



Enhancing the Accuracy of 6T SRAM-Based In-Memory Architecture via Maximum Likelihood Detection

Hyungyo Kim , *Student Member, IEEE*, and Naresh R. Shanbhag , *Fellow, IEEE*

Abstract—This paper presents a statistical signal processing-based algorithmic approach to enhance the compute signal-to-noise ratio (compute SNR) of 6T SRAM-based analog in-memory computing (IMC) architectures which have recently emerged as an attractive alternative to mainstream digital accelerators for machine learning workloads due to their superior energy efficiency and compute densities. However, today, the compute SNR of analog IMCs is limited by device parameter variations and noise. To overcome this limitation, we propose a maximum likelihood (ML)-based statistical error compensation (MLEC) technique to improve the accuracy of binary dot-products (DPs) realized in 6T (six transistor) SRAM-based analog IMC architectures. The MLEC method involves exploiting the symmetric nature of the 6T SRAM bitcell to extract multiple observations efficiently and employ them for detection purposes. MLEC methods involving two (MLEC-2) and four (MLEC-4) observations are proposed along with efficient architectures to realize them in hardware, e.g., distribution-aware and energy-aware approximations of MLEC-4. Simulations in a commercial 28 nm CMOS process demonstrate that the proposed methods increase the compute SNR for the commonly used 144-dimensional DP by 5 dB-to-12 dB. This improvement in the bank-level compute SNR leads to a network-level accuracy improvement of up to 11% when a ResNet-20 (CIFAR-10 dataset) network is implemented on the IMC. Employing energy models of the IMC, the energy overhead of MLEC is estimated to lie between 3%-to-10% resulting in up to 45.6% and 18% increase in energy efficiency (1b-TOPS/W) for a target SNR of 20 dB and ResNet-20 accuracy of 90% on the CIFAR-10 dataset, respectively, compared to a conventional (uncompensated) 6T SRAM-based IMC.

Index Terms—In-memory computing, statistical error compensation, maximum likelihood detection.

I. INTRODUCTION

DEEP neural networks (DNNs), with their unmatched inference capability, continue to transform diverse application areas such as computer vision [1], [2], natural language

Manuscript received 19 September 2023; revised 16 March 2024; accepted 24 April 2024. Date of publication 29 April 2024; date of current version 24 June 2024. This work was supported in part by the DARPA and SRC funded Center for Ubiquitous Connectivity (CUBIC) and in part by the Center for Codesign of Cognitive Systems (COCOSYS). The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Andreas Burg. (*Corresponding author: Hyungyo Kim.*)

The authors are with the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: hyungyo2@illinois.edu; shanbhag@illinois.edu).

Digital Object Identifier 10.1109/TSP.2024.3394656

processing [3], [4], autonomous driving [5], and many others. However, these networks tend to be overparameterized with the largest of these reaching into >100 trillion [6] parameters today. Such large parameter counts lead to high computational complexity, storage requirements, and energy consumption. The energy consumption and latency of DNN-based machine learning workloads are dominated by the energy cost of data movement [7], thereby preventing their deployment on resource-constrained Edge platforms.

In-memory computing (IMC) emerged in 2014 [8], [9] as an alternative to the mainstream digital accelerators and as a means to alleviate memory access costs. IMCs realize their energy efficiency and throughput benefits by embedding analog computations within the bitcell array. Since 2014, numerous IMC integrated circuits [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25] have appeared using a variety of memory types and cell architectures to implement high-dimensional matrix-vector-multiplication (MVM) – a kernel that dominates the computational complexity in DNNs. Due to their ability to realize such high-dimensional MVMs, IMCs have recently been suggested [26], [27] for implementing the baseband processing of massive multi-input multi-output (MIMO) wireless systems.

Of the diverse types of IMCs, SRAM-based ones are most popular due to their clear energy efficiency (13 \times) and compute density (7 \times) advantages over digital accelerators [28]. However, the intrinsic analog nature of SRAM-based IMCs renders their computations susceptible to noise and process variations [11], [12], [13] thereby limiting their computational signal-to-noise-and-distortion ratio (compute SNR). Consequently, many IMC prototype works report a considerable drop of up to 7% in accuracy for an image classification task with a simple dataset as CIFAR-10 [13], [14], [15], [16] over floating-point digital implementations. These limits on the compute SNR of IMCs hinders their deployment in both machine learning and massive MIMO wireless systems in spite of their enormous advantages in energy efficiency and latency over digital accelerators.

To address this loss in compute accuracy of IMCs, noise-training algorithms have been proposed [29], [30], [31] such that the resulting models exhibit resilience to hardware noise. A disadvantage of these methods is that they are network and dataset specific, hence may not generalize to other tasks, and are

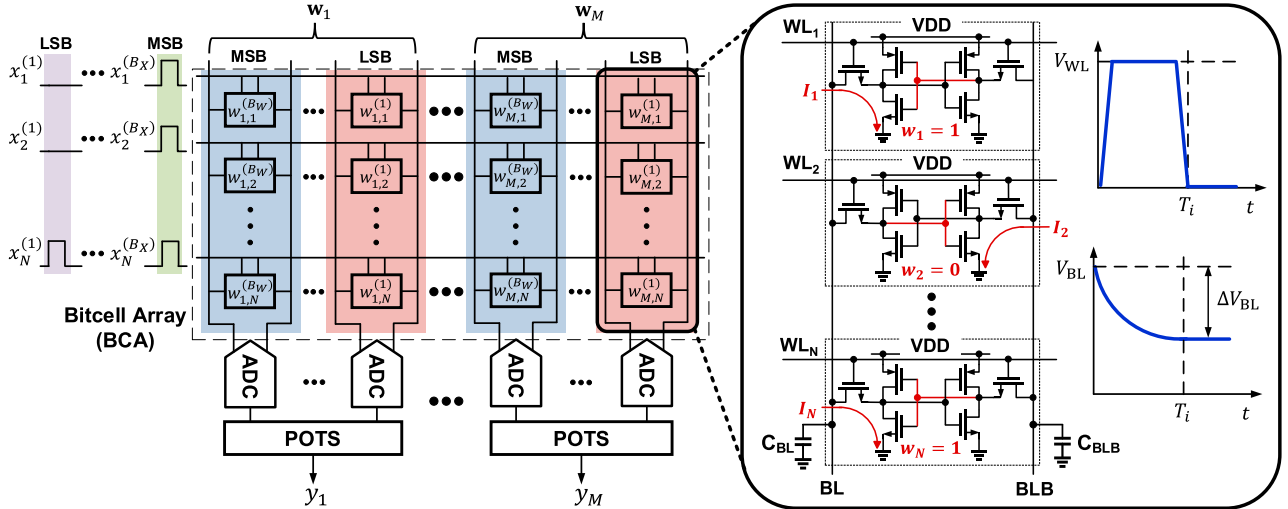


Fig. 1. The ISWP architecture: block diagram of the IMC array (left) and the circuit schematic of a single ADC column with V_{WL} and V_{BL} responses (right).

computationally expensive. As an alternative, signal processing methods [32], [33] have recently been proposed to enhance the compute accuracy of IMC architectures agnostic to the application. These methods obtain SNR-optimal design parameters of existing blocks in the IMC architecture, e.g., determining the optimal clipping levels for the column analog-to-digital converters (ADCs).

In contrast, in this paper, we explore the use of statistical signal processing methods to develop a class of algorithmic approaches to enhance the compute SNR of 6T SRAM-based IMCs. While [32], [33] focus on SNR-optimization by tuning the design parameters of existing blocks, methods proposed in this paper focus on compensating for noise and thereby introduce additional processing. Furthermore, in contrast to noise-aware training methods, our method operates at the level of an MVM, which is a core computation in most AI applications, and hence is general. Referred to as Maximum Likelihood-based statistical Error Compensation (MLEC), these methods leverage the circuit architecture of a 6T SRAM-based IMC to extract multiple observations efficiently and employ them to realize low-complexity maximum likelihood detection schemes. In doing so, this paper goes beyond previous works [32] by introducing an error compensation functionality into the IMC architecture to enhance its accuracy. This paper expands on the preliminary results presented in [34] by: 1) proposing a diversity of MLEC methods with varying accuracy enhancement vs. overhead trade-off, 2) demonstrating accuracy improvement at both MVM- and network-level tasks, and 3) quantifying the accuracy-energy trade-off.

Simulations in a commercial 28 nm CMOS process demonstrate that the proposed methods increase the compute SNR for the commonly used 144-dimensional DP by 5 dB-to-12 dB leading to a network-level classification accuracy improvement of up to 11% for a ResNet-20 (CIFAR-10 dataset) network implemented on the IMC. Employing energy models of the IMC, the energy overhead of MLEC is estimated to be between

3%-to-10% resulting in a significantly improved accuracy-energy trade-off compared to a conventional (uncompensated) 6T SRAM-based IMC.

The rest of this paper is organized as follows. Section II introduces the background, Section III describes our proposed MLEC methods, and Section IV presents simulation results to validate MLEC's effectiveness in both MVM-level task and network-level task and provide the estimated energy overhead and analyze the accuracy-energy trade-off. The Appendix provides the derivations, circuit implementation details, network accuracy details, and energy models. Finally, Section V concludes the paper.

II. BACKGROUND

This section provides the necessary background related to a commonly used IMC architecture referred to as the input-serial weight-parallel (ISWP) architecture [35] for computing an MVM and the impact of process variations on the IMC output.

A. The ISWP Architecture

Fig. 1 shows the ISWP architecture storing B_W -bit N -dimensional column vectors $\mathbf{w} = [\mathbf{w}_1, \dots, \mathbf{w}_M]$ of a $M \times N$ weight matrix \mathbf{w} across B_W consecutive columns and N consecutive rows of the SRAM bitcell array (BCA). Each bitcell stores the k -th bit $w_{i,j}^{(k)}$ of the j -th element of \mathbf{w}_i where $i \in [M]$, $j \in [N]$, $k \in [B_W]$. Note: we employ the notation $[A] = \{1, 2, \dots, A\}$.

The ISWP architecture computes the MVM function $\mathbf{w}^T \mathbf{x}$ between a B_W -bit $M \times N$ weight matrix \mathbf{w} and a B_X -bit N -dimensional activation vector \mathbf{x} . It does so by slicing \mathbf{x} into $B_X N$ -dimensional bit vectors and streaming it into the BCA over B_X clock cycles on the horizontal wordlines (WLs). Thus, $MB_X B_W$ binary dot products (DPs) are computed across MB_W bitline pairs (BLs/BLB) of the BCA over B_X clock

cycles. Since, the DP computation is realized in the analog domain, the BL/BLB voltages are digitized via MB_W analog-to-digital converters (ADCs) followed by a powers-of-two summing (POTS) that combine the binary DPs to generate a multi-bit output.

From the above description, it is clear that a binary DP is a fundamental computation in an IMC. Henceforth, in the rest of this paper, we focus on the computation of a binary DP, given by

$$y_o = \mathbf{w}^T \mathbf{x} = \sum_{i=1}^N w_i x_i, \quad (1)$$

where $y_o \in \{0, 1, 2, \dots, N\}$ is the ideal DP of two N -dimensional binary vectors $\mathbf{w} = [w_1, \dots, w_N]^T$ ($w_i \in \{0, 1\}$) and $\mathbf{x} = [x_1, \dots, x_N]^T$ ($x_i \in \{0, 1\}$).

B. Impact of Process Variations

IMCs are analog computing architectures and hence suffer from spatial variations in the parameters of the CMOS fabrication process, specifically the transistor threshold V_t variations. This section quantifies the impact of process variations on the ISWP architecture's output.

The ISWP architecture realizes a binary DP by mapping the Boolean variables (w_i, x_i) in (1) to corresponding physical circuit variables (I_i, T_i) as follows (see Fig. 1 right):

$$(y_o \rightarrow \Delta V_{BL}) = \sum_{i=1}^N \frac{(w_i \rightarrow I_i)(x_i \rightarrow T_i)}{C_{BL}}, \quad (2)$$

where $\Delta V_{BL} = V_{dd} - V_{BL}$ is the BL voltage discharge, I_i is the cell current in the i -th bitcell (BC), T_i is the pulse width on the i -th WL, and C_{BL} is the BL capacitance.

The cell current I_i in the i -th BC discharges the BL for a time duration T_i whenever the WL voltage is high, i.e., $x_i = 1$, and the BC stores $w_i = 1$ thereby accomplishing a 1-b multiply. Since N BCs are activated simultaneously, the charge drawn from the BL by each BC is accumulated leading to the computation of the N -dimensional binary DP in (2).

However, threshold voltage (V_t) variations in the BC transistors result in a variation in the cell current given by [36]:

$$\frac{\sigma_I}{\mu_I} = \frac{\alpha \sigma_{V_t}}{V_{WL} - V_t} \implies \frac{\sigma_{\Delta V_{BL}}}{\mu_{\Delta V_{BL}}} = \frac{1}{\sqrt{y_o}} \frac{\sigma_I}{\mu_I}, \quad (3)$$

where μ_I and σ_I are the mean and standard deviation, respectively of the cell current I , α is a fitting parameter, V_{WL} is the WL voltage, V_t is the threshold voltage of the access transistor, σ_{V_t} is the standard deviation of V_t , and $\mu_{\Delta V_{BL}}$ and $\sigma_{\Delta V_{BL}}$ are the mean and standard deviation, respectively of the BL voltage discharge ΔV_{BL} .

Equation (3) shows that the impact of V_t variations on V_{BL} increases as V_{WL} reduces. Circuit simulations in a commercial 28 nm CMOS in Fig. 2 shows that transistor V_t variations leads to variations in the IMC's output voltage V_{BL} with a normalized standard deviation $\sigma_{\Delta V_{BL}}/\mu_{\Delta V_{BL}}$ of 2.1% to 7.8% for a DP value y_o of 4 to 32 when $V_{WL} = 0.5$ V. Furthermore, the absolute value of V_{BL} variations increase with the ideal DP value y_o since many more (up to N) BCs get activated and their cell currents contribute to the BL discharge.

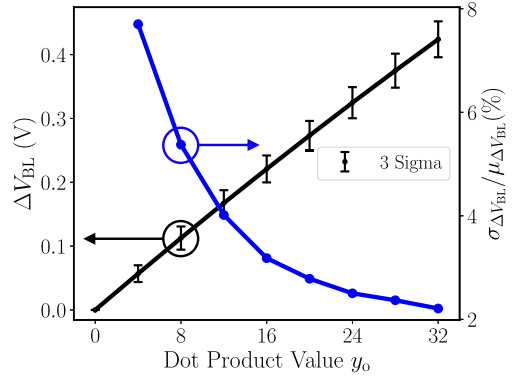


Fig. 2. The impact of process variations on the BL response ΔV_{BL} obtained via circuit simulations in a commercial 28 nm CMOS process when the WL voltage is $V_{WL} = 0.5$ V and $N = 32$.

C. Stochastic Signal Model

The impact of process variations on the BL voltage is stochastic since it is a function of the specific set of BCs activated by the input vector. This makes it challenging to compensate for them using circuit-level methods. In this paper, we develop statistical signal processing methods described in Section III, to compensate for BL voltage variations. To do so, we employ the following signal model of the IMC computation [25]:

$$y_1 = \mathbf{w}_\beta^T \mathbf{x} = \sum_{i=1}^N \beta_i w_i x_i, \quad (4)$$

where y_1 is the IMC output in the presence of spatial variations, $\mathbf{w}_\beta = [\beta_1 w_1, \dots, \beta_N w_N]^T$ is a noisy weight vector reflecting the impact of spatial variations, $\beta_i \sim \mathcal{N}(1, \sigma_\beta^2)$, $w_i \sim \text{Be}(p_w)$ and $x_i \sim \text{Be}(p_x)$ are i.i.d Bernoulli random variables. We further assume that β_i , w_i , and x_i are independent of each other $\forall i$. The value of σ_β ranges from 0.06 to 0.26 in 28 nm CMOS technology for a wordline voltage range of 0.5 V to 0.9 V.

D. Compute SNR

We employ the compute SNR defined as

$$\text{SNR} = \frac{\sigma_{y_o}^2}{\sigma_{y_{IMC} - y_o}^2} \quad (5)$$

to quantify the accuracy of the IMC computation for a binary DP, where $y_{IMC} = Q(y_1)$ is the post-ADC, i.e. the digital output of the IMC and $Q(\cdot)$ denotes the quantization function.

The numerator in (5) can be easily estimated by assuming $w_i \sim \text{Be}(p_w)$, $x_i \sim \text{Be}(p_x)$, and hence $y_o \sim \text{Bi}(N, p_x p_w)$ from (1), i.e., y_o is a binomial random variable with a known variance. The denominator in (5) can be estimated by sampling the distributions of w_i , x_i , and β_i to empirically compute y_o and y_1 (hence $y_{IMC} = Q[y_1]$) from (1) and (4), respectively.

III. PROPOSED STATISTICAL ERROR COMPENSATION METHODS

In this section, we propose maximum likelihood-based statistical error compensation (MLEC) methods to compensate

for the impact of spatial variations on the IMC output. These methods exploit the inherent symmetry in the structure of the 6T SRAM BC (see Fig. 1) to extract multiple observations, and leverage the signal model in (4). In the rest of the paper, we assume that $x_i \sim Be(0.5)$ and $w_i \sim Be(0.5)$ for $i \in \{0, 1, 2, \dots, N\}$ are i.i.d. in order to make the derivation of ML detection rules mathematically tractable. Derivation of all the MLEC methods are provided in the Appendix.

A. Maximum Likelihood (ML) Detection

Given the signal model in (4), the classical ML detection rule using a single BL output y_1 is given by

$$\hat{y} = \arg \max_i P(y_1 | y_0 = i) = \arg \min_i |y_1 - i| = \lfloor y_1 \rfloor, \quad (6)$$

where $i \in \{0, 1, \dots, N\}$ and $\lfloor \cdot \rfloor$ denotes the rounding operation. The rounding operation in (6) is already realized by column ADCs in today's IMCs. Therefore, we explore ML methods based on multiple observations next.

B. Obtaining Multiple Observations

We leverage the symmetric structure of the 6T SRAM BCA to acquire four observations y_1, y_2, y_3 , and y_4 with very low complexity cost as follows:

$$\begin{aligned} y_1 &= \mathbf{w}_\beta^T \mathbf{x} = \sum_{i=1}^N \beta_i w_i x_i, & y_2 &= \overline{\mathbf{w}}_\beta^T \mathbf{x} = \sum_{i=1}^N \beta_i \overline{w}_i x_i, \\ y_3 &= \mathbf{w}_\beta^T \overline{\mathbf{x}} = \sum_{i=1}^N \beta_i w_i \overline{x}_i, & y_4 &= \overline{\mathbf{w}}_\beta^T \overline{\mathbf{x}} = \sum_{i=1}^N \beta_i \overline{w}_i \overline{x}_i, \end{aligned} \quad (7)$$

where the overline notation represents a binary complement. Note that the four observations are statistically independent conditioned on the input vector \mathbf{x} and weight vector \mathbf{w} since each β_i will only appear in one of the four observations depending on the value of \mathbf{x} and \mathbf{w} . Fig. 1 shows that y_2 is computed on the BL-bar (BLB) node simultaneously with y_1 since the i -th BC discharges BLB when $x_i = 1$ and $w_i = 0$. Therefore, y_2 is obtained for free by exploiting the symmetry of the 6T SRAM structure.

Observations y_3 and y_4 in (7) are the \mathbf{x} -flipped versions of y_1 and y_2 , respectively. A brute force approach to obtain y_3 and y_4 is to invoke the IMC with a $\overline{\mathbf{x}}$ input vector. Doing so would incur a doubling of the energy consumption. Instead, we first generate observations $y_1 + y_3$ and $y_2 + y_4$ as follows:

$$y_1 + y_3 = \mathbf{w}_\beta^T \mathbf{x} + \mathbf{w}_\beta^T \overline{\mathbf{x}} = \sum_{i=1}^N \beta_i w_i = n_{\mathbf{w}_\beta}, \quad (8)$$

$$y_2 + y_4 = \overline{\mathbf{w}}_\beta^T \mathbf{x} + \overline{\mathbf{w}}_\beta^T \overline{\mathbf{x}} = \sum_{i=1}^N \beta_i \overline{w}_i = n_{\overline{\mathbf{w}}_\beta}, \quad (9)$$

where $n_{\mathbf{w}_\beta}$ is the sum of all elements of \mathbf{w}_β (soft Hamming weight), which the sampled value can be computed by sending a 1s input vector into the IMC and storing the resulting output. This calibration step incurs a one-time cost when new weights are loaded into the array. This cost is minimal because of the high arithmetic intensity for weights in a matrix-vector multiply

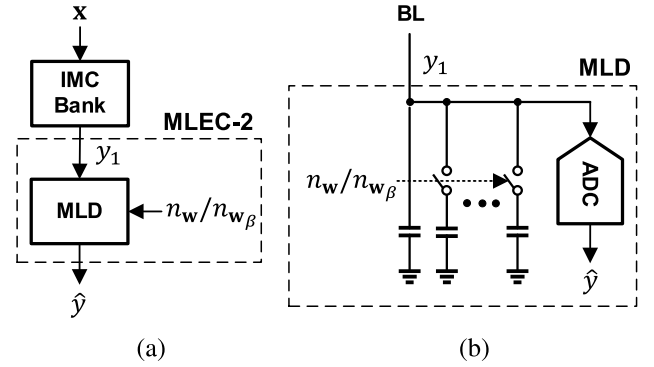


Fig. 3. MLEC-2: (a) architecture, and (b) circuit realization of the MLD block.

operation realized by the IMC. Furthermore, this step can be repeated in order to compensate for the impact of transistor aging, changes in the ambient temperature, and supply voltage noise. As $n_{\mathbf{w}_\beta}$ ($n_{\overline{\mathbf{w}}_\beta}$) and y_1 (y_2) provide sufficient information regarding y_3 (y_4), all four observations can be obtained with negligible computational and hence energy costs.

Note: each of the four observations y_1, y_2, y_3 and y_4 has a different variance, e.g., from (4), the variance of observation y_1 is given by

$$\sigma_{y_1}^2 = \sum_{i=1}^N \sigma_{\beta_i}^2 (w_i x_i)^2 = y_0^{(1)} \sigma_\beta^2, \quad (10)$$

where $y_0^{(1)} = y_0 = \mathbf{w}^T \mathbf{x}$ is the ideal value of y_1 given by (1), i.e., when $\beta_i = 1 \forall i$ in (7). Similarly, y_2, y_3 , and y_4 have a variance proportional to their corresponding ideal DP values $y_0^{(2)}, y_0^{(3)}$, and $y_0^{(4)}$, respectively. ML detection considers these differences in the variance of the observations and optimally combines them to generate a detected output \hat{y} that equals the ideal DP $y_0 (= y_0^{(1)})$ in (1) with high probability as shown next.

C. MLEC with Two Observations (MLEC-2)

There are six possible two observation 2-tuples that can be formed from the four observations y_1, y_2, y_3 , and y_4 defined in (7). Of these, employing y_1 and y_3 requires the least complexity since the closed-form expression of the detection rule has an efficient hardware implementation as shown next.

Employing the two observations y_1 and y_3 and assuming the distribution of inputs and weights $x_i, w_i \sim Be(0.5)$, we obtain the following approximation to the ML rule (see Appendix A):

$$\hat{y} = \arg \max_j P(y_1, y_3 | y_0^{(1)} = j) \approx \left\lfloor y_1 \frac{n_{\mathbf{w}}}{n_{\mathbf{w}_\beta}} \right\rfloor. \quad (11)$$

The overall architecture of MLEC-2 (see Fig. 3(a)) includes an IMC bank and an ML detector (MLD). The product of y_1 and $n_{\mathbf{w}}/n_{\mathbf{w}_\beta}$ can be realized efficiently using a switched capacitor DAC (CDAC) as shown in Fig. 3(b). The input $n_{\mathbf{w}}/n_{\mathbf{w}_\beta}$ is computed once at initialization when the weight parameters are loaded into the BCA. Finally, the rounding operation is realized by the column ADC of the IMC.

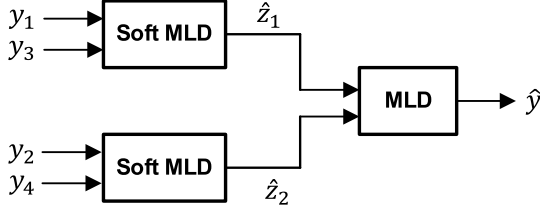


Fig. 4. A two-stage cascaded ML detector architecture based on a tournament tree.

D. MLEC With Four Observations (MLEC-4)

To enhance the detection performance beyond MLEC-2, we present three types of MLEC-4 methods: 1) Exact MLEC-4 (E-MLEC-4); 2) Distribution-aware MLEC-4 (DA-MLEC-4); and 3) Energy-aware MLEC-4 (EA-MLEC-4), all of which employ all four observations y_1, y_2, y_3 , and y_4 , in (7) to generate an ML decision rule. The distribution- and energy-aware MLEC methods trade off accuracy with computational complexity by approximating E-MLEC-4.

1) *Exact MLEC-4 (E-MLEC-4)*: This method obtains the following ML rule (see Appendix B):

$$\begin{aligned} \hat{y} &= \arg \max_j P(y_1, y_2, y_3, y_4 | y_o^{(1)} = j) \\ &= \arg \min_j \left\{ \ln [j(n_x - j)(n_w - j)(N - n_w - n_x + j)] \right. \\ &\quad + \frac{1}{\sigma_\beta^2} \left[\frac{(y_1 - j)^2}{j} + \frac{(n_{w\beta} - y_1 - n_w + j)^2}{n_w - j} \right. \\ &\quad \left. \left. + \frac{(y_2 - n_x + j)^2}{n_x - j} + \frac{(n_{\bar{w}\beta} - y_2 - n_{\bar{w}} + n_x - j)^2}{n_{\bar{w}} - n_x + j} \right] \right\} \end{aligned} \quad (12)$$

where n_x is the Hamming weight of activation vector \mathbf{x} . Further simplification of (12) is mathematically intractable making it difficult to find an efficient hardware architecture. Therefore, we propose a tournament tree approximation of (12) that utilizes a two-stage cascaded ML detector as shown in Fig. 4.

In Section IV, we solve (12) via brute-force search in order to establish the upper bound on the SNR boost provided by approximations (DA-MLEC-4 and EA-MLEC-4) of E-MLEC-4 which are described next. We do not show the performance of the tournament tree architecture of Fig. 4 since DA-MLEC-4 was found to provide an SNR boost very close to the exact solution E-MLEC-4.

2) *Distribution-Aware MLEC-4 (DA-MLEC-4)*: This method exploits weight distribution to reduce the complexity of the solution to (12). We show later in Section IV that the SNR boost provided by DA-MLEC-4 is very close to that of E-MLEC-4. Fig. 5 illustrates the block diagram of the DA-MLEC-4 method. The first stage (MLD1) performs soft ML detection employing two observation pairs (y_1, y_3) and (y_2, y_4) to generate soft outputs \hat{z}_1 and \hat{z}_2 , respectively, as follows:

$$\hat{z}_1 = \frac{y_1 n_w}{n_{w\beta}}; \quad \hat{z}_2 = \frac{y_2 n_{\bar{w}}}{n_{\bar{w}\beta}} \quad (13)$$

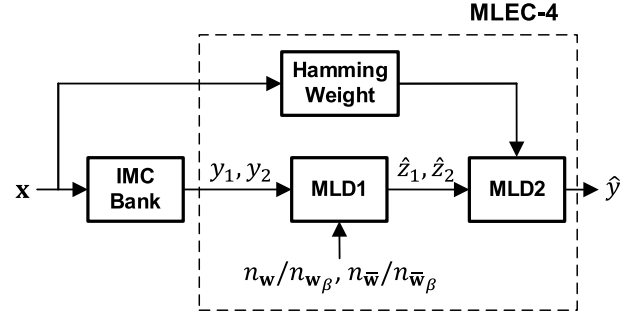


Fig. 5. The proposed MLEC-4 method.

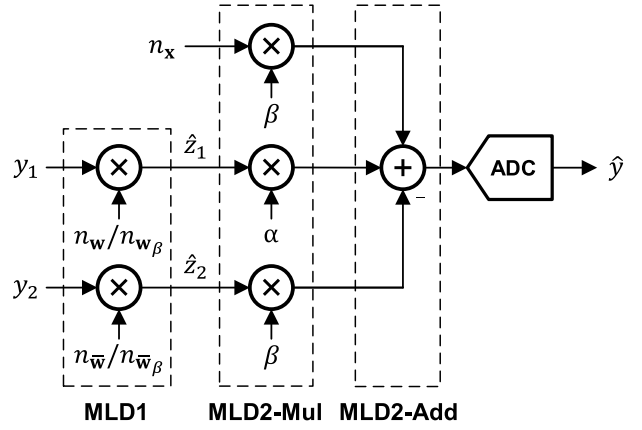


Fig. 6. The DA-MLEC-4 architecture.

where $[\hat{z}_1]$ and $[\hat{z}_2]$ are ML estimates of $y_o^{(1)}$ and $y_o^{(2)}$, respectively, as in the MLEC-2 rule (11). Equation (13) is obtained by employing the relationships: $y_3 = n_{w\beta} - y_1$ and $y_4 = n_{\bar{w}\beta} - y_2$ (see (8) and (9)).

The second stage (MLD2) employs the MLD1 outputs \hat{z}_1 and \hat{z}_2 to generate the final decision \hat{y} as shown below (see Appendix C):

$$\hat{y} = \arg \max_j P(\hat{z}_1, \hat{z}_2 | y_o^{(1)} = j) = \lfloor \beta n_x + \alpha \hat{z}_1 - \beta \hat{z}_2 \rfloor \quad (14)$$

where $\alpha = 1 - \beta = n_{\bar{w}}/N$ assuming that \hat{z}_1 and \hat{z}_2 are normally distributed with variances $\sigma_{\hat{z}_1}^2 = n_w \sigma^2$ and $\sigma_{\hat{z}_2}^2 = n_{\bar{w}} \sigma^2$, respectively, thereby reflecting the impact of weight distribution. Since closed-form expressions of the distributions of \hat{z}_1 and \hat{z}_2 are difficult to obtain, we approximate them, by employing sample statistics obtained from simulation, to normal distributions with a variance ratio $\sigma_{\hat{z}_1}^2 / \sigma_{\hat{z}_2}^2 = n_w / n_{\bar{w}}$.

Fig. 6 shows the DA-MLEC-4 architecture. Multiplication for MLD1 is followed by a multiplication (MLD2-Mul) and addition (MLD2-Add) for MLD2 and realized in analog. The output of MLD2-Add passes through an ADC to convert the final detection output in digital. Detailed analog implementations of MLD1, MLD2-Mul, and MLD2-Add are provided with Fig. 13 in Appendix D.

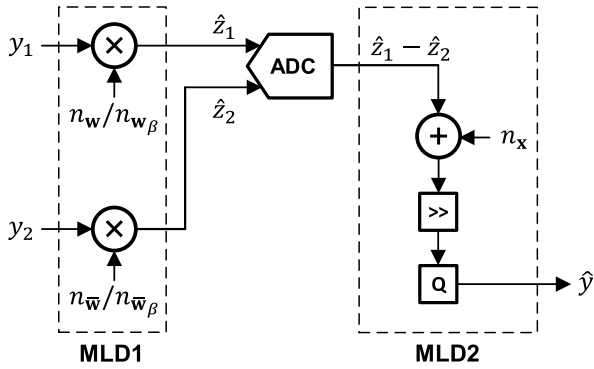


Fig. 7. The EA-MLEC-4 method realized by implementing MLD1 in analog and MLD2 in digital.

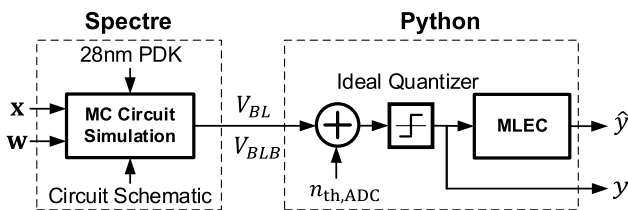


Fig. 8. Block diagram of the validation model comprising of circuit simulation and Python model.

3) *Energy-Aware MLEC-4 (EA-MLEC-4)*: This method is obtained by approximating the MLD2 stage of DA-MLEC-4 as follows:

$$\hat{y} = \arg \max_j P(\hat{z}_1, \hat{z}_2 | y_o = j) \approx \left\lfloor \frac{n_x + \hat{z}_1 - \hat{z}_2}{2} \right\rfloor, \quad (15)$$

where we assume that \hat{z}_1 and \hat{z}_2 have equal variances, i.e., $\sigma_{\hat{z}_1}^2 = \sigma_{\hat{z}_2}^2$, and hence $\alpha = \beta = 0.5$. This assumption is increasingly accurate when $p_w = 0.5$.

EA-MLEC-4 can be realized by dropping the MLD2-Mul block in DA-MLEC-4 (Fig. 6). Another variant of EA-MLEC-4 studied in [34] computes the MLD2 stage in the digital domain at the expense of a higher energy overhead (see Fig. 7). In this paper, we focus on the former where the entire EA-MLEC-4 is implemented in the analog domain.

IV. SIMULATION RESULTS

In this section, we present simulation results to quantify the accuracy enhancement via the proposed MLEC methods, first at the bank level and then at the network level. Then, the accuracy-energy trade-off of each method is studied by combining the energy overhead estimation with the accuracy results.

A. Validation Methodology

Bank-level: The SNR boost provided by the proposed MLEC methods is validated through a circuit-aware behavioral model (see Fig. 8) of a N_R -row and 128-column BCA via circuit simulations in a commercial 28 nm CMOS technology and a behavioral model of the ADC as described below.

The Python model captures the impact of process variations and transient noise seen on the BCA outputs V_{BL} and V_{BLB} via Monte Carlo circuit simulations. The WL pulse width T_{WL} is set to 200 ps which is sufficient to drive 128 SRAM columns. We ensure that the probability of read upsets [11] is minimized by enforcing the condition $\Pr\{V_{BL} < V_{DD} - V_1\} < 10^{-12}$. This is done by controlling either the DP dimension (N) or the number of BCA rows (N_R) for a given value of V_{WL} .

The ADC is modeled as a noisy quantizer with its input noise standard deviation set to the thermal noise value $\sigma_{n_{th,ADC}} = 0.5$ mV observed in a nominal successive approximation (SAR) ADC architecture [37]. Furthermore, we clip the ADC's input range per [32] to maximize the signal-to-quantization noise ratio (SQNR).

The IMC DP accuracy is quantified via the compute SNR, defined in (5), by comparing the IMC output with the output of an ideal binary DP. The activation x vector is obtained by i.i.d. sampling each element from a Bernoulli distribution with $p_x = 0.5$. The compute SNR for each configuration is estimated via Monte Carlo simulations with 200,000 trials.

Network-level: An 8-bit input and 4-bit weight quantized ResNet-20 is obtained through quantization-aware training on the CIFAR-10 dataset and is used as the baseline network. The software baseline test accuracy of the network is 92.51%. The MVM operations of all the residual blocks and the FC layer are mapped to the circuit-aware Python model described above. In order to accelerate the network simulations, this Python model substitutes the circuit simulation-based outputs of the BCA used to obtain the results for bank-level validation, with a voltage-scaled signal model based on (4) that captures the effect of spatial variations. The value of σ_β is extracted from circuit simulations for various wordline voltages V_{WL} . The bitline voltage V_{BL} is constrained to prevent read-upsets as in the bank-level validation setting.

B. Accuracy Results

Bank-level: We study the effectiveness of MLEC in boosting the compute SNR by sweeping the key design parameters: 1) ADC precision B_{ADC} , 2) weight probability p_w , and 3) the wordline voltage V_{WL} .

Fig. 9(a) shows that the compute SNR (for $N = 144$) of all methods, including the uncompensated IMC, improves with the ADC precision B_{ADC} until B_{ADC} equals the SNR-optimal value of $B_{ADC}^* = 6$ -b. This implies that the compute SNR is dominated by ADC quantization noise at lower values of B_{ADC} and eventually by cell current variations and ADC noise when $B_{ADC} \geq B_{ADC}^*$.

MLEC-2 and E-MLEC-4 enhances the compute SNR by 3.3 dB and 7.3 dB, respectively, over the uncompensated IMC, at the optimal ADC precision of $B_{ADC}^* = 6$. The SNR boosts provided by DA-MLEC-4 and EA-MLEC-4 are 6.6 dB and 6.4 dB, respectively, which are less than that of E-MLEC-4. This is to be expected since DA-MLEC-4 and EA-MLEC-4 are approximations of E-MLEC-4 as discussed in Section III.

Results similar to Fig. 9(a) were obtained for different DP dimensions N using the optimal ADC precision B_{ADC}^* and clip

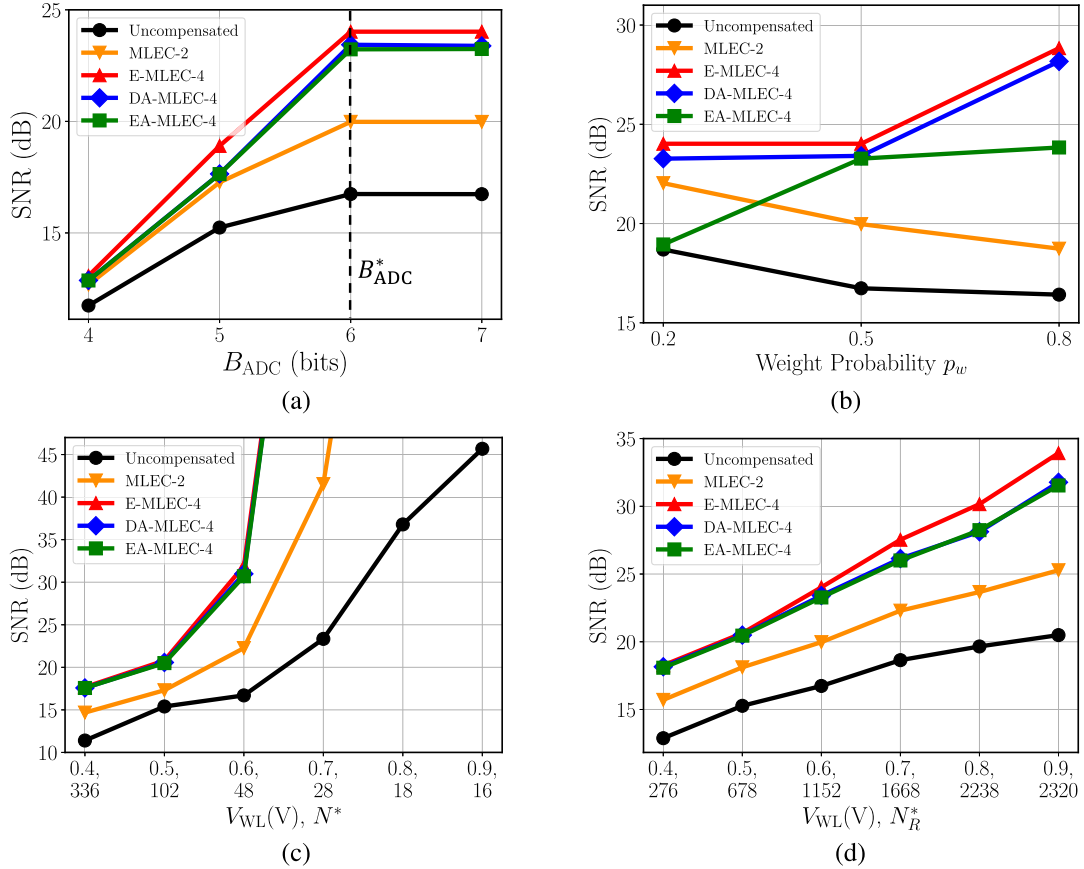


Fig. 9. Compute SNR enhancement via the proposed MLEC methods – MLEC-2, E-MLEC-4, DA-MLEC-4, and EA-MLEC-4 – showing: (a) SNR vs. ADC precision B_{ADC} with $N = 144$ and $N_R = 1152$, (b) SNR vs. weight probability p_w with $N = 144$ and $N_R = 1152$, (c) SNR vs. V_{WL} and N^* with $N_R = 512$, and (d) SNR vs. V_{WL} and N_R^* with $N = 144$. Here $p_w = 0.5$, and $V_{WL} = 0.6$ V are used as the base values if not mentioned. Optimal ADC precision B_{ADC}^* for each DP dimension N and its corresponding clip range is summarized in Table I.

TABLE I
OPTIMAL ADC PRECISION B_{ADC}^* AND CORRESPONDING CLIP RANGE FOR DIFFERENT DOT PRODUCT DIMENSIONS N AND $N_R = 512$

	Dot Product Dimension N						
	336	144	102	48	28	18	16
B_{ADC}^*	7	6	5	5	4	4	4
Clip Range	[20, 148]	[4, 68]	[10, 42]	[0, 32]	[0, 16]	[0, 16]	[0, 16]

ranges listed in Table I. In the rest of the plots, we choose $B_{ADC} = B_{ADC}^*$.

Fig. 9(b) quantifies the impact of weight probability on the SNR boost. It shows that: 1) DA-MLEC-4's SNR boost is close to the optimum, i.e., SNR boost achieved by E-MLEC-4, for all weight probability values, while 2) EA-MLEC-4 shows a considerable gap-to-optimum in SNR for extreme probability values of $p_w = 0.2$ and $p_w = 0.8$. The reason for this difference is that DA-MLEC-4 accounts for the difference between the variances of \hat{z}_1 and \hat{z}_2 via the use of weighing factors α

and β in (14) while EA-MLEC-4 does not (see (15)). Since the variance ratio $\sigma_{\hat{z}_1}^2 / \sigma_{\hat{z}_2}^2 = n_w / n_{\bar{w}}$ is a function of weight probability p_w , SNR boost is improved if the detector accounts for it. Not surprisingly, DA-MLEC-4 and EA-MLEC-4 achieve the same SNR boost at $p_w = 0.5$ where the variances of \hat{z}_1 and \hat{z}_2 are equal.

Fig. 9(c) shows the compute SNR versus WL voltage V_{WL} for a fixed number $N_R = 512$ of BCA rows. The realizable DP dimension N^* , i.e., the maximum value of N at which read-upssets are avoided, increases as V_{WL} reduces (see x-axis of Fig. 9(c)). This trend occurs because a reduced WL voltage reduces the cell current and thereby enabling higher DP values to be computed while satisfying the condition $\Pr\{V_{BL} < V_{DD} - V_t\} < 10^{-12}$.

Fig. 9(c) also shows that the compute SNR of MLEC methods increases exponentially with V_{WL} though at the expense of reduced N^* . Specifically, MLEC-4 methods achieve close to error-free decisions, i.e., $\Pr\{\hat{y} \neq y_o\} < 5 \times 10^{-6}$, when $V_{WL} > 0.6$ V. Similarly, MLEC-2 also shows error-free behavior for $V_{WL} > 0.7$ V. These results indicate that the proposed MLEC methods can emulate a fixed-point digital architecture in spite of employing analog computations provided the V_{WL} is sufficiently high.

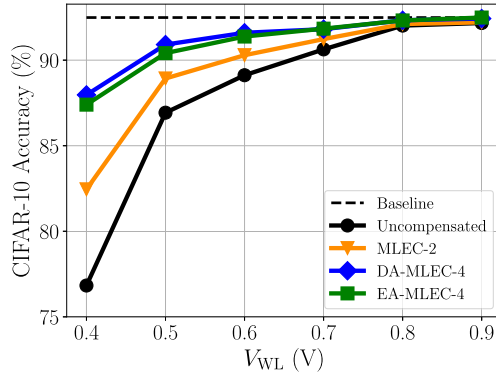


Fig. 10. Effect of the wordline voltage V_{WL} for a fixed DP dimension of $N = 144$ on the inference accuracy IMCs executing the ResNet-20 network on CIFAR-10 dataset. The baseline accuracy of a 8b-activation, 4b-weight quantized fixed-point architecture is 92.49%.

Fig. 9(d) shows that the compute SNR improves with the WL voltage V_{WL} for all methods. For each V_{WL} , the number of BCA rows is set to the minimum value N_R^* that ensures $\Pr\{V_{BL} < V_{DD} - V_t\} < 10^{-12}$ as shown in the x-axis. Specifically, MLEC-2 enhances the compute SNR by 2.9 dB-to-4.8 dB over the uncompensated IMC, whereas DA-MLEC-4 and EA-MLEC-4 provide an SNR boost of 5.2 dB-to-11 dB as V_{WL} is varied, for a fixed DP dimension $N = 144$. The main reason for this trend is seen from (3) which indicates that the impact of spatial V_t variations on the cell current reduces as V_{WL} is increased.

Network-level: Next, we study the impact of the bank-level compute SNR boost provided by MLEC methods on the classification accuracy of a DNN. The values of parameters were set to $B_{ADC}^* = 6$, $\sigma_{m_{th, ADC}} = 0.5mV$, and DP dimension of $N = 144$. The number of physical rows N_R in the BCA was set to N_R^* per Fig. 9(d).

Fig. 10 plots the accuracy of ResNet-20 deployed on uncompensated and error-compensated IMCs while sweeping V_{WL} (see Table II in Appendix for exact values). We find that the classification accuracy of the uncompensated IMC drops from its ideal fixed-point value of 92.51% to $< 90\%$ for $V_{WL} \leq 0.6V$. Similarly, MLEC-2 achieves an accuracy $> 90\%$ for $V_{WL} \geq 0.6V$. On the other hand, MLEC-4 methods achieve an accuracy $> 90\%$ for $V_{WL} \geq 0.5V$. This indicates that MLEC methods achieve iso-network-level accuracy at a reduced energy consumption.

Overall, we observe an exponential decrease in network accuracy with a drop in the bank-level compute SNR. Furthermore, comparing Figs. 9(d) and 10, we find that a compute SNR in the range 20 dB-to-23 dB will ensure that the drop in the classification accuracy is $< 1\%$ w.r.t. that of an ideal fixed-point digital implementation. This result provides a target SNR specification on IMC bank designs.

C. Accuracy vs. Energy Trade-Off

In this subsection, we analyze the energy overhead of each method based on the energy models in Appendix F. The

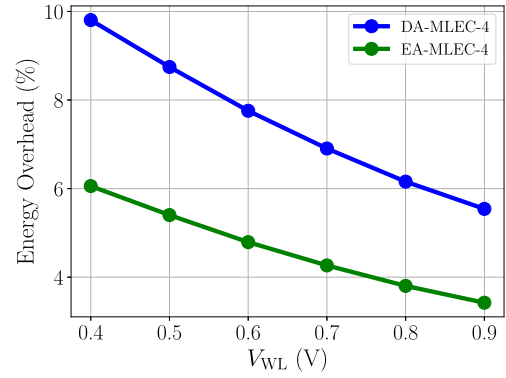


Fig. 11. Energy overhead vs. wordline voltage V_{WL} for DA-MLEC-4 and EA-MLEC-4 methods for a fixed DP dimension of $N = 144$.

MLEC-2 method does not incur any energy overhead as explained in Appendix F, hence we do not consider it any further.

Fig. 11 shows that the energy overhead of the two MLEC-4 methods, DA-MLEC-4 and EA-MLEC-4, ranges from 3.4%-to-6.1% (for EA-MLEC-4), and from 5.5%-to-9.8% (for DA-MLEC-4). Thus, DA-MLEC-4 incurs roughly $1.5\times$ the energy overhead compared to EA-MLEC-4 while providing a slight improvement in network-level accuracy per Fig. 10.

Furthermore, note that the relative energy overhead decreases with an increase in V_{WL} for both methods. This trend is expected because the overall energy consumption of the IMC bank increases while the overhead of MLEC remains constant. Additionally, EA-MLEC-4 has a lower energy overhead compared to the 10%-to-30% energy overhead of a fixed-point digital implementation studied [34], hence it is the preferred method for achieving both high accuracy and high energy efficiency.

Fig. 12 studies the trade-off between accuracy and energy by sweeping the WL voltage V_{WL} for a fixed DP dimension of $N = 144$, at the bank- (Fig. 12(a)) and the network-levels (Fig. 12(b)).

Bank-level: At the bank-level, Fig. 12(a) shows that the MLEC-2 method improves the SNR by 3 dB-to-5 dB for the same energy efficiency while MLEC-4 methods improve the SNR by 3 dB-to-7 dB. For a target SNR of 20 dB, MLEC-2, DA-MLEC-4, and EA-MLEC-4 achieve a 34.1%, 40.7%, and 45.6% increase in energy efficiency, respectively. It can be seen that EA-MLEC-4 exhibits the best accuracy vs. energy-efficiency trade-off compared to other methods.

Network-level: Fig. 12(b) shows the inference accuracy vs. energy efficiency of a ResNet-20 network. For an energy efficiency of 350 TOPS/W, both MLEC-4 methods incur a 1.51% accuracy drop w.r.t. baseline compared to 3.78% for the uncompensated IMC. For an accuracy target of $> 90\%$, EA-MLEC-4 achieves an 18% increased energy efficiency over the uncompensated IMC.

The accuracy vs. energy-efficiency trade-off can be further improved via more efficient implementations of the proposed MLEC methods.

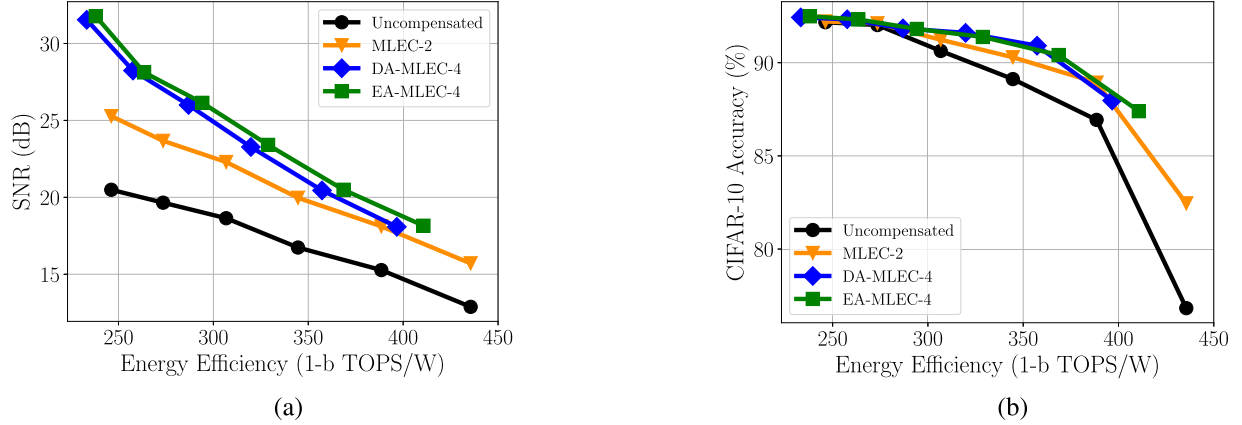


Fig. 12. Accuracy vs. energy efficiency (in terms of 1-bit TOPS/W (teraops/Watt)) of uncompensated and MLEC-compensated IMCs: (a) SNR vs. energy-efficiency for a fixed DP dimension of $N = 144$, and (b) inference accuracy vs. energy-efficiency of a ResNet-20 network (CIFAR-10 dataset) implemented on uncompensated and MLEC-compensated IMCs.

V. CONCLUSION

In this paper, maximum likelihood (ML) detection-based statistical error compensation (MLEC) methods are proposed to enhance the accuracy of 6T SRAM-based IMC architectures. Simulation results show that the proposed MLEC method enables significant gains in the compute SNR with a modest energy cost, leading to an enhanced accuracy vs. energy-efficiency trade-off. Network-level accuracy gains are also validated through ResNet-20 on the CIFAR-10 dataset. The proposed MLEC methods clearly demonstrate improved accuracy-energy trade-off over an uncompensated IMC design. The diversity of IMC architectures is enormous spanning various types of SRAM-based and non-volatile memory (NVM)-based IMCs. Similar ML-based statistical error compensation methods can be developed to enhance the accuracy vs. energy-efficiency trade-offs for these as well. Our work indicates the potential of leveraging statistical signal processing approaches to enhance the accuracy of an important class of machine learning architectures, viz. IMCs.

APPENDIX A DERIVATION OF MLEC-2

Derivation of (11): Recall from (7) that all four observations y_1, y_2, y_3 and y_4 are independent of each other when conditioned on a specific value of the tuple (\mathbf{x}, \mathbf{w}) . Therefore:

$$\begin{aligned} P(y_1, y_3 | y_o^{(1)} = j) &= \sum_{(\mathbf{x}, \mathbf{w}) \in A_j} P(y_1, y_3 | (\mathbf{x}, \mathbf{w})) P((\mathbf{x}, \mathbf{w}) | y_o^{(1)} = j) \\ &= \sum_{(\mathbf{x}, \mathbf{w}) \in A_j} P(y_1 | (\mathbf{x}, \mathbf{w})) P(y_3 | (\mathbf{x}, \mathbf{w})) P((\mathbf{x}, \mathbf{w}) | y_o^{(1)} = j) \\ &= P(y_1 | y_o^{(1)} = j) P(y_3 | y_o^{(1)} = j), \end{aligned} \quad (16)$$

where $A_j = \{(\mathbf{x}, \mathbf{w}) : \mathbf{w}^T \mathbf{x} = j\}$, and we exploit the fact that $P((\mathbf{x}, \mathbf{w}) | y_o^{(1)} = j) = 1/|A_j| \mathbb{1}\{(\mathbf{x}, \mathbf{w}) \in A_j\}$ where $|A_j|$ is the cardinality of set A_j since x_i and w_i are assumed to be i.i.d..

Conditioned on the event $\{y_o^{(1)} = j\}$, y_1 (y_3) becomes a sum of j ($n_w - j$) i.i.d. Gaussian RVs $\beta_i \sim \mathcal{N}(1, \sigma_\beta^2)$, with distribution functions given by

$$P(y_1 | y_o^{(1)} = j) = \frac{1}{\sqrt{2\pi j \sigma_\beta^2}} \exp\left(-\frac{(y_1 - j)^2}{2j \sigma_\beta^2}\right), \quad (17)$$

and

$$\begin{aligned} P(y_3 | y_o^{(1)} = j) &= P(y_3 | y_o^{(3)} = n_w - j) \\ &= \frac{1}{\sqrt{2\pi(n_w - j) \sigma_\beta^2}} \exp\left(-\frac{(y_3 - (n_w - j))^2}{2(n_w - j) \sigma_\beta^2}\right) \end{aligned} \quad (18)$$

since $y_o^{(1)} + y_o^{(3)} = n_w$.

If $\hat{y} = \arg \max_j P(y_1, y_3 | y_o^{(1)} = j)$, then the following inequalities are satisfied:

$$\begin{aligned} \log\left(\frac{P(y_1, y_3 | y_o^{(1)} = \hat{y} + 1)}{P(y_1, y_3 | y_o^{(1)} = \hat{y})}\right) &< 0, \\ \log\left(\frac{P(y_1, y_3 | y_o^{(1)} = \hat{y})}{P(y_1, y_3 | y_o^{(1)} = \hat{y} - 1)}\right) &> 0, \end{aligned} \quad (19)$$

except when $\hat{y} = 0$ or $\hat{y} = n_w$. In the latter case, either one of the inequalities in (19) will be satisfied. Substituting (17), (18), and $y_3 = n_w - y_1$, into (16), to expand the L.H.S of the first inequality in (19) gives

$$\begin{aligned} &\log\left(\frac{P(y_1, y_3 | y_o^{(1)} = \hat{y} + 1)}{P(y_1, y_3 | y_o^{(1)} = \hat{y})}\right) \\ &= \frac{1}{2\sigma_\beta^2} \left[\frac{y_1^2}{\hat{y}(\hat{y} + 1)} - \frac{(n_w - y_1)^2}{(n_w - \hat{y})(n_w - \hat{y} - 1)} \right] \\ &\quad + \frac{1}{2} \log\left[\frac{\hat{y}(n_w - \hat{y})}{(\hat{y} + 1)(n_w - \hat{y} - 1)}\right]. \end{aligned} \quad (20)$$

The second term in the R.H.S. of (20) is negligible since the argument of the log function is close to unity. Therefore, the first inequality in (19) now becomes

$$\frac{1}{2\sigma_\beta^2} \left[\frac{y_1^2}{\hat{y}(\hat{y}+1)} - \frac{(n_{\mathbf{w}\beta} - y_1)^2}{(n_{\mathbf{w}} - \hat{y})(n_{\mathbf{w}} - \hat{y} - 1)} \right] < 0$$

$$\iff \frac{(n_{\mathbf{w}} - \hat{y} - \frac{1}{2})^2 - \frac{1}{4}}{(n_{\mathbf{w}\beta} - y_1)^2} < \frac{(\hat{y} + \frac{1}{2})^2 - \frac{1}{4}}{y_1^2}. \quad (21)$$

Furthermore, we approximate the inequality to

$$\frac{(n_{\mathbf{w}} - \hat{y} - \frac{1}{2})^2}{(n_{\mathbf{w}\beta} - y_1)^2} < \frac{(\hat{y} + \frac{1}{2})^2}{y_1^2} \quad (22)$$

since: 1) $\hat{y}(\hat{y}+1)$, $(n_{\mathbf{w}} - \hat{y})(n_{\mathbf{w}} - \hat{y} - 1) \gg 1/4$ in general and 2) $(n_{\mathbf{w}\beta} - y_1)^2 \approx y_1^2$ since $y_1 \approx n_{\mathbf{w}\beta}/2$ for $p_x = 1/2$. Solving (22) yields to

$$y_1 \frac{n_{\mathbf{w}}}{n_{\mathbf{w}\beta}} - \frac{1}{2} < \hat{y}. \quad (23)$$

The second inequality in (19) simplified in a similar manner, and combined with (23) yields to:

$$y_1 \frac{n_{\mathbf{w}}}{n_{\mathbf{w}\beta}} - \frac{1}{2} < \hat{y} < y_1 \frac{n_{\mathbf{w}}}{n_{\mathbf{w}\beta}} + \frac{1}{2}, \quad (24)$$

which can be re-written as

$$\hat{y} = \text{rnd} \left(y_1 \frac{n_{\mathbf{w}}}{n_{\mathbf{w}\beta}} \right), \quad (25)$$

resulting in (11). Note that (25) covers the exception $\hat{y} = 0$ and $\hat{y} = n_{\mathbf{w}}$ in (19).

APPENDIX B DERIVATION OF MLEC-4

Derivation of (12): All the four observations in (7) given $y_o^{(1)} = j$ can be shown to be statistically independent similar to (16). Thus, the joint conditional probability can be expressed as follows:

$$P(y_1, y_2, y_3, y_4 | y_o^{(1)} = j) = \prod_{k=1}^4 P(y_k | y_o^{(1)} = j) \quad (26)$$

From (7), the following can be easily shown:

$$y_o^{(1)} = n_{\mathbf{x}} - y_o^{(2)} = n_{\mathbf{w}} - y_o^{(3)} = n_{\mathbf{x}} - n_{\overline{\mathbf{w}}} + y_o^{(4)} \quad (27)$$

Combining (26), (27) with $\{y_k | y_o^{(1)} = j\} \sim \mathcal{N}(y_o^{(k)}, y_o^{(k)} \sigma_\beta^2)$

$$\begin{aligned} \log P(y_1, y_2, y_3, y_4 | y_o^{(1)} = j) &= -\frac{1}{2} \ln [j(n_{\mathbf{x}} - j)(n_{\mathbf{w}} - j)(N - n_{\mathbf{w}} - n_{\mathbf{x}} + j)] \\ &\quad - \frac{1}{2\sigma_\beta^2} \left[\frac{(y_1 - j)^2}{j} + \frac{(y_2 - n_{\mathbf{x}} + j)^2}{n_{\mathbf{x}} - j} \right. \\ &\quad \left. + \frac{(y_3 - n_{\mathbf{w}} + j)^2}{n_{\mathbf{w}} - j} + \frac{(y_4 - n_{\overline{\mathbf{w}}} + n_{\mathbf{x}} - j)^2}{n_{\overline{\mathbf{w}}} - n_{\mathbf{x}} + j} \right] + C, \end{aligned} \quad (28)$$

where C is a constant. Substituting y_3 and y_4 using the relationship in (8) and (9), yield the argmin expression in (12) since the argmin of the negative log of the original function is the argmax of the original function.

APPENDIX C DERIVATION OF DA-MLEC-4

Derivation of (14): It can be shown from (13) that the mean of \hat{z}_1 and \hat{z}_2 are j and $n_{\mathbf{x}} - j$, respectively, given $y_o^{(1)} = j$. We assume that \hat{z}_1 and \hat{z}_2 follow a normal distribution with variances of $n_{\mathbf{w}}\sigma^2$ and $n_{\overline{\mathbf{w}}}\sigma^2$ given $y_o^{(1)} = j$, respectively, where σ is a constant.

Then, the joint conditional probability can be expressed as follows:

$$\begin{aligned} P(\hat{z}_1, \hat{z}_2 | y_o^{(1)} = j) &= P(\hat{z}_1 | y_o^{(1)} = j) P(\hat{z}_2 | y_o^{(1)} = j) \\ &= \frac{1}{2\pi\sigma^2 \sqrt{n_{\mathbf{w}}n_{\overline{\mathbf{w}}}}} \exp \left(-\frac{(\hat{z}_1 - j)^2}{2n_{\mathbf{w}}\sigma^2} - \frac{(\hat{z}_2 - (n_{\mathbf{x}} - j))^2}{2n_{\overline{\mathbf{w}}}\sigma^2} \right). \end{aligned} \quad (29)$$

Similar to (19), if $\hat{y} = \arg \max_j P(\hat{z}_1, \hat{z}_2 | y_o = j)$, the following inequalities will be satisfied:

$$\begin{aligned} \log \left(\frac{P(\hat{z}_1, \hat{z}_2 | y_o^{(1)} = \hat{y} + 1)}{P(\hat{z}_1, \hat{z}_2 | y_o^{(1)} = \hat{y})} \right) &< 0, \\ \log \left(\frac{P(\hat{z}_1, \hat{z}_2 | y_o^{(1)} = \hat{y})}{P(\hat{z}_1, \hat{z}_2 | y_o^{(1)} = \hat{y} - 1)} \right) &> 0. \end{aligned} \quad (30)$$

The first inequality can be expanded as follows:

$$\frac{2\hat{z}_2 - 2(n_{\mathbf{x}} - \hat{y}) + 1}{n_{\overline{\mathbf{w}}}} < \frac{2\hat{z}_1 - 2\hat{y} - 1}{n_{\overline{\mathbf{w}}}}, \quad (31)$$

which yields to

$$\frac{n_{\mathbf{w}}n_{\mathbf{x}} + n_{\overline{\mathbf{w}}}\hat{z}_1 - n_{\mathbf{w}}\hat{z}_2}{n_{\mathbf{w}} + n_{\overline{\mathbf{w}}}} - \frac{1}{2} < \hat{y}. \quad (32)$$

By solving the second inequality in (30) and combining the result with (32), we obtain:

$$\hat{y} = \text{rnd} \left(\frac{n_{\mathbf{w}}n_{\mathbf{x}} + n_{\overline{\mathbf{w}}}\hat{z}_1 - n_{\mathbf{w}}\hat{z}_2}{n_{\mathbf{w}} + n_{\overline{\mathbf{w}}}} \right). \quad (33)$$

Since $n_{\mathbf{w}} + n_{\overline{\mathbf{w}}} = N$, the above equation can be re-written as (14).

APPENDIX D ANALOG IMPLEMENTATION OF MLD1 AND MLD2

Fig. 13 shows the analog circuit implementation of the MLD1 and MLD2 blocks. MLD1 follows the same implementation method using CDACs as that of MLEC-2. \hat{z}_1 and \hat{z}_2 are computed in BL and BLB separately as shown in Fig. 13(a). Block MLD2 is realized by sequential stages: MLD2-Mul and MLD2-Add stages. Fig. 13(b) shows the implementation of MLD2-Mul via a charge redistribution-based mixed-signal multiplier (CRM) proposed in [38]. The B bit of the digital input $\alpha = n_{\mathbf{w}}/N$, where $B = \lceil \log_2 N \rceil$, controls the $\phi_{2,i}$ switches and generate the output $V_{\alpha\hat{z}_1} = \alpha V_{\hat{z}_1}$.

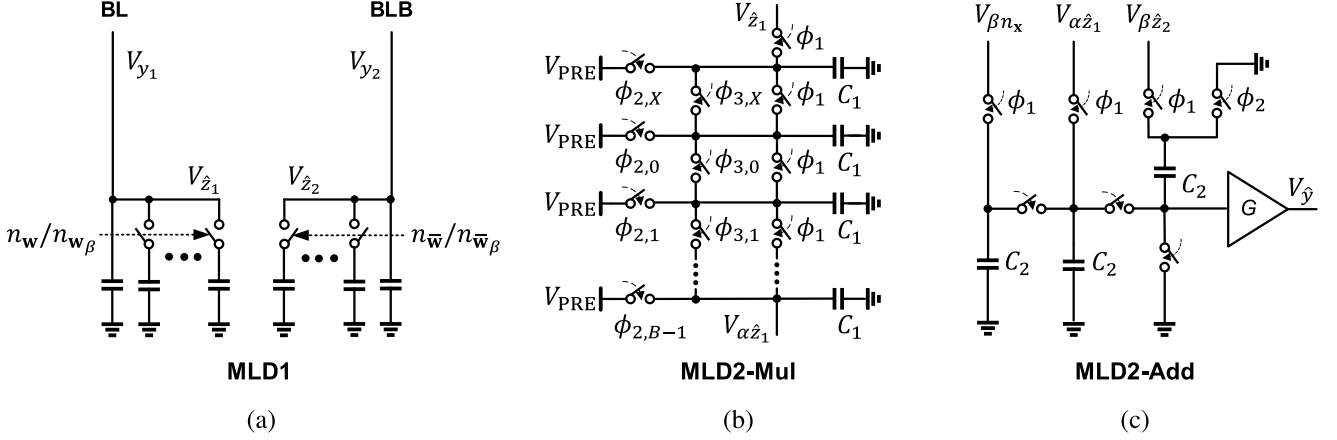


Fig. 13. Circuit schematic of the proposed DA-MLEC-4 in analog: (a) MLD1 block, (b) multiplication stage of the MLD2 block, and (c) addition stage of the MLD2 block.

TABLE II
TEST ACCURACIES FOR RESNET-20 ON CIFAR-10

Baseline 8/4-bit (activation/weight) Quantized ResNet-20				
6T-SRAM based IMC Architecture				
V_{WL}	Uncomp.	MLEC-2	DA-MLEC-4	EA-MLEC-4
0.9 V	92.17 %	92.21 %	92.43 %	92.49 %
0.8 V	92.02 %	92.11 %	92.33 %	92.32 %
0.7 V	90.63 %	91.23 %	91.81 %	91.84 %
0.6 V	89.12 %	90.29 %	91.60 %	91.38 %
0.5 V	86.93 %	88.93 %	90.91 %	90.41 %
0.4 V	76.83 %	82.45 %	87.97 %	87.40 %

Hamming weight n_x of the activation \mathbf{x} is calculated by a dedicated digital circuit as shown in Fig. 5 and converted to V_{n_x} by DAC. Then, $V_{\beta n_x}$ and $V_{\beta z_2}$ are also obtained through their dedicated CRMs. Finally, the MLD2-Add stage is realized by the charge redistribution circuit shown in Fig. 13(c) which takes the structure of the cross bitline processor (CBLP) in [38]. The addition $\beta n_x + \alpha z_1 - \beta z_2$ is computed and passed through an amplifier with a gain of G to restore the reduced signal range [9]. The final output is rounded and digitized by an ADC.

APPENDIX E NETWORK-LEVEL ACCURACY

Table II summarizes the ResNet-20 accuracy on CIFAR-10 dataset.

APPENDIX F ENERGY MODELS

The energy of a single column of the IMC bank and its dedicated MLEC processing unit is estimated in order to calculate the energy overhead of MLEC. Parameters and values used in

TABLE III
PARAMETERS VALUES FOR ENERGY ESTIMATION

Parameter	Value	Parameter	Value
V_{DD}	0.9 V	C_{WL}	0.3 fF
$\mathbb{E}[\Delta V_{BL}]$	144 mV	C_{BL}/N_R	0.6 fF
$\mathbb{E}[\Delta V_{C1}]$	72 mV	C_1	17 fF
$\mathbb{E}[\Delta V_{C2}]$	48 mV	C_2	25 fF
I_{bias}	20 μ A	T_s	2 ns

the estimation are extracted from 28 nm CMOS technology, which is summarized in Table III.

ADC Column: The energy consumption of an ADC column has the following components: 1) wordline driver energy per BC (E_{WLD}), 2) bitcell array energy (E_{BCA}), and 3) ADC energy (E_{ADC}), which are modeled as follows:

$$E_{WLD} = \mathbb{E}[n_x] C_{WL} V_{DD}^2, \quad (34)$$

$$E_{BCA} = \mathbb{E}[\Delta V_{BL}] V_{DD} C_{BL} + \mathbb{E}[\Delta V_{BLB}] V_{DD} C_{BLB}, \quad (35)$$

$$E_{ADC} = k_1 B_{ADC} + k_2 4^{B_{ADC}}, \quad (36)$$

where $\mathbb{E}[n_x]$ is the average number of activated WL rows, C_{WL} is the effective wordline capacitance per bitcell, $\mathbb{E}[\Delta V_{BL}]$ and $\mathbb{E}[\Delta V_{BLB}]$ are the mean voltage discharge on BL and BLB, respectively, and $k_1 = 100$ fJ and $k_2 = 1$ aJ are the empirical fitting parameters for modeling recent ADCs [39]. Optimal ADC precision B_{ADC}^* summarized in Table I is used for the ADC energy precision in (35) for each DP dimension. N_R is set to 4 times the DP dimension N to avoid signal clipping due to the voltage headroom of BL.

IMC Bank: The energy consumption of an IMC bank with N_C ADC columns can be modeled as:

$$E_{IMC} = N_C (E_{WLD} + E_{BCA} + E_{ADC}). \quad (37)$$

MLEC: We divide the energy of the MLD2 block into MLD2 multiplication stage (MLD2-Mul) and MLD2 addition stage (MLD2-Add), which the circuit implementation is shown in Fig. 13 in Appendix. The energy model for each component is as follows:

$$E_{\text{MUL}} = 3(\lceil \log_2 N \rceil + 1) \mathbb{E}[\Delta V_{C_1}] V_{\text{DD}} C_1, \quad (38)$$

$$E_{\text{ADD}} = 3\mathbb{E}[\Delta V_{C_2}] V_{\text{DD}} C_2 + I_{\text{bias}} V_{\text{DD}} T_s, \quad (39)$$

where C_1 and C_2 are the capacitance values used in the multiplication and addition stage of MLD2, respectively, $\mathbb{E}[\Delta V_{C_1}]$ and $\mathbb{E}[\Delta V_{C_2}]$ are the mean voltage discharge on capacitors in the multiplication and addition stage, respectively, I_{bias} is the bias current of the amplifier, T_s is the settling time of the amplifier. C_1 and C_2 values in Table III are chosen to meet the constraint of the thermal noise accumulated at the final output to be less than $0.2V_{\text{res}}$, where $V_{\text{res}} = 4 \text{ mV}$ for $N = 144$. This is to ensure less than 1 dB compute SNR drop for $V_{\text{WL}} = 0.6 \text{ V}$ and $N = 144$ in Section IV-B. I_{bias} and T_s of the amplifier are chosen to drive a 100 fF ADC sampling capacitor.

The energy of the Hamming weight block for MLEC depicted in Fig. 5 is negligible as it amortizes over multiple columns. Also, MLD1 simply adds extra load capacitance to the BL and BLB and the MLD1 computation is merged into the DP computation. This does not increase the energy consumption since the current drawn from each bitcell remains the same for a fixed WL pulse width T_{WL} and WL voltage V_{WL} .

ACKNOWLEDGMENT

The authors acknowledge the financial support from the JUMP 2.0 CUBIC and COCOSYS Centers funded by DARPA and the Semiconductor Research Corporation, and helpful discussions with Saion Roy, Shuo Li, and Hassan Dbouk.

REFERENCES

- [1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.
- [2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
- [3] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Piscataway, NJ, USA: IEEE Press, 2016, pp. 4960–4964.
- [4] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.
- [5] B. Wu, F. Iandola, P. H. Jin, and K. Keutzer, "SqueezeDet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2017, pp. 129–137.
- [6] X. Lian et al., "Persia: An open, hybrid system scaling deep learning-based recommenders up to 100 trillion parameters," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, 2022, pp. 3288–3298.
- [7] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.
- [8] M. Kang, M.-S. Keel, N. R. Shanbhag, S. Eilert, and K. Curewitz, "An energy-efficient VLSI architecture for pattern recognition via deep embedding of computation in SRAM," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Piscataway, NJ, USA: IEEE Press, 2014, pp. 8326–8330.
- [9] M. Kang, S. K. Gonugondla, and N. R. Shanbhag, "Deep in-memory architectures in SRAM: An analog approach to approximate computing," *Proc. IEEE*, vol. 108, no. 12, pp. 2251–2275, Dec. 2020.
- [10] X. Si et al., "15.5 A 28nm 64Kb 6T SRAM computing-in-memory macro with 8b MAC operation for AI edge chips," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, 2020, pp. 246–248.
- [11] J. Zhang, Z. Wang, and N. Verma, "In-memory computation of a machine-learning classifier in a standard 6T SRAM array," *IEEE J. Solid-State Circuits*, vol. 52, no. 4, pp. 915–924, Apr. 2017.
- [12] Z. Chen et al., "CAP-RAM: A charge-domain in-memory computing 6T-SRAM for accurate and precision-programmable CNN inference," *IEEE J. Solid-State Circuits*, vol. 56, no. 6, pp. 1924–1935, Jun. 2021.
- [13] S. Yin, Z. Jiang, J.-S. Seo, and M. Seok, "XNOR-SRAM: In-memory computing SRAM macro for binary/ternary deep neural networks," *IEEE J. Solid-State Circuits*, vol. 55, no. 6, pp. 1733–1743, Jun. 2020.
- [14] J. Kim et al., "Area-efficient and variation-tolerant in-memory BNN computing using 6T SRAM array," in *Proc. Symp. VLSI Circuits*, Piscataway, NJ, USA: IEEE Press, 2019, pp. C118–C119.
- [15] S. Yin et al., "PIMCA: A 3.4-Mb programmable in-memory computing accelerator in 28nm for on-chip DNN inference," in *Proc. Symp. VLSI Technol.*, Piscataway, NJ, USA: IEEE Press, 2021, pp. 1–2.
- [16] Z. Jiang, S. Yin, J.-S. Seo, and M. Seok, "C3SRAM: An in-memory-computing SRAM macro based on robust capacitive coupling computing mechanism," *IEEE J. Solid-State Circuits*, vol. 55, no. 7, pp. 1888–1897, Jul. 2020.
- [17] J. Yue et al., "15.2 A 2.75-to-75.9 TOPS/W computing-in-memory NN processor supporting set-associate block-wise zero skipping and ping-pong CIM with simultaneous computation and weight updating," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, vol. 64, Piscataway, NJ, USA: IEEE Press, 2021, pp. 238–240.
- [18] Q. Dongo et al., "15.3 A 351TOPS/W and 372.4 GOPS compute-in-memory SRAM macro in 7nm FinFET CMOS for machine-learning applications," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, Piscataway, NJ, USA: IEEE Press, 2020, pp. 242–244.
- [19] Y.-D. Chih et al., "16.4 an 89TOPS/W and 16.3 TOPS/mm² all-digital SRAM-based full-precision compute-in memory macro in 22nm for machine-learning edge applications," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, vol. 64, Piscataway, NJ, USA: IEEE Press, 2021, pp. 252–254.
- [20] H. Jia, H. Valavi, Y. Tang, J. Zhang, and N. Verma, "A programmable heterogeneous microprocessor based on bit-scalable in-memory computing," *IEEE J. Solid-State Circuits*, vol. 55, no. 9, pp. 2609–2621, Sep. 2020.
- [21] A. Biswas and A. P. Chandrakasan, "Conv-RAM: An energy-efficient SRAM with embedded convolution computation for low-power CNN-based machine learning applications," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, Piscataway, NJ, USA: IEEE Press, 2018, pp. 488–490.
- [22] X. Si et al., "24.5 A twin-8T SRAM computation-in-memory macro for multiple-bit CNN-based machine learning," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, Piscataway, NJ, USA: IEEE Press, 2019, pp. 396–398.
- [23] R. Guo et al., "A 5.1 pJ/neuron 127.3 us/inference RNN-based speech recognition processor using 16 computing-in-memory SRAM macros in 65nm CMOS," in *Proc. Symp. VLSI Circuits*, Piscataway, NJ, USA: IEEE Press, 2019, pp. C120–C121.
- [24] J.-W. Su et al., "15.2 a 28nm 64Kb inference-training two-way transpose multibit 6T SRAM Compute-in-Memory macro for AI edge chips," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, Piscataway, NJ, USA: IEEE Press, 2020, pp. 240–242.
- [25] S. K. Gonugondla, M. Kang, and N. Shanbhag, "A 42pJ/decision 3.12 TOPS/W robust in-memory machine learning classifier with on-chip training," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, Piscataway, NJ, USA: IEEE Press, 2018, pp. 490–492.
- [26] P. Mannocci, E. Melacarne, and D. Ielmini, "An analogue in-memory ridge regression circuit with application to massive MIMO acceleration," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 12, no. 4, pp. 952–962, Dec. 2022.
- [27] P. Mannocci et al., "Accelerating massive MIMO in 6G communications by analog in-memory computing circuits," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Piscataway, NJ, USA: IEEE Press, 2023, pp. 1–5.
- [28] N. R. Shanbhag and S. K. Roy, "Comprehending in-memory computing trends via proper benchmarking," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Piscataway, NJ, USA: IEEE Press, 2022, pp. 1–7.

- [29] B. Zhang, L.-Y. Chen, and N. Verma, "Stochastic data-driven hardware resilience to efficiently train inference models for stochastic hardware implementations," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Piscataway, NJ, USA: IEEE Press, 2019, pp. 1388–1392.
- [30] C. Zhou, P. Kadambi, M. Mattina, and P. N. Whatmough, "Noisy machines: Understanding noisy neural networks and enhancing robustness to analog hardware errors using distillation," 2020, *arXiv:2001.04974*.
- [31] H. Kim et al., "Direct gradient calculation: Simple and variation-tolerant on-chip training method for neural networks," *Adv. Intell. Syst.*, vol. 3, no. 8, 2021, Art. no. 2100064.
- [32] C. Sakr and N. R. Shanbhag, "Signal processing methods to enhance the energy efficiency of in-memory computing architectures," *IEEE Trans. Signal Process.*, vol. 69, pp. 6462–6472, 2021.
- [33] H.-M. Ou and N. R. Shanbhag, "Enhancing the accuracy of resistive in-memory architectures using adaptive signal processing," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Piscataway, NJ, USA: IEEE Press, 2023, pp. 1–5.
- [34] H. Kim and N. Shanbhag, "Boosting the accuracy of SRAM-Based in-memory architectures via maximum likelihood-based error compensation method," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Piscataway, NJ, USA: IEEE Press, 2023, pp. 1–5.
- [35] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A mixed-signal binarized convolutional-neural-network accelerator integrating dense weight storage and multiplication for reduced data movement," in *Proc. IEEE Symp. VLSI Circuits*, Piscataway, NJ, USA: IEEE Press, 2018, pp. 141–142.
- [36] S. K. Gonugondla, C. Sakr, H. Dbouk, and N. R. Shanbhag, "Fundamental limits on the precision of in-memory architectures," in *Proc. 39th Int. Conf. Comput.-Aided Des.*, 2020, pp. 1–9.
- [37] W. P. Zhang and X. Tong, "Noise modeling and analysis of SAR ADCs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 12, pp. 2922–2930, Dec. 2015.
- [38] M. Kang, S. K. Gonugondla, A. Patil, and N. R. Shanbhag, "A multi-functional in-memory inference processor using a standard 6T SRAM array," *IEEE J. Solid-State Circuits*, vol. 53, no. 2, pp. 642–655, Feb. 2018.
- [39] B. Murmann, "Mixed-signal computing for deep neural network inference," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 29, no. 1, pp. 3–13, Jan. 2021.



Hyungyo Kim (Student Member, IEEE) received the B.S. degree in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2020. He is currently working toward the Ph.D. degree in electrical and computer engineering with the University of Illinois at Urbana-Champaign, Champaign, IL, USA. His research interest includes algorithm hardware co-design for energy-efficient systems for machine learning applications.



Naresh R. Shanbhag (Fellow, IEEE) received the Ph.D. degree in electrical engineering from the University of Minnesota, Minneapolis, MN, USA, in 1993. He is currently the Jack Kilby Professor of Electrical and Computer Engineering with the University of Illinois at Urbana-Champaign, Champaign, IL, USA. From 1993 to 1995, he was with AT&T Bell Laboratories, Murray Hill, NJ, USA, where he led the design of high-speed transceiver chip-sets for very high-speed digital subscriber line, before joining the University of Illinois at Urbana-Champaign in 1995. He has held visiting faculty appointments with the National Taiwan University, Taipei, Taiwan, from 2007, and Stanford University, Stanford, CA, USA, from 2014. He holds 13 US patents, coauthored two books and multiple book chapters, and more than 200 publications in his research field, which include the design of energy-efficient systems for machine learning, communications, and signal processing, spanning algorithms, architectures, and integrated circuits. He was the recipient of the 2018 SIA/SRC University Researcher Award, the 2010 Richard Newton GSRC Industrial Impact Award, the IEEE Circuits and Systems Society Distinguished Lecturership in 1997, the National Science Foundation CAREER Award in 1996, and Multiple Best Paper Awards. In 2000, he co-founded and was the Chief Technology Officer of the Intersymbol Communications, Inc., which introduced mixed-signal ICs for electronic dispersion compensation of OC-192 optical links, and became a part of Finisar Corporation in 2007. From 2013 to 2017, he was the Founding Director of the Systems on Nanoscale Information Fabrics Center, a five-year multi university center funded by DARPA and SRC under the STARnet Program.