Invited Paper: Learned In-Sensor Visual Computing: From Compression to Eventification

Yu Feng*‡, Tianrui Ma†§, Adith Boloor†¶, Yuhao Zhu*‖, Xuan Zhang†**,
*Department of Computer Science, University of Rochester, USA
†Department of Electrical and Systems Engineering, Washington University in St. Louis, USA
Email: ‡yfeng28@ur.rochester.edu, §tianrui.ma@wustl.edu, ¶adith@wustl.edu,
‖yzhu@rochester.edu, **xuan.zhang@wustl.edu,

***Abstract*—Visual computing is vital for numerous applications. In conventional visual computing systems, CMOS image sensors (CIS) act as pure imaging devices for capturing images, however, recent CIS designs increasingly integrate processing capabilities such as Deep Neural Networks (DNN), which give rise to a notion of in-sensor computing. In this paper, we propose a new concept, learned in-sensor visual computing, which exploits end-to-end optimization of in-sensor processing and downstream vision tasks to achieve better overall algorithm accuracy and adopts hardware/algorithm co-design to achieve ultra-low sensor energy consumption. Two examples of the learned in-sensor visual computing, LECA and EDGAZE, are demonstrated.**

I. INTRODUCTION

Visual computing applications on the horizon such as autonomous machines, computational photography, and space exploration all fundamentally rely on image sensing. These applications incur heavy data traffic between image sensors and off-sensor processors, causing significant energy and latency overhead. To alleviate these overheads, recent CMOS image sensors (CIS) expand the capability of image capturing and increasingly integrate *computation* functions, ranging from classic signal pre-processing [9] to Deep Neural Networks (DNN) [5], [18] and spanning over both the analog domain and the digital domain. For instance, a Nikon CIS [18] integrates an image processor for per-tile exposure control; the Sony IMX 500 CIS [12] integrates a near-pixel DNN accelerator for edge visual processing. Such recent CIS designs enable CIS to consume large volumes of pixel data *in-situ* so that the CIS only transmit low-dimensional processed data, thereby significantly reducing the communication overhead. This technique is known as *in-sensor computing*.

In-sensor computing is not a new concept and can be traced back to the early 2010s. Prior works can be chronologically categorized into three: classical descriptor extraction, heuristic compression/decompression, and compressive sensing (CS). The classical descriptor extraction converts a raw image to low-dimensional descriptors, such as Haar [4] and HOG [30], thus the communication overhead is reduced and the off-sensor processor can directly utilize the sensor outputs. However, this method is not directly compatible with many deep learning (DL)-based downstream algorithms. Heuristic compression/decompression and CS, on the other hand, are fundamentally designed for image restoration and thus in theory can work with different downstream algorithms. Nonetheless, constrained by the limited computation that can be implemented in the CIS, they tend to include simple operations such as encoding the neighboring pixel's intensities [41], encoding a block of pixels based on its mean, gradient, and bitmap [8], perturbing pixels to achieve low-resolution quantization [40], encoding pixel gradient to logarithmic representation [39], and skipping pixels with small accumulated gradients [21]. Therefore, the algorithm accuracy of the heuristic methods largely varies among different vision tasks, due to the dependence on manually-chose parameters. Compressive sensing exploits the sparsity of natural images and allows the raw images to be progressively reconstructed with a small number of linear measurements. By leveraging this insight, CS samples fewer signals to save data communication during the pixel readout and has been widely used in applications such as image/video compression and restoration [34]. However, CS typically introduces non-trivial computation overheads due to its use of an iterative optimization method for image reconstruction.

With the advance of artificial intelligence (AI), emerging visual applications become more AI-driven, and prior techniques are no longer sufficient for the next-generation edge-AI applications for two main reasons. From the hardware perspective, the general trend of today's computing is shifting towards domain-specific designs due to the slowdown of Moore's law, mirroring the trajectory of CIS designs. However, prior in-sensor processing techniques are often general-purpose and fail to meet the goal of domain-specific CIS designs. Instead, in-sensor techniques should be also domain-specific to better accommodate today's CIS and downstream tasks. From the algorithm perspective, today's state-of-the-art visual algorithms are largely driven by DL-based methods. To achieve better end-to-end performance, in-sensor computing systems should expand the optimization scope and co-optimize the downstream visual algorithms as well.

To this end, we propose a new concept, learned in-sensor visual computing, which co-optimizes the in-sensor computing algorithm/hardware with the downstream DL algorithms, by fusing task-specific insights into CIS designs. The task-specific insights stem from both domain knowledge and end-to-end optimization of the visual system. By combining these two, in-sensor systems can compress the visual data with the optimal ratio, largely saving the hardware resources while

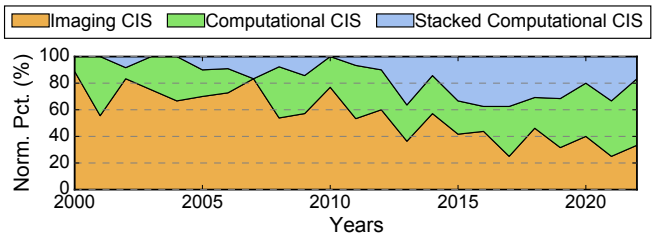


Fig. 1: Percentage of conventional CIS, computational CIS, and stacked computational CIS designs from surveying all ISSCC and IEDM papers published between Year 2000 and 2022. Increasingly more CIS designs are computational.

preserving the vision task accuracy. This paper serves as a guide to learned in-sensor visual computing by proposing a generic framework for learned in-sensor computing and presenting two concrete examples of this concept: learned compressive acquisition (LECA) and learned ROI-based eye-tracking (EDGAZE).

In summary, this paper makes the following contributions:

- We provide a comprehensive survey of the design and scaling trends of CIS, pointing out new opportunities for in-sensor processing.
- We propose the idea, learned in-sensor visual computing, and provide a generic framework to realize our idea.
- We present two concrete examples to guide sensor architects and visual algorithm designers to design their in-sensor visual computing systems.

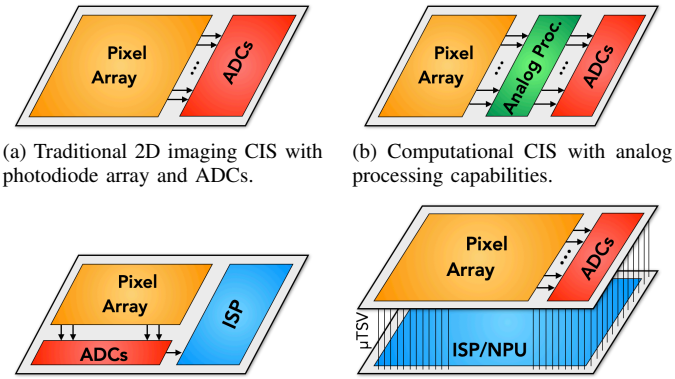
II. MOTIVATION: GLODEN ERA OF SENSOR DESIGN

Recent success in mobile visual computing cannot live without the effort in recent CIS advancement. In this section, we first discuss the main design trend of CIS that underlies this paper: CIS are becoming increasingly computational, moving from human-centric perception to machine-centric perception (Sec. II-A). We then explain the energy benefits of such computational CIS for in-sensor visual computing (Sec. II-B). Finally, we discuss the challenges of reaping the energy benefits, which motivates this work (Sec. II-C).

A. Design Trends

CIS Primer. Fundamentally, a CIS consists of two basic components as illustrated in Fig. 2a: a light-sensitive photodiode array that converts photons to voltages and a readout circuit array that converts voltages to digital values (i.e., raw pixels) through the analog-to-digital converters (ADC). Traditionally, raw pixels are transferred to the host, e.g., a Systems-on-a-Chip (SoC) on a smartphone, through the MIPI CSI-2 interface [17]. A fixed-functional Image Signal Processor (ISP) in the SoC removes sensing artifacts (e.g., denoising) and prepares pixels for the *human-centric perception* (e.g., visual display).

CIS Design Trend. In the past 20 years, a clear trend in edge visual computing is that more mobile devices move from *human-centric perception* to *machine-centric perception*.



(a) Traditional 2D imaging CIS with photodiode array and ADCs.

(b) Computational CIS with analog processing capabilities.

(c) Computational CIS with digital accelerator (ISP here).

(d) Stacked computational CIS with digital accelerators in a separate layer.

Fig. 2: CIS architecture evolution. CIS is moving away from a purely imaging device (a) to integrate computation capabilities (b)(c), sometimes in a 3D stacking fashion (d).

This trend motivates CIS design to move computations into the sensor itself, which gives rise to the notion of *Computational CIS*. Fig. 1 shows the percentage of computational CIS papers in ISSCC and IEDM from Year 2000 to Year 2022 with respect to all the CIS papers during the same time range. Increasingly more CIS designs integrate compute capabilities.

The computations inside a CIS could take place in both the analog and the digital domain. Fig. 2b and Fig. 2c illustrate examples where analog computing and digital computing are integrated into a CIS chip, respectively. Early works primarily implement general-purpose feature descriptors for conventional vision tasks [5], [4], [39]. With recent AI innovations, computational CIS gradually integrate computations for DNN processing [19], [38], [12], [29], moving towards the task-specific *machine-centric perception*.

As edge-AI requires complex processing capabilities, CIS design has embraced 3D stacking technologies, as is evident by the increasing number of stacked CIS in Fig. 1. Fig. 2d illustrates a typical stacked design, where the processing logic is separated from, and stacked with, the pixel array layer. The different layers communicate through hybrid bond or micro Through-Silicon Via (μTSV) [27], [37]. The processing layer typically integrates digital processors, such as ISP [24] and DNN accelerator [12]. Such designs further advance the sensor development towards *machine-centric perception*.

B. Benefits of Computational CIS

It is no coincidence that computational CIS emerge when energy efficiency is critical. From an architecture perspective, computational CIS provides two main energy benefits. First, moving computation inside the sensor allows the pixel data to be consumed closer to where they are generated. Doing so reduces the data transmission energy, which could dominate the overall energy consumption.

Specifically, data communication inside a CIS using a μTSV consumes about 1 pJ/B, whereas the energy cost of transmitting one Byte out of the CIS through the MIPI CSI-

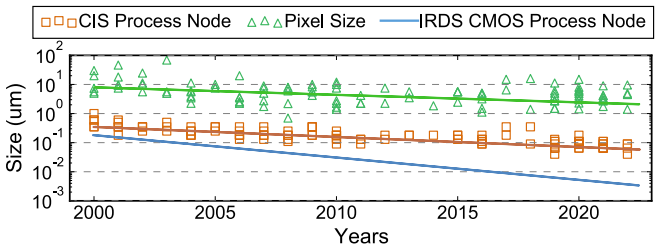


Fig. 3: CIS process node always lags behind conventional CMOS process node. This is because CIS node scaling tracks the pixel size scaling, which does not shrink aggressively due to the fundamental need of maintaining photon sensitivity.

2 interface consumes about 100 pJ of energy [27]. As an example, if a CIS is capable of executing an object detection DNN directly, the data volume that has to be transmitted out of the sensor is simply a few Bytes (object location and label), as opposed to, say, 6 MB, for a 1080p image.

Second, computational CIS also provides a natural platform for analog acceleration, since the pixel data originate from the analog domain to begin with, obviating the need for energy-intensive digital-to-analog converters that often dominate the hardware overheads in conventional analog accelerators. Compare to digital processing, analog processing minimizes energy-intensive data conversion [30], [7] and can reduce both the computation and memory energy consumption.

C. Challenges for In-Sensor Computing

Moving computation inside a CIS, however, does not without challenges. Compared to a mobile SoC, the computational resources inside CIS are far more stringent. The chip area of today's mainstream CIS is ≤ 25 mm^2 as posed to 400 mm^2 of a mobile SoC [36], [35]. Such a resource constraint limits the in-sensor processing capability. Meanwhile, starting at Year 2010 when the AI explodes, the computation cost of AI-related applications increases exponentially [32]. Therefore, it is infeasible to simply deploy an entire visual model or the first several layers into an image sensor.

On the other hand, the in-sensor processing is far less efficient than that the off-sensor processing, fundamentally because the CIS process node significantly lags behind that of the conventional CMOS. Fig. 3 illustrates this difference, where square markers show the process nodes used in CIS designs from all ISSCC papers appeared during Year 2000 and Year 2022, which include leading industry CIS designs at different times. We overlay a trend line regressed from these CIS designs to better illustrate the scaling trend. As a comparison, the blue line at the bottom represents the conventional CMOS technology node scaling laid out by International Roadmap for Devices and Systems (IRDS) [1].

At around Year 2000, the CIS process node started lagging behind that of the conventional CMOS node, and the gap is increasing. CIS design today commonly use 65nm and older process nodes. This gap is not an artifact of the CIS designs we pick; it is fundamental: there is simply no need to aggressively scale down the process node because the pixel size does not shrink much. The triangles in Fig. 3 represent the pixel sizes of all the CIS designs we surveyed. The slope of CIS process node scaling almost follows exactly that of the pixel size scaling. The reason that pixel size does not shrink is to ensure light sensitivity: a small pixel reduces the number of photons it can collect, which directly reduces dynamic range and signal-to-noise ratio (SNR) [3].

These two challenges motivate us to explore a viable framework for in-sensor visual computing in a systematic and principled manner.

III. Framework

No framework to date depicts the computing paradigm of in-sensor visual computing, which can guide the algorithm designers to better leverage in-sensor computation power. To fill this void, we first outline an in-sensor computing paradigm and explain why current visual algorithms fail to meet such requirements (Sec. III-A). Next, we explain the importance of hardware-aware training for in-sensor visual computing (Sec. III-B).

A. In-Sensor Computing Paradigm

An ideal in-sensor algorithm should fully exploit the computational power inside CIS while being aware of the limitations mentioned in Sec. II-C, namely limited computational resources and lagging-behind sensor processing nodes. Therefore, the ideal algorithms that are implemented in-sensor should only consume a lightweight computation inside CIS yet drastically reduce the data communication between the CIS and the off-sensor processor. Fig. 4 shows the data volume changes of four representative algorithms during the course of execution. If we split any of these four algorithms into two halves and deploy the first half inside CIS, only the last example of Fig. 4 fits the ideal in-sensor computing paradigm. In the last example, the first small portion of computation deployed inside CIS does not overwhelm the resources within CIS and largely reduces the data transmission so that the rest of the computation can be offloaded to the off-sensor processor. Meanwhile, the trivial amount of computation inside the sensor also does not overturn the potential energy savings from data communication reduction.

On the other hand, existing visual algorithms are ill-suited for in-sensor computing as shown in the first three examples of Fig. 4. These three examples represent the computing paradigms of widely-used visual algorithms: image processing, convolutional neural network (CNN), and encoder-decoder-like network. Compared to the ideal in-sensor paradigm, image processing retains roughly the same data volume at each processing stage. Any kind of splitting of an image processing pipeline does not result in data communication reduction. As for CNN, its data volume only shrinks at the very end, e.g. predicted data labels, requiring unrealistic computation demands inside CIS. The same situation applies to the encoder-decoder-like network as well.

To conclude, leveraging full-fledged in-sensor power inevitably requires us to rethink the visual algorithms. In Sec. IV,

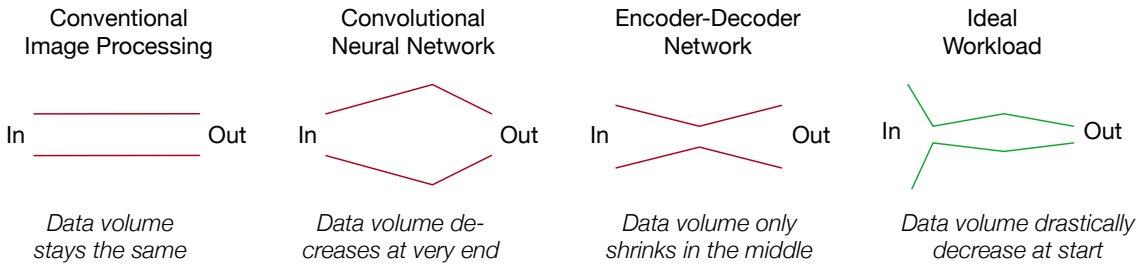


Fig. 4: Examples of representative computing paradigms in today's visual algorithms. The first three, representing conventional image processing, CNN, and encoder-decoder network, are ill-suited for in-sensor visual computing. Conventional image processing has no data transmission savings, while CNNs and encoder-decoder networks require significant computation to achieve communication savings. Only the last one requires a trivial in-sensor computation and drastically reduces the data communication.

two examples are presented to demonstrate how to design an algorithm for in-sensor computing.

B. In-Sensor Training Procedure

To better enhance *machine-centric perception*, the in-sensor training procedure should be both task-specific and hardware-aware. To do so, the task-specific training should optimize the downstream task accuracy rather than the image reconstruction quality. Particularly, our approach of being task-specific is that all parameters in the in-sensor computing algorithms are co-trained simultaneously with the downstream visual algorithms to maximize its end-to-end task accuracy, in contrast to prior works that train to minimize the reconstruction loss between the original and decoded images [28].

Making training procedure to be hardware-aware guarantees that accuracy is maintained after hardware implementation. Such a training procedure should explicitly consider multiple hardware non-idealities in the forward training pass, especially when the computation is implemented in the analog domain. These hardware non-idealities include hardware constraints (e.g. limited signal range, limited precision, and constrained polarity), hardware offsets, and hardware noises and variations.

For the hardware constraints, as an example, the data numerical values in the model should be clamped to be consistent with the real signal range in the hardware, and the model's weight precision should be quantized to the hardware precision. For the hardware offsets, two different types of circuits should be considered respectively: for the circuits with fixed transfer functions (e.g., buffers), we approximate them with analytical regression functions and insert them in the training forward path; for the circuits with programmable transfer functions (e.g., switched-capacitor multiplier), we incorporate the programmable circuit parameters in the training loop by inserting the exact circuit behavior models in the training forward path. For the hardware noises and variations, we specifically model the noise at each circuit stage and insert them into the training forward path stage by stage.

IV. Application Examples

In this section, two examples are given to showcase how to co-design CIS with in-sensor algorithms. Particularly,

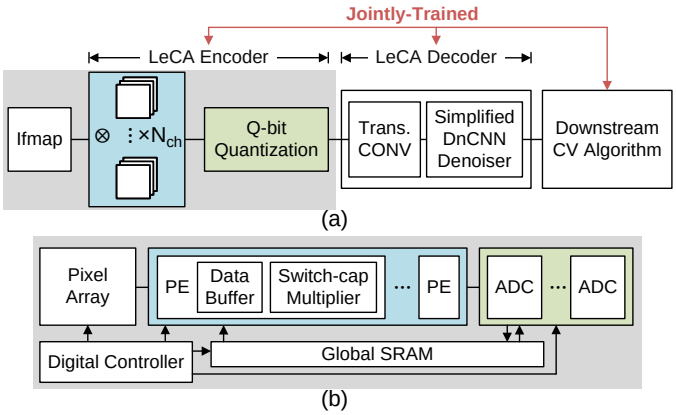


Fig. 5: (a) LeCA vision processing pipeline. (b) LeCA sensor system diagram. LeCA encoder is implemented by PE and quantization is implemented by ADC.

Sec. IV-A presents LeCA which co-optimizes an autoencoder with the downstream DNN models and co-designs the sensor hardware with the autoencoder. Sec. IV-B shows customized in-sensor hardware for an in-sensor algorithm to further improve energy efficiency in eye tracking.

A. LeCA

Application LeCA (learned compressive acquisition) [29] is a machine vision sensor for DL-based downstream algorithms. As illustrated in Fig. 5, LeCA consists of two synergistically-optimized components – an encoder/decoder model that is jointly trained with a backbone DNN for the downstream tasks; and an in-sensor processing architecture that efficiently implements the encoder in the sensor. On the algorithmic side, LeCA adopts an encoder-decoder structure and stacks it before the backbone DNN. The encoder performs a single-layer convolution between the raw RGB image and the encoder's learned kernels. The convolution output is then quantized to a low-resolution feature map, whose bit-depth is allowed to vary between 1.5-bit (ternary) and 4-bit. The decoder reconstructs task-specific features from the encoded feature map to the same size of original image. Both the LeCA encoder and decoder take the form of convolution layers and

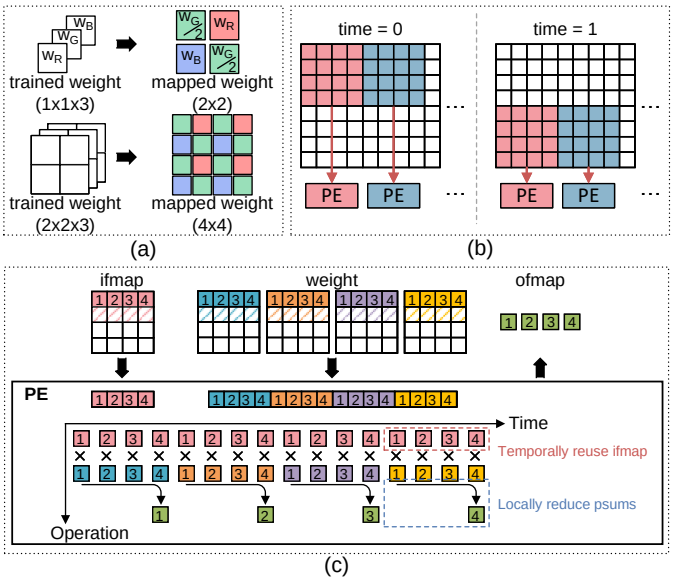


Fig. 6: (a) Kernel flattening. (b) Column-parallel processing. (c) A 4×4 example showing processing dataflow in one PE.

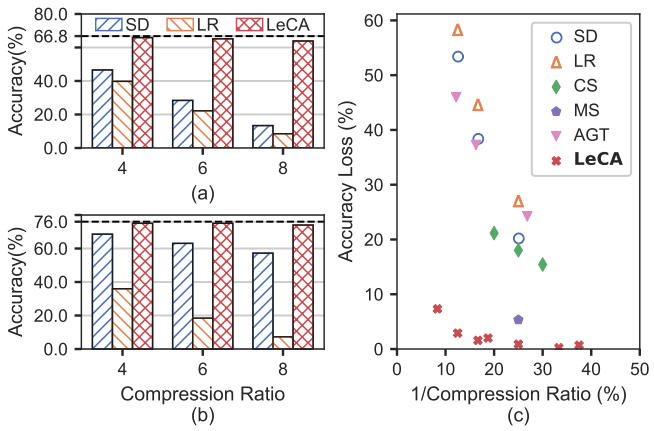


Fig. 7: Downstream classification accuracy comparison on (a) TinyImageNet (ResNet18) and (b) ImageNet (ResNet50) for CR ∈ {4, 6, 8} with SD, LR and LECA. (c) Accuracy loss vs. compression tradeoff based on TinyImageNet (ResNet18).

are jointly-trained with the downstream DNN so that all design parameters are aware of the downstream task accuracy.

Hardware Design We design a novel sensor hardware (Fig. 5(b)) to embed the computation of LECA encoder in the sensor, and implement the computation with analog-domain processing element (PE) array to achieve significant image data compression with high energy efficiency. The sensor architecture comprises five parts: The pixel array contains column-parallel analog pixel readout circuits with row-wise rolling shutter exposure. The PE array receives analog pixel values from the pixel array as input feature maps (ifmap), fetches digital LECA encoder kernels from the global SRAM as weights, and generates analog output feature maps (ofmap) through charge-domain multiply-accumulate (MAC) operations. The ADC array performs digital quantization on the analog ofmap and its resolution is variable from 1.5-bit to 4-bit. The quantized ofmap is stored back into the global SRAM to be transmitted off-chip. The digital controller cooperates with the row scanner, to control data scheduling and operation timing from the start of the exposure to the final readout of the quantized ofmap.

LECA sensor is designed with a pixel array size of 448×448 with the Bayer pattern filter where the green pixel is duplicated. This means that LECA sensor captures a full frame of 224×224×3 color image in which 3 stands for the RGB color channels. Note that the LECA encoder is trained on RGB images. To map each kernel (2×2×3) of the encoder to the kernel on raw images, the trained weights of the green color channel is halved and duplicated, effectively flattening the 2×2×3 convolutional kernel to 4×4, as illustrated in Fig. 6(a). Thus the 448×448 pixel array requires 112 identical PEs to perform column-parallel processing, as each 4 columns sharing one exclusive PE to process the non-overlapping 4×4 pixel block, as shown in Fig. 6(b).

To illustrate the processing dataflow, Fig. 6(c) is a toy example demonstrating how each PE processes a 4×4 pixel block. Here, N_{ch} is 4, thus 4 ofmap elements are generated. Bias in the convolution is ignored here for simplicity. To reduce analog data movement, LECA sensor adopts an input-stationary dataflow: the ifmap is temporally reused and the psum is locally reduced. In the beginning, the 1st row in the ifmap and each weight is buffered in the PE. During the PE processing, 16 MAC operations are sequentially performed by loading the $\texttt{ifmap}_{1,2,3,4}$ cyclically and loading the $\texttt{weight}_{1,2,3,4}$ in kernel 1 to kernel 4 consecutively. The psum generated from every MAC operation is reduced locally: during the MAC operations with $\texttt{ifmap}_{1,2,3,4}$ and $\texttt{weight}_{1,2,3,4}$ in kernel 1, the 4 psums are reduced to $\texttt{psum}_1$; the same process applies to $\texttt{psum}_{2,3,4}$. After 16 MAC operations, $\texttt{psum}_{1,2,3,4}$ are generated and buffered. Then, the 2nd row in the ifmap and each weight is buffered and processed, and the newly generated $\texttt{psum}_{1,2,3,4}$ are accumulated to the previously buffered $\texttt{psum}_{1,2,3,4}$. After processing the 4th row of the ifmap and the weight, 64 MAC operations are totally performed and the $\texttt{ofmap}_{1,2,3,4}$ are generated and popped out of the buffer.

Accuracy-optimized Compression LECA has the ability to retain high downstream accuracy at high compression ratios due to its ability to jointly remove redundancy across the spatial domain, color domain, and bit-depth resolution. In our baselines, spatial down-sampling (SD) and low-resolution quantizer (LR) are typical methods to remove the redundancy in the spatial domain and bit-depth resolution, respectively. In Fig. 7(a) and (b), we compare LECA with SD and LR on ResNet18 and ResNet50 for TinyImageNet and ImageNet, respectively. For SD validation, we use a 2×2, 2×3, and 2×4 average pooling kernel with corresponding up-sampling through bilinear interpolation to acquire compression ratios of 4, 6, and 8, respectively. For LR validation, we perform 3-bit, 1.5-bit (ternary), and 1-bit quantization to achieve compression ratios of 4, 6, and 8 respectively. LECA outperforms its prede-

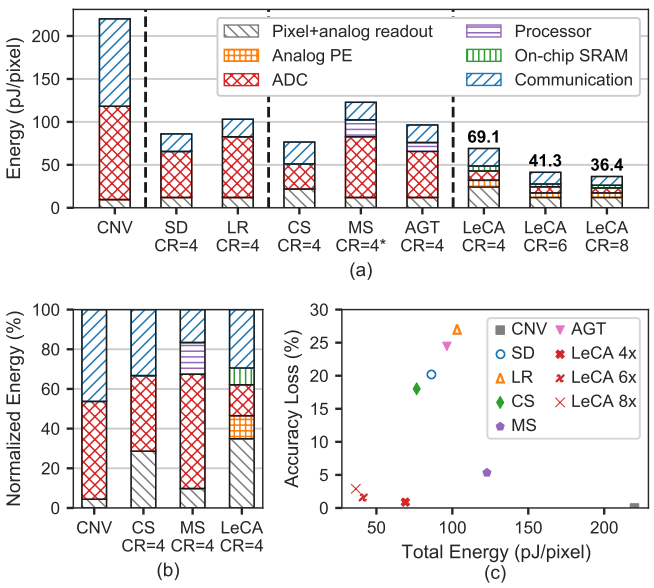


Fig. 8: Energy and accuracy comparison between conventional, compressive, and LECA sensors: (a) absolute energy; (b) relative energy normalized to LECA (CR=4); (c) tradeoff between sensor energy consumption and accuracy loss on the proxy pipeline. *-MS's compression is image dependent, varying between 4× and 5×.

cessors in all three compression ratio categories. LECA attains accuracies of 75.05%, 75.04% and 74.01% for 4×, 6×, and 8× compression, respectively, which translate to 0.97%, 0.98%, and 2.01% accuracy loss with respect to the baseline accuracy of 76.02%. An observation is that LECA overall loses less accuracy on ImageNet, than on TinyImageNet, especially when performing aggressive compression. We hypothesize that this is because ImageNet's larger image sizes (224×224 as compared to 64×64) allows LECA to generate larger encoded images which contain more information.

Fig. 7(c) shows a more thorough comparison of LECA with its counterparts on ResNet18 for TinyImageNet. It shows the compression ratio of LECA can be flexibly changed over a large ratio range by adjusting N_{ch} and Q_{bit}. It also shows that LECA outperforms all the baselines. At a compression of 25% (CR = 4), MS [40] and CS [34] have an accuracy loss of 5.3% and 18% respectively, whereas LECA loses $< 1\%$ accuracy, highlighting the advantage of LECA's task-specific training. A common trend seen is that aggressive compression leads to higher accuracy loss. This is because all models perform lossy compression, meaning that increasing information is irrevocably lost with higher compression.

Hardware Evaluation As shown in Fig. 8(a), LECA sensor achieves extremely-low energy consumption. Compared to the conventional image sensor (CNV), the energy of ADC and communication in LECA sensor (CR = 4) is dramatically reduced by 10.1× and 5×, respectively, due to analog domain image compression and low-resolution ADC. Comparing to the CIS with SD and LR compression techniques under the

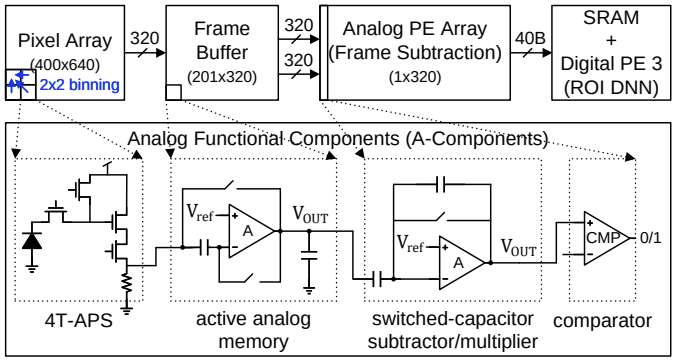


Fig. 9: Mixed-signal CIS design for EDGAZE, which implements the three key operations: 2×2 binning, frame subtraction and ROI DNN.

same compression ratio, the energy of ADC in LECA sensor (CR = 4) is still reduced by 5× and 6.6× because SD only has compression in spatial domain while LR only in bit-depth domain. Comparing to the CIS with learned compression techniques (CS, MS, and AGT [21]) under the same compression ratio, LECA sensor consumes 11%, 57%, and 31% less energy, respectively. Fig. 8(b) shows the normalized energy breakdown of CNV, MS, CS, and LECA. For CS, excessive energy is consumed by ADC due to the requirement on high quantization resolution in CS algorithms [11]. For MS, since it is implemented in the digital domain, pixel-wise A/D conversion consumes excessive energy, even though the quantization resolution is as low as 2-bit. In LECA (CR = 4), neither analog PE nor ADC is the energy bottleneck. LECA (CR = 6) and LECA (CR = 8) gain more energy savings from non-repetitive pixel readout and less off-chip communication. Specifically, LECA (CR = 8) is 6.3× and 2.2× more energy efficient than CNV and CS, respectively. Fig. 8(c) shows the tradeoff between the sensor chip energy and the downstream task accuracy. In line with expectations, lower energy is gained in exchange of higher accuracy loss. However, LECA defines the optimal Pareto frontier by achieving extremely-low energy consumption while maintaining the lowest accuracy loss.

B. EDGAZE

Application Eye tracking is critical in many fields such as medical operations, human-machine interface, and augmented/virtual reality (AR/VR). For those applications, performance and speed both are important to achieve a satisfactory user experience. EDGAZE [14], which is a customized real-time eye-tracking algorithm, leverages the in-sensor computing capability to achieve supreme runtime performance without any gaze estimation accuracy compromise.

The crux of EDGAZE is to generate a small region of interest (ROI) inside the sensor via in-sensor computing, thus, avoiding data transmission of the original full-resolution image. Fig. 9 shows three key operations in EDGAZE ROI prediction. In EDGAZE, the original 640 × 400 image is first downsampled by 2 × 2, and then processed by a pixel-wise subtraction operation with respect to the previous frame to generate an event map, which is then processed by a DNN to

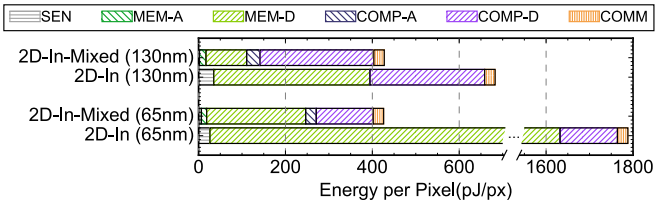


Fig. 10: Energy comparison between mixed-signal in-sensor computation and fully-digital in-sensor computation on EDGAZE. COMP/MEM-D: digital compute and memory; COMP/MEM-A: analog compute and memory.

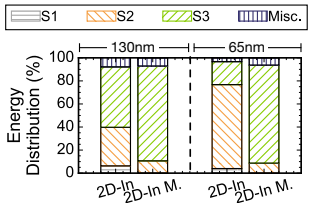


Fig. 11: Normalized energy breakdown among the three stages (S1, S2, S3).

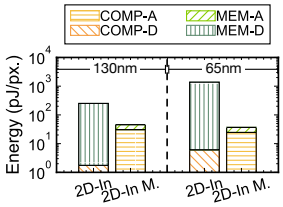


Fig. 12: Energy breakdown of first two stages.

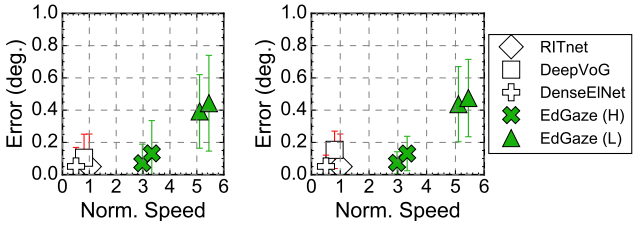
generate the ROI. The ROI, on average, reduces the image size by 25%. The DNN dominates the computation and performs about 5.76×10^7 MAC operations per frame.

Hardware Design Plainly implementing EDGAZE inside a CIS would design customized digital circuits for each individual operation. Here, we design a mixed-signal CIS which achieves better energy efficiency compared to the plainly digital one. Fig. 9 shows how EDGAZE is mapped to a mixed-signal CIS. Inside the pixel array, the 2×2 downsampling is done through pixel binning. The analog frame buffer stores the downsampled analog pixel values, which are read by an analog PE array for frame subtraction. Each analog PE consists of a switched-capacitor subtractor/multiplier for absolute subtraction and a comparator for frame delta digitization. The output of the Analog PE array enters the SRAM array, a dedicated DNN accelerator is implemented for the DNN inference. To demonstrate the efficiency of our mixed-signal CIS design, we compare two CIS implementations:

- **2D-In**: a 2D CIS fabricated; the entire execution is performed in the digital domain inside the CIS.
- **2D-In-Mixed**: a 2D CIS, where the first two stages in the algorithm (Fig. 9), 2×2 downsampling and frame subtraction, are implemented in analog while last stage (ROI DNN) is implemented in the digital domain.

For a fair comparison and to ensure area overhead is well accounted for, we conservatively set all the capacitors to 100fF. Despite the oversizing, the analog design still yields at least 27% less area than the digital counterpart.

Energy Savings Fig. 10 compares **2D-In-Mixed** and **2D-In**. Moving the first stages of the Ed-Gaze algorithm to the analog domain reduces the energy by 38.8% and 77.1%. The energy reduction comes from two sources: removing the ADCs (indicated by lower SEN) and replacing SRAMs in the first



(a) Vertical gaze error vs. speedup. (b) Horizontal gaze error vs. speedup.

Fig. 13: The accuracy and speed comparison of different methods. All the subfigures share the same legend. The speedup values are normalized to the speed of RITNET. EDGAZE(H) and EDGAZE(L) are two configurations optimized for accuracy and performance.

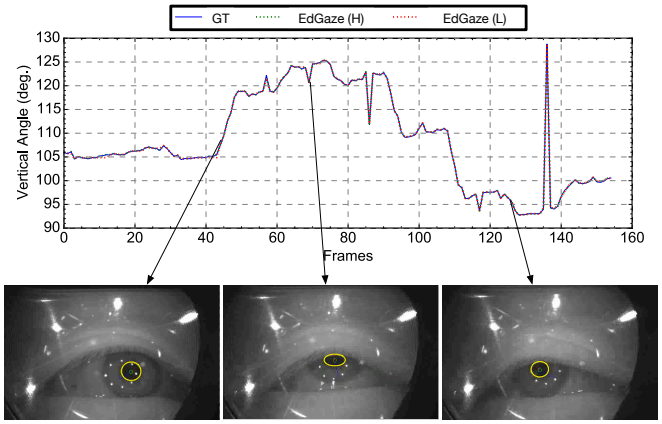


Fig. 14: Gaze estimation results over one sequence of frames. EDGAZE(H) robustly tracks the ground truth. The bottom panel shows three representative cases: eye moves right, just before a blink, and eye moves up-left, respectively.

two stages with analog buffers (indicated by lower MEM-D). The reduction in MEM-D is particularly significant for the 65nm node, where the SRAM leakage power is high. To corroborate the results, Fig. 11 shows the normalized energy breakdown among the three stages (S1, S2, and S3). S3 (DNN) becomes the dominant stage after moving first two stages into analog domain, showing the effectiveness of analog processing.

Interestingly, the energy reduction is obtained when the compute energy of the first two stages slightly increase. Fig. 12 shows the energy breakdown of the first two stages. While the memory energy reduces, the compute energy increases in the mixed-signal mode. This is because to maintain an 8-bit precision the OpAmp consumes too much energy. A caveat is that the analog design presented here, which uses active switched-capacitor circuits, is based on our specific implementation choice. It is conceivable that different designs would yield different efficiency results.

An important finding from this result is: while analog computing is known for reducing ADC and computing energy, the energy saving is also attributed to lower analog memory energy, especially for memory-intensive applications, which

many in-CIS use-cases are.

Accuracy Evaluation EDGAZE achieves a 5.5× speedup over the baselines with a sub-0.5° gaze error. Fig. 13a and Fig. 13b compare the vertical and horizontal gaze errors and the speed of EDGAZE with state-of-the-art algorithms.[1] Different markers of EDGAZE represent different configurations optimized for accuracy and performance. The speedups are normalized to the speed of RITNET, which runs at 5.4 Hz on a mobile Volta GPU. Note that a 1° error is generally acceptable for gaze tracking [20].

RITNET, DENSEELNET, and EDGAZE (H) keep the absolute error rate below 0.1° in both the vertical and horizontal direction. They are all more accurate than DEEPVOG. By operating on ROI images when possible, EDGAZE(H) improves the speedup over RITNET to 3.0×. By further optimizing performance, EDGAZE(L) further improves the speedup to 5.1× and 5.5×, respectively, while both retaining an angular error rate within 0.5°.

To further confirm the robustness of our system, Fig. 14 compares the frame-by-frame gaze results of EDGAZE(H), EDGAZE(L), and the ground truth. EDGAZE(H) virtually matches the ground truth, and EDGAZE(L) has slight deviations (e.g., around frame 10). We show three representative gazes in the bottom panel, where the eye moves right, blinks, and moves up left.

Summary Strategically harnessing computational power inside CIS significantly speedup the end-to-end eye tracking while retraining similar accuracy compared to state-of-the-art algorithms. Further augmenting tailored in-sensor circuits on existing CIS not only maximizes the potential of the in-sensor algorithm but also achieves optimal algorithm-CIS co-design, all with minimum hardware modification.

V. FUTURE PERSPECTIVE

Emerging CIS designs are discussed in Sec. II.

Cross-Domain Modeling Framework To effectively pinpoint the bottleneck of computational CIS designs, a comprehensive CIS modeling would facilitate the design process. Such a modeling framework should cover both the analog domain and the digital domain. Unfortunately, no comprehensive CIS modeling framework exists. Two recent papers from Meta use a first-order analytical model to estimate the energy of their custom CIS design, i.e., 3D-stacked CIS with DPS [16], [26]. It does not provide the level of flexibility to accommodate to general CIS designs and architecture exploration.

LiKamWa et al. [25] provide a coarse-grained CIS power model using the idle and active period/power without considering the hardware implementation details. Kodukula et al. [22] cite coarse-grained component energy of typical CIS designs and build a thermal model. A recent work, CAMJ, models the hardware with finer granularity to achieve fine-grained architectural exploration [31]. Integrating CAMJ with Kodukula et al. can provide more accurate power/energy modeling that helps thermal modeling.

[1]For a more detailed experimental setup, please refer to [14]

However, the aforementioned works overlook the functional modeling of the computaional CIS. iSETCam [13] provides a coarse-grained noise modeling that does not characterize noise generation within individual analog components. Similarly, other works [6], [33] give detailed noise modeling at the component level but for conventional CIS rather than computational CIS. To model in-sensor computing, a demand arises for a detailed functional framework with a flexible interface to support diverse analog computing designs. Designing such a framework would be a potential direction for future research.

End-to-End Visual Optimizations Recent work discusses the possibility of in-sensor processing to reduce the data transmission cost, such as EDGAZE [14], Rhythmic Pixel Regions [23], LECA [29], and SplitNets [10]. However, none of them optimize the entire visual pipeline end-to-end. EDGAZE, Rhythmic Pixel Regions, and SplitNets rely on first-order energy models without considering the CIS details. LECA [29] does simulate the detailed computational circuits within CIS, but only optimizes pixel readout without considering pixel sensing. CAMJ [31] demonstrates that considering optimizations inside CIS allows Ed-Gaze and Rhythmic Pixel Regions to achieve superior performance compared to the original implementations. It is clear that optimizing from pixel sensing to the downstream task end-to-end would open more opportunities and potential improvements.

Many recent visual computing optimizations use motion vectors that can be naturally generated during imaging to simplify downstream vision processing [42], [15]. It is interesting to explore how motion estimation can be integrated into the CIS and optimize the visual pipeline end-to-end. Additionally, hardware-aware neural architecture search (NAS) is already on the horizon [2]. Rather than manually finetuning the CIS design, another intriguing avenue is to integrate the current visual systems with NAS and automate CIS design process.

VI. CONCLUSION

As computational CIS become mainstream in visual computing, inevitably, different CIS designs will emerge to accommodate various applications. Incorporating hardware behavior into the algorithm optimization will gradually gain its population due to increasingly lightweight computing platforms and stringent hardware resources. This paper serves as the initial step towards learned in-sensor visual computing.

REFERENCES

[1] "International roadmap for devices and systems," https://irds.ieee.org/.

[2] H. Benmeziane, K. E. Maghraoui, H. Ouarnoughi, S. Niar, M. Wistuba, and N. Wang, "A comprehensive survey on hardware-aware neural architecture search," *arXiv preprint arXiv:2101.09336*, 2021.

[3] M. Bigas, E. Cabruja, J. Forest, and J. Salvi, "Review of cmos image sensors," *Microelectronics journal*, vol. 37, no. 5, pp. 433–451, 2006.

[4] K. Bong, S. Choi, C. Kim, D. Han, and H.-J. Yoo, "A low-power convolutional neural network face recognition processor and a cis integrated with always-on face detector," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 1, pp. 115–123, 2017.

[5] 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.

[6] A. Boukhayma and A. Boukhayma, *Low-noise CMOS image sensors*. Springer, 2018.

[7] W. Cao, Y. Zhao, A. Boloor, Y. Han, X. Zhang, and L. Jiang, "Neural-pim: Efficient processing-in-memory with neural approximation of peripherals," *IEEE Transactions on Computers*, vol. 71, no. 9, pp. 2142–2155, 2022.

[8] D. G. Chen, F. Tang, M.-K. Law, and A. Bermak, "A 12 pj/pixel analog-to-information converter based 816× 640 pixel cmos image sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 5, pp. 1210–1222, 2014.

[9] C.-C. Cheng, C.-H. Lin, C.-T. Li, and L.-G. Chen, "ivisual: An intelligent visual sensor soc with 2790 fps cmos image sensor and 205 gops/w vision processor," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 127–135, 2008.

[10] X. Dong, B. De Salvo, M. Li, C. Liu, Z. Qu, H. Kung, and Z. Li, "Splitnets: Designing neural architectures for efficient distributed computing on head-mounted systems," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 559–12 569.

[11] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. F. Kelly, and R. G. Baraniuk, "Single-pixel imaging via compressive sampling," *IEEE signal processing magazine*, vol. 25, no. 2, pp. 83–91, 2008.

[12] R. Eki, S. Yamada, H. Ozawa, H. Kai, K. Okuike, H. Gowtham, H. Nakanishi, E. Almog, Y. Livne, G. Yuval *et al.*, "9.6 a 1/2.3 inch 12.3 mpixel with on-chip 4.97 tops/w cnn processor back-illuminated stacked cmos image sensor," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64. IEEE, 2021, pp. 154–156.

[13] J. E. Farrell, P. B. Catrysse, and B. A. Wandell, "Digital camera simulation," *Applied optics*, vol. 51, no. 4, pp. A80–A90, 2012.

[14] Y. Feng, N. Goulding-Hotta, A. Khan, H. Reyserhove, and Y. Zhu, "Real-time gaze tracking with event-driven eye segmentation," in *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 2022, pp. 399–408.

[15] Y. Feng, P. Whatmough, and Y. Zhu, "Asv: Accelerated stereo vision system," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2019, pp. 643–656.

[16] J. Gomez, S. Patel, S. S. Sarwar, Z. Li, R. Capoccia, Z. Wang, R. Pinkham, A. Berkovich, T.-H. Tsai, B. De Salvo *et al.*, "Distributed on-sensor compute system for ar/vr devices: A semi-analytical simulation framework for power estimation," *arXiv preprint arXiv:2203.07474*, 2022.

[17] C. W. Group, "Mipi white paper: An introductory guide to mipi automotive serdes solutions (mass)," 2021. [Online]. Available: https://www.mipi.org/introductory-guide-to-mass

[18] T. Hirata, H. Murata, H. Matsuda, Y. Tezuka, and S. Tsunai, "7.8 a 1-inch 17mpixel 1000fps block-controlled coded-exposure back-illuminated stacked cmos image sensor for computational imaging and adaptive dynamic range control," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64. IEEE, 2021, pp. 120–122.

[19] 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.

[20] A. Kar and P. Corcoran, "A review and analysis of eye-gaze estimation systems, algorithms and performance evaluation methods in consumer platforms," *IEEE Access*, vol. 5, pp. 16 495–16 519, 2017.

[21] A. Kaur, D. Mishra, K. Amogh, and M. Sarkar, "On-array compressive acquisition in cmos image sensors using accumulated spatial gradients," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 2, pp. 523–532, 2020.

[22] V. Kodukula, S. Katrawala, B. Jones, C.-J. Wu, and R. LiKamWa, "Dynamic temperature management of near-sensor processing for energy-efficient high-fidelity imaging," *Sensors*, vol. 21, no. 3, p. 926, 2021.

[23] V. Kodukula, A. Shearer, V. Nguyen, S. Lingutla, Y. Liu, and R. LiKamWa, "Rhythmic pixel regions: multi-resovisual sensing system towards high-precision visual computing at low power," in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2021, pp. 573–586.

[24] M. Kwon, S. Lim, H. Lee, I.-S. Ha, M.-Y. Kim, I.-J. Seo, S. Lee, Y. Choi, K. Kim, H. Lee *et al.*, "A low-power 65/14nm stacked cmos image sensor," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2020, pp. 1–4.

[25] R. LiKamWa, B. Priyantha, M. Philipose, L. Zhong, and P. Bahl, "Energy characterization and optimization of image sensing toward continuous mobile vision," in *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, 2013, pp. 69–82.

[26] C. Liu, A. Berkovich, S. Chen, H. Reyserhove, S. S. Sarwar, and T.-H. Tsai, "Intelligent vision systems–bringing human-machine interface to ar/vr," in *2019 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2019, pp. 10–5.

[27] C. Liu, S. Chen, T.-H. Tsai, B. De Salvo, and J. Gomez, "Augmented reality-the next frontier of image sensors and compute systems," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 65. IEEE, 2022, pp. 426–428.

[28] S. Ma, X. Zhang, C. Jia, Z. Zhao, S. Wang, and S. Wang, "Image and video compression with neural networks: A review," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 6, pp. 1683–1698, 2019.

[29] T. Ma, A. J. Boloor, X. Yang, W. Cao, P. Williams, N. Sun, A. Chakrabarti, and X. Zhang, "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.

[30] T. Ma, W. Cao, F. Qiao, A. Chakrabarti, and X. Zhang, "Hogeye: neural approximation of hog feature extraction in rram-based 3d-stacked image sensors," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, 2022, pp. 1–6.

[31] T. Ma, Y. Feng, X. Zhang, and Y. Zhu, "Camj: Enabling system-level energy modeling and architectural exploration for in-sensor visual computing," in *Proceedings of the 50th Annual International Symposium on Computer Architecture*, 2023, pp. 1–14.

[32] T. S. of Machine Learning, "The exponential growth of ai," 2020. [Online]. Available: https://www.ml-science.com/exponential-growth

[33] J. Ohta, *Smart CMOS image sensors and applications*. CRC press, 2020.

[34] C. Park, W. Zhao, I. Park, N. Sun, and Y. Chae, "A 51-pj/pixel 33.7-db psnr 4× compressive cmos image sensor with column-parallel single-shot compressive sensing," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 8, pp. 2503–2515, 2021.

[35] D. Schor, "Hot chips 30: Nvidia xavier soc," 2018. [Online]. Available: https://fuse.wikichip.org/news/1618/hot-chips-30-nvidia-xavier-soc/#:~:text=Xavier%20consists%20of%209%20billion,die%20itself%20measures%20350%20mm%C3%82%C2%B2.

[36] Z. Shukri, "Apple's newest iphone three camera system is "cam-packed"," 2023. [Online]. Available: https://www.techinsights.com/blog/apple-iphone-14-image-sensor-preliminary-analysis

[37] H. Tsugawa, H. Takahashi, R. Nakamura, T. Umebayashi, T. Ogita, H. Okano, K. Iwase, H. Kawashima, T. Yamasaki, D. Yoneyama, J. Hashizume, T. Nakajima, K. Murata, Y. Kanaishi, K. Ikeda, K. Tatani, T. Nagano, H. Nakayama, T. Haruta, and T. Nomoto, "Pixel/dram/logic 3-layer stacked cmos image sensor technology," in *2017 IEEE International Electron Devices Meeting (IEDM)*, 2017, pp. 3.2.1–3.2.4.

[38] 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*, vol. 69, no. 1, pp. 232–243, 2021.

[39] C. Young, A. Omid-Zohoor, P. Lajevardi, and B. Murmann, "A data-compressive 1.5/2.75-bit log-gradient qvga image sensor with multi-scale readout for always-on object detection," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 11, pp. 2932–2946, 2019.

[40] B. Zhang, P. V. Sander, C.-Y. Tsui, and A. Bermak, "Microshift: An efficient image compression algorithm for hardware," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 11, pp. 3430–3443, 2018.

[41] M. Zhang, A. Bermak *et al.*, "Cmos image sensor with on-chip image compression: A review and performance analysis," *Journal of Sensors*, vol. 2010, 2010.

[42] Y. Zhu, A. Samajdar, M. Mattina, and P. Whatmough, "Euphrates: Algorithm-soc co-design for low-power mobile continuous vision," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE Computer Society, 2018, pp. 547–560.