ViTSen: Bridging Vision Transformers and Edge Computing with Advanced In/Near-Sensor Processing

Sepehr Tabrizchi, Brendan C. Reidy, Deniz Najafi, Shaahin Angizi, Ramtin Zand, and Arman Roohi

***Abstract*—This paper introduces `ViTSen`, optimizing Vision Transformers (ViTs) for resource-constrained edge devices. It features an in-sensor image compression technique to reduce data conversion and transmission power costs effectively. Further, `ViTSen` incorporates a ReRAM array, allowing efficient near-sensor analog convolution. This integration, novel pixel reading, and peripheral circuitry decrease the reliance on analog buffers and converters, significantly lowering power consumption. To make `ViTSen` compatible, several established ViT algorithms have undergone quantization and channel reduction. Circuit-to-application co-simulation results show that `ViTSen` maintains accuracy comparable to a full-precision baseline across various data precisions, achieving an efficiency of ∼3.1 TOp/s/W.**

***Index Terms*—In-sensor processing, vision transformer, IoT.**

I. Introduction

DESPITE the rise of the Internet of Things (IoT), it still lacks inherent intelligence and heavily relies on cloud-based decision-making. Due to challenging issues such as high latency and security, IoT nodes should autonomously and efficiently process the detected data [1]. Edge intelligence has rapidly evolved into an integral part of modern technological ecosystems, fundamentally reshaping how data is processed and interacts with our digital environment. This evolution towards more efficient and localized computing paradigms is vividly exemplified in the workings of an image sensor. An image sensor converts light into electrical signals, which are then stored, processed, transmitted, and utilized. The procedure requires turning every individual pixel into a digital value with a fixed bit depth. It is remarkable that pixel conversion and retention account for over 96% of the power consumption in traditional vision sensors [2]. Moreover, note that one bit's communication energy (e.g., Bluetooth low energy (BLE) needs 1 nJ/bit [3]) is usually orders of magnitude higher than its computation energy (e.g., Multiply-Accumulate (MAC) $<$ 2 pJ/6-bit [4]). In recent years, several studies have been conducted regarding devising CMOS image sensors to accelerate compute-intensive artificial intelligence applications. The smart CMOS imagers can be classified into two types (1) near-sensor processing (NSP), which involves incorporating on-chip processors next to the imager for accelerating digital outputs of pixels [5]; and (2) in-Sensor Processing (ISP) platforms that process pre-Analog-to-Digital Converter (pre-ADC) data before transmitting it to the on/off-chip processor [6]. Due to resource limitations, NSPs and ISPs cannot host all layers of neural networks, yet they significantly contribute to the paradigm shift in low-power IoT image processing. By reducing power requirements, accelerating data transmission, and alleviating memory bottlenecks, they enable the deployment of Vision Transformers (ViTs) in energy-efficient devices. ViTs, while computationally intensive, excel in high-dimensional image analysis due to their self-attention mechanisms and ability to capture long-range dependencies. This extends their applicability from consumer electronics to industrial monitors for real-time, energy-efficient image analysis.

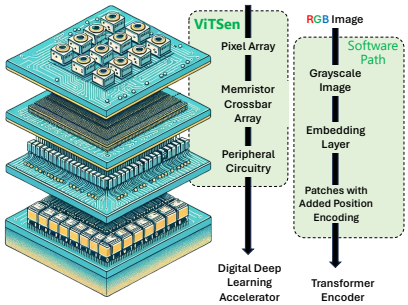


Fig. 1. High-level schematic of `ViTSen`.

This paper proposes the **`ViTSen`** architecture, leveraging advances in hardware and algorithms. The contributions include: (1) Introduce a novel in-sensor image compression technique converting RGB to grayscale to reduce power costs. (2) Develop a highly efficient parallel near-sensor analog convolution that integrates resistive memory to mitigate the overhead of the analog buffer and ADC converter. (3) Optimize pixel read and peripheral circuitry for improved power and delay reduction. (4) Adapt ViT algorithms with quantization and color channel reduction to align with **`ViTSen`** architecture. (5) Evaluation of system accuracy across different ViTs using various data precision and performance metrics. These steps significantly decrease data movement and computational demands while maintaining acceptable accuracy, making **`ViTSen`** a promising solution for intelligent IoTs.

II. Background

A. Vision Transformer (ViT)

Vision Transformers (ViT) represent a shift in computer vision from convolutional neural networks (CNNs) to architectures using self-attention mechanisms akin to transformers in natural language processing. Unlike CNNs, ViTs divide an image into fixed-sized patches and process them via transformer architectures. Given an image I of dimensions $H \times W \times C$, it is decomposed into N patches each of size $p \times p \times C$. For each patch p_i, a linear embedding is applied: $V_i = E \times \text{reshape}(p_i)$, where V_i denotes the embedded vector of patch p_i and E is the embedding matrix with dimensions $D \times (p^2 \times C)$. The architecture comprises layers of multi-headed self-attention and feed-forward neural networks, with positional embedding preserving spatial structure. ViTs excel in learning long-range dependencies and scaling across visual domains [7].

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

S. Tabrizchi and A. Roohi are with the Department of Electrical and Computer Engineering, University of Illinois Chicago, Chicago IL, USA. E-mail: aroohi@uic.edu.

B. Reidy and R. Zand are with the Department of Computer Science and Engineering, University of South Carolina, Columbia, SC, USA. E-mail: ramtin@cse.sc.edu.

D. Najafi and S. Angizi are with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, USA. E-mail: shaahin.angizi@njit.edu.

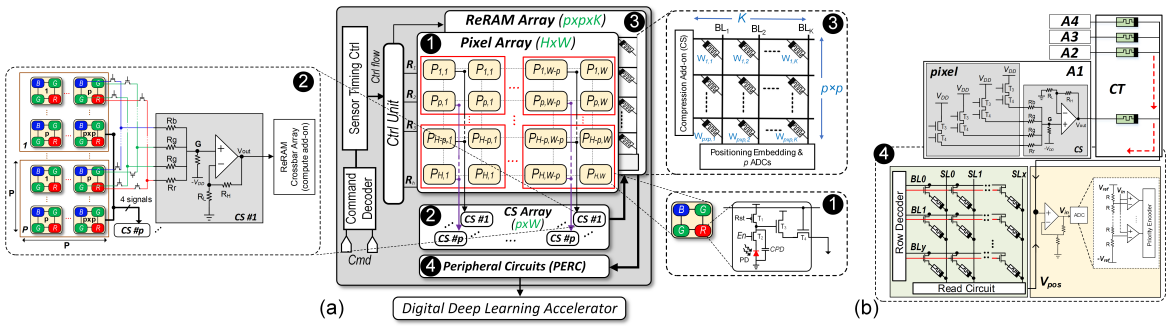


Fig. 2. (a) Overview of **ViTSen** illustrating its key components. (b) The PERC structure, including the analog positional encoding and a flash-ADC.

B. In-Sensor Processing (ISP)

Integrating computing with sensor arrays reduces off-chip data transmission and enhances power efficiency, achieved through near-sensor processing (NSP) and in-sensor processing (ISP) architectures. NSP processes data close to the sensor, minimizing off-chip transfer, whereas ISP incorporates processing within the sensor, enhancing efficiency by reducing data transmission and potential loss. Both architectures diminish data transfer and processing overhead, enhancing computing efficiency for diverse applications [8]–[12].

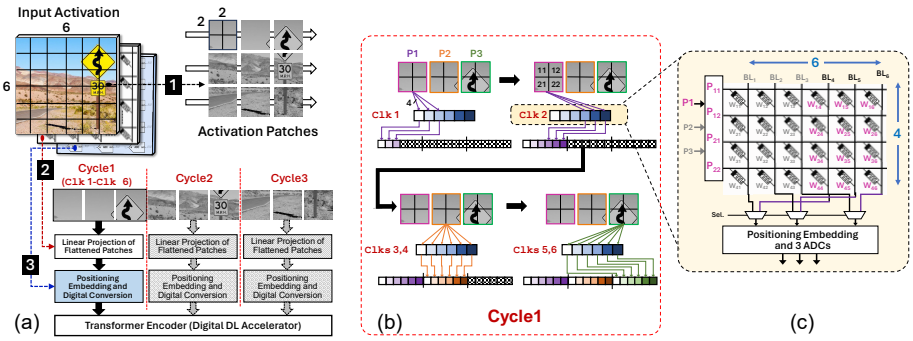
III. ViTSen Architecture

Integrating NSP/ISP with ViT models, as proposed in our **ViTSen**, marks a significant advancement in edge computing for vision systems. **ViTSen** architecture enables the preprocessing of raw image data at the sensor level, effectively reducing data transfer overhead and lessening the computational burden on ViT models. Such synergy allows **ViTSen** to adeptly handle initial image tasks like feature extraction, dimensionality reduction, and patch-based segmentation, aligning perfectly with the input needs of ViT models. Figure 2(a) depicts the architecture of **ViTSen**, consisting of four key components. **ViTSen** captures an image (1), performs in-sensor compression by converting RGB to grayscale (2) and near-sensor non-overlapping convolution (3). Finally, it applies positional embedding (4). This section details the proposed components individually and then integrates them into a unified system to form **ViTSen**. This innovative collaboration fosters a new paradigm of intelligent vision sensors capable of efficiently capturing and interpreting visual data, thereby reducing operation latency, conserving power, and ensuring high accuracy in vision tasks.

Pixel Array: The pixel array size is $H \times W$ indicating height and width dimensions, respectively, where $P_{ij}, i \in \{1, \ldots, H\}$ and $j \in \{1, \ldots, W\}$, shown in Fig. 2(a) (1). A pixel is responsible for acquiring the light intensity of the environment and converting it into an analog voltage. Each pixel comprises four different sensors that absorb the intensities of red, green ($\times 2$), and blue light. The pixel undergoes three phases: charge, evaluate, and read. In the charge phase, the capacitor C_{PD} is charged to V_{DD} through T_1. During the evaluate phase, the photodiode (PD) resistance varies with light intensity, discharging C_{PD} via T_2 and PD. In the read phase, T_3 generates a current based on the remaining voltage on C_{PD}, which is then connected to a compute add-on (CS) through T_4. Two common shutter techniques are global and rolling shutters. Global shutters require #ADCs = $H \times W$, with each pixel having its own ADC. Rolling shutters use one ADC per column, resulting in #ADCs = W. Global shutters offer higher frame rates, while rolling shutters are more power-efficient. **ViTSen** strikes a balance by connecting p rows of the pixel array to the CS components, where $p \times p$ is the transformer-defined patch size, reducing the number of required ADCs. Each column has p buses of different colors connecting pixels to the CSs, as shown in Fig. 2(a) (2).

Compression Add-on (CS): The second component is the CS array with $p \times W$ dimension, which is responsible for converting an RGB image to grayscale in an ISP manner. Figure 2(a) (2) shows the circuit of a CS and proper connectivities among the pixels. The standard equation for converting RGB images to grayscale is expressed by $0.299R + 0.587G + 0.114B$.This conversion necessitates multiplying each color value by a specific coefficient, in the digital domain. Conversely, in our proposed CS, this operation is achieved by utilizing different resistor values. According to the equation, the resistors should have values of $3.34 \times R$, $1.70 \times R$, and $8.77 \times R$ for the red, green, and blue channels, respectively. All sensors are connected through these resistors, followed by an opamp that amplifies the result value. It may cause transistors to turn off. The output of this step (V_{out}) will connect to the ReRAM array, which is elaborated next.

Computation Add-on (CT): The Computation add-on (CT) consists of a non-volatile resistive memory bank known as the ReRAM crossbar (Fig. 2(a) (3). In contrast to ReRAM crossbar arrays that typically use both digital-to-analog converter (DAC) and ADC, ViTSen utilizes integrated ADCs exclusively. ReRAM's resistance changes continuously based on the width of the write voltage. Therefore, we divide the resistance range of ReRAM into 16 (2^4) to 256 (2^8) points and find the appropriate voltage width for each. This writing process only needs to be carried out once. The ReRAM crossbar comprises the 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. In our previous study, we proposed a device-to-system level co-design approach to explore the theoretical limits of distinct weight levels through a large retention experiment. To reduce HRS variability, we employed a read-write-verify method to achieve resistance within a specific window. We captured 256 resistance levels, maintaining each for 10,000 seconds with reads every 100 seconds. The variability of each resistance level was analyzed, showing increased variability in the read current with higher resistance. texttt**ViTSen** uses



Parameters	Conventional Imager		ViTSen
	Global	Rolling	
#ADC	$H \times W \times 3$	$W \times 3$	p
#Clk	1	H	$K \times q^2/p$
Off-Chip Mem.	$H \times W \times 3$	$H \times W \times 3$	$K \times q^2$
On-Chip Mem.	0	0	$K \times p^2 + q^2$
ADC Usage	$H \times W \times 3$	$H \times W \times 3$	$K \times q^2$
Operations	0	0	$K \times p^2 \times q^2$

(d)

Fig. 3. (a) The required steps for a 6×6 RGB image. (b) Illustration of Cycle1, including parallel operations during 6 Clks. (c) The $(2 \times 2) \times 6$ ReRAM cells stores weights and connects to 3 MUXs that select columns properly for ADCs. (d) Performance results of different shutter schemes, where $q = H/p = W/p$.

256 high-confidence experimental resistance states as 8-bit memory states. To increase parallelism, the weights are divided into multiple memory banks, where each bank consists of an ADC to measure the final voltage independently. The size of our crossbar is $p \times p \times K$, i.e., $p \times p$ is 16×16, and K is the hidden size, shown in Fig. 2(a) (3). We limit the number of ADCs to p, i.e., 16, while each $p \times p$ weights are stored in one column of the memory bank. So, if $K > 16$, $\frac{K}{16}$ cycles are required to calculate the final results for one hidden size.

Positional Encoding and Readout Circuitry (PERC): PERC consists of two main elements: positional encoding and readout circuits, as depicted in Fig. 2(b) (4). The readout circuit is configured with $p \times p + 1$ inputs and incorporates an analog sense amplifier. Each input is interconnected via ReRAMs, resulting in the summation of their voltages, which is then amplified by the sense amplifier. Where $p \times p$ inputs are CT's outputs. The final input is designated for positional encoding produced by a $\frac{w}{p} \times \frac{h}{p}$ ReRAM cells. Unlike CT, this storage is the conventional ReRAM bank, allowing only one cell to be activated at any time. The resistors' varying values generate distinct offset voltages, functioning as positional embedding. To address the resistive issue affecting the path from the pixel array's V_{DD} to the ADCs, we analyze the worst-case and best-case scenarios. In the best-case scenario, pixel values are at maximum voltage and ReRAM resistance is at minimum, ensuring the highest input voltage for the ADC. Conversely, the lowest pixel values and highest ReRAM resistance give the ADC's minimum voltage. Determining these extremities allows us to establish optimal values for $\pm V_{\text{ref}}$, thereby correctly adjusting the ADC voltage.

Putting It All Together: To adapt ViTs for **ViTSen**, we modified them to start with a non-overlapping convolutional embedding layer, followed by positional encoding, exploring quantization and grayscale conversion to reduce embedding size. **ViTSen** operates sequentially through image capture, in-sensor compression, near-sensor analog convolution, positional encoding, and digital conversion. Figure 3 depicts an example of a 6×6 RGB image, with an embedding layer (K=6) and a patch (2×2). In step 1, the image is converted to grayscale using CS array, requiring $\frac{H}{p}$=$\frac{6}{2}$=3 cycles, activating p rows per cycle. Step 2 performs parallel analog convolution, requiring $\frac{K}{\#ADC} \times \frac{W}{p}$=$\frac{6}{3} \times \frac{6}{2}$=6 Clks per cycle. The convolution series for the first set of rows and pixel connections to the weight array in each patch are shown in Fig. 3(b) and (c), respectively. In step 3, positional encoding values are applied, and ADCs produce digital results. Figure 3(d) compares the computational and memory costs of conventional imagers using different shutter schemes. **ViTSen** uses fewer ADCs than rolling image sensors and only two on-chip storages for weights and positional encoding, enabling parallel analog computations.

TABLE I
TOP1 IMAGENET-1K ACCURACY FOR DIFFERENT QUANTIZATION AND COLOR SCHEMES.

	RGB				Grayscale			
ViT Model	FP	W8	W6	W4	FP	W8	W6	W4
ViT-B/16*	81.3	81.2	78.2	27.4	71.6	71.3	65.9	11.1
DeiT-T/16‡	68.6	67.1	57.0	17.1	56.0	52.8	36.7	6.0
DeiT-S/16†	78.0	77.4	71.8	32.9	68.7	67.1	57.6	16.2
DeiT-B/16*	82.2	82.3	79.6	55.3	76.2	75.8	70.0	33.2
DINOv2-S/14†	81.0	44.9	1.27	0.14	77.6	25.8	0.52	0.11
DINOv2-B/14*	84.3	73.9	3.63	0.23	82.1	61.6	1.08	0.20

*$K = 768$, ‡$K = 192$, †$K = 384$, where K is the hidden size.

IV. Evaluation Results

A. Framework & Methodology

The proposed evaluation framework employs a bottom-up methodology. At the device level, switching data and resistance levels from our fabricated ReRAM device are experimentally extracted for a Verilog-A model. At the circuit level, **ViTSen**'s array and computational peripheral circuitry are implemented in NCSU 45nm technology using HSPICE. The embedding layer's trained weights are quantized, while other transformer components are processed off-chip. A PyTorch model is used to train and extract the embedding-layer weights. At the architecture level, a Python-based behavioral model measures timing, energy, and area to offer flexibility in array configuration and peripheral circuitry design. After the embedding layer of the transformer, the results are captured, positional encoding values are added, and then the results are used in the Pytorch model to process the encoder-decoder structure and to report accuracy.

B. Accuracy

We compare the Top1 accuracy of popular ViTs on the ImageNet-1k validation dataset, including the original ViT [7], Data-efficient image Transformers (DeiT) [13], and DINOv2 (Self-Distilation with No Labels) [14]. Most models share the same base architecture with a resolution of 224×224, differing in patch size (16×16 and 14×14) and hidden size. The main difference lies in their training methods: ViT models (e.g., ViT-B/16) were pre-trained on ImageNet-21k and fine-tuned on ImageNet-1k; DeiT models were trained once on ImageNet-1k; and DINOv2 used self-supervised learning with a large teacher model (1 billion params) pre-trained on

TABLE II
PERFORMANCE COMPARISON OF VARIOUS NSP/ISP UNITS AND THE PROPOSED **ViTSen**.

Designs	Technology (nm)	Purpose	Precision	Comput. Scheme	Memory	NVM	Pixel Size (μm^2)	Array Size	Power (mW)	Efficiency (TOp/s/W)
[9]	60/90	STP†	NA	row-wise	Yes	No	3.5×3.5	1296×976	230-363	0.386
[11]	180	1^{st}-layer BNN	1-bit	entire-array	Yes	No	110×110	32×32	0.0121	1.32
[5]	180	edge*/TMF‡	NA	row-wise	Yes	No	32.6×32.6	256×256	1230	0.535
[10]	65	1^{st}-layer BNN	1-bit	entire-array	Yes	Yes	55×55	128×128	0.0088-0.025	1.745
[6]	180	1^{st}-layer CNN	8-bit	entire-array	No	No	10×10	128×128	0.45 - 1.83	1.41 - 3.37
[12]	45	1^{st}-layer CNN	2-bit	entire-array	Yes	Yes	38×38	32×32	0.00096 - 0.0028	1.37 - 4.12
ViTSen	45	Embedding + Positional encoding	8-bit	multiple rows***	Yes	Yes	38×38	224×224	1370	3.1

†Spatial Temporal Processing. ‡Thresholding Median Filter. ** Region Of Interest. ***The number of rows is defined by the patch's height.

ImageNet-22k, followed by fine-tuning the student models on ImageNet-1k. To reduce the size of the embedding layer, we investigate the effects of quantization as well as the effects of collapsing the color channel, i.e., making the images grayscale. For our experiments, we use post-training quantization with 100 calibration images. Since the inputs will be analog, we leave the input of the model in full precision; we quantize the output to 8 bits and the weights to 8 bits, 6 bits, and 4 bits. We report our findings in Table I. The table shows a small accuracy drop from the baseline full-precision model to the 8-bit quantized model for all cases except DINOv2, which becomes unusable below 8 bits. For 6-bit quantized models, tiny models have the sharpest accuracy decrease, while base models drop $\sim$3%. Extreme quantization (4-bit) causes significant accuracy degradation for all models. Quantization-aware training could mitigate accuracy loss in these cases. Grayscale images follow a similar trend, with accuracy decreasing more quickly. Comparing grayscale to RGB images, supervised learning models show a 7%-12% accuracy drop in grayscale, while DINOv2 experiences a 3.5%-2.2% drop. This is likely due to DINOv2's semi-supervised pre-training for feature extraction, resulting in a more generalized model compared to those trained in a supervised manner for a specific task.

C. Performance Evaluation

Table II 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. To evaluate **ViTSen**'s performance, we chose the DeiT-T16 model, which has an input size of 224×224, $K = 192$, and a patch size of 16×16, resulting 5M parameters in total. Due to the **ViTSen** features and capability, roughly 19.5M operations are performed close to the sensor, where data resides in the analog domain. However, to ensure a fair comparison, we estimated the power consumption and performance of computing units when performing the same task. Our observations are summarized below: (i) The only designs that support a fully parallel and entire-array computation scheme are the accelerators mentioned in [10]–[12]; (ii) Except the design in [6], **ViTSen** and all the others incorporate integrated memory components. Among these designs, our design and [10], [12] are the only ones that utilize NVMs, realizing intermittent and instant computing paradigms. Moreover, thanks to the proposed multi-level ReRAM, **ViTSen** supports up to 8-bit (4, 6, and 8 - bit) weight precision; (iii) In terms of power consumption, **ViTSen** shows the worst results due to the large number of hidden sizes required by the DeiT-T16 model yet still meeting the IoT power budget; (iv) Regarding efficiency, the CNN accelerator with 8-bit support mentioned in [6] achieves a rate of 3.37 TOp/s/W. On the other hand, AppCiP [12] with 2-bit weight precision achieves a higher efficiency of 4.12 TOp/s/W. Although our design consumes a relatively large amount of power, it achieves a comparable computation efficiency ($\sim$3.1 TOp/s/W); (v) In almost all of the examined ViT models, **ViTSen** compressed the ADCs' outputs, reducing the required bandwidth and storage.

V. CONCLUSION

The paper presented an efficient integrated sensing and computing engine called **ViTSen**, incorporating a novel software/hardware co-design approach. **ViTSen** enables in-sensor compression and near-sensor highly parallel analog processing, facilitating a low-precision quantized weight embedding layer while optimizing the need for ADCs. By integrating these steps for enhanced computation parallelism to decrease data movement and computational demands, the system achieves comparable accuracy to a full-precision baseline for object classification tasks with an efficiency of 3.1 TOp/s/W.

REFERENCES

[1] T.-H. Hsu *et al.*, "AI edge devices using computing-in-memory and processing-in-sensor: from system to device," in *IEDM*, 2019.

[2] J. Choi *et al.*, "An energy/illumination-adaptive cmos image sensor with reconfigurable modes of operations," *IEEE JSCC*, vol. 50, no. 6, pp. 1438–1450, 2015.

[3] S. Sen, "Context-aware energy-efficient communication for iot sensor nodes," in *2016 53nd DAC*. IEEE, 2016, pp. 1–6.

[4] T. Bouguera *et al.*, "Energy consumption model for sensor nodes based on lora and lorawan," *Sensors*, vol. 18, no. 7, p. 2104, 2018.

[5] S. J. Carey *et al.*, "A 100,000 fps vision sensor with embedded 535gops/w 256× 256 simd processor array," in *Symposium on VLSI*. IEEE, 2013.

[6] R. Song *et al.*, "A reconfigurable convolution-in-pixel cmos image sensor architecture," *IEEE TCSVT*, 2022.

[7] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[8] A. Roohi *et al.*, "Pipsim: A behavior-level modeling tool for cnn processing-in-pixel accelerators," *IEEE TCAD*, 2023.

[9] T. Yamazaki *et al.*, "4.9 a 1ms high-speed vision chip with 3d-stacked 140gops column-parallel pes for spatio-temporal image processing," in *ISSCC*. IEEE, 2017, pp. 82–83.

[10] S. Angizi *et al.*, "Pisa: A non-volatile processing-in-sensor accelerator for imaging systems," *IEEE Transactions on Emerging Topics in Computing*, 2023.

[11] H. Xu *et al.*, "Macsen: A processing-in-sensor architecture integrating mac operations into image sensor for ultra-low-power bnn-based intelligent visual perception," *IEEE TCAS II*, vol. 68, pp. 627–631, 2020.

[12] S. Tabrizchi *et al.*, "Appcip: Energy-efficient approximate convolution-in-pixel scheme for neural network acceleration," *IEEE JETCAS*, vol. 13, pp. 225–236, 2023.

[13] H. Touvron *et al.*, "Deit iii: Revenge of the vit," in *European Conference on Computer Vision*. Springer, 2022, pp. 516–533.

[14] M. Oquab *et al.*, "Dinov2: Learning robust visual features without supervision," *arXiv preprint arXiv:2304.07193*, 2023.