy required during inference is 162.64 (J) while APOT and 8-bit quantization require 531.33 and 674.45 (J), respectively. In addition, ternary quantization consume least energy at the expense of lower accuracy. Therefore, BitS-Net is more energy efficient than the baseline methods.

VI. Conclusion

In this paper, we proposed BitS-Net, an algorithm-hardware joint approach to attain superior system-level energy efficiency by addressing the bit-level sparsity of weights while considering energy per bit in CIM. Since $E_{00} < E_{01} < E_{10} < E_{11}$, we sparse the weights and quantized the networks to the values that have bits with lower energies required for multiplication in CIM architecture. As a result, we are able to develop a highly energy efficient system to run DNNs during inference. We demonstrate that BitS-Net improves the energy efficiency by up to 5x compared to the 8-bit network for ResNet model on ImageNet dataset. It should be noted that in order to efficiently utilize the proposed RRAM macro for executing actual neural networks in a system, the system architecture and the compiler, and related system-level costs such as hardware utilization and off-chip memory access should be taken in account. Considering the fact that the energy efficiency of all the methods is estimated under the same condition, we believe that the comparison in Table III is apples-to-apples to address the efficiency improvement of the proposed BitS-Net.

References

[1] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Comput. Intell. Mag.*, vol. 13, no. 3, pp. 55–75, Aug. 2018.

[2] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Comput. Intell. Neurosci.*, vol. 2018, pp. 1–13, Feb. 2018.

[3] S. Ma, X. Zhang, C. Jia, Z. Zhao, S. Wang, and S. Wang, "Image and video compression with neural networks: A review," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 6, pp. 1683–1698, Jun. 2020.

[4] R. Boostani, F. Karimzadeh, and M. Nami, "A comparative review on sleep stage classification methods in patients and healthy individuals," *Comput. Methods Programs Biomed.*, vol. 140, pp. 77–91, Mar. 2017.

[5] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *J. Field Robot.*, vol. 37, no. 3, pp. 362–386, 2020.

[6] S. Han *et al.*, "EIE: Efficient inference engine on compressed deep neural network," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 243–254.

[7] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2704–2713.

[8] F. Karimzadeh, N. Cao, B. Crafton, J. Romberg, and A. Raychowdhury, "A hardware-friendly approach towards sparse neural networks based on LFSR-generated pseudo-random sequences," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 2, pp. 751–764, Feb. 2021.

[9] F. Karimzadeh and A. Raychowdhury, "Memory and energy efficient method toward sparse neural network using LFSR indexing," in *Proc. IFIP/IEEE 28th Int. Conf. Very Large Scale Integr. (VLSI-SOC)*, Oct. 2020, pp. 206–207.

[10] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. De Freitas, "Predicting parameters in deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 2148–2156.

[11] F. Karimzadeh, N. Cao, B. Crafton, J. Romberg, and A. Raychowdhury, "Hardware-aware pruning of DNNs using LFSR-generated pseudo-random indices," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Oct. 2020, pp. 1–5.

[12] J. Faraone, N. Fraser, M. Blott, and P. H. W. Leong, "SYQ: Learning symmetric quantization for efficient deep neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4300–4309.

[13] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[14] S. Zhang *et al.*, "Cambricon-X: An accelerator for sparse neural networks," in *Proc. 49th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2016, p. 20.

[15] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*.

[16] G. Kour and R. Saabne, "Real-time segmentation of on-line handwritten Arabic script," in *Proc. 14th Int. Conf. Frontiers Handwriting Recognit.*, Sep. 2014, pp. 417–422.

[17] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "Cacti 6.0: A tool to model large caches," *HP Laboratories*, vol. 27, p. 28, Apr. 2009.

[18] D. Goldberg, "What every computer scientist should know about floating-point arithmetic," *ACM Comput. Surv.*, vol. 23, no. 1, pp. 5–48, Mar. 1991.

[19] J.-H. Yoon, M. Chang, W.-S. Khwa, Y.-D. Chih, M.-F. Chang, and A. Raychowdhury, "A 40 nm 100 Kb 118.44TOPS/W ternary-weight compute in-memory RRAM macro with voltage-sensing read and write verification for reliable multi-bit RRAM operation," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Apr. 2021, pp. 1–2.

[20] J.-H. Yoon, M. Chang, W.-S. Khwa, Y.-D. Chih, M.-F. Chang, and A. Raychowdhury, "29.1 A 40 nm 64 Kb 56.67TOPS/W read-disturb-tolerant compute-in-memory/digital RRAM macro with active-feedback-based read and *in-situ* write verification," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2021, pp. 404–406.

[21] W. He *et al.*, "2-bit-per-cell RRAM-based in-memory computing for area-/energy-efficient deep learning," in *IEEE Solid-State Circuits Lett.*, vol. 3, pp. 194–197, 2020.

[22] B. Crafton, S. Spetalnick, G. Murali, T. Krishna, S.-K. Lim, and A. Raychowdhury, "Breaking barriers: Maximizing array utilization for compute in-memory fabrics," 2020, *arXiv:2008.06741*.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[24] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1135–1143.

[25] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless CNNs with low-precision weights," 2017, *arXiv:1702.03044*.

[26] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients," 2016, *arXiv:1606.06160*.

[27] Y. Li, X. Dong, and W. Wang, "Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks," 2019, *arXiv:1909.13144*.

[28] A. Parashar *et al.*, "SCNN: An accelerator for compressed-sparse convolutional neural networks," in *Proc. 44th Annu. Int. Symp. Comput. Archit.*, Jun. 2017, pp. 27–40.

[29] Y.-H. Chen, T.-J. Yang, J. S. Emer, and V. Sze, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 2, pp. 292–308, Jun. 2019.

[30] M. Le Gallo and A. Sebastian, "An overview of phase-change memory device physics," *J. Phys. D, Appl. Phys.*, vol. 53, no. 21, May 2020, Art. no. 213002.

[31] O. Golonzka *et al.*, "MRAM as embedded non-volatile memory solution for 22FFL FinFET technology," in *IEDM Tech. Dig.*, Dec. 2018, pp. 1–18.

[32] S. Mittal, J. S. Vetter, and D. Li, "A survey of architectural approaches for managing embedded DRAM and non-volatile on-chip caches," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 6, pp. 1524–1537, Jun. 2015.

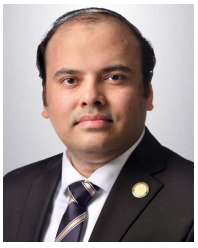
[33] C. Nail *et al.*, "Understanding RRAM endurance, retention and window margin trade-off using experimental results and simulations," in *IEDM Tech. Dig.*, Dec. 2016, pp. 4–5.

[34] W.-H. Chen *et al.*, "A 65 nm 1 Mb nonvolatile computing-in-memory ReRAM macro with sub-16 ns multiply-and-accumulate for binary DNN AI edge processors," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2018, pp. 494–496.

[35] C. Xue *et al.*, "Embedded 1-mb ReRam-based computing-in-memory macro with multibit input and weight for CNN-based AI edge processors," *IEEE J. Solid-State Circuits*, vol. 55, no. 1, pp. 203–215, Jan. 2020.
[36] C.-X. Xue *et al.*, "15.4 A 22 nm 2 Mb ReRAM compute-in-memory macro with 121-28TOPS/W for multibit MAC computing for tiny AI edge devices," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2020, pp. 244–246.
[37] W. Wan *et al.*, "33.1 A 74 TMACS/W CMOS-RRAM neurosynaptic core with dynamically reconfigurable dataflow and *in-situ* transposable weights for probabilistic graphical models," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2020, pp. 498–500.
[38] R. Mochida *et al.*, "A 4M synapses integrated analog ReRAM based 66.5 TOPS/W neural-network processor with cell current controlled writing and flexible network architecture," in *Proc. IEEE Symp. VLSI Technol.*, Jun. 2018, pp. 175–176.
[39] B. Chen, F. Cai, J. Zhou, W. Ma, P. Sheridan, and W. D. Lu, "Efficient in-memory computing architecture based on crossbar arrays," in *IEDM Tech. Dig.*, Dec. 2015, pp. 5–17.
[40] T. F. Wu *et al.*, "Brain-inspired computing exploiting carbon nanotube FETs and resistive RAM: Hyperdimensional computing case study," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2018, pp. 492–494.
[41] S. Yin, X. Sun, S. Yu, and J.-S. Seo, "High-throughput in-memory computing for binary deep neural networks with monolithically integrated RRAM and 90-nm CMOS," *IEEE Trans. Electron Devices*, vol. 67, no. 10, pp. 4185–4192, Oct. 2020.
[42] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," 2013, *arXiv:1308.3432*.
[43] J. Faraone *et al.*, "AddNet: Deep neural networks using FPGA-optimized multipliers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 1, pp. 115–128, Sep. 2019.
[44] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep. 7, 2009.

Foroozan Karimzadeh (Graduate Student Member, IEEE) received the B.S. degree in electrical and computer engineering and the M.Sc. degree in biomedical engineering from Shiraz University, Iran, in 2012 and 2016, respectively. She is currently pursuing the Ph.D. degree with the Electrical and Computer Engineering Department, Georgia Institute of Technology, under supervision of Dr. Raychowdhury. Her research interests mainly include developing novel algorithms and hardware for energy efficient machine learning techniques. She was awarded a Semiconductor Research Corporation (SRC) Graduate Fellowship, which is awarded in partnership with Texas Instruments.

Jong-Hyeok Yoon (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Republic of Korea, in 2012, 2014, and 2018, respectively. From 2018 to 2020, he was a Post-Doctoral Fellow at the Georgia Institute of Technology, Atlanta, GA, USA. In 2021, he joined the Daegu Gyeongbuk Institute of Science and Technology (DGIST), Daegu, Republic of Korea, where he is currently an Assistant Professor with the Department of Information and Communication Engineering. His research interests include non-volatile memory (NVM)-based processing-in-memory architectures for deep learning, neuromorphic circuits for edge intelligence, high-speed wireline communications, and mixed-signal circuit designs. He was a recipient of the Best Regular Paper Award at the IEEE Custom Integrated Circuits Conference (CICC) in 2021.



Arijit Raychowdhury (Fellow, IEEE) received the Ph.D. degree in electrical and computer engineering from Purdue University in 2007.

He joined Georgia Tech in January 2013, where he was an Associate Professor and held the ON Semiconductor Junior Professorship at the department from 2013 to 2019. Prior to joining Georgia Tech, he held research positions at Intel Corporation for six years and at Texas Instruments for one and a half years. He is the Steve W. Chaddick Chair and a Professor with the School of Electrical and Computer Engineering, Georgia Institute of Technology. He holds more than 27 U.S. and international patents and has published over 250 articles in journals and refereed conferences. His research interests include low power digital and mixed-signal circuit design, design of power converters, signal-processors, and exploring interactions of circuits with device technologies. He is currently a Distinguished Lecturer of the IEEE Solid State Circuits Society (SSCS) and a Mentor for IEEE Young Professionals and IEEE Women in Circuits. He serves on the technical program committee for key circuits and design conferences, including ISSCC, VLSI Symposium, DAC, and CICC. He is the winner of several prestigious awards, including the SRC Technical Excellence Award 2021, the Qualcomm Faculty Award 2020, the IEEE/ACM Innovator under 40 Award, the NSF CISE Research Initiation Initiative Award (CRII) 2015, the Intel Labs Technical Contribution Award 2011, the Dimitris N. Chorafas Award for outstanding doctoral research and best thesis 2007, and several fellowships. He and his students have won 14 best paper awards over the years.