PISA: A Non-Volatile Processing-In-Sensor Accelerator for Imaging Systems

Shaahin Angizi, *Senior Member, IEEE,* Sepehr Tabrizchi, *Student Member, IEEE,* David Z. Pan, *Fellow, IEEE,* and Arman Roohi, *Senior Member, IEEE*

Abstract—This work proposes a Processing-In-Sensor Accelerator, namely PISA, as a flexible, energy-efficient, and high-performance solution for real-time and smart image processing in AI devices. PISA intrinsically implements a coarse-grained convolution operation in Binarized-Weight Neural Networks (BWNNs) leveraging a novel compute-pixel with non-volatile weight storage at the sensor side. This remarkably reduces the power consumption of data conversion and transmission to an off-chip processor. The design is completed with a bit-wise near-sensor in-memory computing unit to process the remaining network layers. Once the object is detected, PISA switches to typical sensing mode to capture the image for a fine-grained convolution using only a near-sensor processing unit. Our circuit-to-application co-simulation results on a BWNN acceleration demonstrate minor accuracy degradation on various image datasets in coarse-grained evaluation compared to baseline BWNN models, while PISA achieves a frame rate of 1000 and efficiency of ∼1.74 TOp/s/W. Lastly, PISA substantially reduces data conversion and transmission energy by ∼84% compared to a baseline.

Index Terms—Magnetic memories, processing-in-sensor, accelerator.

1 Introduction

Internet of Thing (IoT) devices are projected to attain an $1100B market by 2025, with a web of interconnection projected to comprise approximately 75+ billion IoT devices, including wearable devices, smart cities, and smart industry [1], [2]. Intelligent IoT (IIoT) nodes consist of sensory systems, which enable massive data collection from the environment and people to process with on-/off-chip processors (10^{18} bytes/s or ops). In most cases, large portions of the captured sensory data are redundant and unstructured. Data conversion and transmission of large raw data to a back-end processor impose high energy consumption, high latency, a memory bottleneck, and low-speed feature extraction on the edge [1] as shown with the pixel-only architecture in Fig. 1(a). To overcome these issues, computing architectures will need to shift from a cloud-centric approach to a thing-centric (data-centric) approach, where the IoT node processes the sensed data. Nonetheless, the processing demands of artificial intelligence tasks such as Convolutional Neural Networks (CNNs) spanning hundreds of layers face serious challenges for their tractability in computational and storage resources. Effective techniques in both software and hardware domains have been developed to improve CNN efficiency by alleviating the "power and memory wall" bottleneck.

In algorithm-based approaches, the use of shallower but wider CNN models, quantizing parameters, and network binarization has been explored thoroughly [3], [4]. Recently, low bit-width weights and activations reduce computing complexity and model size. For instance, in [3], authors performed bit-wise convolution between the inputs and low bit-width weights by converting the conventional Multiplication-And-Accumulate (MAC) into the corresponding AND-bitcount operations. In an extreme quantization method, binary convolutional neural networks have achieved acceptable accuracy on both small [5] and large datasets [4] by relaxing the demands for some high precision calculations. Instead, they binarize weight and/or input feature map while processing the forward path, providing a promising solution to mitigate the aforementioned bottlenecks in storage and computational components [6].

From the hardware point of view, the underlying operations should be realized using efficient mechanisms. However, the conventional processing elements are developed based on the von-Neumann computing model with separate memory and processing blocks connecting via buses, which imposes serious challenges, such as long memory access latency, limited memory bandwidth, energy-hungry data transfer, and high leakage power consumption restricting the edge device's efficiency and working hours [2], [7]. Besides, at the upper level, this causes several significant issues such as communication bandwidth and security. Therefore, as a potential remedy, smart image sensors with instant image preprocessing have been extensively explored for object recognition applications [2], [8]–[11]. This paves the way for new sensor paradigms such as a Processing-Near-Sensor (PNS), in which digital outputs of a pixel are accelerated near the sensor leveraging an on-chip processor. Another solution to alleviate the above-mentioned challenges is a Processing-in-Memory (PIM) architecture, which is extensively studied in [6], [7], [12]. By inspiring the PNS and PIM techniques, two promising alternatives are the Processing-in-Sensor (PIS) that works on pre-Analog-to-Digital Converters (ADC) data [9], [13], [14] and a hybrid PIS-PNS platform [1] to improve vision sensor functionality

- *S. Angizi is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, USA. E-mail: shaahin.angizi@njit.edu.*
- *S. Tabrizchi and A. Roohi are with the School of Computing, University of Nebraska–Lincoln, Lincoln NE, USA. E-mail: aroohi@unl.edu.*
- *D. Z. Pan is with the Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX 78712, USA. E-mail: dpan@mail.utexas.edu.*

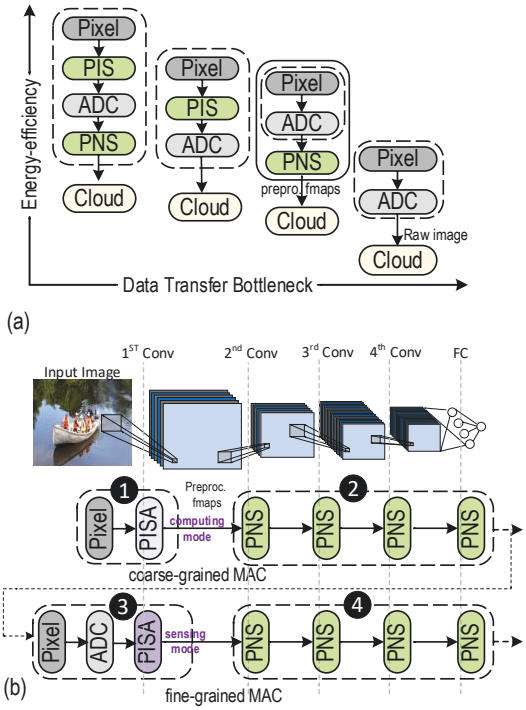


Fig. 1: (a) Various visual system architectures, (b) The proposed hybrid architecture.

and eliminate redundant data output, as shown in Fig. 1(a). However, the computational capabilities of these sensors have been limited to specific applications. This includes specific feature extraction applications less supporting MAC-based image classification [1], [8] to meet both resiliency and efficiency such as Haar-like image filtering [15], sharpening, blurring [10], and local binary pattern [16]. In general, the PIS units are designed to process the image before transmitting the raw data to the on-chip memory unit to be processed by a PNS (PIM) unit. Such data transfer in traditional designs (from CMOS image sensors to the memory) imposes a serious bottleneck and reduces the feature extraction speed remarkably. Therefore, having a coarse-grained computation with a PIS unit can (i) reduce the power consumption of data conversion from photo-currents to pixel values in the image processing tasks, (ii) increase the data processing speed, and (iii) alleviate the memory bottleneck issue [1], [2].

In this paper, we propose a new **P**rocessing-**I**n-**S**ensor **A**ccelerator (PISA) as an energy-efficient PIS paradigm co-integrating always-on sensing and processing capabilities working with a near-sensor PIM unit (PNS) that is categorized as a new hybrid design as shown in Fig. 1(b). The proposed design features a real-time programmable coarse-grained convolution to reduce the power consumption of data conversion from photo-currents to pixel values in the image processing task. Once the object is detected, PISA switches to a typical sensing mode to capture the image for fine-grained convolution using a PNS unit. The contributions of this paper are as follows:

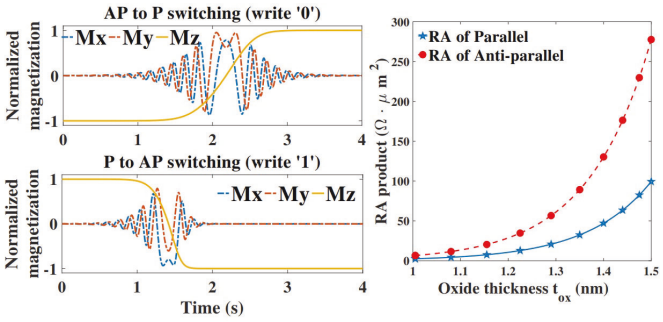
1) We develop a PIS architecture based on a set of innovative microarchitectural and circuit-level schemes optimized to process the 1^{st}-layer of Binarized-Weight Neural Networks (BWNN) with weights stored in non-volatile memory components that offers energy-efficiency and speed-up.
2) We complete the design with a bit-wise near-sensor PIM-enabled unit to process the remaining network layers. The presented bulk bit-wise computation operations are supported by most PIM architectures.
3) We present a solid bottom-up evaluation framework and a PIM assessment simulator to analyze the performance of the whole system.
4) We extensively assess PISA's performance and energy-efficiency co-integrated with the PNS unit compared with recent sensory platforms.

The remainder of the paper is designed as follows. Section 2 discusses the state-of-the-art near-sensor, in-sensor processing designs, and Magnetic Random Access Memory (MRAM). Section 3 delineates the proposed PISA architecture and the supported operations. Section 4 presents the near-sensor PIM unit. Section 5 gives the proposed bottom-up evaluation framework and simulation results. Section 6 discusses the future work and finally, Section 7 concludes this work.

2 Background & Motivation

2.1 Near-Sensor & In-Sensor Processing

Systematic integration of computing and sensor arrays has been widely studied to eliminate off-chip data transmission and reduce ADC bandwidth by combining CMOS image sensor and processors in one chip as known as PNS [2], [10], or even integrating pixels and computation unit so-called PIS [9], [13], [17], [18]. In [10], photocurrents are transformed into pulse-width modulation signals and a dedicated analog processor is designed to execute feature extraction reducing ADC power consumption. In [2], 3D-stacked column-parallel ADCs and Processing Elements (PE) are implemented to run spatiotemporal image processing. In [19], a CMOS image sensor with dual-mode delta-sigma ADCs is designed to process 1^{st}-conv, layer of BWNNs. RedEye [20] executes the convolution operation using charge-sharing tunable capacitors. Although this design shows energy reduction compared to a CPU/GPU by sacrificing accuracy, to achieve high accuracy computation, the required energy per frame increases dramatically by 100×. MACSEN [9] as a PIS platform processes the 1^{st}-convolutional layer of BWNNs with the correlated double sampling procedure achieving 1000fps speed in computation mode. However, it suffers from humongous area-overhead and power consumption mainly due to the SRAM-based PIS method. In [21], a pulse-domain algorithm uses fundamental building blocks, photodiode arrays, and an ADC to perform near-sensor image processing that reduces design complexity and enhances both cost and speed. Putting all together, there are three main bottlenecks in IoT imaging systems that this work explores and aims to solve: (1) The conversion and storage of pixel values consume most of the power (>96% [9]) in conventional image sensors; (2) the computation imposes a large area-overhead and power consumption in more recent PNS/PIS units and requires extra memory for intermediate data storage; and (3) the system is hardwired



(a) (b)

Fig. 2: (a) The normalized magnetization switching in x-, y- and z-axis. (b) The Resistance-Area product w.r.t the thickness of MTJ tunnel oxide (t_{ox}).

so the functionality is limited to simple pre-processing tasks such as 1^{st}-layer BWNN computation and cannot go beyond that.

2.2 MRAM as a High-Performance Non-Volatile Memory

With the great advancement of fabrication technology and commercialization of MRAM (e.g., IBM [22] and Everspin [23]), it is becoming a next-generation universal Non-Volatile Memory (NVM) technology, with potential applications in both last-level cache and main memory [24]. Particularly, recent current-induced Spin-Transfer Torque (STT) and Spin-Orbit Torque (SOT)-based MRAMs have greatly changed the state-of-the-art memory hierarchy due to their non-volatility, zero leakage power in un-accessed bit-cell [25], high integration density (2× more than SRAM), high speed (sub-nanosecond) [26], excellent endurance ($\sim 10^{15}$ cycles [27]), and compatibility with the CMOS fabrication process (back end of the line) [25]. A standard 1-transistor 1-resistor (1T1R) STT-MRAM bit-cell consists of an access transistor and a Magnetic Tunnel Junction (MTJ). A typical MTJ structure consists of two ferromagnetic layers with a tunnel barrier sandwiched between them [28]. One of the layers is a pinned magnetic layer, while the other one is a free magnetic layer. Due to the tunneling magnetoresistance (TMR) effect [28], the resistance of MTJ is high (/low) when the magnetization of two ferromagnetic layers is in anti-parallel (/parallel). The free layer magnetization could be manipulated by applying a current-induced STT [29]. For the STT-MRAM modeling in this work, the Non-Equilibrium Green's Function (NEGF) and Landau-Lifshitz-Gilbert(LLG) equation are used before the circuit-level simulation. The magnetization dynamics of MTJ's Free Layer-FL (m) can be modeled as [30]:

$$\frac{dm}{dt} = -|\gamma| m \times H_{\text{eff}} + \alpha\left(m \times \frac{dm}{dt}\right) + |\gamma|\beta(m \times m_{\text{P}} \times m) - |\gamma|\beta\epsilon'(m \times m_{\text{P}}) \quad (1)$$

$$\beta = |\frac{\hbar}{2\mu_0 e}|\frac{I_{\text{c}} P}{A_{\text{MTJ}} t_{\text{FL}} M_s} \quad (2)$$

where $\hbar$ is the reduced plank constant, γ is the gyromagnetic ratio, I_{c} is the charge current flowing through MTJ, t_{FL} is the thickness of the free layer, ϵ' is the second Spin transfer torque coefficient, and H_{eff} is the effective magnetic field. P is the effective polarization factor, A_{MTJ} is the cross-sectional area of MTJ, and m_{P} is the unit polarization direction. Figure 2(a) shows the normalized magnetization dynamics of the free layer in x, y, and z-axes when performing the STT-MRAM write scheme. Based on the simulation parameters listed in Table 1, the magnetization dynamic from the LLG equation can provide the relative angle θ between the magnetization of Pinned Layer-PL ($\hat{z}$) and Free Layer-FL (m). Therefore, the real-time conductance of MTJ (G_{MTJ}) is given by:

$$G_{\text{MTJ}} = \frac{G_{\text{P}} + G_{\text{AP}}}{2} + \frac{G_{\text{P}} - G_{\text{AP}}}{2}\cos\theta \quad (3)$$

where G_{P} and G_{AP} are the conductance of MTJ in parallel ($\theta = 0$) and anti-parallel ($\theta = 180$) configurations. Both G_{P} and G_{AP} are obtained from the atomistic level simulation framework based on Non-Equilibrium Green's Function (NEGF) [31], while the Resistance-Area Product with respect to the thickness of MTJ tunnel oxide is shown in Fig. 2(b).

TABLE 1: Simulations Parameters for MTJ.

Parameter	Value
Free layer dimension $(W \times L \times t)_{FL}$	$65 \times 65 \times 2\ nm^3$
Polarization factor, P	0.4
Gilbert Damping Factor, α	0.007
Saturation Magnetization, M_s	$850\ kA/m$
Oxide thickness, t_{ox}	$1.5\ nm$
Resistance-Area product, RA_p / TMR	$10.58\ \Omega \cdot \mu m^2$ / 171.2%
Supply voltage	$1\ V$
STT-MRAM cell area	$48F^2$
Access transistor width	$9F$
Cell aspect Ratio	1.34

3 PISA Architecture

Figure 1(b) shows an overview of the proposed hybrid architecture's data flow regarding a simple network structure with four convolutional layers and one Fully-Connected (FC) layer. Similarly, our proposed approach can be extended to accelerate much more complex CNN models. We first propose PISA as a flexible, energy-efficient, and high-performance solution for real-time and smart image processing in AI devices. PISA will integrate sensing and processing phases and can intrinsically implement a coarse-grained convolution operation (Fig 1(b) ❶) required in a wide variety of image processing tasks such as *classification* by processing the 1^{st}-layer in BWNNs. The design will be completed with a PNS unit to perform a low bit-width coarse-grained convolution on the remaining layers. Once the object is roughly detected at the end of step-❷, PISA will switch to typical sensing mode ❸ to capture the image for a fine-grained convolution using the PNS unit ❹.

Overview: At a high level, the PISA array consists of an $m \times n$ Compute Focal Plane (CFP), row and column controllers (Ctrl), command decoder, sensor timing ctrl, and sensor I/O operating in two modes, i.e., sensing and processing as shown in Fig. 3(a). The CFP is designed to co-integrate sensing and processing of the 1^{st}-layer of BWNN targeting a low-power and coarse-grained classification. To enable this, the conventional pixel unit is upgraded to a Compute Pixel (CP). The *Ri* (Row) signal is controlled by the Row Ctrl and shared across pixels located in the same row to enable access during the row-wise sensing mode. However, the *CR* (ComputeRow) is a unique controlling signal connected to entire CP units activated during processing

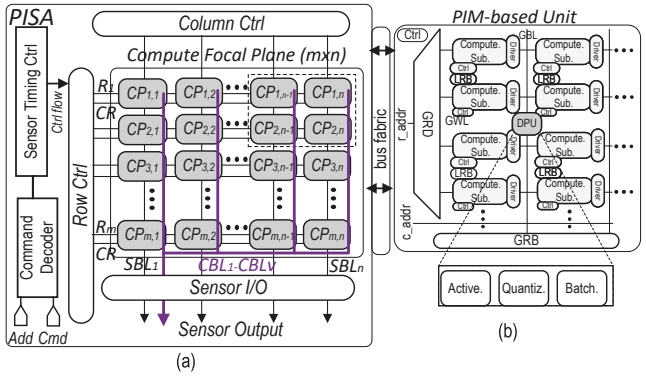


Fig. 3: (a) The overview of PISA architecture, (b) PNS architecture.

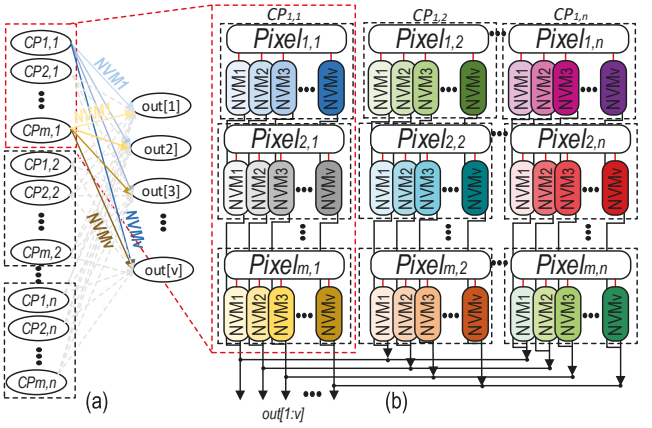


Fig. 4: (a) An example of a fully-connected network with v output, (b) PISA's mapping scheme for an m×n CFP.

mode. The core part of PISA is the CP unit consisting of a pixel connected to v Non-Volatile Memory (NVM) elements as shown in Fig. 4. A Sense Bit-line (SBL) is shared across the pixels on the same column connected to sensor I/O for sensing mode (Fig. 3(a)). Moreover, CPs share v Compute Bit-lines (CBL), each connected to a sense amplifier for processing as indicated by the purple line in Fig. 3(a). The 1^{st}-layer binarized weight corresponding to each pixel is pre-stored into NVMs and an efficient coarse-grained MAC operation is then accomplished in a voltage-controlled cross-bar fashion. Accordingly, the output of the first layer, as shown in Fig. 1(b), is transmitted to a PNS unit that enables the computation of the remaining BWNN layers. Fig. 4(a) depicts a sample neural network, wherein $CP_{1,1}$-$CP_{m,n}$ are linked to out1 via NVM_1's weight. Similarly, every pixel is connected to out2-out$v$. To maximize MAC computation throughput and fully leverage PISA's parallelism, we propose a hardware mapping scheme and connection configuration between CP elements and corresponding NVM add-ons shown in Fig. 4(b) to implement the target neural network. From a design perspective, the total number of CPs in PISA is given by m×n as the input spatial dimension. The total number of NVMs can be given by m×n×v where v represents the number of output layer's nodes as depicted in Fig. 4(a). PISA hardware is developed and fixed at design time based on the target application and the number of CPs has a direct impact on the accuracy at the end. The more CPs are considered to fully cover the input feature map translated to the higher number of weight parameters that can be stored in NVMs and this leads to a higher accuracy at the cost of higher energy consumption.

3.1 Compute-Pixel Element

The CP is composed of a pixel (three transistors and one Photodiode (PD)) as shown in Fig. 5, and v compute add-ons. The compute add-on consists of three transistors of which T4 and T5 work as deep triode region current sources and a 2:1 MUX controlled by an NVM element. We selected STT-MRAM as the NVM unit as depicted in Fig. 5(b) due to its high speed (sub-nanosecond), long-endurance (10 years), and less than fJ/bit memory write energy (close to SRAM) [30]. Thus, the binary weight data is stored as the magnetization direction in the MTJ's free layer, which could be programmed through the current-induced STT by the NVM write driver. A reference resistor is then used to realize a voltage divider circuit to read out the weight value from the memory. Fig. 5(a) illustrates a 2×1 CP array implementation. Please note that we keep the number of compute add-ons in each CP to a maximum of 64 according to our simulations to keep the pixel sensitivity high.

3.2 Sensing Mode

In sensing mode, by initially setting Rst='high', the photodiode (PD) connected to the T1 transistor (see Fig. 5(b)) turns into inverse polarization. In this way, turning on the access transistor T3 and k_1 switch (see Fig. 5(c)) at the Sensor I/O allows the C_1 capacitor to fully charge through SBL. By turning off T1, PD generates a photo-current with respect to the external light intensity which in turn leads to a voltage drop (V_{PD}) at the gate of T2. Once again by turning on the T3 and this time k_2 switch, C_2 is selected to record the voltage drop. Therefore, the voltage values before and after the image light exposure, i.e., V_1 and V_2, are sampled by the CP, and the difference between the two voltages is sensed with an amplifier. This value is proportional to the voltage drop on V_{PD}. In other words, the voltage at the cathode of PD can be read at the pixel output. It is worth pointing out that each ADC samples when the voltage drops, then it subtracts the pixel reset voltage and converts the output signal. Accordingly, the ADC can skip to the next row of the array. Please note that in sensing mode, the CR signal is grounded.

3.3 Integrated Sensing-Processing Mode

In this mode, as shown in a sample 2×1 CFP array in Fig. 5(a), the C_{PD} capacitor is initialized to the fully-charged state by setting Rst='high', similar to the sensing mode. During an evaluation cycle, by turning off T1, the row ctrl activates the CR signal, while the R_i signals are deactivated. This will activate the entire array for a single-cycle MAC operation. The core idea behind compute add-on shown in Fig. 5(b) is to leverage pixel's V_{PD} as a sampling voltage for T4(/T5) in v-NVM units to simultaneously generate(/pull) current from the CBLs. To implement multiplications between the pixel value identified by V_{PD} and the binary weight stored in NVM, a 2:1 MUX unit was devised in every CP taking the T4's source and T5's drain signals as

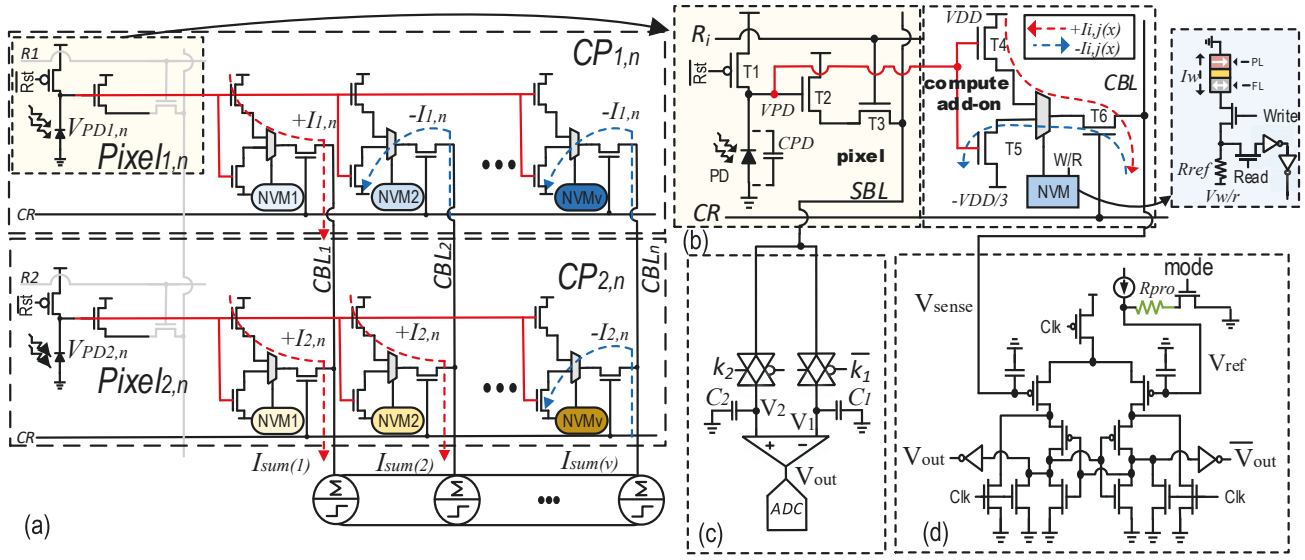


Fig. 5: (a) A 2×1 CFP array in processing mode, (b) Compute pixel, (c) CP's read and conversion circuit in sensing mode realizing correlated double sampling procedure, (d) Sense amplifier design in processing mode based on StrongArm latch.

inputs and NVM sensed data as the selector. Note that T4 and T5 are connected to V_{DD} and $\frac{-V_{DD}}{3}$, respectively. After exposure, the set of input sensor voltages V_{PD}= [$V_{PD_{1,1}}$, $V_{PD_{1,2}}$,..., $V_{PD_{m,n}}$] is applied to the gate of T4s and T5s generating current set I= [$I_{1,1(1)}$, $I_{1,1(2)}$,..., $I_{1,1(v)}$,..., $I_{m,n(1)}$, $I_{m,n(2)}$,..., $I_{m,n(v)}$] for the entire array. If the binary weight equals '1' (Wi=+1), T4 acts as a current source and generates a current with $I_{i,j(x)}$ magnitude on the shared CBL as shown by the red dashed line in Fig. 5(b). However, if the binary weight equals '0' (Wi=-1), the T5 transistor acts as a negative current source and pulls a current with the same magnitude as $I_{i,j(x)}$ in the opposite direction from the shared CBL as indicated by the blue dashed line in Fig. 5(b). Please note that T4's and T5's gate capacitors as well as parasitic capacitors will be fully charged to VDD through T1 in the pre-charge cycle, this will significantly keep the pixel sensitivity when the number of compute add-ons increases. This mechanism converts every input pixel value to a weighted current according to the NVM that is interpreted as the multiplication in BWNNs. Mathematically, let $G_{j,i}$ be the conductance of the synapse connecting i^{th} to the j^{th} node, the current through that synapse is $G_{j,i}V_i$ and the collection of the current through each CBL represents the MAC result ($I_{sum,j}$=$\sum_i G_{j,i}V_i$), according to Kirchhoff's law. This is readily calculated by measuring the voltage across a sensing resistor. For the activation function, we designed and tuned a sense circuit connected to each CBL based on StrongARM latch to realize an in-sensor $sign$ function [32], [33] as shown in Fig. 5(d). The sense amplifier requires two clock phases: pre-charge (Clk 'high') and sensing (Clk 'low'). During sensing, $I_{sum(x)}$ flows from every CBL to the ground and generates a sense voltage (V_{sense}) at the input of the sense amplifier. This voltage is compared with the reference voltage by applying a proportional current over a processing reference resistor (R_{pro}) activated by the mode signal. The binary activation is then transmitted through the bus fabrics to the PNS unit for storage.

4 Near-sensor Bit-wise PIM Unit (PNS)

Besides 1^{st}-layer, there are other convolutional and FC layers in BWNNs that can be accelerated close to the sensor without sending the activated feature maps to off-chip processors. The general memory organization of the PNS unit is shown in Fig. 3(b). The memory unit is divided into multiple banks consisting of computational sub-arrays. Every two sub-arrays share a Local Row Buffer (LRB) and the entire array shares a Digital Processing Unit (DPU) to pre-process the data by quantization and post-process outputs with linear batch normalization and activation. For the micro-architecture and circuit-level implementation of the near-sensor PIM unit, we adopt DRISA-1T1C [12] and ReDRAM [34] techniques. Fig. 6 gives an overview of the BWNN bit-wise acceleration steps. In the first step, the preprocessed data from PISA is mapped into the computational sub-arrays. In the second step, parallel computational sub-arrays perform bulk bit-wise operations between tensors and generate the output. Accordingly, the output is activated by DPU's activation unit and saved back into memory. From a computation perspective, every convolutional layer can be similarly implemented by exploiting *logic AND*, *bitcount*, and *bitshift* as rapid and parallelizable operations [3]. Assume I is a sequence of M-bit input integers (3-bit as an example in Fig. 6) located in input fmap covered by the sliding kernel of W, such that $I_i \in I$ is an M-bit vector representing a fixed-point integer. Now, we index the bits of each I_i element from LSB to MSB with $m = [0, M-1]$, such that $m = 0$ and $m = M-1$ are corresponding to LSB and MSB, respectively. Accordingly, we represent a second sequence denoted as $C_m(I)$ including the combination of m^{th} bit of all I_i elements (shown by colored elliptic). For instance, $C_0(I)$ vector consists of LSBs of all I_i elements "0110". Considering W as a sequence of N-bit weight integers (3-bit, herein) located in a sliding kernel with the index of $n = [0, N-1]$. The second sequence can be similarly generated as $C_n(W)$. Now, by considering the set of all m^{th} value sequences, the I can be represented like $I = \sum_{m=0}^{M-1} 2^m c_m(I)$.

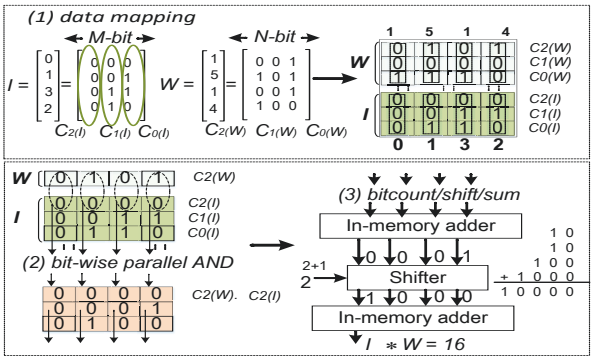


Fig. 6: Acceleration steps of the PNS convolver.

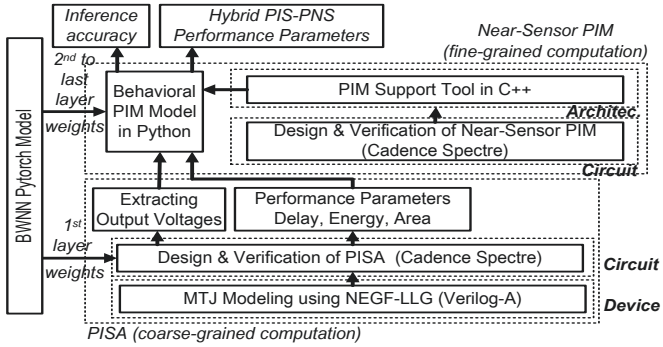


Fig. 7: Evaluation framework.

Likewise, W can be represented like $W = \sum_{n=0}^{N-1} 2^n c_n(W)$. In this way, the convolution between I and W can be defined as $\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} 2^{m+n} bitcount(and(C_n(W), C_m(I)))$. As shown in the data mapping step of Fig. 6, $C_2(W)$-$C_0(W)$ are consequently mapped to the designated sub-array. $C_2(I) - C_0(I)$ are mapped in the following memory rows in the same way. Now, computational sub-array can perform bit-wise parallel AND operation of $C_n(W)$ and $C_m(I)$ as depicted in Fig. 6. The results stored within the sub-array will be accordingly processed using DPU's bit-counter. Bit-counter readily counts the number of "1"s in each resultant vector and passes it to the Shifter unit. As depicted in Fig. 6, "0001", as a result of Bit-Counter is left-shifted by 3-bit ($\times 2^{2+1}$) to "1000". Eventually, the PIM adds the shifter unit's outputs to produce output fmaps for every layer. Note that the PNS unit supports multi-bit convolution so the various configurations of weight:input can be achieved at the edge. Due to the lack of space, we refer the readership to the above-mentioned papers for details on in-memory logic implementation techniques. The total number and size of the compute sub-arrays are predefined to fit the preprocessed data from PISA and to accelerate the bit-wise operations. Considering an N-input feature map layer that needs to be mapped to PNS with N_s activated sub-arrays of the size of $x \times y$, each sub-array can process n input feature maps ($n \leq f | n \in N, f = min(x, y)$). In this way, the number of sub-arrays for processing an N input feature map layer can be formulated as $N_s = \left\lceil \frac{N}{f} \right\rceil$.

5 Performance Evaluation

5.1 Framework & Methodology

To assess the performance of the proposed design, we developed a simulation framework from scratch consisting of two main components as shown in Fig. 7. First, for coarse-grained computation, at the circuit level, we fully implemented PISA with peripheral circuity with TSMC 65nm-GP in Cadence to achieve the performance parameters. For the NVM elements, we jointly use the NEGF and LLG equations to model MTJ [30]. A Verilog-A model of the NVM element is then developed to co-simulate with interface CMOS circuits in Cadence Spectre and SPICE. PISA requires binarizing the 1^{st}-layer weights as discussed while the rest of the layers processed with the PNS unit have various bit-length. We trained a PyTorch BWNN model inspired by [35], [36] extracting the 1^{st}-layer weights. PISA's NVM elements are then programmed at the circuit-level by the binary weights. After 1^{st}-layer computation, the results are recorded and fed into a behavioral-level PIM simulator to simulate the PNS platform. Second, for fine-grained computation, at the circuit level, we fully implemented DRISA-1T1C [12] and ReDRAM [34] with TSMC 65nm-GP in Cadence to achieve the performance parameters. A custom architecture-level PIM support tool is developed based on our previous simulator (PIMA-SIM [37]) to model the timing, energy, and area based on the circuit-level data. This tool offers the same flexibility in memory configuration regarding bank/mat/subarray organization and peripheral circuitry design as Cacti [38] while supporting PIM-level configurations. Based on the circuit level results, it can alter the configuration files (.cfg) with different array organizations and add-ons such as DPU and report performance for PIM operations. We then configure the PNS unit with 1024 rows and 256 columns, 4×4 mats per bank organized in an H-tree routing manner, and 16×16 banks in each memory group. The behavioral PIM model developed in Python then takes coarse-grained computation voltage results, 2^{nd}-to-last layer trained weights, and the PIM architecture-level data and processes the BWNN. It calculates the latency and energy that the whole system spends executing the network.

5.2 Results

Functionality: Fig. 8 shows the post-layout transient simulation waveforms of a 4×4 PISA array with eight NVM units (v=8) storing binary weights with V_{Clk}, V_{Rst}, V_{PD}, I_{CBL}, and V_{Out} signals. PISA executes global shutter in processing mode and conducts all computations in parallel. As shown, periodically, by precharging V_{PD} to VDD, the computation takes place at every falling edge of the clock, i.e., ∼100μs. In this way, I_{CBL} carries the summation current corresponding to V_{PD}s. As can be seen, when I_{CBL} is positive (e.g., the case of 32μA and 39μA) meaning the MAC result is larger than zero and the output $sign$ function results in "1" and vice-versa.

Robustness: PISA operates in the mixed-signal domain, which is vulnerable to non-ideal factors, such as variations, noises, and leakage. We simulated the PISA's circuit-level variations and noises with equivalent post-layout parasitic at 300K with 10000 Monte-Carlo runs. This includes a variation in the width/length of transistors and CBL capacitance. The impact of thermal noises was modeled as the additive Gaussian noise on the dynamic capacitance along with 1/f noise of CMOS transistors from the source-follower in pixels. Our study shows that the percentage

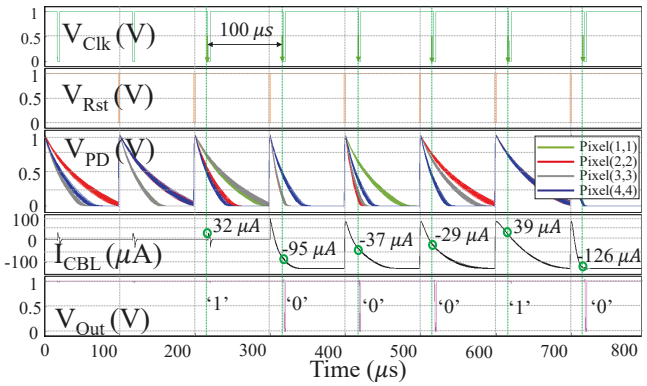


Fig. 8: Post-layout transient simulation result for a sample 4×4 PISA array.

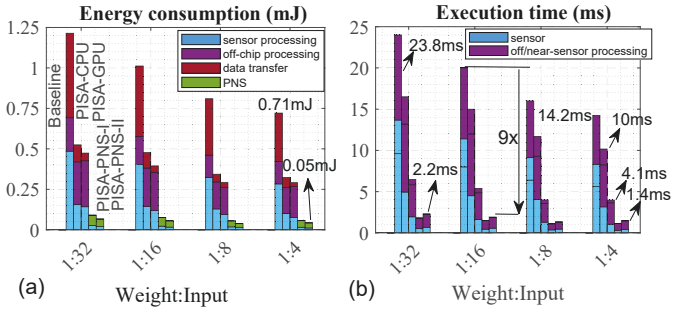


Fig. 9: (a) Energy consumption, and (b) Execution time of under-test PISA-based platforms in various PNS configurations compared with the baseline. From left to right in each bar group: Baseline, PISA-CPU, PISA-GPU, PISA-PNS-I, and PISA-PNS-II.

of failure upon a considerable variation/noise (10%) across 10000 iterations is 0% as plotted V_{PD} in Fig. 8. It is worth mentioning that T4's and T5's gate capacitors as well as parasitic capacitors will be fully charged to VDD through T1 in the pre-charge cycle, this will significantly keep the pixel sensitivity when the number of compute add-ons increases. For variations above 10%, a noise-aware training technique [39] is used injecting multiplicative noise onto the weights in the training to increase BWNN robustness. For the NVM element, we added a $\sigma = 2\%$ variation to the Resistance-Area product, and a $\sigma = 5\%$ process variation (typical MTJ conductance variation [30]) on the TMR and verified a sense margin of 70mV between parallel and anti-parallel cases. In the noise-aware training technique, input features are augmented with a noise estimate before training. Originally, such a technique was demonstrated in [40] for noise-resistant speech recognition. The basic CNN training did not specifically utilize the noise information of each utterance. Thus, noise awareness is enabled by feeding the CNN with noisy speech samples augmented with noise estimates. Using additional noise information online, CNN can predict more accurately.

Energy & Performance: We analyze the PISA's utility in processing the 1^{st}-convolutional layer for continuous mobile vision in three scenarios, i.e., assisting mobile CPU (PISA-CPU), assisting mobile GPU (PISA-GPU), and PISA-PNS, and comparing it with a baseline sensor-CPU platform. For this goal, a BWNN model with 6 binary-weight convolutional layers and 2 FC layers to process the SVHN dataset is adopted. The energy consumption and latency results of the under-test platforms are then reported for four various weight/input configurations in PNS (W:I= 1:32, 1:16, 1:8, 1:4) in Fig. 9. The rationale behind this experiment is to show how weight/input configuration's precision will impact the whole deep neural network acceleration performance in various platforms, especially in the PISA-PNS designs. The under-test platforms in each experiment from left to right include the baseline design consisting of a conventional 128×128 image sensor and an Intel(R) Core i7-6700 at 3.4GHz CPU with 16GB RAM where the CPU plays the main role in processing all layers after receiving the raw data from the sensor's ADC. The second platform consists of the same CPU connected to a 128×128 PISA array, where PISA processes 1^{st} convolutional layer, and the remaining layers are processed by the CPU. The third design replaces the previous CPU with an NVIDIA GTX 1080Ti Pascal GPU with 3584 CUDA cores running at 1.5GHz (11 TFLOPs peak performance). For CPU/GPU platforms, we use the open-source algorithm DoReFa-Net [3] where the rest of the layers can be accelerated using the bit-wise convolution of fixed-point integers. The last two designs (fourth and fifth columns in each configuration in Fig. 9) take advantage of PISA and its PNS-support to process the whole BWNN. When the 1^{st} convolutional layer is processed by PISA, we adopted two alternative PIM techniques, i.e., DRISA [12] and ReDRAM [34] in PNS unit to compute the 2^{nd}-6^{th} convolutional and 2 FC layers near the sensor. Note that, any bit-wise PIM techniques could be adopted. We report the breakdown of energy consumption into sensor processing, off-chip processing, data transfer, and PNS. We find that PISA performs favorably against conventional CMOS image sensors. To have a fair comparison with the PISA-PNS platforms, we also considered the fact that the 2^{nd}-layer input matrices on the processor side need to be transposed and processed cycle-by-cycle before any computation. Therefore, the energy/latency cost of such an extra operation was taken into account. First, PISA substantially reduces the data transmission energy by ∼84%. paired with the CPU and GPU. The PISA-CPU platform saves 58% energy on average compared with the baseline as shown in Fig. 9(a). While the PISA-GPU does not show a remarkable energy-saving over PISA-CPU but is still ∼62.5% more energy-efficient than the baseline. Besides the reduction in data transfer, the other reason behind such a striking energy saving is eliminating energy-hungry ADC units in PISA's processing mode. Second, we observe that PISA-PNSs (PNS-I and PNS-II denote the adopted DRISA-1T1C and ReDRAM techniques, respectively) reduce the energy consumption of edge devices dramatically. The PISA-PNS-II requires ∼50-170μJ energy depending on the PNS configuration to process the whole BWNN on the edge, which is a safe choice for power-constrained IoT sensor devices. The PISA-PNS designs almost eliminate the data transmission energy. Fig. 9(b) illustrates the execution time corresponding to various W:I configurations. This figure reports the sensor and off-sensor/near-sensor processing time as the time taken by the computing cores (i.e., CPU in the baseline, PISA plus CPU in the PISA-CPU platform, PISA plus GPU in the PISA-GPU platform, and PISA plus PNS in PISA-PNS). We observe that

TABLE 2: Performance comparison of various PIS units.

Designs	Technology (nm)	Purpose	Comput. Scheme	Memory	NV*	Pixel Size (μm^2)	Array Size	Frame Rate ($frame/s$)	Power (mW)	Efficiency ($TOp/s/W$)
[41]	180	2D optic flow est.	raw-wise	Yes	No	28.8×28.8	64×64	30	0.029	0.0041
[10]	180	edge*/blur/sharpen/ 1st layer CNN	raw-wise	No	No	7.6×7.6	128×128	480	sensing: 0.077 processing: 0.091	0.777
[2]	60/90	STP†	raw-wise	Yes	No	3.5×3.5	1296×976	1000	sensing: 230 processing:363	0.386
[9]	180	1st layer BNN	entire-array	No	No	110×110	32×32	1000	0.0121	1.32
[8]	180	edge*/TMF‡	raw-wise	Yes	No	32.6×32.6	256×256	100,000	1230	0.535
PISA	65	1st layer BNN	entire-array	Yes	Yes	55×55	128×128	1000	sensing: 0.025 processing: 0.0090	1.745

* Edge extraction. † Spatial Temporal Processing. ‡ Thresholding Median Filter.

PISA-PNS designs achieve ∼9-11× speed-up in processing input frames compared with the baseline.

Resource Utilization: To explore the impact of PISA in reducing memory bottleneck in executing the 1^{st}-layer of BWNN, we measured the time fraction at which on-/off-chip data transfer limits the performance. This evaluation was accomplished through experimentally extracted results of each platform with the number of memory access. We observe the PISA spends less than 5% of time for data conversion and memory access, whereas the baseline design spends over 76% of its time waiting to load data from memory. The PISA-PNS platforms obtain the highest ratio utilizing up to 95% computation resources.

Comparison: Table 2 compares the structural and performance parameters of selective PIS implementations in the literature. As different implementations are developed for specific domains, for an impartial comparison, we estimated and normalized the power consumption when all PIS units execute the similar task of processing the 1^{st}-layer of CNN. In our evaluation, we followed the method in state-of-the-art papers and intentionally didn't scale down the process nodes to 65nm as we didn't have access to all technical configurations of the counterpart implementations. It is noteworthy that the reported numbers for Top/s/W and frame-rate are technology-dependent. The PISA achieves the frame rate of 1000 and the efficiency of ∼1.745 TOp/s/W as an efficient implementation. This comes from the massively-parallel CFP and eliminating ADC for coarse-grained detection. However, the implementation in [8] achieves the highest frame-rate and the implementation in [2] imposes the least pixel size enabling in-sensor computing. As for the area, our simulation results reported in Table 2 show a PISA's compute-pixel occupies ∼55x55 μm^2 in 65nm. As we do not have access to the other layouts' configurations, it is very hard to have a fair comparison between area overheads. However, we believe a ballpark assessment can be made by comparing the number of minimum-size transistors in previous SRAM-based implementations and PISA's lower-overhead compute add-on. We reimplemented MACSen [9] at circuit-level as the only BWNN accelerator developed with the same purpose. Our evaluation showed that with the same PNS unit based on DRISA [12], PISA consumes ∼40% less power consumption. Putting everything together, PISA offers 1) a low-overhead, dual-mode, and reconfigurable design to keep the sensing performance and realize a processing mode to remarkably reduce the power consumption of data conversion and transmission; 2) single-cycle in-sensor processing mechanism to improve image processing speed; 3) highly parallel in-sensor processing design to achieve ultra-high-throughput; 4) exploiting NVM which reduces standby power consumption during idle time and offers instant wake-up time, and resilience to power failure to achieve high performance.

As discussed earlier, MRAM is not the only type of NVM that can be adopted in PISA. The efficiency and speed-up of the PISA compared to the recently-proposed near-sensor and in-sensor designs mainly come from the innovative compute focal plane architecture consisting of compute pixels that can be readily implemented with various types of NVMs such as ReRAM, PCM, FeRAM, etc. To further explore the performance of PISA using other types of NVM, we simulated the design with the $TiN/Ti/HfO_2/TiN$ RRAM device integrated with CMOS n-channel Field-Effect Transistor (nFET) in [11] to realize a 1T1R unit cell as the primary NVM element. For this experiment, we considered Muli-Level Cells (MLC) with 4 resistance levels. We used the same evaluation framework in Fig. 7 only replacing the device measurements. The only hardware modification is the read/write voltages in the voltage driver that changes the power budget of the system. We observe that the processing power consumption will increase by 28% compared to the STT-MRAM. Besides, we get a lower overall efficiency of ∼1.41 TOP/s/W as opposed to the 1.74 TOP/s/W reported for PISA's initial configuration.

Table 3 mainly discusses the worst-, average-, and best-case power consumption with respect to four under-test datasets. As can be seen, depending on the input dataset, the processing power (mW) of PISA changes. In Table 2, we only reported the average power consumption across the whole test images in the SVHN dataset.

TABLE 3: PISA's processing power consumption (mW) for various datasets.

Configuration	MNIST	SVHN	CIFAR-10	CIFAR-100
worst	0.0104	0.0114	0.0210	0.0231
average	0.0072	0.0090	0.0175	0.0184
best	0.0063	0.0071	0.0139	0.0181

Accuracy: In the original BWNN topology, all the layers, except the first and last, are implemented with binarized weights [33], [42], [43]. Since, in image classification tasks, the number of input channels is relatively smaller than the number of internal layers' channels, the required parameters and computations are small. Thus, converting the input layer will not be a significant issue [33]. We conduct experiments on several datasets, including MNIST, SVHN, CIFAR-10, and CIFAR-100. Fig. 10 shows the validation error versus the number of epochs for the four datasets in a worst-case scenario, i.e., with a 1:4 configuration for 2^{nd} to the last layer. The comparison of classification accuracy is summarized in Table 4. We find that the PISA-MRAM shows an acceptable

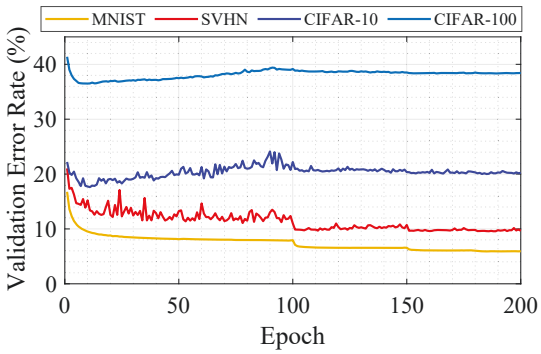


Fig. 10: Validation error curves of four different datasets using the proposed BWNN configuration.

accuracy while providing significant energy-delay-product reduction as discussed earlier. The accuracy drop of PISA is mainly because of two reasons: (1) PISA processes the 1^{st} convolutional layer with binarized weights stored in the magnetic NVM; (2) PISA implements the Sign function as an activation function, which easily saturates. Thus it degrades the accuracy compared to ReLU. To enhance the accuracy, possible solutions are outlined:
(a) In this work, we readily mapped the full precision trained weights to binary_weights $\in \{-1, +1\}$ by only taking the sign function, expressed in Equation 4:

$$\text{Forward:} \quad b = sign(y) = \begin{cases} +1, & if\, y \geq 0 \\ -1, & otherwise \end{cases} \tag{4}$$

In this case, the sign function is non-convex, which results in the gradient becoming zero. Thus, a standard backpropagation approach will be impractical due to the vanishing gradient problem. One practical technique to mitigate this issue is Normalization. A good normalization is a basis for neural network convergence; thus, the weights should be normalized first, then binarized. The accuracy results associated with this experiment indicated by PISA-m1 are listed in Table 4 for four under-test data-sets, which show slight improvement over the baseline PISA implementation.
(b) Another approach could be adding more layers to the neural network, which allows it to learn more abstract features and transform the input space more. Although it may increase accuracy, it increases power consumption and reduces throughput. The accuracy results associated with adding 3 additional layers indicated by PISA-m2 are listed in Table 4 for four under-test data-sets.
(c) As a practical hardware method to enhance the accuracy, we reimplemented the PISA's NVM blocks with the TiN/Ti/HfO_2/TiN RRAM device integrated with CMOS n-channel Field-Effect Transistor (nFET) in [11] that offers 4 different levels of resistance. As can be seen in Table 4, the PISA-m3 with 2-bit weight resolution can increase the accuracy slightly over four different datasets. For example, for CIFAR-10, we observe ∼0.45% improvement compared to the MRAM-based implementation. It is worth pointing out that the PISA architecture enabling edge AI is crucial in applications that necessitate real-time responses, such as predictive maintenance, environmental monitoring, smart home systems, and agricultural AI applications (e.g., crop health monitoring systems) where the lower accuracy can be offset. It's also pivotal in environments with connectivity constraints, where privacy concerns demand local data processing, and where power and cost efficiency are vital.

TABLE 4: PISA's Accuracy (%) on various datasets.

Configuration	MNIST	SVHN	CIFAR-10	CIFAR-100
[33]	96.0	97.47	89.85	70.9
[42]	98.25	97.00	86.98	69.5
[43]	98.4	94.9	80.1	68.8
[35]	–	96.9	88.61	71.5
PISA-MRAM	95.12	90.35	79.80	61.6
PISA-m1	95.89	91.20	82.85	62.10
PISA-m2	95.70	91.34	81.76	64.61
PISA-m3	95.72	90.55	80.25	62.5

6 Discussion and Future Work

Although almost all the state-of-the-art image sensor designs utilize effective methods to reduce dynamic energy consumption, including clock gating and low-voltage operation, an increasing number of modern intelligent sensors and more application scenarios, making the standby power dissipation of such systems a critical issue, which can limit the wider sensors' applications. The emergence of energy harvesting systems as a promising approach for battery-less IoTs suffers from intermittent behavior, leading to data and environmental inconsistencies. For example, captured data by sensors become unstable if they are held for a long time without intermittent resilient architectures and/or harvestable sources. Moreover, since concurrency with sensors is relatively interrupt-driven, intermittency makes this concurrency control much more complex. To solve the data consistency, PISA utilizes NVM elements, which reduce standby power consumption during idle time, instant wake-up time, and resilience to power failure, leading to high throughput and high performance at the cost of minor accuracy degradation. We plan to extend our future work to investigate image sensors' challenges in the presence of power failure for energy-harvested systems, and more thoroughly discuss PISA's power failure resiliency.

7 Conclusion

In summary, this work proposed an efficient processing-in-sensor accelerator, namely PISA, for real-time edge-AI devices. PISA intrinsically performs a coarse-grained convolution operation on the 1^{st}-layer of binarized-weight neural networks leveraging a novel compute-pixel with non-volatile weight storage. The design was then completed by a near-sensor processing-in-memory unit to perform a fine-grained convolution operation over the remaining layers. Our results demonstrate acceptable accuracy on various datasets, while PISA achieves the frame rate of 1000 and the efficiency of ∼1.74 TOp/s/W.

Acknowledgment

This work is supported in part by the National Science Foundation under Grant No. 2228028, 2216772, and 2216773.

References

[1] T.-H. Hsu, Y.-C. Chiu, W.-C. Wei, Y.-C. Lo, C.-C. Lo, R.-S. Liu, K.-T. Tang, M.-F. Chang, and C.-C. Hsieh, "Ai edge devices using computing-in-memory and processing-in-sensor: from system to device," in *2019 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2019, pp. 22–5.

[2] T. Yamazaki, H. Katayama, S. Uehara, A. Nose, M. Kobayashi, S. Shida, M. Odahara, K. Takamiya, Y. Hisamatsu, S. Matsumoto, L. Miyashita, Y. Watanabe, T. Izawa, Y. Muramatsu, and M. Ishikawa, "4.9 a 1ms high-speed vision chip with 3d-stacked 140gops column-parallel pes for spatio-temporal image processing," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE, 2017, pp. 82–83.

[3] 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," *arXiv preprint arXiv:1606.06160*, 2016.

[4] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European conference on computer vision*. Springer, 2016, pp. 525–542.

[5] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or-1," *arXiv preprint arXiv:1602.02830*, 2016.

[6] S. Angizi, Z. He, A. S. Rakin, and D. Fan, "Cmp-pim: an energy-efficient comparator-based processing-in-memory neural network accelerator," in *Proceedings of the 55th Annual Design Automation Conference*. ACM, 2018, p. 105.

[7] L. Song, X. Qian, H. Li, and Y. Chen, "Pipelayer: A pipelined reram-based accelerator for deep learning," in *High Performance Computer Architecture (HPCA), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 541–552.

[8] S. J. Carey, A. Lopich, D. R. Barr, B. Wang, and P. Dudek, "A 100,000 fps vision sensor with embedded 535gops/w 256× 256 simd processor array," in *2013 Symposium on VLSI Circuits*. IEEE, 2013, pp. C182–C183.

[9] H. Xu, Z. Li, N. Lin, Q. Wei, F. Qiao, X. Yin, and H. Yang, "Macsen: A processing-in-sensor architecture integrating mac operations into image sensor for ultra-low-power bnn-based intelligent visual perception," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 2, pp. 627–631, 2020.

[10] T.-H. Hsu, Y.-R. Chen, R.-S. Liu, C.-C. Lo, K.-T. Tang, M.-F. Chang, and C.-C. Hsieh, "A 0.5-v real-time computational cmos image sensor with programmable kernel for feature extraction," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 5, pp. 1588–1596, 2020.

[11] M. Abedin, A. Roohi, M. Liehr, N. Cady, and S. Angizi, "Mr-pipa: An integrated multilevel rram (hfo x)-based processing-in-pixel accelerator," *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, vol. 8, no. 2, pp. 59–67, 2022.

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

[13] H. Xu, N. Lin, L. Luo, Q. Wei, R. Wang, C. Zhuo, X. Yin, F. Qiao, and H. Yang, "Senputing: An ultra-low-power always-on vision perception chip featuring the deep fusion of sensing and computing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2021.

[14] S. Tabrizchi, S. Angizi, and A. Roohi, "Tizbin: A low-power image sensor with event and object detection using efficient processing-in-pixel schemes," in *2022 IEEE 40th International Conference on Computer Design (ICCD)*. IEEE, 2022, pp. 770–777.

[15] K. Bong, S. Choi, C. Kim, S. Kang, Y. Kim, and H.-J. Yoo, "14.6 a 0.62 mw ultra-low-power convolutional-neural-network face-recognition processor and a cis integrated with always-on haar-like face detector," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE, 2017, pp. 248–249.

[16] X. Zhong, Q. Yu, A. Bermak, C.-Y. Tsui, and M.-K. Law, "A 2pj/pixel/direction mimo processing based cmos image sensor for omnidirectional local binary pattern extraction and edge detection," in *2018 IEEE Symposium on VLSI Circuits*. IEEE, 2018, pp. 247–248.

[17] H. Xu, Z. Liu, Z. Li, E. Ren, M. Nazhamati, F. Qiao, L. Luo, Q. Wei, X. Liu, and H. Yang, "A 4.57 μw@ 120fps vision system of sensing with computing for bnn-based perception applications," in *2021 IEEE Asian Solid-State Circuits Conference (A-SSCC)*. IEEE, 2021, pp. 1–3.

[18] S. Tabrizchi, A. Nezhadi, S. Angizi, and A. Roohi, "Appcip: Energy-efficient approximate convolution-in-pixel scheme for neural network acceleration," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2023.

[19] W.-T. Kim, H. Lee, J.-G. Kim, and B.-G. Lee, "An on-chip binary-weight convolution cmos image sensor for neural networks," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 8, pp. 7567–7576, 2020.

[20] R. LiKamWa, Y. Hou, J. Gao, M. Polansky, and L. Zhong, "Redeye: analog convnet image sensor architecture for continuous mobile vision," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 255–266, 2016.

[21] F. Taherian and D. Asemani, "Design and implementation of digital image processing techniques in pulse-domain," in *2010 IEEE Asia Pacific Conference on Circuits and Systems*. IEEE, 2010, pp. 895–898.

[22] W. J. Gallagher and S. S. Parkin, "Development of the magnetic tunnel junction mram at ibm: From first junctions to a 16-mb mram demonstrator chip," *IBM Journal of Research and Development*, vol. 50, no. 1, pp. 5–23, 2006.

[23] "Everspin announces sampling of the world's first 1-gigabit mram product. 2016." [Online]. Available: https://www.everspin.com

[24] W. Kang, Y. Ran, Y. Zhang, W. Lv, and W. Zhao, "Modeling and exploration of the voltage-controlled magnetic anisotropy effect for the next-generation low-power and high-speed mram applications," *IEEE Transactions on Nanotechnology*, vol. 16, no. 3, pp. 387–395, 2017.

[25] S. Fukami, T. Anekawa, C. Zhang, and H. Ohno, "A spin-orbit torque switching scheme with collinear magnetic easy axis and current configuration," *Nature nanotechnology*, 2016.

[26] G. E. Rowlands, T. Rahman, J. A. Katine, J. Langer, A. Lyle, H. Zhao, J. G. Alzate, A. A. Kovalev, Y. Tserkovnyak, Z. M. Zeng, H. W. Jiang, K. Galatsis, Y. M. Huai, P. K. Amiri, K. L. Wang, I. N. Krivorotov, and J.-P. Wang, "Deep subnanosecond spin torque switching in magnetic tunnel junctions with combined in-plane and perpendicular polarizers," *Applied Physics Letters*, vol. 98, no. 10, p. 102509, 2011.

[27] J. J. Kan, C. Park, C. Ching, J. Ahn, L. Xue, R. Wang, A. Kontos, S. Liang, M. Bangar, H. Chen, S. Hassan, S. Kim, M. Pakala, and S. H. Kang, "Systematic validation of 2x nm diameter perpendicular mtj arrays and mgo barrier for sub-10 nm embedded stt-mram with practically unlimited endurance," in *Electron Devices Meeting (IEDM), 2016 IEEE International*. IEEE, 2016, pp. 27–4.

[28] T. Kawahara, "Challenges toward gigabit-scale spin-transfer torque random access memory and beyond for normally off, green information technology infrastructure," *Journal of Applied Physics*, vol. 109, no. 7, p. 07D325, 2011.

[29] M. D. Stiles and A. Zangwill, "Anatomy of spin-transfer torque," *Physical Review B*, vol. 66, no. 1, p. 014407, 2002.

[30] X. Fong, S. K. Gupta, N. N. Mojumder, S. H. Choday, C. Augustine, and K. Roy, "Knack: A hybrid spin-charge mixed-mode simulator for evaluating different genres of spin-transfer torque mram bit-cells," in *2011 International Conference on Simulation of Semiconductor Processes and Devices*, 2011, pp. 51–54.

[31] G. Panagopoulos, C. Augustine, and K. Roy, "A framework for simulating hybrid mtj/cmos circuits: Atoms to system approach," in *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2012, pp. 1443–1446.

[32] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," *arXiv preprint arXiv:1511.00363*, 2015.

[33] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," *Advances in neural information processing systems*, vol. 29, 2016.

[34] S. Angizi and D. Fan, "Redram: A reconfigurable processing-in-dram platform for accelerating bulk bit-wise operations," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2019, pp. 1–8.

[35] P. Guo, H. Ma, R. Chen, P. Li, S. Xie, and D. Wang, "Fbna: A fully binarized neural network accelerator," in *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2018, pp. 51–513.

[36] W. Tang, G. Hua, and L. Wang, "How to train a compact binary neural network with high accuracy?" in *Thirty-First AAAI conference on artificial intelligence*, 2017.

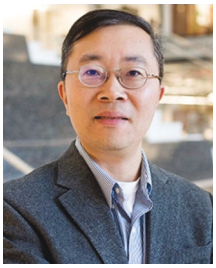
[37] S. Angizi, N. Khoshavi, A. Marshall, P. Dowben, and D. Fan, "Mefram: A new non-volatile cache memory based on magneto-electric fet," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 27, no. 2, pp. 1–18, 2021.

[38] S. Thoziyoor, N. Muralimanohar, J. H. Ahn, and N. P. Jouppi, "Cacti 5.1," Technical Report HPL-2008-20, HP Labs, Tech. Rep., 2008.

[39] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, "A regression approach to speech enhancement based on deep neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 1, pp. 7–19, 2014.
[40] M. L. Seltzer, D. Yu, and Y. Wang, "An investigation of deep neural networks for noise robust speech recognition," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 7398–7402.
[41] S. Park, J. Cho, K. Lee, and E. Yoon, "7.2 243.3 pj/pixel bio-inspired time-stamp-based 2d optic flow sensor for artificial compound eyes," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. IEEE, 2014, pp. 126–127.
[42] M. Ghasemzadeh, M. Samragh, and F. Koushanfar, "Rebnet: Residual binarized neural network," in *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2018, pp. 57–64.
[43] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "Finn: A framework for fast, scalable binarized neural network inference," in *Proceedings of the 2017 ACM/SIGDA international symposium on field-programmable gate arrays*, 2017, pp. 65–74.

Shaahin Angizi (SM'22) is currently an Assistant Professor in the Department of Electrical and Computer Engineering, New Jersey Institute of Technology (NJIT), Newark, NJ, USA, and the director of the Advanced Circuit-to-Architecture Design Laboratory. He completed his doctoral studies in Electrical Engineering at the School of Electrical, Computer and Energy Engineering, Arizona State University (ASU), Tempe, AZ in 2021. His primary research interests include ultra-low-power in-memory computing based on volatile & non-volatile memories, in-sensor computing for IoT, brain-inspired (neuromorphic) computing, and accelerator design for deep neural networks and bioinformatics. He has authored and co-authored +90 research papers in top-ranked journals and EDA conferences. He is the recipient of the Best Ph.D. Research Award (1st-place) of Ph.D. Forum at IEEE/ACM DAC in 2018, two Best Paper Awards of IEEE ISVLSI in 2017 and 2018, and Best Paper Award of ACM GLSVLSI in 2019.

Sepehr Tabrizchi (S'21) is currently a Ph.D. student at the School of Computing, University of Nebraska-Lincoln, USA. He received a masters' degree in computer systems architecture from Azad University, Science and Research Branch, Tehran, Iran, in 2018 and a Bachelor of Engineering degree in computer engineering from the Islamic Azad University Najafabad Branch, Iran. His research interests include ternary logic, VLSI design, non-volatile memories, and brain-inspired (neuromorphic) computing.



David Z. Pan (F'14) received the B.S. degree from Peking University, Beijing, China, in 1992, and the M.S. and Ph.D. degrees from the University of California at Los Angeles (UCLA), Los Angeles, CA, USA, in 1994 and 2000, respectively. From 2000 to 2003, he was a Research Staff Member with the IBM T. J. Watson Research Center, Yorktown Heights, NY, USA. He is currently a Professor and a Holder of the Silicon Laboratories Endowed Chair in Electrical Engineering, The University of Texas at Austin, Austin, TX, USA. He has published over 420 journal articles and refereed conference papers. He holds eight U.S. patents. He has graduated 40 Ph.D. students/post-docs who are holding key academic and industry positions. His research interests include electronic design automation, design for manufacturing, machine learning, hardware acceleration, design/CAD for analog/mixed-signal design, and emerging technologies.

Arman Roohi (SM'23) is currently an assistant professor with the School Computing, University of Nebraska-Lincoln, USA, where he is the director of the Intelligent Device-2–Applications (iDEA) Laboratory. Before joining UNL in 2020, he was a postdoctoral research fellow with UT Design Automation Laboratory, the University of Texas at Austin. He received the Ph.D. degree in Computer Engineering at the University of Central Florida, Orlando, FL, USA, in 2019. His research interests span the areas of cross-layer (device/ circuit/ architecture) co-design approaches for implementing data/compute-intensive tasks, secure computation, reconfigurable and adaptive computer architectures, and beyond CMOS computing. He has completed over 50 publications on these topics, including, book chapters, STEM curricular development, and best paper recognition in IEEE Transactions on Emerging Topics in Computing in 2019, and paper of the month at IEEE Transactions on Computers in 2017.