APRIS: Approximate Processing ReRAM In-Sensor Architecture Enabling Artificial-Intelligence-Powered Edge

Sepehr Tabrizchi, *Student Member, IEEE,* Rebati Gaire, *Student Member, IEEE,* Mehrdad Morsali, *Student Member, IEEE,* Maximilian Liehr, *Member, IEEE,* Nathaniel Cady, *Member, IEEE,* Shaahin Angizi, *Senior Member, IEEE*, and Arman Roohi, *Senior Member, IEEE*

Abstract—Artificial-intelligence-powered edge devices are inspiring interest in always-on, intelligent, and self-powered visual perception systems. Due to the high energy cost of converting raw data and the limited computing and energy resources available, designing energy-efficient and low bandwidth CMOS vision sensors is vital as these emerging systems require continuous sensing and instant processing. This paper proposes a low-power integrated sensing and computing engine, namely `APRIS`, including a novel software/hardware co-design technique. This method provides a highly parallel analog multiplication and accumulation-in-pixel scheme, which realizes low-precision quantized weight neural networks to mitigate the overhead of analog-to-digital converters and analog buffers. Moreover, in order to reduce the size and power consumption, we propose the implementation of an approximate ADC in the readout circuit. Our system utilizes eight memory banks to increase computation parallelism, which has a dramatic effect on its speed and efficiency. Moreover, the proposed structure supports a zero-skipping scheme to reduce power consumption further. Our circuit-to-application co-simulation results demonstrate a comparable accuracy for our platform to the full-precision baseline on various object classification tasks while reaching an efficiency of ∼3.48 TOp/s/W.

Index Terms—Approximate computing, processing in-sensor, ReRAM, multilayer perception.

1 Introduction

Edge AI has rapidly evolved into an integral part of modern technological ecosystems, fundamentally reshaping the way we process data and interact with our digital environment. The significance of this technology cannot be overstated, as it pushes the boundaries of traditional cloud-based AI, moving towards more localized, low-latency, and power-efficient solutions. The advent of Edge AI has brought forth a paradigm shift that promotes processing at or near the data source, often directly within the sensors themselves. This transformation, largely driven by the surge in Internet of Things (IoT) devices, has been essential in dealing with the data deluge that these devices generate. As we stand on the cusp of a world teeming with billions of interconnected smart devices, the need for edge processing has become increasingly crucial. These changes have ushered in the era of sensor-based processing, where sensors not only capture data but also have the intelligence to process it. This emergence of processing near and in the sensor has significant implications for privacy, speed, bandwidth, and energy consumption. By processing data locally, the need to transmit vast quantities of raw data to the cloud is eliminated, resulting in more efficient use of network resources and a substantial reduction in latency. The ability of sensors to process data on the edge increases the responsiveness of real-time applications, making them more effective and reliable. From autonomous vehicles and drones to smart home systems and wearable technology, processing at the edge enhances the functionality of these applications.

However, there are significant obstacles to the tractability of the processing requirements of artificial intelligence tasks in terms of computational and storage resources. By removing "power and memory wall" bottlenecks, efficient methods have been developed in both the software and hardware domains. Algorithm-focused methods have extensively examined the application of quantized parameters and pruned/compressed networks [1], [2], which have helped decrease computational complexity and model size. From a hardware perspective, efficient mechanisms have been exploited to mitigate the bottlenecks of von Neumann computing models caused by separate memory and processing blocks. This setup presents significant challenges, such as lengthy memory access latency, limited memory bandwidth, and energy-intensive data transfers that limit edge device efficiency and operational time [3], [4]. As a potential solution, researchers have extensively investigated Processing-in-Memory (PIM) architecture, which has been widely studied in [3], [5], [6]. Inspired by the PIM concept, smart image sensors with preprocessing capability [4], [7]–[10] have been studied extensively. This exploration has led to new sensor paradigms, like Processing-Near-Sensor

This work is supported in part by the National Science Foundation under Grant No. 2216772, 2216773 and 2247156, and Semiconductor Research Corporation (SRC).

S. Tabrizchi, R. Gaire, and A. Roohi are with the Department of Electrical and Computer Engineering, University of Illinois Chicago, Chicago, IL, 60606 USA e-mail: (aroohi@uic.edu).

R. Gaire is with the School of Computing, University of Nebraska-Lincoln, Lincoln, NE, 68588 USA.

M. Morsali and S. Angizi are with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology (NJIT), Newark, NJ, USA e-mail: (shaahin.angizi@njit.edu).

Ma. Liehr and N. Cady are with the College of Nanoscale Science and Engineering, SUNY Polytechnic Institute, Albany, NY 12222 USA.

(PNS), where an on-chip processor accelerates digital pixel outputs near the sensor. Building on PNS and PIM techniques, two promising alternatives are Processing-in-Sensor (PIS), which operates on pre-Analog-to-Digital Converter (ADC) data [8], [11]–[13], and a hybrid PIS-PNS platform [14] that enhances vision sensor functionality and eliminates redundant data output. In [15], all the computations are performed in the analog domain. The authors in [16] replaced ADC with a novel component to implement the spike neural network (SNN) within the sensor. Generally, PIS units process images before transmitting raw data to the on-chip memory unit for processing by a PNS unit. This data transfer in traditional designs (from CMOS image sensors to memory) creates a severe bottleneck and significantly reduces feature extraction speed. Therefore, incorporating coarse-grained computation within a sensor can decrease power consumption for data conversion from photocurrents to pixel values, enhance data processing speed, and mitigate the memory bottleneck issue [4], [14].

In this paper, to make continuous sensing and instant processing suitable for resource-limited sensor devices, advancements from both algorithm and hardware architecture perspectives are deployed. In summary, our major contributions in this paper can be listed as follows: 1) We introduce two weight grouping approaches that provide an effective performance regarding accuracy loss. Further improvement is achieved by applying a 5-bit quantization method, which massively reduces the required storage and computational resources in the inference path with minimum performance degradation compared to the full precision model; 2) We develop a PIS architecture, namely **APRIS**, based on a set of innovative microarchitectural and circuit-level schemes optimized to process the 1^{st}-layer of quantized-weight MLP with weights stored in our recently fabricated and experimentally measured non-volatile resistive random-access memory (ReRAM) components that offer energy-efficiency and speed-up. Using ReRAM computational sub-arrays and an ultra-low power activation function, a highly parallel analog multiplication and accumulation-in-pixel scheme are provided, which mitigate the overhead of analog-to-digital converters and analog buffers; 3) We present a solid bottom-up evaluation framework and a PIS assessment simulator to analyze the whole system's performance. Our framework extensively assesses APRIS's performance, energy efficiency, and accuracy in different data precisions and datasets. Applying these approximations results in an extensive reduction in energy and area while an acceptable accuracy is achieved.

2 Background

2.1 Multilayer Perceptron

Artificial Neural Networks (ANNs) have revolutionized predictive applications, particularly in the field of image recognition, with Multilayer Perceptron Neural Networks (MLPs) and Convolutional Neural Networks (CNNs) emerging as highly effective architectures. MLPs, renowned for their simplicity and lower computational requirements compared to CNNs, are particularly suitable for intelligent edge devices. Moreover, MLPs generate smaller spatial features in their initial layers than CNNs, minimizing the need for transferring large amounts of data to remote servers or the cloud if enabled localized computations within the sensor. In this work, we focus on MLPs for image classification tasks, as they offer dense connectivity and a large number of learnable parameters, making them suitable for processing high-dimensional inputs like images. However, the presence of a substantial number of learnable parameters in MLPs results in a significant memory footprint and high computational demands, posing challenges when deploying models in resource-constrained environments such as mobile devices or embedded systems. Furthermore, the abundance of parameters increases redundancy and makes generalization to unseen data more difficult [17]. To overcome the computational and storage limitations of MLPs, a range of approximate computing paradigms has been investigated, including pruning [18], quantization [19], and weight sharing [20]–[22]. These paradigms aim to balance model accuracy and efficiency, making them particularly beneficial for resource-constrained devices and real-time performance requirements.

2.2 Processing Near/In Sensor

Integrating computing with sensor arrays can diminish off-chip data transmission and ADC bandwidth. Such a setup promotes power efficiency and yields higher sampling rates and superior data acquisition resolution. Efficient strategies for executing embedded signal/image processing and computer vision algorithms directly on-chip, before off-chip data transfer, include near-sensor and in-sensor processing, PNS, and PIS architectures, respectively.

The PNS architecture is a design used in image processing, where processing occurs close to the sensor, either on the same chip or nearby, before transferring data to an external processor. The PNS unit accepts raw data from the sensor, carrying out essential calculations or processing before dispatching the data to the main system. This design finds its use in scenarios where data is harvested from multiple sources/sensors. PNS diminishes the volume of data transferred off-chip and processed, enabling superior signal processing and vision performance. As per the reference [23], the CMOS imager is equipped with dual-mode delta-sigma ADCs designed to handle the first convolutional (conv.) layer of binary-weight neural networks (BWNNs). RedEye [24] carries out convolution using charge-sharing tunable capacitors. By sacrificing precision for energy savings, this design curtails energy consumption. This system employs a custom image sensor integrated with a low-power digital signal processor to execute image processing tasks. In [4], vertically stacked column-parallel ADCs and processing elements are adopted and used to conduct spatiotemporal image processing. To lower the power consumed by the ADC, [9] transforms photocurrents into pulse-width modulated signals, subsequently processed by a dedicated analog processor. Conversely, PIS includes processing capabilities directly within the sensor. This configuration allows sensors to carry out essential calculations or processing before dispatching the data to the main system. One of the most significant benefits of PIS is the potential for increased precision and accuracy in vision results. This is achieved by enabling complex image processing functions

to be performed directly within the sensor itself [25]. This direct approach mitigates the need for data transmission to a separate processor, reducing the potential for data loss or degradation. Furthermore, PIS offers on-chip memory capabilities, enabling data to be stored and processed on-chip [26]. This can significantly reduce data transfer times and increase overall processing efficiency, which is particularly beneficial in applications where data needs to be collected rapidly, such as in industrial automation and security systems. The PIS platform MACSen, as discussed by Xu et al. [8], is a prime example of the benefits of PIS. MACSen integrates multiply-accumulate (MAC) operations directly into the image sensor, utilizing double sampling to process the first conv. layer of BWNNs. This allows for efficient real-time processing of visual data at the point of acquisition without the need for additional power-hungry devices. However, this method suffers from high power consumption and a significant overhead due to the SRAM-based design.

Although PNS can offer more flexibility and scalability, it may also be more complex and costly to implement [9], [27], [28]. PIS may be more compact and efficient but might be more restricted in processing capabilities [10], [29], [30]. PNS is a suitable choice for lower-end processing needs, whereas PIS is preferable for applications requiring more advanced algorithms. In CMOS image sensors, either of these architectures can diminish data transfer and processing overhead, thus boosting computing efficiency.

3 Proposed HW/SW Co-design Approach

3.1 Proposed Weight Compression Approaches

Herein, a novel compression approach, particularly for an MLP network, is introduced. Specifically, the focus is on compressing the first layer of the network, which contains a substantial portion of the overall weight parameters. Our proposed method revolves around the concept of parameter sharing among multiple interconnected nodes in a neural network. By leveraging this approach, we effectively curtail the number of unique parameters necessary for training, storage, and computation. We refer to this technique as "grouping" and represent the group size by g. Subsequently, we employ quantization, which imposes a constraint on the parameters, compelling them to assume 5-bit integer values, thereby further diminishing memory requirements. Through the synergistic utilization of these techniques, we successfully mitigate the intrinsic limitations of MLPs, ensuring their efficient deployment on resource-constrained devices while ensuring minimal performance degradation.

3.1.1 Weight sharing approaches

The initial compression strategy in our proposed approach involves grouping, aka weight sharing. This step entails organizing multiple nodes within a network so that all nodes within a group share the same weight or parameter value. By implementing this grouping strategy, a significant reduction in the number of parameters to be trained and stored is achieved, leading to faster computations and decreased memory requirements. Figure 1 depicts a 2-layer MLP network comprising four nodes in the input layer, six nodes in the hidden layer, and two nodes in the output layer. Each hidden node performs multiplication and accumulation operations on all input nodes to produce an output. The non-activated output of the first hidden node (h_j) can be represented as follows:

$$y_j = b_j + (w_{1j}.p_1 + w_{2j}.p_2 + w_{3j}.p_3 + w_{4j}.p_4) \quad (1)$$

where b_j is the bias term associated with h_j hidden node and w_{ij} is weight associated with the p_i input node connected with h_j hidden node. Two approaches can be employed to group the weights, as illustrated in Fig. 1 (b) and (c). The first approach involves grouping weights associated with multiple hidden nodes connected to an input node, as shown in Fig. 1 (b). By grouping six weights from the p_4 node connected to g number of hidden nodes, where $g = 6$ represents the group degree, the equation simplifies to:

$$\begin{aligned} y_j &= b_j + (w^{G_1}.p_1 + w^{G_2}.p_2 + w^{G_3}.p_3 + w^{G_4}.p_4) \\ w^{G_j} &= \frac{1}{g}\Sigma_{i=1}^{g} w_{ji} \end{aligned} \quad (2)$$

In the second approach, weights associated with multiple input nodes connected to a hidden node are grouped together, as depicted in Fig. 1 (c). In this scenario, four weights related to the four input nodes connected to a hidden node are assigned the same value. By grouping the weights in this manner with a group size of $g = 4$, the equation simplifies to:

$$\begin{aligned} y_j &= b_j + w^{G_j}(p_1 + p_2 + p_3 + p_4) \\ w^{G_j} &= \frac{1}{g}\Sigma_{i=1}^{g} w_{ij} \end{aligned} \quad (3)$$

ReRAM-based weight representations of two grouping approaches are depicted in Fig. 2, where by using Approach 2, one ReRAM can be shared among several inputs.

3.1.2 Validation

A performance comparison between the two approaches for various datasets and group degrees is presented in Fig. 3. The graph clearly illustrates that Approach 1, in most cases, maintains a consistently high performance across a wide range of group degrees. In contrast, the performance of Approach 2 experiences a sharp decline after a comparatively

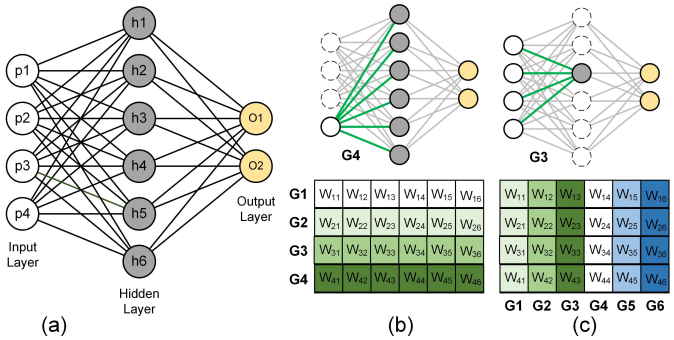

Figure 1: (a) A 2-layer MLP network. Two group sharing methods and their matrices, (b) multiple hidden nodes connected to an input, where the number of groups equals the input nodes, and (c) multiple inputs connected to a hidden node and the number of groups equal to the number of hidden nodes.

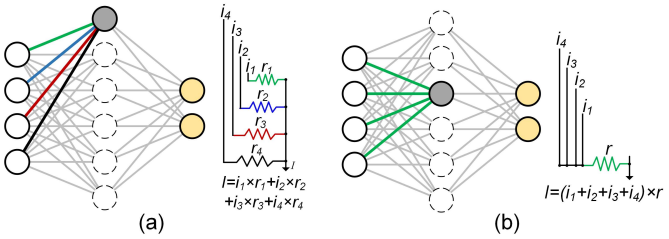

Figure 2: ReRAM-based weight representations of two grouping approaches, (a) method 1, and (b) method 2, where one ReRAM can be shared among several inputs.

smaller group size, with some exceptions. The circles in the graph represent various group degrees considered to ensure comparable results between the two grouping strategies. The reason behind this discrepancy lies in the fundamental differences between the two approaches. In Approach 1, although multiple weights are assigned the same value, each input pixel retains its unique parameter value. As a result, the network is able to effectively learn from all individual input features, leading to superior performance. On the other hand, Approach 2 combines multiple input pixels and assigns them a single parameter value. Consequently, the network is required to learn multiple input features with a single parameter, resulting in a diminished learning capacity. As the group size increases, the network's ability to learn further deteriorates, causing a rapid decline in performance beyond a certain group size. Although Approach 2 may lack flexibility in terms of group size, it offers several hardware-related benefits, as depicted in Fig. 2 and Table 1, which summarizes the comparison of the number of operations among the baseline and the two approaches. It is evident that both approaches entail an equal number of operations and are smaller than the uncompressed method. However, referring to Fig. 2(a), it can be observed that in Approach 1, each pixel needs to be multiplied by a specific weight. Consequently, the design of the architecture for Approach 1 necessitates a separate weight and hardware for each pixel, resulting in a higher number of interconnections and transistors within the sensor area, which is perceived from Fig. 2. This increased density in Approach 1 leads to a significant reduction in the fill factor of the design. Upon analyzing this figure, it is evident that in Approach 2, the individual elements i_{1-4} can be amalgamated and connected to a single weight, allowing for the utilization of a single hardware component to perform the multiplication operation on them collectively.

Table 1: Computational and hardware cost analysis of three approaches.

Method	Mul	Sum	Accuracy	Sensor area overhead
Normal	$m \times n \times h$	$m \times n \times h$	best	n
Method 1	$\frac{m\times n\times h}{g}$	$\frac{m\times n\times h}{g}$	better	n
Method 2	$\frac{m\times n\times h}{g}$	$\frac{m\times n\times h}{g}$	good	$\frac{n}{g}$

m and n are the pixel array dimension, h is the number of first layer hidden nodes, and g is the group number.

3.1.3 Quantization

In the next step, we apply quantization to the shared weights in each group, representing the weights as 5-bit in the sign system, which results in integers in the range of $[-15, 15]$. This quantization reduces the bit-width of the weights, further reducing the model's memory requirements and computational complexity. To quantize the weights, we use a range-based linear quantization [35], where we first scale the full-precision (FP) weights by a scaling factor (q_x), which is calculated as $q_x = \frac{2^n-1}{max_{xf}-min_{xf}}$, where n is the number of bits to use for encoding, which in our case is 5. We then multiply the weights to this scaling factor (q_x) and then quantize to the nearest integers in the desired range.

3.2 Proposed APRIS Architecture

The ultra-low-power sensor integrating sensing and computing to realize TinyML applications, namely APRIS, is proposed and illustrated in Fig. 4. It takes pre-trained MLP, including the quantized grouped weights. Thus, to make it a generic architecture, designers should consider the group size at design time to satisfy both accuracy and efficiency metrics. The APRIS architecture consists of an array of proposed pixels (❶), compute add-ons (CA) (❷), a controller, memory components (❸), and a readout circuit (❹). This architecture is designed to implement the first layer of MLP networks in the analog domain effectively. To perform the remaining layers, the results are then converted to digital using a digital deep learning accelerator (DLA) (❺).

In order to reduce the number of wights in the first layer and also generate a more efficient structure, the proposed Approach 2 is utilized, which is expressed by Eq. 4 The parameter g is the number of pixels that are in the same group and is determined by the application. For example, for a simpler dataset, a higher number for g, or group degree. Since the output of each pixel is current, the summation can be performed readily by connecting the outputs of each pixel group together, the parenthesis on the right side of $=$ in Eq. 4. To complete a MAC operation, the weight multiplication can be executed by the proposed compute add-on (CA). If the size of a pixel array is $m \times n$, by using this technique, rather than $m \times n$ CAs for each pixel, we only need $\lceil \frac{m\times n}{g} \rceil$ CAs. This method also yields an advantage by connecting the output of pixels in a group, allowing us to uncouple CAs from the pixel area and place them before the ADCs and near memory, resulting in a higher fill factor. It should be mentioned the proposed architecture needs more bus lines to transfer data to a CA, which is defined using $\frac{m}{g}$. Based on the equation, the best value for g is m. In this case, there is no need for any additional bus line to transfer data. Hereafter, for simplicity, we consider a size of 32×32 for the pixel array.

$$p_1.w_1 + p_2.w_2 + ... + p_g.w_g = w.(p_1 + p_2 + ... + p_g) \quad (4)$$

3.2.1 Pixel array

The pixel array is composed of 32×32 proposed pixels. The transistor-level and layout demonstrations of one pixel are shown in Fig. 5(a) and (b), respectively. One of the important parameters in designing CMOS imagers is the fill factor,

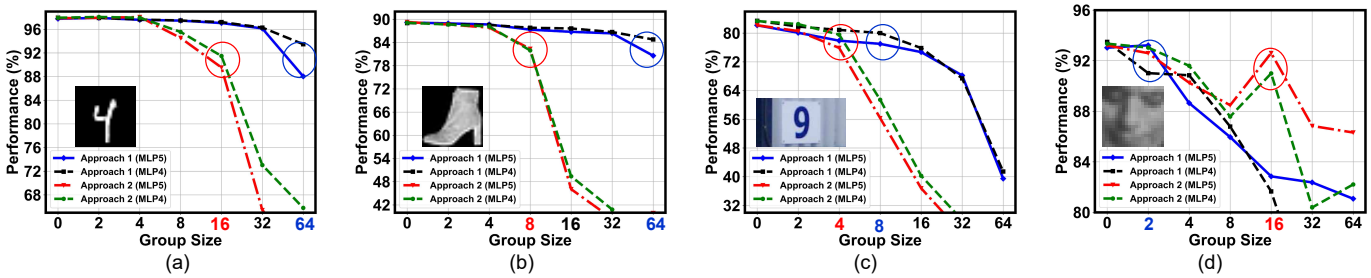

Figure 3: Accuracy on (a) MNIST [31], (b) Fashion-MNIST [32], (c) SVHN [33], and (d) CBCL Face [34] datasets leveraging two weight-sharing approaches.

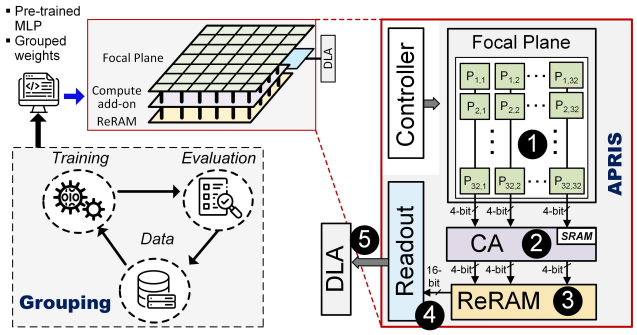

Figure 4: The proposed design flow and the APRIS architecture.

which refers to the percentage of each pixel area that can be used to collect light, i.e., the photosensitive part. The main difference between this structure and widely used pixels is in the row selector, where in conventional image sensors, the values of pixels are read row by row, while in MLP networks, all the pixels should contribute simultaneously. Thus, there is no need for the row selector. Due to the structure of the proposed pixel, a transistor and a control line are removed, resulting in a higher fill factor.

The proposed sensor's dimensions, which include all buses and interconnections, measure $60\lambda \times 60\lambda$, with a fill factor of approximately 0.556. Notably, removing just one additional metal line in each pixel (row controller) results in an increasing fill factor of around 0.62, an increase of 10%. This improvement is for the $g = 32$, which is the best case. However, in our current work, we are considering a different g, specifically set to eight. Consequently, connecting all groups of pixels to the CA's now requires four bus lines, which introduces an overhead that further decreases the fill factor of the pixel area to approximately 0.42. In previous studies, the number of buses in each column was substantially higher. For example, in [8], [12], the number of interconnections in each column was determined by the

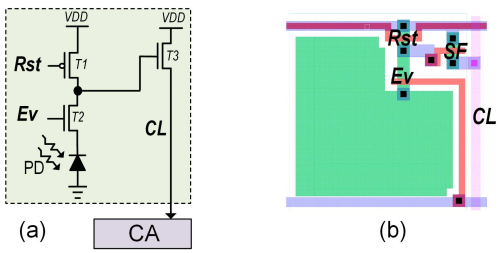

Figure 5: The proposed (a) structure of the pixel, including three transistors, and (b) its layout.

product of $m \times s$, where m and s representing the number of pixel array rows and the number of nodes in the first hidden layer, respectively. We can assert the proposed pixel array has the best area in comparison to similar standard architectures [12], [29]. The number of buses to a CA in each column of the pixel array equals $\frac{32}{g}$. On the other hand, based on Fig. 2(b), each group of sensors needs a bus to transfer their current to a CA. In this case, we can save more area by increasing the number of sensors in each group, which can be determined based on accuracy and power consumption trade-off. We consider the group number to be eight, meaning four lines in each column of the pixel array to the output are required. The functionality of eight connected pixels is conducted using the HSPICE simulator for a 45nm CMOS technology, and their transient vector is shown in Fig. 6. In this figure, green vectors are the inputs, and blue ones are the outputs. The *Rst* signal is used to charge the pixels' internal capacitor. In ❶, the sensor's capacitor charges to V_{DD} after that, in ❷, when this signal equals zero, the discharge signal becomes one, and the value of the pixel based on light intensity is evaluated. In the last step in ❸ and ❹, the current of sensors stays constant on the current line (CL).

3.2.2 *Compute add-on*

This component is responsible for performing MAC operations between pixel values and their weights. The current for each group of the pixel array is connected to one CA. This component converts the current to voltage and also produces the same negative voltage using analog op-amps, as shown in Fig. 7 (a). One of the amplifiers is in inverting mode. The amplifiers' outputs, depicted in Fig. 6, are *Neg (v)* and *Pos (v)*, corresponding to negative and positive voltages, respectively. These amplifiers not only isolate the CL line from the memory section but also can amplify the input signals. APRIS supports the sign system for storing the weights. In this presentation, the most significant bit of the weights determines whether the weight is positive or negative, 0 or 1, respectively. This sign bit is stored on an SRAM and connected to a multiplexer in CAs, and its output is connected to the ReRAM crossbar array. Based on the value of the weight, positive or negative voltages are connected to the *Out* signal in ❸ and ❹, illustrated in Fig 6, respectively.

In our architecture, zero skipping is integrated as an intrinsic input feature. Since APRIS utilizes resistors and currents for computation, an input value of zero naturally does not generate any current, leading to no power consumption in those instances. Additionally, as all operations

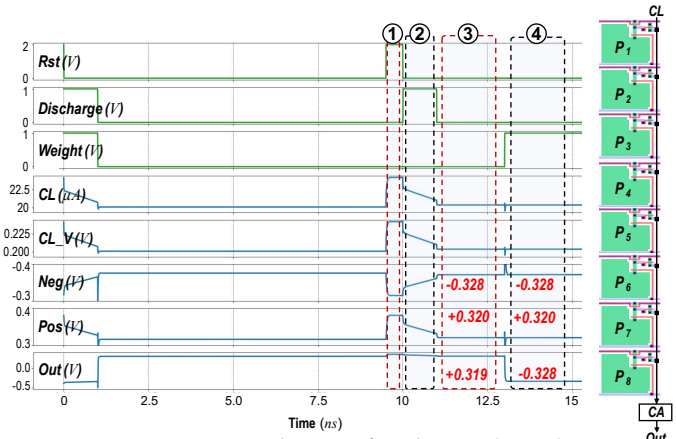

Figure 6: Transient simulation of eight pixels and one compute add-on

within a single patch of our architecture occur concurrently, there is no need for individual detection of zero inputs, simplifying the process and boosting efficiency.

3.2.3 Memory components

The memory part of APRIS contains both volatile (SRAM) and non-volatile (ReRAM) components. If we consider that weight precision is b-bit (here $b = 5$), only one bit, which is the sign bit, is stored in SRAM memory, and the remaining $b - 1$ bits are held in ReRAMs. ReRAM's resistance changes in a continuous range based on the width of the write voltage. Therefore, we divide the resistance range of ReRAM into 16 points and find the appropriate voltage width for each of them. This writing process only needs to be carried out once. In the ReRAM part, we use a $x \times y$ memory crossbar where x is the number of first hidden layer nodes, and y is the total number of pixels divided by the g. The ReRAM crossbar comprises our recently fabricated and experimentally measured ReRAM device that stores data in varying resistive states by creating and rupturing a conductive filament within the metal oxide insulator. Figure 8 (inner image) illustrates a Transmission Electron Micrograph (TEM) of our fabricated TiN/Ta/TaO$_x$/TiN ReRAM device integrated with CMOS n-channel Field-Effect Transistor (nFET) in 65nm CMOS technology to realize a 1T1R unit cell as a primary storage element in the proposed APRIS. In the set phase, the conductive filament connects the top and bottom electrodes, leading to a Low Resistance State (LRS), whereas, in the reset phase, the filament breaks, and the resistance of the device increases, yielding a High Resistance State (HRS), as shown in Fig. 8 (inner schematic).

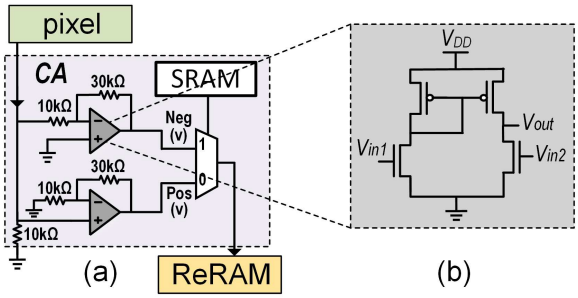

Figure 7: The proposed (a) compute add-on and biasing considerations and (b) op-amp structure.

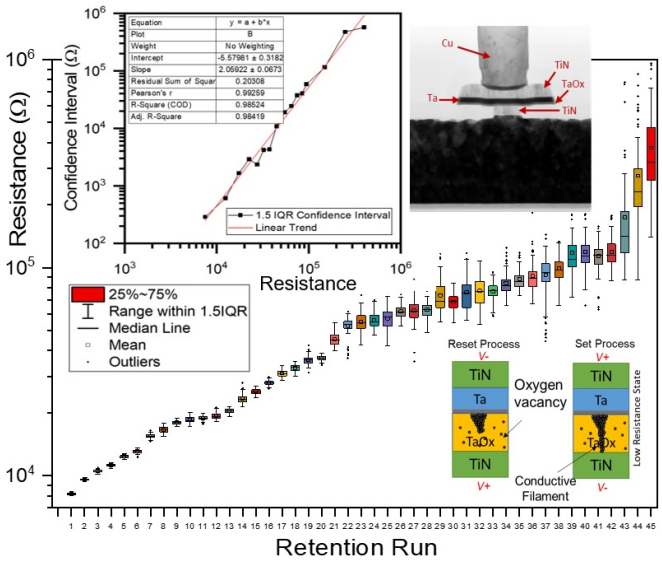

Figure 8: Box-and-Whisker plot showing the variability of 45 retention runs at different resistance levels. Inlet showing the comparison of the confidence interval of retention to median resistance. Linear trendline represents expected variability at specific resistance values.

Here, we propose a promising device-to-system level co-design approach to understand better the theoretical limits on the number of distinct weight levels that ReRAM devices can obtain using a large retention experiment. To reduce HRS variability, we adopt a read-write-verify approach to achieve resistance in a specific window. A total of 45 resistance levels are captured, where each resistance state is held for 10,000 seconds, with reads taken every 100 seconds. The variability of each resistance level is plotted in Fig. 8, where an increasing resistance leads to an increase in the variability of the read current. This is further emphasized in Fig. 8's inlet that shows the clear resistance levels confidence interval, i.e., variability, and the linear trend that shows an Adj. R-squared value of 0.984. This trendline can be used to estimate the total theoretical number of distinct weight levels at 1.5 IQR (Interquartile Range) to be at least 32 when ranging the resistances from 3-500 kΩ. Therefore, 16 high-confidence experimental resistance states are selected to serve as the 4-bit memory states for APRIS. This results in a total quantization of 5 bits. To increase the parallelism, the weights are distributed into multiple memory banks, where each bank consists of an independent ADC to measure the final voltage. Herein, the memory banks are set to 16. As a result, we have 16× speed up in compared to one memory bank and ADC. With this technique, the total number of ReRAM cells is the same, while the number of ADCs increased to 16.

3.2.4 Readout circuit

All the operations in the first layer of MLP are in the analog domain; therefore, in the last part of the architecture, data should be converted to the digital domain to perform the remaining layers using a digital deep learning accelerator (DLA). Accordingly, an approximate ADC, as shown in Fig. 9, is proposed and positioned at the end of each memory bank. The proposed ADC includes a resistor and a capacitor

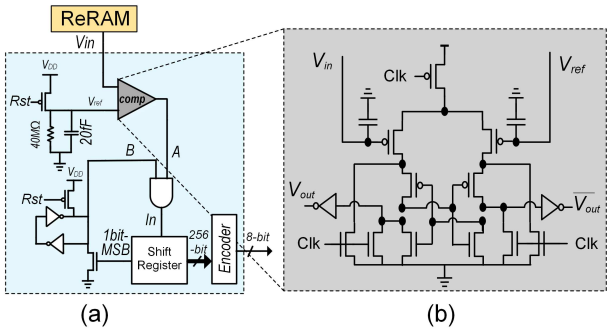

Figure 9: The proposed readout circuits: (a) an ADC and (b) a comparator.

as a substitute for using large resistance divination to generate reference voltages. The transient vector of the proposed ADC is depicted in Fig. 10. In this figure, the voltage of the capacitor is charged to V_{DD} through the *Rst* signal, and after that, it gradually discharges through the resistor (*Ref* signal). During discharge time, *Clk* signal is responsible for pre-charge the comparator to compare the reference voltage *Ref* with the input voltage *In* and stores the result in a shift register. The frequency of *Clk* is $\times 256$ faster than the *Rst* clock; therefore, each time reset occurs, we can measure 256 values and store them in the shift register. It should be mentioned the RC circuit, depicted in Fig. 9(a), generates only positive voltage, which means the output of the ADC for negative inputs is always zero. Due to this, the ReLU activation function is inherently embedded in our ADC. Conventional ADCs use a priority encoder to convert the comparator outputs to a digital value, while in our design, thanks to using an AND gate and a modified 1-bit SRAM, the bulky priority encoder is replaced by a basic encoder.

In the reset phase (*Rst* = 0), the value of the SRAM (*B* signal) sets to V_{DD}; thus, the output of the AND gate is determined by the other input (*A*) and passes through the shift register. This step repeats until we get the first "1" in the AND's output. One cycle after writing "1" in the shift register (*MSB*), the MSB sets *B* to zero. Because inputs *B* of the AND is zero, regardless of input *A*, the output is always zero. With this technique, we can guarantee that only one bit in the shift register is '1', and the rest of them are equal to zero. As a result, there is no need for a priority encoder. The proposed ADC is much smaller but slower compared to conventional ADC, e.g., flash ADC. In the proposed ADC,

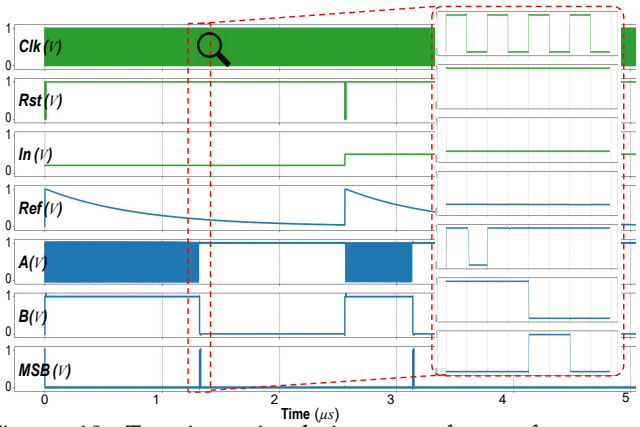

Figure 10: Transient simulation waveform of proposed ADC.

256 clock cycles are required to produce a digital value. Thus, we increase the number of memory banks in order to increase parallelism. Note that the transient results only exhibit the *MSB* bit, not the entire shift register's bits.

4 Evaluation Results

4.1 Framework & Methodology

The proposed evaluation framework is created through a bottom-up methodology, as shown in Fig. 11. At the device level, the proposed ReRAM device is fabricated, and the switching data and resistance levels are experimentally extracted to be used in a Verilog-A model. At the circuit level, we implement the APRIS's array and computational peripheral circuitry in NCSU 45nm technology in HSPICE. To exploit APRIS, the 1^{st}-layer weight parameters need to be quantized while the remaining layers are processed with the off-chip processors. We train a PyTorch to extract the 1^{st}-layer weights at the application level. At the architecture level, a Python-based behavioral APRIS model is then developed, taking input from the circuit level with these quantized weights based on a previous simulator (NVSIM [36]) to model the timing, energy, and area. This tool offers flexibility in array configuration and peripheral circuitry design. Based on the circuit level results, it can alter the configuration files (.cfg) with different array organizations and add-ons and report performance. After the 1^{st}-layer of convolutions, the results are captured and used in the Pytorch model to process the 2^{nd}-to-last and to report accuracy.

4.2 Performance Evaluation

Table 2 presents a comprehensive comparison of the structural and performance features of recent in-sensor/near-sensor designs. Each design has a distinct target application, as indicated in the table. However, to ensure a fair comparison, we estimated the power consumption and performance of computing units when performing the same task of processing the 1^{st}-layer of a CNN. Our observations are summarized below: (i) The only designs that support a fully parallel and fast entire-array computation scheme are APRIS and the accelerators mentioned in [8], [11], [25], [29], [37]; (ii) APRIS and the designs in [7], [8], [11], [25], [29] incorporate integrated memory components. Among these designs, our design and [10], [25], [29] are the only ones that utilize NVMs for normally-off and instant computing. Moreover, thanks to the proposed multi-level ReRAM,

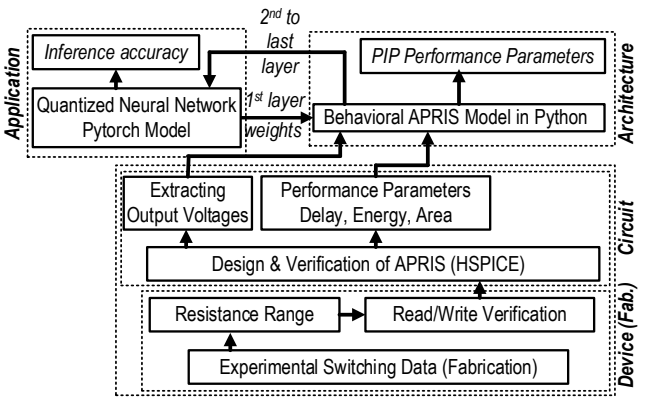

Figure 11: The proposed bottom-up evaluation framework.

APRIS is the only platform that supports up to 5-bit (4+1-bit SRAM) weight precision; (iii) APRIS achieves remarkable power saving compared to several designs, e.g., compared to [37], it consumes about two orders of magnitude less power, making our designs one of the most power-efficient designs available; (iv) In terms of efficiency, the CNN accelerator with 8-bit support mentioned in [37] achieves a rate of 3.37 TOp/s/W. On the other hand, AppCiP [29] with 2-bit weight precision achieves a higher efficiency of 4.12 TOp/s/W. Herein, our design achieves a comparable computation efficiency (∼3.48 TOp/s/W). We also simulated the APRIS at the circuit level, including peripheral circuitry, using the IBM 65nm CMOS10LPe PDK in Cadence to achieve the performance parameters using our bottom-up evaluation framework. APRIS consumes 0.0038 mW and achieves a power efficiency of 2.01 TOp/s/W. Detailed breakdown of area and power consumption of APRIS is shown in Fig. 12. We observe that the majority of power and area consumption is attributed to the ReRAM crossbar component. This finding highlights the crucial importance of our grouping technique, which significantly reduces the ReRAM crossbar size, leading to enhanced efficiency and overall performance improvements.

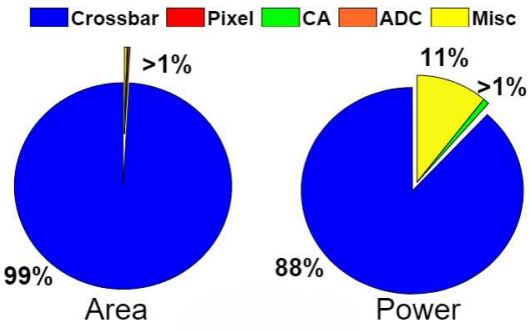

Figure 12: Area and power breakdown of APRIS.

4.3 Accuracy

Datasets: Our model was extensively evaluated using four publicly available datasets: MNIST [31], FMNIST [32], SVHN [33], and CBCL Face [34]. The MNIST dataset is a standard benchmark for image classification consisting of 70,000 28×28 grayscale images of handwritten digits (0-9), split into 60,000 training images and 10,000 test images. Like MNIST, Fashion MNIST (FMNIST) has ten classes but is more challenging than MNIST, focusing on fashion-related classification. SVHN (Street View House Number) includes over 60,000 32×32 color images, commonly used for digit recognition tasks. CBCL Face is used for face recognition, including 2,429 19×19 grayscale images of faces and 4,548 non-face object images, including various lighting conditions. Table 4 summarizes the key information from these datasets. Notably, our approach, APRIS, only supports grayscale images, so RGB SVHN samples were first converted to grayscale before being fed into the networks.

NN Architecture: Our study demonstrates the efficacy of our proposed methodology in addressing image classification tasks across the aforementioned datasets. Specifically, we constructed two distinct multilayer perception architectures, one with five hidden layers (MLP5) and the other with four hidden layers (MLP4). Notably, these architectures were designed as fully-connected networks, characterized by an initial hidden layer comprising 512 nodes. Following this, each subsequent layer's nodes were halved until the output layer was achieved. Remarkably, the number of trainable parameters in the first hidden layer constituted a minimum of 70% of the overall network parameters. Detailed specifications of the architectures, including the number of nodes and parameters per layer, inference time and size, are provided in Table 5. These architectural configurations enabled a comprehensive exploration of the network's capacity to extract discriminative features and facilitate accurate classification.

Training Details: The PyTorch framework served as the cornerstone of our experimentation, facilitating network creation, training, and testing processes. To optimize our models, we employed the stochastic gradient descent (SGD) optimization technique. Our initial learning rate was set at 0.001, which underwent a reduction by 10% after 60 epochs to enhance convergence. The training was conducted using a batch size of 32, and all networks underwent training for 100 epochs to ensure sufficient convergence. In order to identify the most effective model, we preserved the best-performing checkpoint based on its validation set performance and subsequently evaluated and reported its performance on the test set. Our methodology involved the partitioning of the collective set of learnable weights residing within the initial layer of the network into distinct groups. While we quantized the first-layer weight parameters and mapped them into the 16 distinct resistive values obtained from Fig. 8 in the inference process, throughout the training process, we employed FP weights and utilized the ReLU activation function. During each iteration of training, we implemented a weight clipping mechanism [43] to confine the weights of individual nodes within a prescribed range of $(-c, c)$. By setting the value of c to 0.05, we effectively attenuated the influence of outlier weight values, thereby bolstering the numerical stability of the training procedure, $w' \leftarrow \text{clip}(w, -c, c)$. Following the completion of a full epoch, we proceeded to compute the average weight value encompassing all nodes within a given group, assigning this computed value to all nodes within the group. We then performed post-training quantization (PTQ) of the first hidden layer into integer values between -15 and 15. As an alternative, quantization-aware training (QAT) [44] can be exploited in order to minimize accuracy loss. However, leveraging APRIS, the experimental results show that the primary factor influencing accuracy loss is the group size. The difference in performance for proper group size (the group size that gives an acceptable accuracy) is negligible, regardless of the quantization method employed PTQ or QAT. In addition, because of the inflexibility in current implementations of QAT to choose custom layers and bit-precision for quantization, we utilized PTQ in our method.

Results: A summary of the evaluation results across the four diverse test sets is presented in Table 3. The FP column presents the performance outcomes achieved by employing full-precision (FP) weights with varying degrees of weight sharing in the initial hidden layer. Conversely, the 5-bit column exhibits the results obtained by subjecting the FP model to 5-bit quantization. Notably, the values highlighted

Table 2: Performance comparison of various PIS/PNS/PIP units.

Designs	Technology (nm)	Purpose	Precision	Comput. Scheme	Memory	NVM	Pixel Size (μm^2)	Array Size	Power (mW)	Efficiency (TOp/s/W)
[38]	180	2D optic flow est.	NA	row-wise	Yes	No	28.8×28.8	64×64	0.029	0.0041
[9]	180	edge*/blur/sharpen/ 1^{st}-layer CNN	1∼8-bit	row-wise	No	No	7.6×7.6	128×128	sensing: 77 processing: 91	0.777
[4]	60/90	STP†	NA	row-wise	Yes	No	3.5×3.5	1296×976	sensing: 230 processing:363	0.386
[8]	180	1^{st}-layer BNN	1-bit	entire-array	Yes	No	110×110	32×32	0.0121	1.32
[7]	180	edge*/TMF‡	NA	row-wise	Yes	No	32.6×32.6	256×256	1230	0.535
[25]	65	1^{st}-layer BNN	1-bit	entire-array	Yes	Yes	55×55	128×128	sensing: 0.025 processing: 0.0088	1.745
[11]	180	1^{st}-layer BNN	1-bit	entire-array	Yes	No	35×35	32×32	0.00014 - 0.00053	9.4-34.6
[39]	65	2 - 64 Conv/ROI**	8-bit	row-wise	No	No	9×9	160×128	0.042 - 0.206	0.15 - 3.64
[37]	180	1^{st}-layer CNN	8-bit	entire-array	No	No	10×10	128×128	0.45 - 1.83	1.41 - 3.37
[29]	45	1^{st}-layer CNN	2-bit	entire-array	Yes	Yes	38×38	32×32	0.00096 - 0.0028	1.37 - 4.12
APRIS	45	1^{st}-layer MLP	5-bit***	entire-array	Yes	Yes	3.2×3.2	32×32	0.0022	3.48
APRIS	65	1^{st}-layer MLP	5-bit***	entire-array	Yes	Yes	3.2×3.2	32×32	0.003	2.01

*Edge extraction. †Spatial Temporal Processing. ‡Thresholding Median Filter. **Region Of Interest. ***4-bit ReRAM + 1-bit SRAM (denotes the sign).

Table 3: Summary of quantitative performance comparison of our approach on four test sets.

Group Size	MLP5								MLP4							
	MNIST		FMNIST		SVHN		CBCL Face		MNIST		FMNIST		SVHN		CBCL Face	
	FP	5-bit	FP	5-bit	FP	5-bit	FP	5-bit	FP	5-bit	FP	5-bit	FP	5-bit	FP	5-bit
0	97.85*	97.88	89.38*	89.3	83.78*	82.19	97.75*	93.21	98.05*	97.96	89.48*	89.11	84.37*	83.35	97.77*	93.35
2	97.93	97.94	89.22	88.65	83.79	80.57	97.8	92.6	98.01	98.09	89.17	88.64	84.13	82.51	97.8	93.04
4	98.14	98.08	88.15	87.85	80.88	75.87	97.47	90.26	98.03	97.97	88.42	88.15	82.16	79.57	97.35	91.6
8	94.67	94.54	84.38	82.36	68.38	56.33	95.67	88.48	95.52	95.55	83.41	81.95	69.76	61.44	95.62	87.58
16	90.47	89.54	59.53	46.07	52.24	36.79	93.16	92.61	92.68	91.49	59.34	49.4	53.83	40.27	92.98	91
32	73.09	65.44	46.15	38.44	38.45	24.22	92.48	86.84	74.86	73.04	46.54	40.81	39.77	27.94	92.35	80.39
64	71.44	57.27	43.65	39.89	38.34	23.92	92.56	86.32	75.11	65.8	45.28	33.04	39.11	23.44	92.46	82.21
128	64.74	55.72	35.28	30.54	33.41	20.29	92.9	88.93	66.54	46.69	38.28	34.16	36.8	23.34	92.41	84.51
256	36.02	16.91	27.81	30.69	23.55	13.31	92.71	90.31	34.01	20.64	24.94	24.51	23.72	13.53	91.56	83.38

* The full-precision (FP) baseline result without grouping and quantization.

Table 4: Summary of dataset information.

Dataset	# Samples				# Classes	Resolution	ImageType
	Train	Val	Test	Total			
MNIST [40]	50,000	10,000	10,000	70,000	10	28×28	Greyscale
FMNIST [32]	50,000	10,000	10,000	70,000	10	28×28	Greyscale
SVHN [41]	65,931	7,325	26,032	99,289	10	32×32	RGB
CBCL Face [42]	5,581	698	698	6,977	2	19×19	Greyscale

Table 5: Summary of Network Specifications.

Networks	MLP5					MLP4			
Layers	I	II	III	IV	V	I	II	III	IV
# Node	512	256	128	64	10	512	256	128	10
# Param	401,920	131,328	32,896	8,256	650	401,920	131,328	32,896	1,290
Total Param	575,050					567,434			
IT* (μs)	8.738					7.388			
Model Size	5.166 KB					2.635 KB			

*Inference time.

with green backgrounds denote the baseline performance attained when utilizing FP weights without any weight sharing. Furthermore, the values marked with an orange background represent the preferred group degree for each classification task, representing the degree of weight sharing that minimizes performance loss when employing our approach. For the MNIST classification task, employing a group degree of 16 yields acceptable performance, resulting in a meager 7.6% accuracy loss compared to the baseline performance while concurrently reducing the number of parameters by an impressive 93.75%. Similarly, in the FMNIST classification, a group degree of 8 manifests as the proper choice, yielding an accuracy loss of 8.1% while reducing parameters by 87.5%. Furthermore, in the context of SVHN classification, both the FP and 5-bit approaches showcase optimum performance with group degrees of 4 and 2, respectively, resulting in a negligible accuracy loss of 3% for each approach. Surprisingly, the presence of imbalanced samples in the CBCL Face datasets introduces challenges when applying weight grouping techniques, leading to fluctuating results. Consequently, even when employing a group degree as high as 256 (99.6% reduction in weights) in the CBCL face classification, an appropriate performance loss of 11% is achieved. These outcomes underscore the flexibility of our approach in accommodating diverse degrees of weight sharing while minimizing accuracy degradation.

The comparison of classification accuracy between APRIS and the state-of-the-art is summarized in Table 6, where all the designs quantized the 1^{st}-layer convolution. The results show that the APRIS architecture without the grouping scheme provides higher accuracy than almost all the previously published PNS and PIS designs.

Table 6: Accuracy (%) comparison on MNIST, FMNIST, SVHN, and CBCL Face dataset.

Approaches	Configurations	MNIST	FMNIST	SVHN	CBCL Face
MACSen [8]	1-bit MLP	91.20	82.30	–	–
PISA [25]	1-bit CONV	95.12	–	90.35	–
MR-PIPA [10]	2-bit CONV	97.26	85.68	91.05	92.30
TizBin [26]	2-bit CONV	97.38	85.68	91.05	92.30
AppCiP [29]	3-bit CONV	–	–	96.49	98.30
APRIS (Ours) I*	5-bit MLP	97.88	89.30	82.19	93.21
APRIS (Ours) II**	5-bit MLP	89.54	82.36	80.57	90.31

*without and **with grouping.

5 Conclusion

The paper introduces a low-power integrated sensing and computing engine called APRIS, incorporating a novel software/hardware co-design approach. APRIS enables highly parallel analog multiplication and accumulation-in-pixel, facilitating low-precision quantized weight neural networks while minimizing the need for ADC and analog buffers. The system includes an approximate ADC in the readout

circuit to reduce area and power consumption. With eight memory banks for enhanced computation parallelism and a zero-skipping scheme to decrease power consumption, the system achieves comparable accuracy to a full-precision baseline for object classification tasks with an efficiency of 3.48 TOp/s/W.

REFERENCES

[1] S. Zhou *et al.*, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.

[2] 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.

[3] 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.

[4] T. Yamazaki, H. Katayama, S. Uehara, A. Nose, M. Kobayashi, S. Shida, M. Odahara, K. Takamiya, Y. Hisamatsu, S. Matsumoto *et al.*, "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.

[5] 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.

[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] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] T. Ma *et al.*, "Leca: In-sensor learned compressive acquisition for efficient machine vision on the edge," in *Proceedings of the 50th Annual International Symposium on Computer Architecture*, 2023, pp. 1–14.

[14] 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.

[15] B. Zambrano, S. Strangio, T. Rizzo, E. Garzón, M. Lanuzza, and G. Iannaccone, "All-analog silicon integration of image sensor and neural computing engine for image classification," *IEEE Access*, vol. 10, pp. 94 417–94 430, 2022.

[16] Z. Li, Q. Zheng, Y. Chen, and H. Li, "Spikesen: Low-latency in-sensor-intelligence design with neuromorphic spiking neurons," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2023.

[17] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," *arXiv preprint arXiv:1412.6115*, 2014.

[18] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," *Advances in neural information processing systems*, vol. 28, 2015.

[19] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2704–2713.

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

[21] J. Wu, Y. Wang, Z. Wu, Z. Wang, A. Veeraraghavan, and Y. Lin, "Deep k-means: Re-training and parameter sharing with harder cluster assignments for compressing deep convolutions," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5363–5372.

[22] E. Dupuis, D. Novo, I. O'Connor, and A. Bosio, "On the automatic exploration of weight sharing for deep neural network compression," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 1319–1322.

[23] 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.

[24] 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.

[25] S. Angizi, S. Tabrizchi, D. Z. Pan, and A. Roohi, "Pisa: A non-volatile processing-in-sensor accelerator for imaging systems," *IEEE Transactions on Emerging Topics in Computing*, 2023.

[26] 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.

[27] Q. Li *et al.*, "Ns-fdn: Near-sensor processing architecture of feature-configurable distributed network for beyond-real-time always-on keyword spotting," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 5, pp. 1892–1905, 2021.

[28] M. Morsali, S. Tabrizchi, A. Marshall, A. Roohi, D. Misra, and S. Angizi, "Design and evaluation of a near-sensor magneto-electric fet-based event detector," *IEEE Transactions on Electron Devices*, 2023.

[29] 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.

[30] A. Roohi, S. Tabrizchi, M. Morsali, D. Z. Pan, and S. Angizi, "Pipsim: A behavior-level modeling tool for cnn processing-in-pixel accelerators," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 1, pp. 1–13, 2023.

[31] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[32] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[33] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[34] "Cbcl face database," http://www.ai.mit.edu/projects/cbcl.old/software-datasets/FaceData2.html, accessed: 2022-09-30.

[35] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," *arXiv preprint arXiv:2103.13630*, 2021.

[36] X. Dong *et al.*, "Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE TCAD*, vol. 31, no. 7, pp. 994–1007, 2012.

[37] R. Song, K. Huang, Z. Wang, and H. Shen, "A reconfigurable convolution-in-pixel cmos image sensor architecture," *IEEE Transactions on Circuits and Systems for Video Technology*, 2022.

[38] 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.

[39] M. Lefebvre, L. Moreau, R. Dekimpe, and D. Bol, "7.7 a 0.2-to-3.6 tops/w programmable convolutional imager soc with in-sensor current-domain ternary-weighted mac operations for feature extraction and region-of-interest detection," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64. IEEE, 2021, pp. 118–120.
[40] Y. LeCun *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[41] Y. Netzer *et al.*, "Reading digits in natural images with unsupervised feature learning," 2011.
[42] E. Maggio and A. Cavallaro, "Hybrid particle filter and mean shift tracker with adaptive transition model," in *Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, vol. 2, 2005, pp. 221–224.
[43] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*. PMLR, 2017, pp. 214–223.
[44] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. Van Baalen, and T. Blankevoort, "A white paper on neural network quantization," *arXiv preprint arXiv:2106.08295*, 2021.

Sepehr Tabrizchi (S'22) is currently a Ph.D. student at the Department of Electrical and Computer Engineering, University of Illinois Chicago, USA. He received a master's 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.

Rebati Gaire is currently a Masters student at the School of Computing, University of Nebraska-Lincoln, USA. He completed his Bachelor's degree in Computer Engineering from IOE, Pulchowk Campus, Nepal, in 2021. With a focus on Machine Learning and Computer Vision applications tailored for resource-constrained edge devices, Rebati is deeply engaged in research aimed at optimizing computation models for various applications, ensuring their efficient deployment on edge devices.

Mehrdad Morsali received the bachelor's degree in electrical engineering from the Ur- mia University, Urmia, Iran, in 2014, and the master's degree in electrical engineering from Shahid Beheshti University (SBU), Tehran, Iran, in 2020. He is currently working toward the PhD degree with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology (NJIT), Newark, NJ, USA. His cur- rent research interests include in-sensor com- puting and non-volatile memories.

Maximilian Liehr received the B.A. and M.Eng. degrees from the Rensselaer Polytechnic Institute, Troy, NY, USA, and the Ph.D. degree from the College of Nanoscale Science and Engineering, SUNY Polytechnic Institute, Albany, NY, USA. He is currently a Post-Doctoral Research Associate with the SUNY Polytechnic Institute. He has active research interests in development of non- volatile nanoelectronics, including resistive random access memory (ReRAM) devices and neuromorphic computing.

Nathaniel C. Cady received the B.A. and Ph.D. degrees from Cornell University, Ithaca, NY, USA. He is currently an Empire Inno- vation Professor in nanobioscience and the Interim Vice President of Research with the College of Nanoscale Science and Engineering, SUNY Poly- technic Institute. He has active research interests in the development of novel biosensor technolo- gies and biology-inspired nanoelectronics, including novel hardware for neuromorphic computing. He is also the Executive Director of the SUNY Applied Materials Research Institute (SAMRI) that funds collaborative research efforts between SUNY faculty and industry partner applied materials (AMAT).

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 (ACAD-Lab). 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, and accelerator design for deep neural networks and bioinformatics. He has authored and co-authored 120+ research papers in top-ranked journals and EDA conferences. He is the recipient of the Best Poster Award for Ph.D. Forum at IEEE/ACM DAC in 2018, two Best Paper Awards of IEEE ISVLSI in 2017 and 2018, and two Best Paper Awards of ACM GLSVLSI in 2019 and 2023.

Arman Roohi (SM'23) is currently an Associate Professor in the Department of Electrical and Computer Engineering, University of Illinois Chicago, Chicago, IL, USA. Before joining UIC, he was an assistant professor at the University of Nebraska-Lincoln (2020-2024) and a postdoctoral research fellow at the University of Texas at Austin (2019-2020). He received his Ph.D. in Computer Engineering at the University of Central Florida, Orlando, FL, USA, in 2019. His research interests span the areas of cross-layer co-design for implementing complex machine learning tasks and secure computation, including hardware security and the security of artificial intelligence, reconfigurable and adaptive computer architectures, and beyond CMOS computing. He has completed over 50 publications on these topics, including best paper recognition, book chapters, and STEM curricular development. He received Ph.D. Forum at DAC 2018 Scholarship, Frank Hubbard Engineering Endowed Scholarship in 2018, 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.