LeCA: In-Sensor Learned Compressive Acquisition for Efficient Machine Vision on the Edge

Tianrui Ma*
tianrui.ma@wustl.edu
Washington University in St. Louis
St. Louis, MO, USA

Adith Jagadish Boloor*
adith@wustl.edu
Washington University in St. Louis
St. Louis, MO, USA

Xiangxing Yang
yangxx@utexas.edu
University of Texas at Austin
Austin, TX, USA

Weidong Cao
weidong.cao@wustl.edu
Washington University in St. Louis
St. Louis, MO, USA

Patrick Williams
patrickwilliams@wustl.edu
Washington University in St. Louis
St. Louis, MO, USA

Nan Sun
nansun@tsinghua.edu.cn
Tsinghua University
Beijing, China

Ayan Chakrabarti
ayan@email.wustl.edu
Washington University in St. Louis
St. Louis, MO, USA

Xuan Zhang
xuan.zhang@wustl.edu
Washington University in St. Louis
St. Louis, MO, USA

ABSTRACT

With the rapid advances of deep learning-based computer vision (CV) technology, digital images are increasingly consumed, not by humans, but by downstream CV algorithms. However, capturing high-fidelity and high-resolution images is energy-intensive. It not only dominates the energy consumption of the sensor itself (i.e. in low-power edge devices), but also contributes to significant memory burdens and performance bottlenecks in the later storage, processing, and communication stages. In this paper, we systematically explore a new paradigm of in-sensor processing, termed "learned compressive acquisition" (LeCA). Targeting machine vision applications on the edge, the LeCA framework exploits the joint learning of a sensor autoencoder structure with the downstream CV algorithms to effectively compress the original image into low-dimensional features with adaptive bit depth. We employ column-parallel analog-domain processing directly inside the image sensor to perform the compressive encoding of the raw image, resulting in meaningful hardware savings, and energy efficiency improvements. Evaluated within a modern machine vision processing pipeline, LeCA achieves 4×, 6×, and 8× compression ratios *prior to any digital compression*, with minimal accuracy loss of 0.97%, 0.98%, and 2.01% on ImageNet, outperforming existing methods. Compared with the conventional full-resolution image sensor and the state-of-the-art compressive sensing sensor, our LeCA sensor is 6.3× and 2.2× more energy-efficient while reaching a 2× higher compression ratio.

*Both authors contributed equally to this research.

ISCA '23, June 17–21, 2023, Orlando, FL, USA

ACM ISBN 979-8-4007-0095-8/23/06.
https://doi.org/10.1145/3579371.3589089

CCS CONCEPTS

• **Computer systems organization** → **Neural networks**; • **Hardware** → **Sensor devices and platforms**.

KEYWORDS

CMOS image sensor; image compression; autoencoder

ACM Reference Format:
Tianrui Ma, Adith Jagadish Boloor, Xiangxing Yang, Weidong Cao, Patrick Williams, Nan Sun, Ayan Chakrabarti, and Xuan Zhang. 2023. LeCA: In-Sensor Learned Compressive Acquisition for Efficient Machine Vision on the Edge. In *Proceedings of the 50th Annual International Symposium on Computer Architecture (ISCA '23), June 17–21, 2023, Orlando, FL, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3579371.3589089

1 INTRODUCTION

Motivation. The modern world craves for rich contextual information, much of which is driven by diverse vision applications, thanks to the expansion of various consumer camera devices and image sensors in the past decades. Apart from serving the growing demand of social networks, image sensors also play vital roles in many industrial and scientific applications, such as security monitoring [22], environmental sensing [46], and medical imaging [77]. In these first-generation vision applications, humans are often the end-consumers of the images, therefore faithful capture and reconstruction of the original light scene become an important quality measure. Nonetheless, recent accelerated advancements of deep learning (DL)-based computer vision have unleashed the second wave of machine vision. In this second wave, voluminous vision data are increasingly generated by intelligent edge devices and consumed, not by humans, but by downstream computer vision (CV) algorithms to perform sophisticated tasks such as classification, recognition, and machine perception [13, 20, 53], as shown in Fig. 1. It presents a unique opportunity for innovative image sensor systems— *Given that images are destined for the downstream vision algorithms without the need for high-fidelity reconstruction, it is now possible to compress and preserve the "task-specific" information to reduce energy consumption and save hardware costs.*

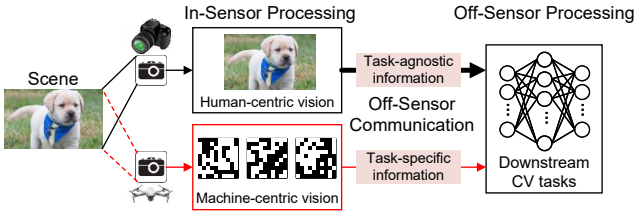

Figure 1: Human-centric vision processing pipeline and LeCA-adopted machine-centric vision processing pipeline.

Generally, image sensors perform the fundamental utility of converting light to electrical signals for later storage, processing, communication and consumption. In conventional image sensors, all pixels are indiscriminately converted to a pre-defined digital format with a fixed bit depth (e.g., 8-bit). Considerable energy and resources of the overall sensor system are dedicated to the readout peripheral and analog-to-digital conversion (ADC) circuits, as well as the on-chip storage and off-chip transmission of the raw image frame after its capture and digitization. These components significantly contribute to the silicon area, power, and latency of the image sensor, which grows proportionally to image resolution. For example, a survey on state-of-the-art image sensors [12, 15–17, 34, 37, 41, 42, 51, 65, 72, 73] shows that the ADC and output buffer circuits alone consume 69% of the sensor's power, 34% of the pixel row's readout time, and more than 60% of pixel array area.

Current limitations. Image compression has been studied extensively in the past, and there exists a body of research on efficient compression/encoding methods that range from classic discrete cosine transform (DCT)/wavelet-based algorithms (i.e. JPEG) to emerging end-to-end learned image compression [1, 33, 62]. However, current compression schemes, by and large, are performed in the digital domain. It demands the same amount of sensor resources and energy to convert the raw pixels to their digital bit representations during initial image acquisition before applying any compression, while requiring dedicated power-hungry digital compression engines in its image processing pipeline. Therefore, the reduced image size from digital compression does not directly benefit the image sensor itself and cannot be readily translated to meaningful resource and energy savings. Alternatively, the concept of compressive sensing [20] and compressive acquisition [87] have been explored to relieve the image capture and digitization cost at the sensor front-end. However, existing schemes of compressive sensing and compressive acquisition are task-agnostic, resulting in a modest compression ratio with limited task accuracy. They often also require compute-intensive iterative optimization at the decoding stage to reconstruct the image [85] and thus are unsuitable for latency-sensitive machine vision applications.

Our work. In this paper, we seek to systematically explore a new paradigm of image sensor design which we term ***LeCA*** – it employs a **Le**arning-based **C**ompressive **A**cquisition method to extract condensed task-relevant features instead of defaulting to the fixed quantization scheme universally adopted by existing compressive sensing/acquisition solutions[49, 50, 63]. LeCA exploits the opportunity in the modern machine vision pipeline where the image data are consumed by the deep neural network (DNN) based downstream CV algorithms, obviating the need to reconstruct the original image based on human-centric visual quality metrics. The proposed LeCA framework is a hardware/algorithm co-design approach that is made feasible by the combination of three key techniques: ❶ LeCA stacks an autoencoder before the downstream CV algorithm that allows the joint learning of the task-specific features in an end-to-end manner. The autoencoder comprises a single encoding layer with lightweight decoder layers, facilitating an in-sensor implementation of the compressive encoding layer. ❷ A hardware-aware noise-tolerant training process that incorporates the analytical behavioral and noise models of the analog-domain multiplier and buffer circuits to properly account for their circuit-level nonidealities, leading to more precise hardware instantiation and superior accuracy of the trained LeCA models. ❸ Our LeCA sensor system employs a column-parallel processing element (PE) array using switched-capacitor multipliers (SCMs) to enable compressive feature extraction and variable low-resolution quantization directly at the sensor front end. In addition to improving the energy efficiency of the image sensor itself, LeCA reduces the image size right off its source, which can be translated to memory storage and computing power savings for later-stage processing. With column-parallel PE arrays and programmable encoder weights and channel dimensions, the proposed LeCA system can flexibly scale with the image resolution and adapt to varying compression ratios, making it a practical solution for energy-efficient machine vision applications.

Contributions. Our main contributions are:

- We propose LeCA, a hardware/algorithm co-design framework that exploits the joint learning of a sensor autoencoder with the downstream CV algorithms to compress the original pixel-wise image data into task-specific low-dimensional features with adaptable bit depth and minimal task accuracy loss.
- A hardware-aware and noise-tolerant training process is developed and tailored for the LeCA framework where we fully account for the circuit-level behaviors and non-idealities of LeCA's analog-domain hardware.
- We design an efficient implementation in standard CMOS 65nm technology employing column-parallel analog-domain PE arrays with variable-resolution ADCs to perform the single-layer LeCA encoder.
- We validate LeCA's superior compression-accuracy tradeoffs against alternative compression methods using comprehensive benchmark datasets (ResNet-50 on ImageNet).

2 BACKGROUND

We first provide an introduction to CMOS image sensors and highlight the hardware overheads that are directly associated with image size (Sec. 2.1). Next, various image compression/encoding schemes are reviewed (Sec. 2.2). Finally, we present previously-proposed in-sensor processing architectures and the associated circuit-level techniques (Sec. 2.3).

2.1 CMOS Image Sensor Primer

CMOS image sensor (CIS) is one of the most popular vision frontends. It typically consists of a pixel plane, column-parallel readout and ADC circuits, output buffers, and a serial communication interface to transmit the image data off-chip, as shown in Fig. 2(a). The 2D pixel plane extends vertically and horizontally with $V \times H$ pixels.

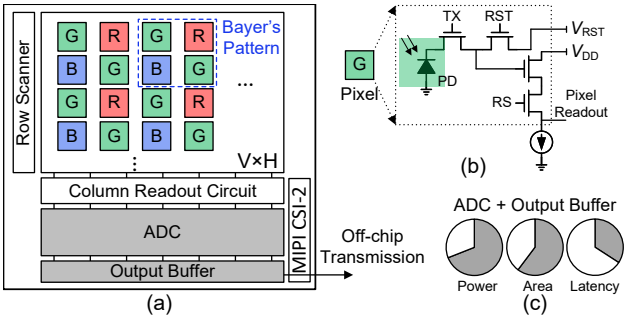

Figure 2: (a) Conventional CIS floorplan. (b) 4-T pixel structure. (c) Overhead of ADC and output buffer based on CIS survey.

A typical active pixel sensor (APS) design employs a 4-T pixel cell structure (Fig. 2(b)). In color image sensors, the color filter array is placed on top of the pixel plane to multiplex visible light with different wavelengths. The filter array is usually placed in a Bayer pattern [25] as shown in Fig. 2(a): a 2×2 pixel block (two green, one red, and one blue) is grouped together, where the number of green filters is as twice as the others to emulate human vision sensitivity. This Bayer pattern of the raw image is later processed digitally by demosaicing through color interpolation to recover the full-color image for display.

Under regular frame rate operations, CIS commonly adopts a rolling shutter by exposing the pixel plane row by row. It allows the pixels in the same column to share one set of readout circuit and ADC. The number of ADCs is thus determined by the image width (H) in a rolling-shutter CIS. After the ADC, the digitized image is stored in the output buffer and streamed out through a serial interface (e.g. MIPI CSI-2). We survey 37 CIS papers published between 2010 and 2022 and find that the ADC and output buffer account for a significant proportion of the sensor's power, latency, and area (Fig. 2(c)) and the energy consumed by the serial communication link can be significant.

2.2 Sensor-side Image Compression

Compression is an effective method to alleviate the large image storage and transmission overheads caused by high-resolution image data. Standard compression techniques such as lossy predictive coding [71], variable length-coding [79], and JPEG encoding [78] exploit abundant spatial redundancy in natural images for compression. Apart from these classic methods, learned image compression has recently been explored, which learns only the most important features to aggressively compress the images, and recover the images with minimal perceptual loss, such as probability methods [60], generative adversarial networks [1], and autoencoders [14, 90]. General techniques that compress neural network feature maps such as sparsity[2, 24] and quantization[8, 91] can also be applied to reduce the input image size. However, all these schemes are exclusively performed in the digital domain after acquiring the digital images, hence they provide no resource or power-saving opportunity to the sensor chip. Moreover, digital compression requires dedicated processing engines whose power consumption often dwarfs that of the image sensor itself. For example, efficient JPEG engines consume on the order of nJ/pixel to compress the image[3, 67], several times the power of the conventional image sensor.

Table 1: Comparison of Image Compression Methods

Compression Method	Encoding Domain	Objective Function	Quality Metric	Hardware Overhead
Standard [71, 78, 79]	Digital	Task Agnostic	PSNR	High
Learned [1, 14, 60, 90]	Digital	Task Agnostic	PSNR	Medium
Heuristic Acquisition [39, 84, 87]	Mixed	Task Agnostic	PSNR	Medium
Compressive Sensing [64]	Analog	Task Agnostic	PSNR	Low
Ours - LeCA	Analog	Task Specific	Accuracy	Low

Alternatively, image compression can be achieved during the acquisition process, often using heuristic algorithms. Constrained by the limited computation that can be implemented inside the sensor chip, these heuristic algorithms tend to include simple operations such as encoding the neighboring pixel's intensities [88], encoding a block of pixels based on its mean, gradient, and bitmap [11], perturbing pixels to achieve low-resolution quantization [84], encoding pixel gradient to logarithmic representation [82] and skipping pixels with small accumulated gradients [39].

Compressive sensing (CS) is another notable approach to reduce the sensor cost associated with image capture. It exploits the sparsity of natural images and allows the raw images to be progressively reconstructed with a small number of linear measurements. When CS is applied to image sensors, these measurements are often obtained by multiplying the image with a random binary/ternary matrix and use the weighted sum of one or more blocks of pixel values to encode and represent the acquired images [64]. A downside of CS is its use of an iterative optimization method for image reconstruction that converges slowly, making it unsuitable for real-time machine vision tasks.

What existing compressive acquisition and CS solutions share in common is that they are all task-agnostic methods optimized and evaluated not by specific vision task performance, but by general image quality factors such as peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM) [30]. Table 1 summarizes the characteristics of different approaches to image compression. We propose LeCA that not only translates effective compression to meaningful hardware resource and energy savings but also delivers superior end-to-end task accuracy and performance.

2.3 In-Sensor Processing

In conventional image processing pipeline, the digitized image captured by the sensor is fed to a digital image signal processor (ISP) chip for post-processing to improve the image quality[29]. However, our reviews on image compression methods (Sec 2.2) suggest that if compression or lower-dimensional feature extraction of the image can be performed directly inside the sensor, preferably in the analog domain, then fewer data have to be explicitly digitized and transmitted off-chip for later processing. Such in-senor architectures have recently been explored with several possible implementations – pixel-level, column-level, and chip-level processing according to

the location of the processing elements (PEs) [61]. Due to the stringent pixel size, pixel-level PE can only employ a few transistors and perform limited computations to avoid severe degradation of the fill factor[57, 59, 68, 81]. Chip-level PE is placed next to the pixel array and processes the pixel readouts sequentially, resulting in low computational parallelism [83]. A variant of chip-level processing is to stack the sensor chip onto the processing chip with through-silicon-vias[45, 75] or hybrid bonds[54], which incurs higher fabrication and packaging cost in exchange for smaller pixel size and higher frame rate[40, 76]. In column-level processing, the PE resides with the column readout circuit that is shared by the pixels in one or multiple adjacent columns[5, 6]. It provides a middle ground for trading off between the area/complexity of the in-sensor circuitry and the processing parallelism.

A number of in-sensor processing circuits have been proposed to perform various pixel-weight operations such as max/min, logarithm, multiplication, and summation with current-[13, 35, 56, 70], voltage-[82], or charge-domain [48] implementations. These analog-domain circuits allow in-sensor pre-processing before signal digitization. In particular, vector multiplication is one of the atomic arithmetic operations that are commonly used in many pre-processing tasks. Our LeCA sensor adopts column-level processing with charge-domain multipliers to perform the learned compressive encoding on the raw pixel values.

3 LECA CO-DESIGN FRAMEWORK

We follow two central tenets in the development of the LeCA framework. First, LeCA is designed for resource-constrained image sensors. In such sensors, it is critical to reduce energy consumption and limit the area overhead of the sensor chip by itself, prompting the need to perform image encoding/compression in the analog domain before digitization. Second, LeCA targets the modern machine vision processing pipeline, where images are consumed by downstream CV algorithms rather than human visual inspection, allowing us to adopt an end-to-end measure of success instead of the traditional image quality metrics.

An overview of the proposed LeCA framework (Sec. 3.1) is first provided, followed by introducing the image processing pipeline (Sec. 3.2) and the encoder design (Sec. 3.3). We then elaborate on LeCA's customized training methodology that accounts for its unique combination of analog-domain encoder and digital-domain decoder with a pre-trained CV model as the backbone for the downstream tasks (Sec. 3.4).

3.1 LeCA Overview

As illustrated in Fig. 3, LeCA is a hardware/algorithm co-design framework consisting of two synergistically-optimized components – an encoder/decoder model that is jointly trained with a backbone DNN (i.e. ResNet50) for the downstream CV tasks; and an in-sensor processing architecture that efficiently implements the LeCA encoder layer directly inside the sensor chip. On the algorithmic side, LeCA adopts an encoder-decoder structure commonly seen in variational autoencoder literature [44] and stacks it before the downstream DNN model. The encoder performs a single-layer convolution between the raw RGB image and the LeCA encoder's learned kernels. The convolution output is then quantized to a

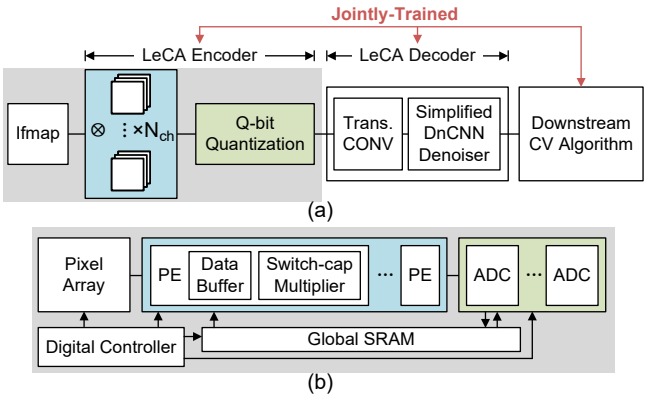

Figure 3: (a) LeCA vision processing pipeline. (b) LeCA sensor system diagram. LeCA encoding is implemented by PE and quantization is implemented by ADC.

low-resolution feature map. And we allow the bit depths 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 take the form of convolution layers and are jointly-trained with the downstream DNN so that all design parameters are aware of the downstream task accuracy.

In extreme low-power edge machine vision applications (e.g. always-on surveillance [16], wellness monitoring [52]), it is paramount to reduce the sensor energy beyond the conventional CIS implementation. We design a novel sensor architecture (Fig. 3(b)) to embed the computation of LeCA encoder into the sensor, and implement the computation with analog-domain PE arrays 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.

3.2 LeCA Machine Vision Processing Pipeline

We intentionally choose the encoder-decoder structure, as it enables simple plug-and-play of LeCA on top of a learning-based machine vision pipeline without modifying the backbone DNN structures or retraining the entire pipeline. Although LeCA could further reduce the energy/latency benefits of the downstream CV models due to image compression [19], we set the scope of our evaluation only to the sensor chip energy and performance.

The LeCA encoder is deliberately simplified: it has only one convolution layer with non-overlapping kernels and a limited number of output channels (N_{ch}) as shown in TABLE 2. In this way, each LeCA kernel condenses a $K \times K \times C$ pixel block in the ifmap into a

Table 2: Network Structure of LeCA Encoder and Decoder

Layer	Ifmap Dimensions	Weight Dimensions	OfMap Dimensions
LeCA Encoder			
CONV	$W \times H \times C$	$K \times K \times C \times N_{\text{ch}}$	$\frac{W}{K} \times \frac{H}{K} \times N_{\text{ch}}$
LeCA Decoder			
CONV Transpose	$\frac{W}{K} \times \frac{H}{K} \times N_{\text{ch}}$	$K \times K \times N_{\text{ch}} \times C$	$W \times H \times C$
CONV +ReLU	$W \times H \times C$	$K \times K \times N_{\text{ch}} \times C$	$W \times H \times C$
(M layers) CONV + BatchNorm + ReLU	$W \times H \times F$	$K_d \times K_d \times C \times F$	$W \times H \times F$
CONV	$W \times H \times F$	$K_d \times K_d \times F \times C$	$W \times H \times C$

single element in the ofmap, thereby achieving $K^2\times$ and $C\times$ compression in the ifmap's spatial domain and input-channel domain, respectively. Here, K is both the kernel size and stride length, and C is the number of input channels ($C = 3$ for RGB colorspace). After the encoder, the ofmap is hard-truncated and uniformly quantized to its low-resolution (Q_{bit}) representation, thereby achieving $\frac{Q_{\text{full}}}{Q_{\text{bit}}}\times$ compression in the ofmap's bit depth domain where $Q_{\text{full}} = 8$ represents the full resolution of typical images. Therefore, the compression ratio (CR) achieved by the LeCA encoder is:

$$CR = \frac{K \times K \times C \times Q_{\text{full}}}{N_{\text{ch}} \times Q_{\text{bit}}} \tag{1}$$

By intentionally choosing a small N_{ch}, we ensure that the compression gained through ifmap down-sampling and quantization is not offset by a large number of ofmaps. The simplicity of the LeCA encoder and the low-resolution quantization also enables energy-efficient hardware implementation (discussed later in Sec. 4).

After the image data are transmitted off-chip, the quantized ofmap is first upsampled to the size of the original ifmap through a transposed convolution block, as shown in TABLE 2; then it is denoised through a simple DnCNN denoiser [86] consisting of M stacked convolutional blocks. We find that complicated decoder designs used for image quality enhancement (e.g. PSNR, SSIM) are not necessary, as the LeCA decoder aims to retrieve the salient information from the quantized ofmap that contributes to the downstream task accuracy. Quantitatively, our evaluation suggests that sufficient accuracy is achieved when the number of DnCNN layers is $M = 15$ and the number of convolutional filters is $F = 64$, which takes only a fraction of the parameter sizes used in the state-of-the-art DNN backbone models (e.g., ResNet18/50).

3.3 LeCA Encoder Design

According to Eq. (1), the level of compression in LeCA is determined by three key parameters associated with the encoder: (1) the encoder's convolution kernel size K, (2) the number of encoded features N_{ch}, and (3) the bit depth of the encoded features Q_{bit}. These parameters participate in the tradeoff between compression ratio, hardware complexity, and downstream task accuracy. Here, we investigate this tradeoff and identify an optimal setting of the encoder parameters using a *proxy* machine vision pipeline: the TinyImageNet [47] dataset on a ResNet18 [28] downstream model.

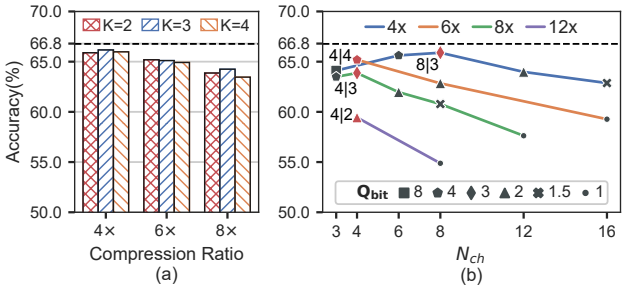

Figure 4: Proxy pipeline (a) accuracy under various kernel size K; (b) accuracy for N_{ch} and Q_{bit} (markers) parameter sweep across compression ratios of {4, 6, 8, 12} (colors) for K = 2. Best accuracies are marked in $N_{\text{ch}}|Q_{\text{bit}}$ notation for each compression ratio.

In Fig. 4(a), the effect of K is explored under three different compression ratios on the proxy pipeline. It shows that while high compression leads to some accuracy degradation, choosing $K \in \{2, 3, 4\}$ gives similar inference accuracy. From a hardware perspective, a smaller K means that fewer consecutive MAC operations are needed and a smaller portion of the ofmap is to be buffered, resulting in lower hardware complexity. $K = 1$, which does not perform spatial downsampling is not included because it require aggressive Q_{bit} and N_{ch} to achieve adequate compression, which leads to poor accuracy. Therefore, we fix $K = 2$ out of hardware efficiency consideration.

The remaining design space of the LeCA encoder is completely characterized by N_{ch} and Q_{bit} and there exists a clear and critical tradeoff between LeCA encoder's compression ratio (CR) and the downstream task accuracy. We investigate the inference accuracy by sweeping over N_{ch} and Q_{bit} combinations under different CRs. In Fig. 4(b), it shows that increasing the CR with lower N_{ch} and Q_{bit} values leads to degradation in end-to-end task accuracy. For a fixed CR of 4×, as N_{ch} increases and Q_{bit} decreases, the best performance is observed at the middle, suggesting that too few N_{ch}s or too aggressive Q_{bit}s leads to poor accuracy. The optimal N_{ch} and Q_{bit} combination that gives the highest inference accuracy varies for different CRs. Empirically, we observe in Fig. 4(b) that for CRs of 4×, 6×, 8×, $N_{\text{ch}}|Q_{\text{bit}}$ of 8|3, 4|4 and 4|3 are the optimal configurations, which are used as starting points for later full evaluation in Sec. 6. The LeCA hardware is designed to support programmable $N_{\text{ch}}|Q_{\text{bit}}$ configurations.

3.4 Training Methodology

Given the analog-domain implementation of the encoding layer and the stacked nature of the LeCA machine vision pipeline, we develop a customized training methodology to tackle a number of unique challenges that differentiate LeCA from the typical DNN-based CV model training process.

Joint training with backbone DNN. All parameters in the LeCA encoder-decoder are learned simultaneously with the downstream CV model (e.g., ResNet in this work) to maximize its end-to-end task accuracy. We train the entire pipeline with a cross-entropy loss that is typical for image classification tasks, in contrast to prior works that train to minimize the reconstruction loss between the raw and decoded images [58]. Particularly, the training is performed by freezing the backbone DNN with its pre-trained weights. This means that during backpropagation, the gradients are calculated

for each layer of the DNN, but its weights are not updated. Instead, those gradients are propagated back to update the weights in the LeCA encoder and decoder. This joint training allows LeCA to extract task-specific information in its encoding layer that emphasizes end-to-end task accuracy over the conventional visual reconstruction quality. Freezing the backbone weights is a deliberate choice as it enables us to easily swap the backbone for other models without retraining the entire end-to-end network.

Hardware-aware and noise-tolerant training. Although we can obtain learned weights from the digital training, transferring them to the hardware model is not trivial due to various hardware non-idealities. In order to minimize the accuracy degradation after the software-to-hardware mapping, we must consider comprehensive hardware non-idealities in the training forward path, including hardware constraints (e.g. limited signal range, limited precision, and constrained polarity), hardware offsets, and hardware noise and variations. Specifically, the convolution of the LeCA encoder layer is implemented by the analog PE consisting of three circuit stages – a PMOS source follower (PSF) buffer, a switched-capacitor multiplier (SCM), and a flipped voltage follower (FVF) buffer (Sec. 4).

Unlike ideal buffers and multipliers, PSF and FVF cause linear scaling and offsets, and SCM performs precision-limited multiplication with gain error. To deal with hardware constraints, we clamp the numerical values of the data in the encoder to be consistent with the real signal range in the PE circuit, and we quantize the encoder's weight to the hardware precision. We model the hardware offsets in two different ways. First, for the circuits with fixed transfer functions (PSF and FVF), we approximate them with analytical regression functions and insert them in the training forward path. Specifically, both transfer functions in PSF and FVF are modeled as linear functions. Second, for the circuits with programmable transfer functions (SCM and ADC), we innovatively incorporate the programmble circuit parameters in the training loop by inserting the exact circuit behavior models in the training forward path. Specifically, SCM takes both ifmap and weight for MAC operations. Instead of finding a mapping between the weight and the real circuit parameter in the SCM that represents the weight after training, we directly train that circuit parameter during backpropagation. Similarly, we directly train the ADC's quantization boundary.

To deal with hardware noise and variations, we specifically model the noise at each circuit stage from pixel acquisition to the end of ofmap digitization, and insert them into the training forward path stage by stage. Direct training on the full noise model leads to poor convergence. Instead, we first pre-train a noise-free pipeline, and then finetune it by incorporating the various noise models. The detailed modeling is discussed in Sec. 5.3. A key aspect in our training method is how to deal with hardware offsets and noises in an iterative manner due to the temporally-multiplexed weights unique to LeCA operation, unlike prior hardware-aware training that solely applies to spatially-multiplexed weights[9, 38, 89].

To summarize, there are three training modalities: ❶ *soft training* – training a convolutional layer without any hardware non-idealities, ❷ *hard training* – replace the software computation by the circuit analytical models with hardware constraints and offsets, and ❸ *noisy training* – replace the software computation by the actual circuit behaviors with hardware constraints, offset, and noise/variations. In Sec. 6, we show the performance of these different training modalities.

Differentiable backpropagation and incremental training. In the training pipeline, we model the ADC as an uniform quantizer, however, it prevents gradients from flowing during backpropagation due to the non-differentiable quantization function. To solve this, we employ the well-studied straight through estimator (STE) [4] technique. Specifically, during training we use Eq. (2) to ensure proper gradient flow:

$$f(x) = q(x) + x - \text{stop-gradient}(x). \tag{2}$$

where $q(\cdot)$ denotes the quantization function, and stop-gradient$(\cdot)$ means that the variable is included in the forward path but excluded from the gradient calculation. Eq. (2) ensures that in the forward path only the quantized values are propagated while during backpropagation, the actual value of x is used for gradient calculation.

In addition to non-differentiable quantizer, we observe that directly training with aggressive quantization (e.g., $Q_{\text{bit}} \leq 4$) generally leads to sub-optimal convergence. To alleviate this issue, we first train a LeCA model with more lenient quantization (e.g., Q_{bit}=8), and then use these weights to initialize the model that is trained with lower Q_{bit}. This strategy helps the model converge faster. Note that since the decoder comes after the ADC, we use full-precision for its weights and activations.

4 LECA SENSOR ARCHITECTURE AND CIRCUITS

In this section, we present the energy-efficient hardware implementation of the LeCA sensor system by embedding the computation of the encoder layer directly inside a CIS. We first describe the processing dataflow with a simplified example of a 4×4 pixel block (Sec. 4.1) and then present the timing sequence of the system operations (Sec. 4.2). We build the full sensor system with detailed circuit-level implementation (Sec. 4.3) and perform the transistor-level simulation using SPICE (Sec. 4.4).

4.1 LeCA Processing Dataflow

The LeCA sensor is designed with a pixel array size of 448×448 with the Bayer pattern filter where the green pixel is duplicated (Sec. 2.1). 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. 5(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. 5(b).

To illustrate the processing dataflow, Fig. 5(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. As mentioned in Sec. 3.1, in LeCA sensor the ifmap and the partial sum (psum)/ofmap are in the analog domain while the weight is in the digital domain. To reduce analog data movement, LeCA sensor adopts an input-stationary dataflow:

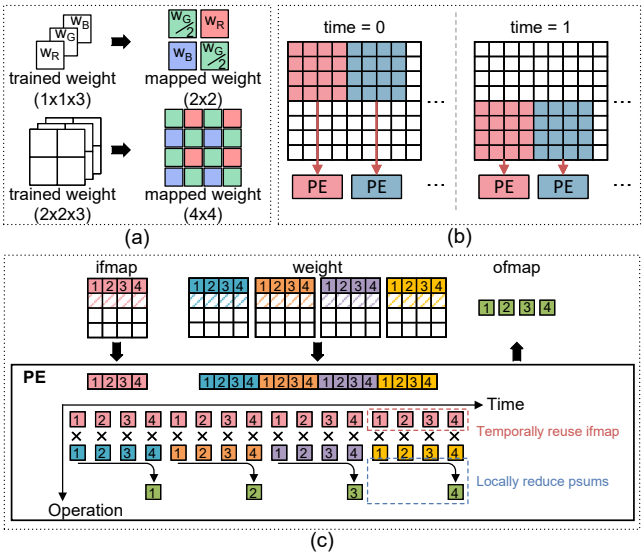

Figure 5: (a) Kernel flattening. (b) Column-parallel PE in LeCA. (c) A 4×4 example showing processing dataflow inside one PE.

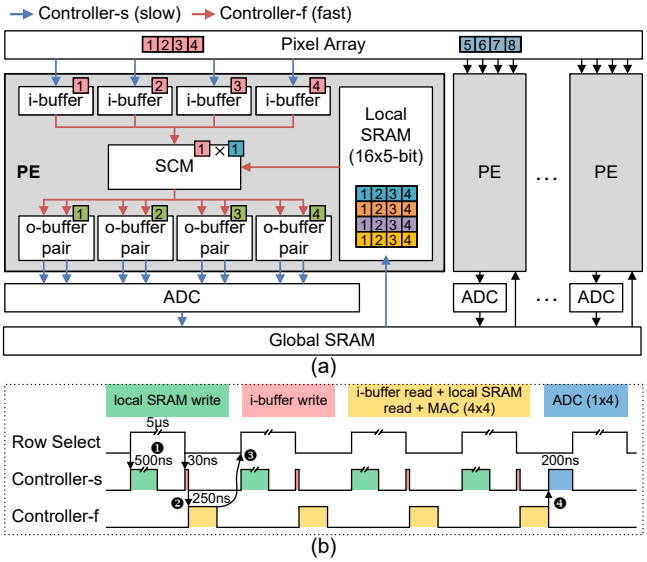

Figure 6: (a) Hardware signal path and (b) controller timing diagram.

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.

4.2 Sensor Operation Sequence

Fig. 6(a) shows how the toy example unfolds in hardware in the first PE. All PEs share the same processing dataflow except each of them receives different ifmap. For example, the first PE receives $\texttt{ifmap}_{1,2,3,4}$ and the second PE receives $\texttt{ifmap}_{5,6,7,8}$. Each PE contains 4 ifmap buffers (i-buffers) for ifmap storage, a 16×5-bit local SRAM for weight storage, a SCM to perform consecutive MAC operations, and 4 ofmap buffers (o-buffers) for psum accumulation and ofmap storage. Each PE can at most process 4 ofmap elements, corresponding to 4 kernels. When the number of kernels (N_{ch}) is larger than 4, e.g., N_{ch}=8, after popping out $\texttt{ofmap}_{1,2,3,4}$, the $\texttt{ifmap}_{1,2,3,4}$ is buffered to the PE again together with the $\texttt{weight}_{1,2,3,4}$ in kernel 5 to kernel 8, generating $\texttt{ofmap}_{5,6,7,8}$.

To take advantage of the SCM's fast operation without imposing high network-on-chip (NoC) bandwidth requirements, a hybrid strategy is applied where the timing of the LeCA sensor is coordinated by two controllers in different clock domains – a slow controller-s at 100MHz and a fast controller-f at 400MHz. In Fig. 6(a) the blue arrows are synchronized by the controller-s while the red ones are by the controller-f. The operation sequence among each component of the LeCA sensor is shown in Fig. 6(b) in four steps:

❶ Once the readout for the 1st row of pixels starts (ROWSEL is on), the row scanner triggers controller-s to enable writing 16×5-bit weights from the global SRAM to the local SRAM. The local SRAM write consumes 500ns and the pixel readout typically takes ~ μs, thus the latency of the local SRAM write is hidden behind that of the pixel readout. When pixel readout is finished (ROWSEL off), controller-s enables writing of 4 analog pixel values (ifmap) to 4 i-buffers, consuming 30ns.

❷ Controller-s triggers controller-f to consecutively read weights from the SRAM to the SCM, and cyclically read the 4 ifmap elements from the i-buffers to the SCM. The generated psums are written to the o-buffers. This step takes 250ns.

❸ After 16 psums are accumulated to 4 o-buffers, controller-f triggers the row scanner to readout the 2nd row of pixels.

❹ After processing 4 rows of pixels, controller-s is triggered to fetch the 4 ofmap elements from the o-buffers to the ADC, and finally to the global SRAM, which consumes 200ns. Depending on N_{ch}, the row scanner can either trigger the readout of the 5th row (if $N_{ch} \leq 4$), or trigger the readout of the 1st row again for **repetitive readout** (if $N_{ch} > 4$).

As the LeCA encoder processes the image row by row, the frame latency is determined by the encoder processing latency of each 4 row accumulated over the height of the pixel array. The row processing latency is dominated by pixel readout prior to computation, especially when the repetitive readout is needed. Based on the timing diagram in Fig. 6(b), we estimate the frame rate to reach 209fps with 448×448 resolution.

4.3 Circuit Implementation

The main building blocks of the LeCA sensor system include pixel array, analog-domain PE array, and ADC array. Fig. 7 illustrates how the raw pixel values, as voltage signals, are processed in the

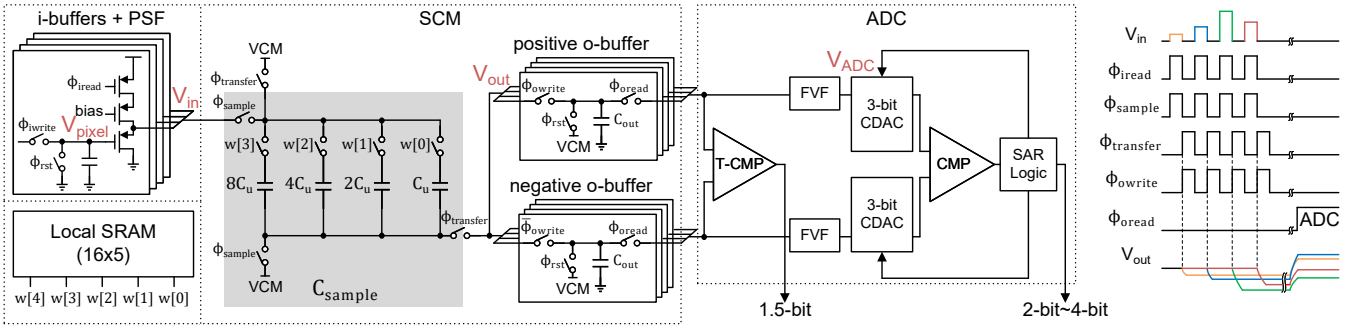

Figure 7: Circuit schematic of the mixed-signal PE and resolution-reconfigurable ADC and the operation timing diagram.

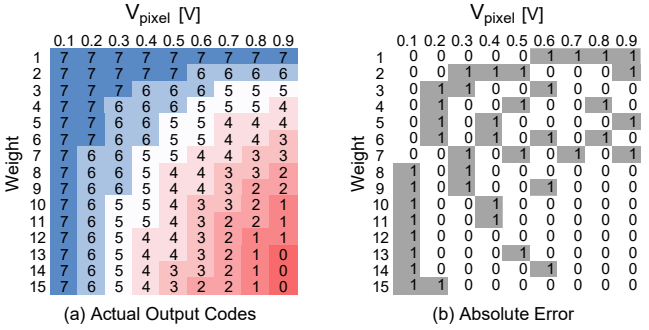

Figure 8: (a) Simulation output and (b) simulation error compared with ideal output.

analog domain inside the PE, as they go through i-buffer, SCM, o-buffer, and ADC to complete the LeCA encoding operation.

Pixel and analog readout. We adopt standard 4-T pixel (Fig. 2(b)). The pixel array size is set to 448×448 to match ImageNet's image resolution (224×224×3) with ifmap/weight flattening. The energy of pixel exposure and readout is estimated as 12.1pJ/pixel based on previous work [74].

I-buffer. As the first PE stage, the i-buffer is implemented using metal-oxide-metal (MOM) capacitors. The analog pixel readout is stored as voltage (V_{pixel}) at the capacitor through ϕ_{iwrite} and reset through ϕ_{rst}. To drive the SCM, the i-buffer is followed by a PSF which reads $V_{\text{i-buffer}}$ out as V_{in} through ϕ_{iread}. The capacitance of the i-buffer is 109fF.

Switched-capacitor multiplier (SCM). The four i-buffers share one SCM for MAC operation and the computation precision is ±4-bit. The SCM consists of a 4-bit sampling capacitor (C_{sample}) for magnitude multiplication, and differential o-buffers for sign operation. The trained weight $w[4{:}0]$ is read out from the local SRAM, with the magnitude bits $w[3{:}0]$ setting how much capacitance is connected in the C_{sample}, and the sign bit determining if the C_{sample} is connected to the positive o-buffer or the negative o-buffer.

The SCM performs MAC operations in a time-multiplexing manner. After the local SRAM sets C_{sample}, V_{in} is sampled to C_{sample}'s top plate through ϕ_{sample}. Then with ϕ_{sample} off and ϕ_{transfer} on, the sampled charge is transferred to C_{sample}'s bottom plate, and redistributed between C_{sample} and C_{out} in one of the o-buffers. In this way, the multiplication of the V_{in} and the weight $w[4{:}0]$ is finished, and the psum is stored as voltage (V_{out}) at C_{out}. Time-multiplexed on/off of the ϕ_{sample}-ϕ_{transfer} with different V_{in} and $w[4{:}0]$ updates the V_{out} cycle by cycle, realizing the consecutive MAC operations, as illustrated in the timing diagram in Fig. 7. Analytically, after the i^{th} cycle of ϕ_{sample}-ϕ_{transfer}, the V_{out} is as Eq. (3):

$$V_{\text{out}}[i] = \frac{C_{\text{sample}}[i]\,(2V_{\text{CM}} - V_{\text{in}}[i]) + C_{\text{out}}V_{\text{out}}[i-1]}{C_{\text{out}} + C_{\text{sample}}[i]} \tag{3}$$

where $V_{\text{out}}[i-1]$ is the voltage on the C_{out} after the first $i-1$ cycles; $V_{\text{in}}[i]$ is the input voltage to SCM at the i^{th} cycle; $C_{\text{sample}}[i]$ is the connected capacitance in the SCM at the i^{th} cycle; and V_{CM} is a constant voltage. The total capacitance of the sampling capacitance is $C_{\text{sample,tot}}$=135fF.

O-buffer. Conventionally, the o-buffer has much larger capacitance ($C_{\text{out}} \gg C_{\text{sample,tot}}$) to reduce the incomplete charge transfer which incurs large area overhead [48]. However, with our hardware-aware training technique introduced earlier (Sec 3.4), extremely-low $\frac{C_{\text{out}}}{C_{\text{sample,total}}}$ ratio can be tolerated, allowing us to set C_{out} to 135fF (ratio=1) to save notable area.

Variable-resolution ADC. The 4 pairs of V_{out} are sequentially quantized by a differential-input ADC. The ADC's resolution is reconfigurable to accommodate different Q_{bit}. When Q_{bit}=1.5-bit (ternary), the differential V_{out} is connected to a ternary comparator (T-CMP) [26]. For higher bit depth ($Q_{\text{bit}} > 1.5$), the differential V_{out} is sampled to a successive approximation register ADC [55] through a pair of FVFs [10]. The ADC is configurable to 8-bit resolution to support normal sensing mode. In the normal mode, after each row exposure, the digital controller enables the pixels to bypass the PE and be quantized by the ADC through four quantization cycles.

4.4 Full System Simulation

We perform full system transistor-level simulation in a standard CMOS 65nm process and and plot the $\{V_{\text{pixel}}, w\}$ against digitized output code to validate the correctness of the system. We choose 65nm, as the CIS technology scaling has to balance the degraded photon sensitivity in smaller pixel size and therefore typically lags behind the more aggressive scaling of the digital process. Since the sign operations are performed on independent o-buffers, the output code with negative weight is central symmetric to the one with positive weight. Without loss of generality, in the simulation, the ADC's resolution is set to 4-bit and the weights are always positive, so the output code ranges from 0 to 7. The simulation results are shown in Fig. 8(a), and the output code correctly changes from 7 to 0 along with increased $\{V_{\text{pixel}}, w\}$. Comparing to the results from analytical circuit model where the PSF/FVF/ADC are linear and

the SCM exactly follows Eq. (3), the absolute error in the actual output codes is within 1 LSB, as show in Fig. 8(b). The absolute error comes from the PSF/FVF/ADC's nonlinearity, the SCM's offset, and the ADC's offset. The ADC's nonlinearity and offset can be easily calibrated digitally, and all other nonidealities are considered in the LeCA hardware-aware training using stage-wise, fine-grained look-up-tables, which will be introduced in Sec. 5.3.

5 EXPERIMENTAL METHODOLOGY

5.1 Baseline Compression Methods

To rigorously evaluate the LeCA method, we compare it with five alternative compression methods, as well as the conventional full-precision sensor:

- **Conventional sensor (CNV)**. Pixel-wise uniform quantization with 8-bit precision.
- **Spatial down-sampling (SD)**. Block-wise spatial averaging of the pixel values with 8-bit uniform quantization.
- **Low-resolution quantizer (LR)**. Pixel-wise uniform quantization with low precision.
- **Compressive sensing (CS) [64]**. Block-based compressive sensing using random matrix for measurement and $\mathcal{L}_0$ normalization for reconstruction.
- **Microshift (MS) [84]**. Fixed value-shifting pattern is performed to each block of pixels, and each pixel is quantized to low resolution.
- **Accumulated gradient thresholding (AGT) [39]**. Pixel gradients are accumulated over the neighboring pixels and the pixels are skipped until the sum crosses a threshold.

We evaluate the task accuracy of all these methods using a *frozen* ResNet-style network as the downstream DNN.

5.2 Datasets and Training

We validate the LeCA algorithm on TinyImageNet [47] and ImageNet [18]. TinyImageNet is a subset of the ImageNet dataset down-sampled to 64×64 with 200 classes. For TinyImageNet, we use a random rotation of 20 degrees and random horizontal flipping during training. We use PyTorch [66] with its provided pre-trained weights for ResNet-like [28] downstream DNN models. We use Adam [43] as the optimizer to train LeCA for 100 epochs for TinyImageNet and 25 epochs for ImageNet while keeping the downstream weights frozen. We start the learning rate at 10^{-3} and decay it by a factor of 0.1 every 30 epochs for tiny, and every 10 epochs for ImageNet with a batch size of 256.

5.3 Hardware Non-idealities Modeling

Comprehensive noise sources and non-ideality effects in the LeCA sensor are modeled and added to the training pipeline to fine-tune the pre-trained LeCA encoder and decoder.

Pixel array noise. The pixel array noise is added to the images to emulate real CIS sensing effect, including shot noise and read noise, which are formulated as Poisson and Gaussian distribution, respectively. We first convert the digital image to its voltage intensity, add the equivalent noise in the voltage domain, and finally convert it back to the digital image.

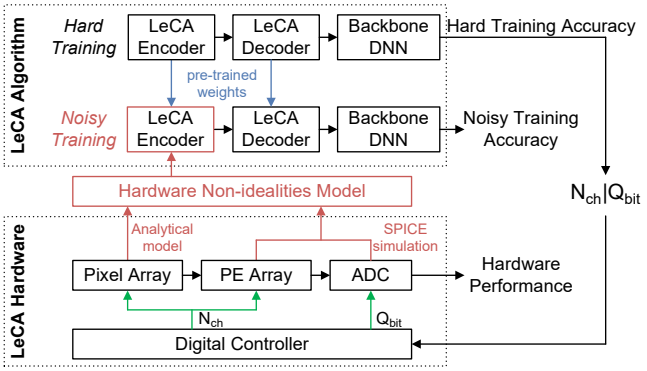

Figure 9: Evaluation methodology of LeCA system.

Analog circuit non-ideality. The analog non-ideality includes three parts, starting from pixel readout to ADC:

❶ PSF's non-linear transfer function and mismatch. 200-sample Monte-Carlo simulation is conducted to obtain the PSF's transfer function with mismatch variation incorporated. The PSF's readout effect is thus modeled as a look-up-table (LUT) with input-related Gaussian disturbance: $V_{\text{in}}[i] = \mathcal{N}(\text{LUT}_{\text{PSF}}(V_{\text{pixel}}[i]), \sigma_{\text{PSF}}[i])$.

❷ SCM's incomplete charge transfer and mismatch. The SCM's output is calculated by the ideal analytical model (Eq. (3), LUT_{SCM}) and superimposed by an input/weight-related error term with Gaussian disturbance, which is obtained from 200-sample Monte-Carlo simulation. The SC multiplier's computation effect is thus modeled as: $V_{\text{out}}[i] = \text{LUT}_{\text{SCM}}(V_{\text{in}}[i], \text{weight}[i]) - \mathcal{N}(\epsilon_{\text{SCM}}[i], \sigma_{\text{SCM}}[i])$.

❸ FVF's non-linear transfer function and mismatch. Similar to the PSF, the FVF's readout effect is modeled as a LUT with input-related Gaussian disturbance: $V_{\text{ADC}}[i] = \mathcal{N}\left(\text{LUT}_{\text{FVF}}\left(V_{\text{out}}[i]\right), \sigma_{\text{FVF}}[i]\right)$.

Our hardware non-ideality model lumps the effects of time-invariant process variations (e.g. spatial mismatch) with time-variant fluctuations due to supply, temperature, and aging as random statistical variables. In this way, we comprehensively capture all non-ideal behaviors in the training process without the need to retrain for each hardware instantiation.

5.4 System-level Accuracy and Performance Analysis

Our evaluation methodology is shown in Fig. 9. First, the performance of LeCA algorithm is validated by *hard training*, from which we determine the value/range of $N_{\text{ch}}|Q_{\text{bit}}$ that achieves optimal tradeoff between compression ratio and downstream accuracy. Second, the LeCA hardware is configured by the $N_{\text{ch}}|Q_{\text{bit}}$ from the *hard training*. Specifically, the N_{ch} is used to configure the pixel array and the PE array for repetitive readout, and the Q_{bit} is used to configure the ADC's quantization resolution. The hardware performance is then evaluated through transistor-level simulation and the hardware non-idealities model is extracted. Third, the *noisy training* is set up to get *noisy training* accuracy by initializing with the pre-trained weights from the *hard training* and incorporating the extracted hardware non-idealities model. Note that the parameters of the backbone DNN are always frozen.

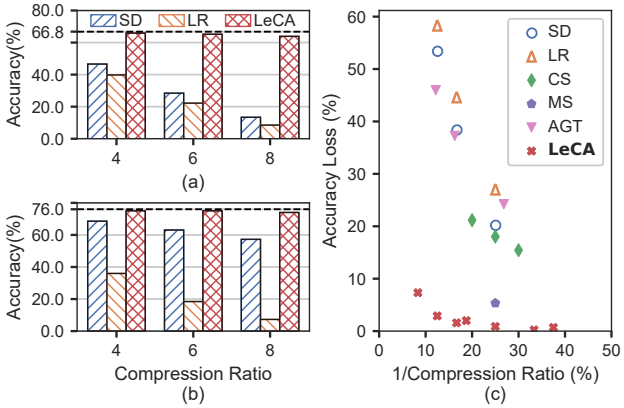

Figure 10: Downstream classification accuracy comparison on (a) proxy and (b) ImageNet for CR ∈ {4, 6, 8} with SD, LR and LeCA. (c) Accuracy loss vs. compression tradeoff based on the proxy pipeline.

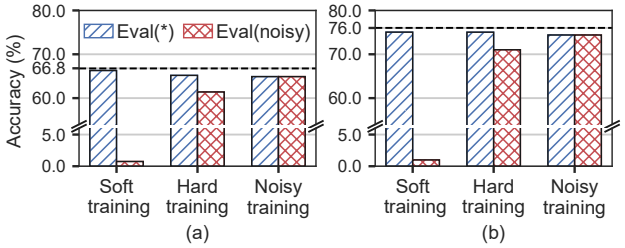

Figure 11: Accuracy of different training modes with respect to hardware non-idealities for (a) proxy and (b) ImageNet. With our rigorous hardware-aware training, most of the lost accuracy is recovered.

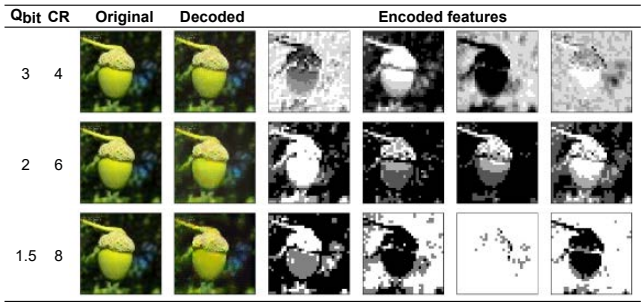

Figure 12: Visualization of encoded (last 4) and decoded features.

6 EVALUATION

In this section, we thoroughly evaluate the LeCA method on its accuracy, noise resilience, and sensor performance. Additionally, we provide an in-depth discussion about task accuracy improvement, image resolution flexibility, standard compression comparison and system deployment.

6.1 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, SD and LR are typical methods to remove the redundancy in the spatial domain and bit-depth resolution, respectively. In Fig. 10(a) and (b), we compare our LeCA method 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 upsampling through bilinear interpolation to acquire compression ratios of 4, 6, and 8, respectively (using Eq. (1)). 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 predecessors 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 important 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. 10(c) shows a more thorough comparison of LeCA with its counterparts on the proxy pipeline. 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 and CS 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 trivial because all models perform lossy compression, meaning that increasing information is irrevocably lost with higher compression.

6.2 Hardware-aware Noise Resilience

Fig. 11 shows the performance of different training modalities - *soft*, *hard* and *noisy* (see Sec. 3.4). The goal is to acquire the best hardware-aware accuracy for a given compression ratio. The left - *Eval(*)* and right - *Eval(noisy)* bars show the validation accuracy on the corresponding modalities and on the full hardware with non-idealities, respectively. *Soft* training clearly shows good performance with <1% accuracy loss on both datasets. However, when learned *soft* weights are mapped to the *hard* (analytical hardware) modality, there is a large accuracy drop. This implies that there is no trivial method to map *soft* to *hard*. With *hard* training we get accuracies that are on par with the purely *soft* training (-1.1% and -0.01% accuracy loss for proxy and ImageNet with respect to *soft* respectively). However, we lose some accuracy (~4% in both cases) when we use these learned weights and map them to the *noisy* modality. The final pair of bars (*noisy training*) show that finetuning the *hard* model helps recover most of the accuracy lost due to noise. We find that directly training the *hard* model is insufficient for LeCA to converge to a good optimum on the *noisy* modality and that finetuning is indeed important. Overall, this highlights the novelty and noise-resilience of our training method, where even with most of the hardware non-idealities, we are able to achieve negligible accuracy loss.

Fig. 12 shows a sample image from TinyImageNet, which demonstrates qualitatively what the encoded and decoded features look like. A key observation of this visualization is that despite LeCA encoder-decoder being trained on cross-entropy loss to maximize

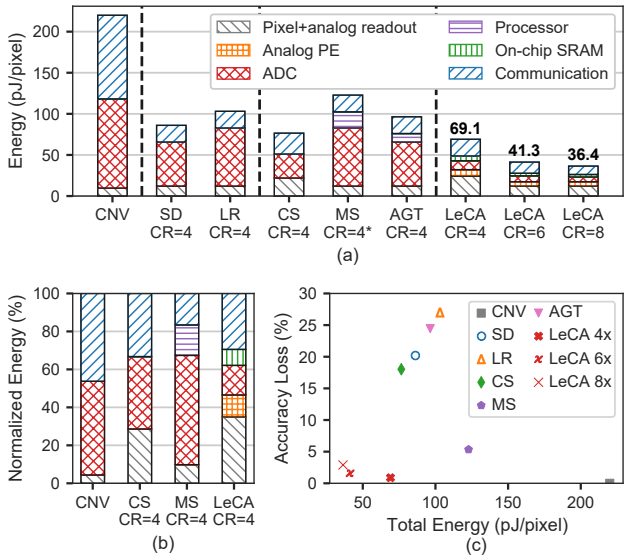

Figure 13: Energy and accuracy comparison between conventional, compressive, and LeCA sensors: (a) absolute energy; (b) relative energy normalized to LeCA (CR=4); (c) tradeoff between sensor energy consumption and accuracy loss on the proxy pipeline. *-MS's compression is image dependent, varying between 4× and 5×.

the downstream accuracy, the decoded image structurally looks similar to the original image. The visual quality decays as more aggressive quantization is used.

6.3 Sensor System Evaluation

As shown in Fig. 13(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 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) under the same compression ratio, LeCA sensor consumes 11%, 57%, and 31% less energy, respectively. Fig. 13(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 [20]. 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. 13(c) shows the tradeoff between sensor chip energy and 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. LeCA encoder circuit occupies $1.1mm^2$ ($0.85mm^2$ is ADC area) based on layout estimate. Considering that the conventional CIS would minimally include the pixel array ($5mm^2$ for $5\mu m$ pitched pixel) and the ADC, the area overhead of LeCA is less than 5%.

6.4 Discussion and Future Work

Task accuracy. Our evaluations assume a downstream model with frozen weights. However, by unfreezing the downstream model, experiments show 0.02% and 0.78% accuracy loss for 4× and 8× compression, respectively, which are well within standard metrics (<1%) proposed by benchmarks like MLPerf[69]. This suggests that there are avenues to further improve and close the gap with the state-of-the-art task accuracy, at the cost of longer training time and the complexity to adapt the weights of entire vision pipeline.
Image resolution. Both LeCA's hardware and algorithm support higher-resolution images. LeCA adopts column-parallel PE arrays where the physical width of the PE and ADC matches four pixel columns, allowing LeCA to scale with the pixel array width to accommodate higher-resolution inputs. We also validate that LeCA can achieve up to 86fps frame rate with 1080p resolution, comfortably supporting moving-object recording (at 60fps). Our results indicate comparable performance from TinyImageNet (64×64) and ImageNet (224×224), suggesting LeCA algorithm works across varying resolutions and the trend continues for high-resolution inputs.
Standard compression. Our evaluation has included standard compression techniques such as spatial gradients (AGT), run-length encoding (MS), and quantization (LR). We also perform JPEG and see a 0.51% accuracy loss at 5.07× compression, with LeCA's 0.98% accuracy at 6×. Critically, we emphasize that LeCA achieves outstanding compression/accuracy performance on top of sizable sensor energy saving, while standard digital compression invariably requires significant additional hardware and energy to perform.
System deployment. LeCA can adapt to downstream tasks beyond image classification by following the same training/finetuning process with no change to the hardware. It allows for configurable number of feature channels and quantization levels to provide flexible compression/accuracy tradeoff. The trained encoding parameters instantiated in the PE are re-programmable according to the downstream task. Therefore, it is a practical solution with broad applications. Intuitively, capturing a smaller image directly translates to reduced memory storage and fewer communication and computing powers in the later processing stages of the machine vision pipeline. Nonetheless, we limit the scope of our study to the sensor chip and leave the full system analysis for future work.

7 RELATED WORKS

Computational CIS for DNN. Many prior computational CIS works offload the first layer [32, 81] or the first few layers [31, 53] of a DNN in the sensor chip. However, these works do not explicitly leverage data compression brought by the DNN offloading to improve the sensor's energy efficiency. Recent study has also considered optimizing the ISP design jointly with the downstream CV tasks[7, 27, 80]. Nonetheless, these computational approaches after

CIS digitization do not translate to sensor resource/energy savings, whereas LeCA sensor implements an encoder to highly compress the data in the analog domain, which results in compression-dependent extremely-low energy consumption.

In-sensor compression. Common in-sensor compression works either adopt heuristic compression method [39, 84, 88] or are based on compressive sensing [63, 64]. However, these compression methods are task-agnostic and the compression ratio is related to the PSNR of the reconstructed images and independent of downstream tasks. Instead, LeCA sensor encodes the image to a task-specific representation so that there exists a clear tradeoff between the compression ratio and the downstream task accuracy.

Learned compression. Learning-based image compression has been increasingly popular since the advent of neural networks [21, 36]. Most models use an autoencoder structure that can compress an image by 80~100× while maintaining high visual quality [14, 60, 90]. Adjacent to the autoencoder approach, generative adversarial networks [1, 23] have been proposed to synthesize details the model cannot afford to store. However, these models use computation-intensive encoder networks which are infeasible to be incorporated into the CIS. LeCA also uses an encoder-decoder structure but implements the encoder in the analog domain with extremely-low overhead while also maintaining the decoder in lightweight.

8 CONCLUSION

We propose a new in-sensor processing paradigm – LeCA – that targets machine vision applications on the edge. By jointly learning the sensor acquisition function with the downstream CV algorithms, LeCA effectively compresses the original image into informative condensed feature maps. Co-designed with LeCA algorithm, LeCA sensor adopts analog-domain in-sensor processing to translate the compression into meaningful hardware savings. Evaluated on ImageNet, LeCA shows both high compression ratio (6×) and minimal accuracy loss (0.98%). Transistor-level simulation shows LeCA sensor is 6.3× and 2.2× more energy efficient than conventional sensor and compressive sensing sensor with negligible area overhead.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. Tianrui Ma, Adith Jagadish Boloor, Weidong Cao, and Xuan Zhang were partially supported by NSF CCF-1942900 and NSF CNS-1739643.

REFERENCES

[1] Eirikur Agustsson, Michael Tschannen, Fabian Mentzer, Radu Timofte, and Luc Van Gool. 2019. Generative adversarial networks for extreme learned image compression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 221–231.

[2] Alessandro Aimar, Hesham Mostafa, Enrico Calabrese, Antonio Rios-Navarro, Ricardo Tapiador-Morales, Iulia-Alexandra Lungu, Moritz B Milde, Federico Corradi, Alejandro Linares-Barranco, Shih-Chii Liu, et al. 2018. Nullhop: A flexible convolutional neural network accelerator based on sparse representations of feature maps. *IEEE transactions on neural networks and learning systems* 30, 3 (2018), 644–656.

[3] Evgeny Belyaev, Kai Liu, Moncef Gabbouj, and YunSong Li. 2014. An efficient adaptive binary range coder and its VLSI architecture. *IEEE transactions on circuits and systems for video technology* 25, 8 (2014), 1435–1446.

[4] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432* (2013).

[5] Wissam Benjilali, William Guicquero, Laurent Jacques, and Gilles Sicard. 2019. An analog-to-information vga image sensor architecture for support vector machine on compressive measurements. In *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–5.

[6] Kyeongryeol Bong, Sungpill Choi, Changhyeon Kim, Donghyeon Han, and Hoi-Jun Yoo. 2017. A low-power convolutional neural network face recognition processor and a CIS integrated with always-on face detector. *IEEE Journal of Solid-State Circuits* 53, 1 (2017), 115–123.

[7] Mark Buckler, Suren Jayasuriya, and Adrian Sampson. 2017. Reconfiguring the imaging pipeline for computer vision. In *Proceedings of the IEEE International Conference on Computer Vision*. 975–984.

[8] Shijie Cao, Lingxiao Ma, Wencong Xiao, Chen Zhang, Yunxin Liu, Lintao Zhang, Lanshun Nie, and Zhi Yang. 2019. Seernet: Predicting convolutional neural network feature-map sparsity through low-bit quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11216–11225.

[9] Weidong Cao, Yilong Zhao, Adith Boloor, Yinhe Han, Xuan Zhang, and Li Jiang. 2021. Neural-PIM: Efficient processing-in-memory with neural approximation of peripherals. *IEEE Trans. Comput.* 71, 9 (2021), 2142–2155.

[10] Ramón González Carvajal, Jaime Ramírez-Angulo, Antonio J López-Martín, Antonio Torralba, Juan Antonio Gómez Galán, Alfonso Carlosena, and Fernando Muñoz Chavero. 2005. The flipped voltage follower: A useful cell for low-voltage low-power circuit design. *IEEE Transactions on Circuits and Systems I: Regular Papers* 52, 7 (2005), 1276–1291.

[11] Denis Guangyin Chen, Fang Tang, Man-Kay Law, and Amine Bermak. 2014. A 12 pJ/pixel analog-to-information converter based 816× 640 pixel CMOS image sensor. *IEEE Journal of Solid-State Circuits* 49, 5 (2014), 1210–1222.

[12] Denis Guangyin Chen, Fang Tang, Man-Kay Law, Xiaopeng Zhong, and Amine Bermak. 2014. A 64 fJ/step 9-bit SAR ADC array with forward error correction and mixed-signal CDS for CMOS image sensors. *IEEE Transactions on Circuits and Systems I: Regular Papers* 61, 11 (2014), 3085–3093.

[13] Zhe Chen, Huifeng Zhu, Erxiang Ren, Zheyu Liu, Kaige Jia, Li Luo, Xuan Zhang, Qi Wei, Fei Qiao, Xinjun Liu, et al. 2019. Processing near sensor architecture in mixed-signal domain with CMOS image sensor of convolutional-kernel-readout method. *IEEE Transactions on Circuits and Systems I: Regular Papers* 67, 2 (2019), 389–400.

[14] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. 2018. Deep convolutional autoencoder-based lossy image compression. In *2018 Picture Coding Symposium (PCS)*. IEEE, 253–257.

[15] Jaehyuk Choi, Seokjun Park, Jihyun Cho, and Euisik Yoon. 2015. An energy/illumination-adaptive CMOS image sensor with reconfigurable modes of operations. *IEEE Journal of Solid-State Circuits* 50, 6 (2015), 1438–1450.

[16] Jaehyuk Choi, Jungsoon Shin, Dongwu Kang, and Du-Sik Park. 2015. Always-on CMOS image sensor for mobile and wearable devices. *IEEE Journal of Solid-State Circuits* 51, 1 (2015), 130–140.

[17] K. D. Choo, L. Xu, Y. Kim, J. Seol, X. Wu, D. Sylvester, and D. Blaauw. 2019. Energy-Efficient Motion-Triggered IoT CMOS Image Sensor With Capacitor Array-Assisted Charge-Injection SAR ADC. *IEEE Journal of Solid-State Circuits* 54, 11 (2019), 2921–2931.

[18] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[19] Piotr Dollár, Mannat Singh, and Ross Girshick. 2021. Fast and accurate model scaling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 924–932.

[20] Marco F Duarte, Mark A Davenport, Dharmpal Takhar, Jason N Laska, Ting Sun, Kevin F Kelly, and Richard G Baraniuk. 2008. Single-pixel imaging via compressive sampling. *IEEE signal processing magazine* 25, 2 (2008), 83–91.

[21] Michael Egmont-Petersen, Dick de Ridder, and Heinz Handels. 2002. Image processing with neural networks—a review. *Pattern recognition* 35, 10 (2002), 2279–2301.

[22] Sven Fleck and Wolfgang Straßer. 2008. Smart camera based monitoring system and its application to assisted living. *Proc. IEEE* 96, 10 (2008), 1698–1714.

[23] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Commun. ACM* 63, 11 (2020), 139–144.

[24] Jie Gui, Zhenan Sun, Shuiwang Ji, Dacheng Tao, and Tieniu Tan. 2016. Feature selection based on structured sparsity: A comprehensive study. *IEEE transactions on neural networks and learning systems* 28, 7 (2016), 1490–1507.

[25] Bahadir K Gunturk, John Glotzbach, Yucel Altunbasak, Ronald W Schafer, and Russel M Mersereau. 2005. Demosaicking: color filter array interpolation. *IEEE Signal processing magazine* 22, 1 (2005), 44–54.

[26] Wenjuan Guo and Nan Sun. 2016. A 12b-ENOB 61μW noise-shaping SAR ADC with a passive integrator. In *ESSCIRC Conference 2016: 42nd European Solid-State Circuits Conference*. IEEE, 405–408.

[27] Patrick Hansen, Alexey Vilkin, Yury Khrustalev, James Imber, David Hanwell, Matthew Mattina, and Paul N Whatmough. 2019. ISP4ML: Understanding the role of image signal processing in efficient deep learning vision systems. *arXiv preprint arXiv:1911.07954* (2019).

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[29] Felix Heide, Markus Steinberger, Yun-Ta Tsai, Mushfiqur Rouf, Dawid Pająk, Dikpal Reddy, Orazio Gallo, Jing Liu, Wolfgang Heidrich, Karen Egiazarian, et al. 2014. Flexisp: A flexible camera image processing framework. *ACM Transactions on Graphics (ToG)* 33, 6 (2014), 1–13.

[30] Alain Hore and Djemel Ziou. 2010. Image quality metrics: PSNR vs. SSIM. In *2010 20th international conference on pattern recognition*. IEEE, 2366–2369.

[31] Tzu-Hsiang Hsu, Guan-Cheng Chen, Yi-Ren Chen, Chung-Chuan Lo, Ren-Shuo Liu, Meng-Fan Chang, Kea-Tiong Tang, and Chih-Cheng Hsieh. 2022. A 0.8 V intelligent vision sensor with tiny convolutional neural network and programmable weights using mixed-mode processing-in-sensor technique for image classification. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, Vol. 65. IEEE, 1–3.

[32] Tzu-Hsiang Hsu, Yi-Ren Chen, Ren-Shuo Liu, Chung-Chuan Lo, Kea-Tiong Tang, Meng-Fan Chang, and Chih-Cheng Hsieh. 2020. A 0.5-V real-time computational CMOS image sensor with programmable kernel for feature extraction. *IEEE Journal of Solid-State Circuits* 56, 5 (2020), 1588–1596.

[33] Yueyu Hu, Wenhan Yang, Zhan Ma, and Jiaying Liu. 2021. Learning end-to-end lossy image compression: A benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[34] Sun-Il Hwang, Jae-Hyun Chung, Hyeon-June Kim, Il-Hoon Jang, Min-Jae Seo, Sang-Hyun Cho, Heewon Kang, Minho Kwon, and Seung-Tak Ryu. 2018. A 2.7-M pixels 64-mW CMOS image sensor with multicolumn-parallel noise-shaping SAR ADCs. *IEEE Transactions on Electron Devices* 65, 3 (2018), 1119–1126.

[35] Laurent Jacques, Pierre Vandergheynst, Alexandre Bibet, Vahid Majidzadeh, Alexandre Schmid, and Yusuf Leblebici. 2009. CMOS compressed imaging by random convolution. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 1113–1116.

[36] J Jiang. 1999. Image compression with neural networks–a survey. *Signal processing: image Communication* 14, 9 (1999), 737–760.

[37] Yun-Rae Jo, Seong-Kwan Hong, and Oh-Kyong Kwon. 2015. A multi-bit incremental ADC based on successive approximation for low noise and high resolution column-parallel readout circuits. *IEEE Transactions on Circuits and Systems I: Regular Papers* 62, 9 (2015), 2156–2166.

[38] Vinay Joshi, Manuel Le Gallo, Simon Haefeli, Irem Boybat, Sasidharan Rajalekshmi Nandakumar, Christophe Piveteau, Martino Dazzi, Bipin Rajendran, Abu Sebastian, and Evangelos Eleftheriou. 2020. Accurate deep neural network inference using computational phase-change memory. *Nature communications* 11, 1 (2020), 2473.

[39] Amandeep Kaur, Deepak Mishra, KM Amogh, and Mukul Sarkar. 2020. On-array compressive acquisition in cmos image sensors using accumulated spatial gradients. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 2 (2020), 523–532.

[40] Masaya Kawano, Xiangy-Yu Wang, and Qin Ren. 2019. New Cost-Effective Via-Last Approach by" One-Step TSV" After Wafer Stacking for 3D Memory Applications. In *2019 IEEE 69th Electronic Components and Technology Conference (ECTC)*. IEEE, 1996–2002.

[41] Hyeon-June Kim. 2021. 11-bit column-parallel single-slope ADC with first-step half-reference ramping scheme for high-speed CMOS image sensors. *IEEE Journal of Solid-State Circuits* 56, 7 (2021), 2132–2141.

[42] Hyeon-June Kim, Sun-Il Hwang, Ji-Wook Kwon, Dong-Hwan Jin, Byoung-Soo Choi, Sang-Gwon Lee, Jong-Ho Park, Jang-Kyoo Shin, and Seung-Tak Ryu. 2016. A delta-readout scheme for low-power CMOS image sensors with multi-column-parallel SAR ADCs. *IEEE Journal of Solid-State Circuits* 51, 10 (2016), 2262–2273.

[43] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[44] Diederik P Kingma, Max Welling, et al. 2019. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning* 12, 4 (2019), 307–392.

[45] Minho Kwon, Seunghyun Lim, Hyeokjong Lee, Il-Seon Ha, Moo-Young Kim, Il-Jin Seo, Suho Lee, Yongsuk Choi, Kyunghoon Kim, Hansoo Lee, et al. 2020. A Low-Power 65/14nm Stacked CMOS Image Sensor. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–4.

[46] Nicholas D Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew T Campbell. 2010. A survey of mobile phone sensing. *IEEE Communications magazine* 48, 9 (2010), 140–150.

[47] Ya Le and Xuan Yang. 2015. Tiny imagenet visual recognition challenge. *CS 231N* 7, 7 (2015), 3.

[48] E. H. Lee and S. S. Wong. 2017. Analysis and Design of a Passive Switched-Capacitor Matrix Multiplier for Approximate Computing. *IEEE Journal of Solid-State Circuits* 52, 1 (2017), 261–271.

[49] Hyunkeun Lee, Woo-Tae Kim, Jinho Kim, Myonglae Chu, and Byung-Geun Lee. 2020. A compressive sensing CMOS image sensor with partition sampling technique. *IEEE Transactions on Industrial Electronics* 68, 9 (2020), 8874–8884.

[50] Hyunkeun Lee, Donghwan Seo, Woo-Tae Kim, and Byung-Geun Lee. 2017. A Compressive Sensing-Based CMOS Image Sensor With Second-Order ΣΔ ADCs. *IEEE Sensors Journal* 18, 6 (2017), 2404–2410.

[51] Junan Lee, Himchan Park, Bongsub Song, Kiwoon Kim, Jaeha Eom, Kyunghoon Kim, and Jinwook Burm. 2015. High frame-rate VGA CMOS image sensor using non-memory capacitor two-step single-slope ADCs. *IEEE Transactions on Circuits and Systems I: Regular Papers* 62, 9 (2015), 2147–2155.

[52] Sheng Li, Zhong Ma, Zhonglin Cao, Lijia Pan, and Yi Shi. 2020. Advanced wearable microfluidic sensors for healthcare monitoring. *Small* 16, 9 (2020), 1903822.

[53] R. LiKamWa, Y. Hou, Y. Gao, M. Polansky, and L. Zhong. 2016. RedEye: Analog ConvNet Image Sensor Architecture for Continuous Mobile Vision. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. 255–266.

[54] Chiao Liu, Song Chen, Tsung-Hsun Tsai, Barbara De Salvo, and Jorge Gomez. 2022. Augmented Reality-The Next Frontier of Image Sensors and Compute Systems. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, Vol. 65. IEEE, 426–428.

[55] Chun-Cheng Liu, Soon-Jyh Chang, Guan-Ying Huang, and Ying-Zu Lin. 2010. A 10-bit 50-MS/s SAR ADC with a monotonic capacitor switching procedure. *IEEE Journal of Solid-State Circuits* 45, 4 (2010), 731–740.

[56] Zheyu Liu, Erxiang Ren, Fei Qiao, Qi Wei, Xinjun Liu, Li Luo, Huichan Zhao, and Huazhong Yang. 2020. NS-CIM: A current-mode computation-in-memory architecture enabling near-sensor processing for intelligent IoT vision nodes. *IEEE Transactions on Circuits and Systems I: Regular Papers* 67, 9 (2020), 2909–2922.

[57] Yi Luo and Shahriar Mirabbasi. 2017. Always-on CMOS image sensor pixel design for pixel-wise binary coded exposure. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–4.

[58] Siwei Ma, Xinfeng Zhang, Chuanmin Jia, Zhenghui Zhao, Shiqi Wang, and Shanshe Wang. 2019. Image and video compression with neural networks: A review. *IEEE Transactions on Circuits and Systems for Video Technology* 30, 6 (2019), 1683–1698.

[59] Vahid Majidzadeh, Laurent Jacques, Alexandre Schmid, Pierre Vandergheynst, and Yusuf Leblebici. 2010. A (256× 256) pixel 76.7 mW CMOS imager/compressor based on real-time in-pixel compressive sensing. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. IEEE, 2956–2959.

[60] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. 2018. Conditional probability models for deep image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4394–4402.

[61] Laurent Millet, Stephane Chevobbe, Caaliph Andriamisaina, Lamine Benaissa, Edouard Deschaseaux, Edith Beigne, Karim Ben Chehida, Maria Lepecq, Mehdi Darouich, Fabrice Guellec, et al. 2019. A 5500-frames/s 85-gops/w 3-d stacked bsi vision chip based on parallel in-focal-plane acquisition and processing. *IEEE Journal of Solid-State Circuits* 54, 4 (2019), 1096–1105.

[62] David Minnen, Johannes Ballé, and George D Toderici. 2018. Joint autoregressive and hierarchical priors for learned image compression. *Advances in neural information processing systems* 31 (2018).

[63] Yusuke Oike and Abbas El Gamal. 2012. CMOS image sensor with per-column ΣΔ ADC and programmable compressed sensing. *IEEE Journal of Solid-State Circuits* 48, 1 (2012), 318–328.

[64] Chanmin Park, Wenda Zhao, Injun Park, Nan Sun, and Youngcheol Chae. 2021. 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* 56, 8 (2021), 2503–2515.

[65] Injun Park, Woojin Jo, Chanmin Park, Byungchoul Park, Jimin Cheon, and Youngcheol Chae. 2019. A 640×640 Fully Dynamic CMOS Image Sensor for Always-On Operation. *IEEE Journal of Solid-State Circuits* 55, 4 (2019), 898–907.

[66] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[67] Tommaso Polonelli, Daniele Battistini, Manuele Rusci, Davide Brunelli, and Luca Benini. 2020. An energy optimized jpeg encoder for parallel ultra-low-power processing-platforms. In *Applications in Electronics Pervading Industry, Environment and Society: APPLEPIES 2019 7*. Springer, 125–133.

[68] Christoph Posch, Daniel Matolin, and Rainer Wohlgenannt. 2010. A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS. *IEEE Journal of Solid-State Circuits* 46, 1 (2010), 259–275.

[69] Vijay Janapa Reddi, Christine Cheng, David Kanter, Peter Mattson, Guenther Schmuelling, Carole-Jean Wu, Brian Anderson, Maximilien Breughe, Mark Charlebois, William Chou, et al. 2020. Mlperf inference benchmark. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 446–459.

[70] Ryan Robucci, Jordan D Gray, Leung Kin Chiu, Justin Romberg, and Paul Hasler. 2010. Compressive sensing on a CMOS separable-transform image sensor. *Proc. IEEE* 98, 6 (2010), 1089–1101.

[71] Amir Said and William A Pearlman. 1996. An image multiresolution representation for lossless and lossy compression. *IEEE Transactions on image processing* 5, 9 (1996), 1303–1310.

[72] Min-Woong Seo, Myunglae Chu, Hyun-Yong Jung, Suksan Kim, Jiyoun Song, Junan Lee, Sung-Yong Kim, Jongyeon Lee, Sung-Jae Byun, Daehee Bae, et al. 2021. A 2.6 e-rms low-random-noise, 116.2 mW low-power 2-Mp global shutter CMOS image sensor with pixel-level ADC and in-pixel memory. In *2021 Symposium on VLSI Technology*. IEEE, 1–2.

[73] Min-Seok Shin, Jong-Boo Kim, Min-Kyu Kim, Yun-Rae Jo, and Oh-Kyong Kwon. 2012. A 1.92-megapixel CMOS image sensor with column-parallel low-power and area-efficient SA-ADCs. *IEEE Transactions on Electron Devices* 59, 6 (2012), 1693–1700.

[74] Rituraj Singh, Stevo Bailey, Phillip Chang, Ashkan Olyaei, Mohammad Hekmat, and Renaldi Winoto. 2021. 34.2 a 21pJ/frame/pixel imager and 34pJ/frame/pixel image processor for a low-vision augmented-reality smart contact lens. In *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, Vol. 64. IEEE, 482–484.

[75] H Tsugawa, H Takahashi, R Nakamura, T Umebayashi, T Ogita, H Okano, K Iwase, H Kawashima, T Yamasaki, D Yoneyama, et al. 2017. Pixel/DRAM/logic 3-layer stacked CMOS image sensor technology. In *2017 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 3–2.

[76] Vincent C Venezia, Alan Chih-Wei Hsiung, Wu-Zang Yang, Yuying Zhang, Cheng Zhao, Zhiqiang Lin, and Lindsay A Grant. 2018. Second generation small pixel technology using hybrid bond stacking. *Sensors* 18, 2 (2018), 667.

[77] L Verger, MC Gentet, L Gerfault, R Guillemaud, C Mestais, O Monnet, G Montemont, G Petroz, JP Rostaing, and J Rustique. 2004. Performance and perspectives of a CdZnTe-based gamma camera for medical imaging. *IEEE Transactions on Nuclear Science* 51, 6 (2004), 3111–3117.

[78] Gregory K Wallace. 1991. The JPEG still picture compression standard. *Commun. ACM* 34, 4 (1991), 30–44.

[79] Jiangtao Wen and John D Villasenor. 1997. A class of reversible variable length codes for robust image and video coding. In *Proceedings of International Conference on Image Processing*, Vol. 2. IEEE, 65–68.

[80] Chyuan-Tyng Wu, Leo F Isikdogan, Sushma Rao, Bhavin Nayak, Timo Gerasimow, Aleksandar Sutic, Liron Ain-kedem, and Gilad Michael. 2019. VisionISP: Repurposing the image signal processor for computer vision applications. In *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 4624–4628.

[81] Han Xu, Ningchao Lin, Li Luo, Qi Wei, Runsheng Wang, Cheng Zhuo, Xunzhao Yin, Fei Qiao, and Huazhong Yang. 2021. 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* 69, 1 (2021), 232–243.

[82] Christopher Young, Alex Omid-Zohoor, Pedram Lajevardi, and Boris Murmann. 2019. 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* 54, 11 (2019), 2932–2946.

[83] Bin Zhang, Kuizhi Mei, and Nanning Zheng. 2012. Reconfigurable processor for binary image processing. *IEEE transactions on circuits and systems for video technology* 23, 5 (2012), 823–831.

[84] Bo Zhang, Pedro V Sander, Chi-Ying Tsui, and Amine Bermak. 2018. Microshift: An efficient image compression algorithm for hardware. *IEEE Transactions on Circuits and Systems for Video Technology* 29, 11 (2018), 3430–3443.

[85] Jian Zhang, Chen Zhao, Debin Zhao, and Wen Gao. 2014. Image compressive sensing recovery using adaptively learned sparsifying basis via L0 minimization. *Signal Processing* 103 (2014), 114–126.

[86] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. 2017. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE transactions on image processing* 26, 7 (2017), 3142–3155.

[87] Milin Zhang and Amine Bermak. 2009. Compressive acquisition CMOS image sensor: from the algorithm to hardware implementation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 18, 3 (2009), 490–500.

[88] Milin Zhang and Amine Bermak. 2010. CMOS image sensor with on-chip image compression: A review and performance analysis. *Journal of Sensors* 2010 (2010).

[89] Chuteng Zhou, Fernando Garcia Redondo, Julian Büchel, Irem Boybat, Xavier Timoneda Comas, SR Nandakumar, Shidhartha Das, Abu Sebastian, Manuel Le Gallo, and Paul N Whatmough. 2021. Analognets: ML-HW co-design of noise-robust TinyML models and always-on analog compute-in-memory accelerator. *arXiv preprint arXiv:2111.06503* (2021).

[90] Lei Zhou, Chunlei Cai, Yue Gao, Sanbao Su, and Junmin Wu. 2018. Variational autoencoder for low bit-rate image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2617–2620.

[91] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. 2016. Trained ternary quantization. *arXiv preprint arXiv:1612.01064* (2016).