SNAPPIX: Efficient-Coding–Inspired In-Sensor Compression for Edge Vision

Weikai Lin[1,*], Tianrui Ma[2,3,*], Adith Boloor[3], Yu Feng[4,#], Ruofan Xing[5], Xuan Zhang[3,5], Yuhao Zhu[1]
[1]University of Rochester [2]Institute of Computing Technology, CAS [3]Washington University in St. Louis
[4]Shanghai Jiao Tong University [5]Northeastern University
wlin33@ur.rochester.edu, matianrui@ict.ac.cn, adith@wustl.edu, y-feng@sjtu.edu.cn,
{xing.ruo, xuan.zhang}@northeastern.edu, yzhu@rochester.edu

2025 62nd ACM/IEEE Design Automation Conference (DAC) | 979-8-3315-0304-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/DAC63849.2025.11132672

***Abstract*—Energy-efficient image acquisition on the edge is crucial for enabling remote sensing applications where the sensor node has weak compute capabilities and must transmit data to a remote server/cloud for processing. To reduce the edge energy consumption, this paper proposes a sensor-algorithm co-designed system called SNAPPIX, which compresses raw pixels in the analog domain inside the sensor. We use coded exposure (CE) as the in-sensor compression strategy as it offers the flexibility to sample, i.e., selectively expose pixels, both spatially and temporally. SNAPPIX has three contributions. First, we propose a task-agnostic strategy to learn the sampling/exposure pattern based on the classic theory of efficient coding. Second, we co-design the downstream vision model with the exposure pattern to address the pixel-level non-uniformity unique to CE-compressed images. Finally, we propose lightweight augmentations to the image sensor hardware to support our in-sensor CE compression. Evaluating on action recognition and video reconstruction, SNAPPIX outperforms state-of-the-art video-based methods at the same speed while reducing the energy by up to 15.4×. We have open-sourced the code at: https://github.com/horizon-research/SnapPix.**

I. INTRODUCTION

Energy-efficient imaging is essential for sensing scenarios where the sensor node has weak compute capability and must transmit data to a nearby edge server or even cloud for processing. This includes scenarios such as traffic monitoring [1], remote satellite sensing [2], or even smartphone/extended reality devices that offload computations [3]. In all these scenarios, the sensing node energy is dominated by two components: 1) the image sensor itself, which is in turn dominated by the readout circuitry [4], [5] , and 2) the wireless data transmission from the sensing node, whose energy is known to outweigh computation and increases with transmission distance [6].

To address this energy bottleneck, recent research has explored in-sensor compression, where raw pixels are aggressively sampled before being digitized and transmitted out of the sensor [4], [5], [7]–[9]. While promising, in-sensor compression faces two challenges.

- First, it must address a fundamental dilemma, where the sensor output is both an *energy* bottleneck and an *information* bottleneck, which are directly in contention with each other. Reducing energy consumption requires aggressive in-sensor compression, but over-compression can lead to significant losses in task accuracy.
- Second, the compressed data must support diverse downstream tasks. For instance, a smart-city camera might need to process the same video stream for both action recognition and traffic monitoring. Similarly, in augmented reality (AR) applications, a single camera stream must handle tasks such as hand recognition and scene reconstruction [10].

*Both authors contributed equally to this research.
#Work done while at University of Rochester.

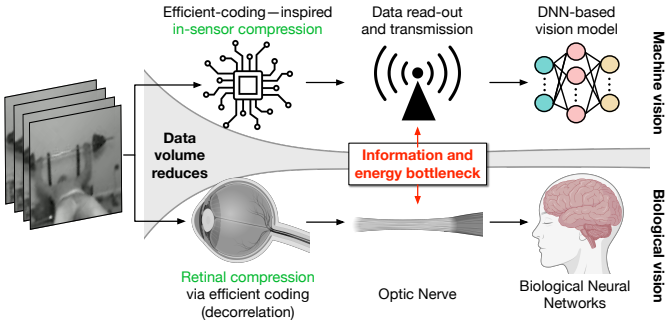

Fig. 1. SNAPPIX reduces edge sensing energy through in-sensor compression by decorrelating output pixel values. This is inspired by the mammalian visual system, where the retina compresses information by decorrelating the retinal output neurons; signals carried through the optic nerve, while at a much lower bandwidth than at the initial stage of the retina, encodes essential information that permits the downstream visual cortex to effectively perform visual tasks.

The paper proposes SNAPPIX, a general in-sensor compression architecture that significantly reduces end-to-end system energy while maintaining high accuracy. SNAPPIX achieves this through three key innovations, summarized below.

General-Purpose Sampling via Decorrelation. To design a general compression strategy, SNAPPIX leverages the principle of decorrelation to minimize redundancy among output pixels (Sec. III). This approach is inspired by the classic theory of efficient coding in neuroscience [11]–[14], which suggests that the retina efficiently transmits information to the brain by reducing redundancy and increasing decorrelation among the Retina Ganglion Cells (RGCs) — the output neurons of the retina [15]. This concept is visualized in Fig. 1.

Similarly, the compression pattern in SNAPPIX is trained to maximize information density in the sensor output, as opposed to be tailored to a particular task. The core mechanism for

compression in SNAPPIX is *sampling*. In particular, SNAPPIX samples through *coded exposure* (CE) [16], where the sensor selectively exposes pixels both spatially and temporally; pixel values are integrated across exposure slots into a single coded image before being read out [7].

Co-Designed Vision Model and Coded Pattern. To maximize the benefits of CE, ideally there should not be any constraint imposed on each pixel's exposure pattern. That, however, means that pixels would carry varying amounts of information and should be treated differently in the downstream model, introducing performance overhead [17], [18].

Instead, SNAPPIX constrains the sampling pattern to be tile-repetitive — pixels within a tile can have different exposure patterns, but the pattern repeats across tiles — and thus constrains the pixel variation within a tile (Sec. IV). To accommodate this tile-repetitive structure, we use Vision Transformers (ViTs) [19] as the backbone. ViTs naturally process inputs tile-by-tile, and the processing of each tile is trained based on the (offline obtained) within-tile pixel variations. With a carefully designed ViT architecture and tailored pre-training, our constrained sampling does not degrade accuracy.

Hardware Support. To support our in-sensor sampling strategy, we propose a set of minimal hardware augmentations to the stacked sensor architecture that is common in modern image sensors [20], where the first layer is the pixel array and the second layer implements per-pixel digital logic (Sec. V). The augmentations to each pixel are limited to a single-bit storage and two transistors to save and apply the CE pattern. These augmentations are integrated beneath the pixel array, resulting in negligible area overhead.

Results. On various edge sensing scenarios (that differ in edge compute capabilities and in data transmission technologies), SNAPPIX achieves 1.4× to 15.4× energy saving compared to existing systems. Compared to other compression methods with a similar compression rate, SNAPPIX delivers better accuracy on multiple tasks.

II. BACKGROUND

A. Energy Bottleneck in Edge Sensing

This paper examines edge sensing scenarios where the edge comprises an image sensor and a lightweight compute unit, such as a microcontroller. In these setups, data must be offloaded to a server for computation-intensive processing. This paradigm is common in remote sensing applications, including urban sensing [21] and satellite imaging [2].

The main energy bottleneck in such systems comes from the cost of in-sensor data read-out and sensor-host data transmission. The read-out energy cost is dominated by the ADC. A recent survey shows that the ADC costs about 66% of an image sensor's energy on average [5]. The data transmission includes both the MIPI CSI-2 interface, which transfers data from the sensor to the edge processor, and the subsequent transmission of data from the processor to the cloud. Energy-wise, transmitting one byte via the MIPI CSI-2 interface is approximately 300 times more expensive than performing a one-byte MAC operation [22]. Wireless transmission exacerbates energy consumption, adding an order of magnitude to the cost, even with passive transmission methods like backscatter [23].

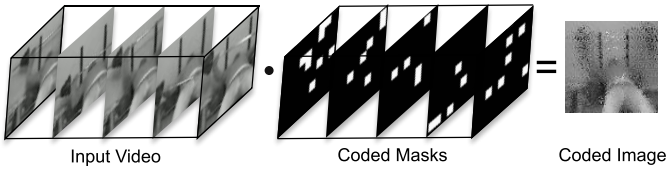

Fig. 2. Coded exposure with 5 exposure slots. In each slot, pixels are selectively exposed, controlled by a coded mask. In the end, the values at all the exposure slots are integrated pixel-wise to form one single coded image.

B. Coded Exposure (CE)

CE compresses image data by selectively exposing pixels across spatial dimensions and multiple frames, integrating the exposures into a single coded image before readout [7]. Fig. 2 illustrates the fundamental operation of CE. Let Y represent the sequence of images that would have been captured using a conventional image sensor, with dimensions $T \times H \times W$, where T is the number of frames in Y, and H and W are the height and width of each image, respectively.

In the CE process, an exposure mask is applied to each frame (or exposure slot) t to select a subset of pixels to expose during that frame (e.g., 5 exposure slots as shown in Fig. 2). The exposed pixel values are then integrated over time for each pixel, resulting in a single coded image of dimensions $1 \times H \times W$ being read out from the sensor. This approach achieves a data reduction factor of T, as T frames are compressed into one. This process can be formulated as follows:

$$X(i,j) = \sum_{t=1}^{T} M(i,j,t) \cdot Y(i,j,t) \tag{1}$$

where X is the final coded image, M is the binary masks controlling exposure, i, j, and t index the spatial dimensions and the temporal dimension (frames), respectively.

III. GENERAL-PURPOSE SAMPLING VIA DECORRELATION

Our CE pattern design maximizes decorrelation among pixels in coded images. Proximal pixels often exhibit high similarity and strong correlation, while pixels farther apart tend to be less correlated. To focus on reducing redundancy among highly correlated pixels, we decorrelate pixels within a tile of P pixels. This is achieved by minimizing the following loss:

$$\mathcal{L}_{\text{Cor}} \triangleq \frac{1}{P(P-1)} \sum_{i=1}^{P} \sum_{j \neq i} C_{ij}^2 \tag{2}$$

where C_{ij} is the Pearson correlation coefficient between two distinct coded pixels i and j within a tile, quantifying redundancy between them.

Estimating C_{ij} requires multiple samples for each coded pixel, and Fig. 3 illustrates this process. First, we sample a batch of images from the dataset and apply the CE operation (Eqn. 1) to generate a batch of coded images. Then we divide each coded image into $N \times N$ tiles, yielding $S = B \times N^2$ samples per coded pixel, where B is the number of coded

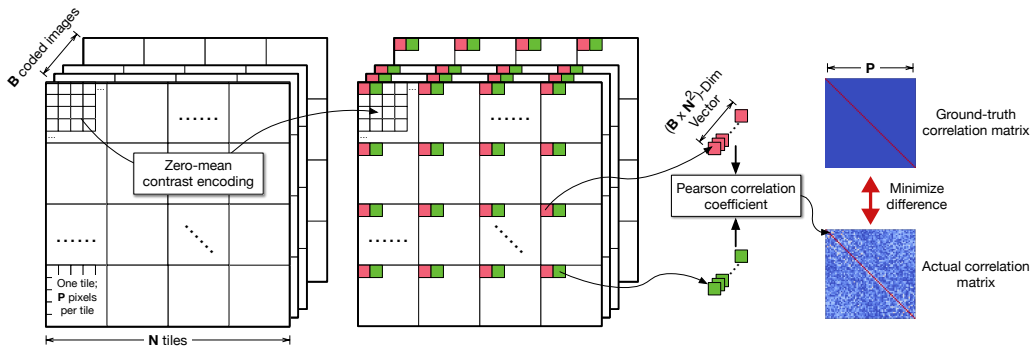

Fig. 3. Illustration of training for pixel decorrelation in coded images. A coded image is divided into tiles, each containing P pixels. The CE mask (not shown) is optimized to decorrelate any pair of pixels within a tile. Zero-mean contrast encoding is applied, ensuring the mean pixel value of each tile is zero.

images in one batch and N^2 is the number of tiles per coded image. As a result, each coded pixel in the batch is represented by an S-dimensional vector, which is used to calculate C_{ij} for any pair of coded pixels i and j.

Before forming the S-dimensional vectors, we preprocess each raw coded tile to have a zero mean, a step referred to as "zero-mean contrast encoding" in Fig. 3. Mathematically, for each tile in a raw coded image, we subtract the average pixel value of the tile from every pixel in the tile. The average value is computed by averaging across all the corresponding tiles in a dataset. This zero-mean preprocessing is important, as proximal pixels naturally exhibit similar values and are thus highly correlated. Without accounting for this inherent correlation, the training process can collapse, where all exposure slots remain closed for all pixels.

We learn the optimal exposure mask (M in Eqn. 1) by minimizing $\mathcal{L}_{Cor}$. This training is task-agnostic: the exposure mask is optimized for a given dataset irrespective of the downstream task. We use straight-through estimation [24] to propagate gradients through the binary masking operation.

IV. CE-Optimized Vision Model and Training

Motivation. Pixels in coded images carry varying amounts of information due to their differences in exposure. A pixel exposed in more slots retains richer temporal information and carries more motion blur. Downstream models that ignore these variations experience accuracy degradation.

In particular, standard convolution operations treat all pixels uniformly, applying the same kernel to every pixel regardless of its exposure. Prior work [17] proposed Shift-Variant Convolution (SVC) layer to address this issue by using different convolution kernels for different coded pixels. However, our profiling shows that SVC slows inference by 4× due to a lack of optimized implementations and inefficient GPU utilization. Due to this performance issue, SVC has only been applied to the first layer in previous work [17], [18], negatively impacting the overall accuracy.

CE-Optimized Vision Transformer. If the exposure pattern is allowed to vary across all the pixels in a frame, then all the pixels are necessarily different. Instead, we propose a tile-repetitive exposure pattern: the frame is divided into tiles, with pixel exposure variations allowed within each tile but constrained to be identical across all tiles. As shown in the left part of Fig. 4: the exposure pattern is consistent across tiles for each time slot (three slots are illustrated), but it can change across slots. As a result, the downstream models only have to deal with pixel variations within a tile while maintaining performance comparable to unconstrained patterns [18].

Building on this tile-repetitive pattern, we propose using Vision Transformers (ViTs) [19] as the backbone for the downstream vision model. ViTs naturally divide an image into patches of $M \times M$ pixels and process pixels within each patch differently via patch-wise embedding (PE) and multi-layer perceptrons (MLPs), while enabling information sharing across patches through multi-head attention (MHA). This makes ViTs well-suited for handling localized pixel variations, as illustrated in the middle part of Fig. 4.

We set the CE tile size to match the ViTs' patch size. This ensures MLPs can learn to address pixel-wise variations within each tile, which are determined by the decorrelation-based exposure pattern. To further facilitate ViT training, each pixel value is normalized by the number of exposure slots.

CE-Optimized Reconstruction Pre-training. Inspired by recent reconstruction-based pre-training [25], [26], we propose a tailored pre-training procedure for CE-compressed inputs.

As illustrated in the right part of Fig. 4, the pre-training process begins with a CE-coded image X, sampled and integrated from a sequence of images. We randomly mask a large portion (e.g., 85%) of its tiles, and set the objective of pre-training to reconstruct the original video sequence. Unlike previous image-to-image [25] or video-to-video [26] pre-training approaches, our pre-training is designed to perform a "coded image-to-video" prediction. This requires the model to both predict the masked tiles, capturing spatial scene structure, and upsample temporal signals from CE-coded information, learning temporal dynamics.

The pre-training process can be formulated as follows:

$$\hat{Y} = \mathrm{D}(\mathrm{E}(\text{random\_masking}(f(Y)))) \tag{3}$$

where Y is the original video, $f(\cdot)$ is CE function as defined in Eqn. 1, $E(\cdot)$ and $D(\cdot)$ are the pre-trained ViT encoder and decoder, and $\hat{Y}$ is the reconstructed video. We use the Mean Squared Error (MSE) loss for pre-training, and predict only 50% of the video frames to accelerate pre-training [26].

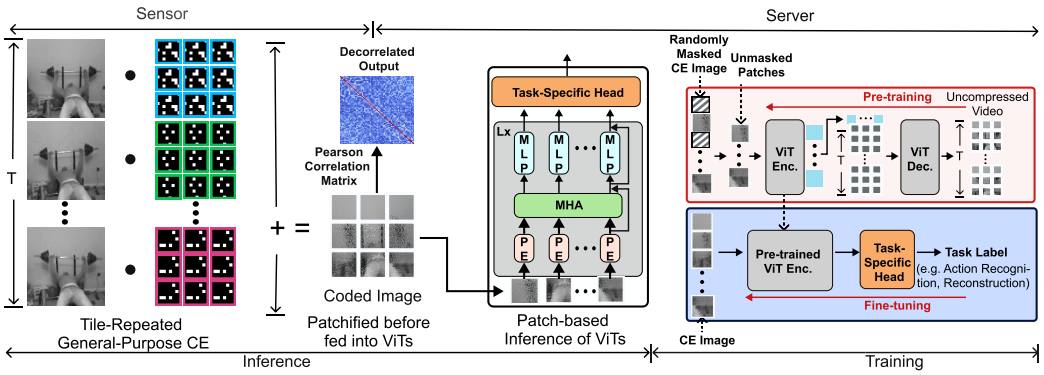

Fig. 4. The end-to-end pipeline of SnapPix with in-sensor CE for compression and a ViT-based vision model for downstream tasks. The CE pattern is trained task-independently using decorrelation, while the downstream model is co-designed with CE patterns and pre-trained specifically for CE-encoded inputs.

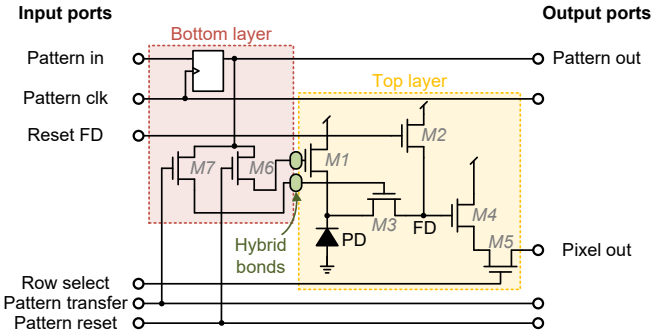

Fig. 5. Schematic of the proposed CE pixel. It is based on a stacked design that is commonly used in modern CMOS image sensors [20].

V. Hardware Support for In-Sensor CE

Rationale and Overview. To support the CE function, previous solutions heavily augment each pixel, placing multiple floating diffusion (FD) nodes [27], [28], using an arithmetic logic unit with mixed-signal memory [29], or communicating high-resolution CE pattern with off-chip controller [30]. They result in a larger pixel area and a smaller pixel fill factor.

In contrast, our sensor utilizes the characteristic of the tile-repetitive decorrelation pattern to reduce both the control footprint and control power. In our design, we replace the global control with local, per-pixel storage to store the CE pattern. We then use a die stacking design, where the top layer is the pixel array and the bottom layer hosts the digital logic/storage. Such a stacking design is common in consumer CMOS image sensors [20], and our design represents a novel usage of the bottom layer to support CE. By stacking, the per-pixel storage is completely hidden beneath the pixel's exposure circuits, so the area of our proposed pixel is approximately the same as that of the conventional, non-CE pixel.

Design. The proposed pixel design is illustrated in Fig. 5, which contains two layers. The top layer is pixel array layer, where each pixel is based on the classic 4T Active Pixel Sensor (APS) design that consists of a photodiode (PD) to acquire incident light and five transistors to reset PD (`M1`), reset floating diffusion (FD) node (`M2`), transfer photocharge (`M3`), and read out photocharge as voltage when the pixel is selected (`M4`, `M5`). Different to the conventional 4T pixel design, the additional transistor `M1` decouples the reset of PD and FD such that the PD can be exposed across multiple exposure slots but only transfer the photocharge once to the FD, thereby realizing CE function.

At the bottom layer, each pixel is equipped with a D-Flip-Flop (DFF) to buffer the one-bit CE pattern of a given exposure slot. The bit represents whether to accumulate the pixel's exposure during the corresponding exposure slot. The DFFs of all the pixels in a tile are connected in a shift-register style: the *pattern in* wire of the second pixel is connected to the *pattern out* wire of its preceding pixel. Each pixel is also augmented with two additional transistors: `M6` (pattern reset) and `M7` (pattern transfer).

At the start of every exposure slot, the CE bits of each pixel in a tile are streamed in through the *pattern in* wire and buffered in the corresponding DFFs . Then, `M6` is turned on for all the pixels (through the *pattern reset* wire), which allows the CE bit in the DFF to control the reset of the PD. If the CE bit is 1, the PD is reset via `M1` (charges accumulated so far is cleared), getting ready for exposure; if the CE bit is 0, the PD is not reset (since `M1` will be open). The DFFs can then be powered-gated.

After the exposure, the same CE bits are streamed in and buffered again in the DFFs. We then turn on `M7` for all the pixels (through the *pattern transfer* wire). If the CE bit is 1, the charge is transferred from the PD to the FD through `M3`; otherwise, `M3` is open and the FD does not accumulate the charge from the previous exposure. The DFFs are then power-gated again until the next exposure slot. While the DFFs are power-gated, logic 0 is given to both `M1` and `M3` via a simple reset logic (not shown in Fig. 5 for simplicity). The delay between the *pattern reset* and *pattern transfer* signals physically creates the exposure time for the pixel [7].

Area Overhead. We synthesize the digital logic at the bottom layer in TSMC 65 nm and obtain the area estimation of 30 μm^2. According to DeepScale tool [31], it translates to 3.2 μm^2 in 22 nm, which is much smaller than commercial stacked digital pixel sensors (DPS) [32], [33]. Therefore, the pixel area is constrained by the APS at the top layer rather than the introduced logic.

An alternative design is to broadcast the CE pattern to all

the pixels in a tiles via dedicated wires. This approach would remove the need for per-pixel digital logic, but require $2N$ signal wires per pixel given a tile size of $N \times N$. In contrast, the number of wires in our design is constant: only four (*pattern in*, *pattern clk*, *pattern transfer*, and *pattern reset*), regardless of tile size. Our synthesis results show that when $N = 8$, the signal wires occupy $2.24\ \mu m \times 2.24\ \mu m$; as N increases to 14, the wire area grows to $3.92\ \mu m \times 3.92\ \mu m$, exceeding the area of the state-of-the-art APS.

VI. Evaluation

A. Experimental Setup

Dataset. We evaluate our approach on three datasets: Something-Something v2 (SSV2) [34], Kinetics-400 (K400) [35], and UCF-101 [36], using the first train-test split in UCF-101. For pre-training, we use SSV2 and the larger K710 dataset [26]. We downsample each video's shorter dimension to 112 pixels and convert the videos to grayscale in linear space. We simulate a CE dataset using $T = 16$ (recall T is the number of exposure slots in Eqn. 1). Each frame is center-cropped to produce a 112×112 resolution input, which is also the input resolution to SnapPix. The video baselines accepts 16 consecutive, uncoded frames as their inputs (i.e., $16 \times 112 \times 112$ in resolution).

Tasks. We evaluate two distinct downstream tasks: action recognition (AR), a high-level task producing a single classification output, and reconstruction (REC), a low-level task generating a video. REC evaluation addresses scenarios where videos are stored for future, undefined tasks. AR is tested on SSV2, K400, and UCF101 using clip-1 crop-1 accuracy, while REC is evaluated on SSV2, which reconstructs the original $16 \times 112 \times 112$ frames from a 112×112 coded image, with Peak Signal-to-Noise Ratio (PSNR) as the metric.

Variants. We provide two SnapPix variants: SnapPix-B and SnapPix-S, differing only in the backbone. SnapPix-B uses ViT-B (87M parameters), for higher accuracy but at a slower speed. SnapPix-S uses ViT-S (22M parameters) for faster performance, similar to prior CE-based models [17], [18]. Both variants use an 8×8 patch size.

Baselines. We compare our decorrelated CE pattern with the following task-agnostic CE patterns ($T = 16$ in all cases):

- Long Exposure: All pixels exposed in all slots.
- Short Exposure: All pixels exposed every 8th frame.
- Random: Each pixel exposed randomly with a 50% probability per exposure slot.
- Sparse Random: Each pixel exposed once randomly across T exposure slots.

For AR, we also compare against three prior systems:

- SVC2D [17]: A CE-based AR model with SVC and an end-to-end learned CE pattern.
- C3D [37]: A strong video-based AR model, treated as an upper bound in prior CE works [17], [18].
- VideoMAEv2-ST [26]: A state-of-the-art ViT-based video AR model, adjusted to match SnapPix-B's speed.

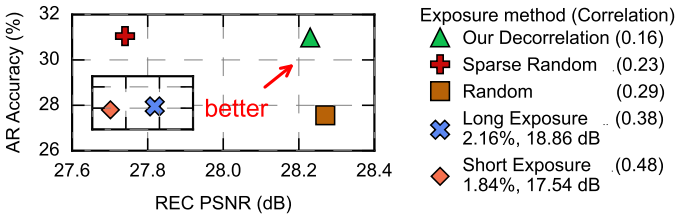

Fig. 6. Comparison of task-agnostic CE patterns using AR and REC results. The legend also shows the corresponding Pearson correlation coefficients.

To ensure a fair comparison, we reproduced all baselines using our data preprocessing, frame sampling strategy, and training recipe, modifying only the learning rate.

Training Recipe. To avoid overfitting to smaller datasets, we first train the decorrelated CE pattern on the large pre-training dataset for 5 epochs. We then fix the CE pattern and perform pre-training and task-specific training for downstream vision models. For training from scratch, REC is trained for 100 epochs, AR for 200 epochs on SSV2, and 400 epochs on K400 and UCF-101. For fine-tuning after pre-training, epochs are halved. Epochs are computed as repeated augmentations$\times$epochs, same as prior work [26]. Learning rates are tuned for different models.

B. Decorrelation Outperforms Other Task-Agnostic CEs

To evaluate the effectiveness of our decorrelation-based CE (Sec. III), we compare it against other task-agnostic CE patterns in Fig. 6, where reports the AR accuracy (y-axis) and the REC quality (x-axis) on the SSV2 dataset using our CE-optimized ViT (Sec. IV) trained from scratch. The legend on the right also reports the Pearson correlation coefficient between coded pixels.

Our decorrelation-based CE excels in both AR and REC, demonstrating its effectiveness as a general task-agnostic CE pattern. While Random performs slightly better in REC, its AR accuracy is significantly lower than ours. Conversely, Sparse Random achieves a high AR accuracy but performs poorly in REC. Long Exposure and Short Exposure perform significantly worse in both tasks. The relative performance across these strategies corresponds roughly to their respective correlation coefficient, suggesting the benefit of using decorrelation to learn the CE pattern.

C. SnapPix Outperforms Task-Specific Methods

We now evaluate SnapPix against prior systems that are specifically tuned for AR, including CE-based and video-based methods. Inference speed is measured in inference per second on a system with a Ryzen 9 7900X3D CPU and a RTX 4090 GPU using a batch size of 64, simulating server-side computations. Results are summarized in Tbl. I. SnapPix-B and SnapPix-S achieve the highest AR accuracy across all datasets, nearly doubling the performance of CE-based baselines. They also surpass video models, which were previously considered upper bounds for CE-based methods [17], [18]. At the same time, SnapPix executes faster than video-based methods due to a reduced input data.

TABLE I
COMPARISON WITH PREVIOUS SYSTEMS. WE HIGHLIGHT THE TOP 3 VARIANTS FOR EACH METRIC (RED, ORANGE, YELLOW).

Model	Input	Accuracy (↑)			Inference /sec (↑)
		UCF-101	SSV2	K400	
SNAPPIX-S (ours)	CE	74.65%	42.38%	47.58%	2282
SNAPPIX-B (ours)	CE	79.14%	45.21%	54.11%	760
SVC2D [17]	CE	41.16%	23.05%	26.09%	2135
C3D [37]	Video	62.70%	33.48%	41.66%	541
VideoMAEv2-ST [26]	Video	72.54%	39.84%	41.99%	750

It might initially seem surprising that SNAPPIX outperforms even the video baselines (last two rows in Tbl. I). This is primarily because SNAPPIX uses a single image as the model input, which significantly reduces the compute overhead and, in turn, allows us to use larger models.

SVC2D is a prior CE-based AR method that learns the CE pattern jointly with SVC in the downstream model [17], [18]. When we apply its CE pattern to our ViT model, it underperforms our decorrelation-based pattern by 1.35% in AR accuracy (not shown in the table). This highlights a key limitation of task-specific, end-to-end learned patterns: they are model-dependent and become suboptimal even for the same task when the model architecture evolves.

D. SNAPPIX *Significantly Reduces Edge Energy*

Our energy evaluation is mainly based on edge-server scenarios where the edge device has to transmit all the data to the cloud/server. In this case, the edge energy is the sum of the sensing and data transmission energy. We use CamJ [22], a sensor energy model calibrated against real silicon to model the sensing energy. The total sensing energy is 220 pJ per pixel (8 bits), of which 95.6% is contributed to by the ADC and MIPI energy [5]. The energy overhead introduced by supporting CE is 9 pJ per pixel with 20 MHz pattern stream clock according to our synthesis results.

The wireless energy depends on the technology used for data transmission. We model short-range (about 10 meters) transmission using passive WiFi, with a transmission energy of 43.04 pJ per pixel [38]. Long-range (over 100 meters) transmission usually uses backscatter Long-Range (LoRa) technology [23], whose energy is 7.4 μ J per pixel.

Under $T = 16$, SNAPPIX reduces the ADC/MIPI and wireless transmission energy by 16×. SNAPPIX achieves a 7.6× edge energy saving in the short-range scenario and a 15.4× saving in the long-range scenario.

We also evaluate a scenario where the edge node has a mobile GPU that can execute the downstream model. For that we measure the energy consumption of the mobile Volta GPU on a Jetson Xavier SoC [39] with a batch size of 1. In this case the mobile GPU energy dominates the total energy consumption. Compared to executing video baselines: VIDEOMAEV2-ST and C3D on the edge server, SNAPPIX-S achieves 1.4× and 4.5× energy saving respectively. This is primarily because SNAPPIX uses single (coded) images to drive the vision model rather than an entire video.

Finally, we compare with a simple compression baseline that spatially downsamples each frame by 16× (the same compression rate as SNAPPIX) using 4×4 average filtering and then processing the compressed data with VIDEOMAEV2-ST. The resulting AR accuracy is 9.83%, 6.24%, and 16.45% *lower* than SNAPPIX-B on UCF-101, SSV2, and K400, respectively.

E. Ablation Study

We perform an ablation study by removing various components from Sec. III and Sec. IV, using SNAPPIX-S as a baseline and the SSV2 dataset and AR task for evaluation:

- Removing pre-training reduces accuracy by 11.39%.
- Replacing the decorrelated pattern with a random pattern further decreases accuracy by 3.43%.
- Replacing the tile-repetitive CE pattern with a global pattern reduces accuracy by 23.74%.

VII. RELATED WORK

Compressed Edge Sensing. Digital-domain compression [40], [41] could achieve high compression rate. However, even with dedicated hardware, classic digital compression consumes nJ/pixel [42], several orders of magnitude higher than the energy of sensing itself. This issue is further exacerbated in deep-learning-based compression [41]. Moreover, digital compression operates after sensor read-out without sensing energy saving. Instead, in-sensor compression [4], [5], [7]–[9] performs compression before read-out, saving both sensing and transmission energy. Our method falls under the category of in-sensor compression.

Sensors with CE Support. Coded Exposure (CE) is a computational imaging technique used for enhancing imaging quality (e.g. High Dynamic Range (HDR) [27], [43] and high-speed imaging [7]). We propose a novel use of CE — for in-sensor compression. Algorithmically compared to prior CE-based methods [16]–[18], we are the first to propose using decorrelation to learn the CE patterns, and have shown that such a pattern out-perform prior methods across multiple tasks.

In-Sensor Computation. Vision sensors increasingly integrate computational capabilities, shifting computations to the analog domain. Examples include selective reading [4], [8], feature extraction [44], computing temporal derivatives [45], and even lightweight neural networks [5], [9], [46]. SNAPPIX supports in-sensor CE for compression, and introduces only minor augmentations to the hardware design.

VIII. CONCLUSION

In-sensor compression through CE, which turns a sequence of video frames into a single coded image, leads to an order of magnitude energy saving on edge sensing while maintaining a task quality comparable with existing video-based methods. Task-agnostic training of CE patterns through decorrelation can generalize to downstream tasks of different nature, outperforming other CE patterns.

ACKNOWLEDGMENT

The work is partially supported by NSF Award #2416375.

REFERENCES

[1] J. Klotz and S. K. Nayar, "Minimalist vision with freeform pixels," in *ECCV*, pp. 329–346, 2024.

[2] A. Gadre, Z. Machester, and S. Kumar, "Adapting lora ground stations for low-latency imaging and inference from lora-enabled cubesats," *TOSN*, vol. 20, no. 5, pp. 1–30, 2024.

[3] XREAL, "Xreal air 2." https://us.shop.xreal.com/products/xreal-air-2.

[4] Y. Feng, T. Ma, Y. Zhu, and X. Zhang, "Blisscam: Boosting eye tracking efficiency with learned in-sensor sparse sampling," in *ISCA*, pp. 1262–1277, 2024.

[5] 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 *ISCA*, pp. 1–14, 2023.

[6] H. Desai, M. Nardello, D. Brunelli, and B. Lucia, "Camaroptera: A long-range image sensor with local inference for remote sensing applications," *TECS*, vol. 21, no. 3, pp. 1–25, 2022.

[7] M. Yoshida, T. Sonoda, H. Nagahara, K. Endo, Y. Sugiyama, and R.-i. Taniguchi, "High-speed imaging using cmos image sensor with quasi pixel-wise exposure," *TCI*, vol. 6, pp. 463–476, 2019.

[8] T. Zhang, K. Kasichainula, D.-W. Jee, I. Yeo, Y. Zhuo, B. Li, J.-s. Seo, and Y. Cao, "Improving the efficiency of cmos image sensors through in-sensor selective attention," in *ISCAS*, pp. 1–4, IEEE, 2023.

[9] G. R. Nair, P. S. Nalla, G. Krishnan, J. Oh, A. Hassan, I. Yeo, K. Kasichainula, M. Seok, J.-s. Seo, Y. Cao, *et al.*, "3d in-sensor computing for real-time dvs data compression: 65nm hardware-algorithm co-design," *SSC-L*, 2024.

[10] H. Kwon, K. Nair, J. Seo, J. Yik, D. Mohapatra, D. Zhan, J. Song, P. Capak, P. Zhang, P. Vajda, *et al.*, "Xrbench: An extended reality (xr) machine learning benchmark suite for the metaverse," *MLSys*, vol. 5, pp. 1–20, 2023.

[11] H. B. Barlow *et al.*, "Possible principles underlying the transformation of sensory messages," *Sensory communication*, vol. 1, no. 01, pp. 217–233, 1961.

[12] F. Attneave, "Some informational aspects of visual perception.," *Psychological review*, vol. 61, no. 3, p. 183, 1954.

[13] X. Pitkow and M. Meister, "Decorrelation and efficient coding by retinal ganglion cells," *Nature neuroscience*, vol. 15, no. 4, pp. 628–635, 2012.

[14] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, "Barlow twins: Self-supervised learning via redundancy reduction," in *ICML*, pp. 12310–12320, PMLR, 2021.

[15] Y. Dan, J. J. Atick, and R. C. Reid, "Efficient coding of natural scenes in the lateral geniculate nucleus: experimental test of a computational theory," *Journal of neuroscience*, vol. 16, no. 10, pp. 3351–3362, 1996.

[16] D. Reddy, A. Veeraraghavan, and R. Chellappa, "P2c2: Programmable pixel compressive camera for high speed imaging," in *CVPR*, pp. 329–336, IEEE, 2011.

[17] T. Okawara, M. Yoshida, H. Nagahara, and Y. Yagi, "Action recognition from a single coded image," in *ICCP*, pp. 1–11, IEEE, 2020.

[18] S. Kumawat, T. Okawara, M. Yoshida, H. Nagahara, and Y. Yagi, "Action recognition from a single coded image," *TPAMI*, vol. 45, no. 4, pp. 4109–4121, 2022.

[19] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *ICLR*, 2021.

[20] Y. Oike, "Evolution of image sensor architectures with stacked device technologies," *IT-ED*, vol. 69, no. 6, pp. 2757–2765, 2021.

[21] J. Adkins, B. Ghena, N. Jackson, P. Pannuto, S. Rohrer, B. Campbell, and P. Dutta, "The signpost platform for city-scale sensing," in *IPSN*, pp. 188–199, IEEE, 2018.

[22] T. Ma, Y. Feng, X. Zhang, and Y. Zhu, "Camj: Enabling system-level energy modeling and architectural exploration for in-sensor visual computing," in *ISCA*, pp. 1–14, 2023.

[23] V. Talla, M. Hessar, B. Kellogg, A. Najafi, J. R. Smith, and S. Gollakota, "Lora backscatter: Enabling the vision of ubiquitous connectivity," *IMWUT*, vol. 1, no. 3, pp. 1–24, 2017.

[24] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[25] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *CVPR*, pp. 16000–16009, 2022.

[26] L. Wang, B. Huang, Z. Zhao, Z. Tong, Y. He, Y. Wang, Y. Wang, and Y. Qiao, "Videomae v2: Scaling video masked autoencoders with dual masking," in *CVPR*, pp. 14549–14560, 2023.

[27] R. Gulve, N. Sarhangnejad, G. Dutta, M. Sakr, D. Nguyen, R. Rangel, W. Chen, Z. Xia, M. Wei, N. Gusev, *et al.*, "39 000-subexposures/s dual-adc cmos image sensor with dual-tap coded-exposure pixels for single-shot hdr and 3-d computational imaging," *JSSC*, vol. 58, no. 11, pp. 3150–3163, 2023.

[28] Y. Luo and S. Mirabbasi, "A 30-fps 192× 192 cmos image sensor with per-frame spatial-temporal coded exposure for compressive focal-stack depth sensing," *JSSC*, vol. 57, no. 6, pp. 1661–1672, 2022.

[29] J. N. Martel, L. K. Mueller, S. J. Carey, P. Dudek, and G. Wetzstein, "Neural sensors: Learning pixel exposures for hdr imaging and video compressive sensing with programmable sensors," *TPAMI*, vol. 42, no. 7, pp. 1642–1653, 2020.

[30] K. Kagawa, M. Horio, A. N. Pham, T. Ibrahim, S.-i. Okihara, T. Furuhashi, T. Takasawa, K. Yasutomi, S. Kawahito, and H. Nagahara, "A dual-mode 303-megaframes-per-second charge-domain time-compressive computational cmos image sensor," *Sensors*, vol. 22, no. 5, p. 1953, 2022.

[31] S. Sarangi and B. Baas, "Deepscaletool: A tool for the accurate estimation of technology scaling in the deep-submicron era," in *ISCAS*, pp. 1–5, IEEE, 2021.

[32] R. Ikeno, K. Mori, M. Uno, K. Miyauchi, T. Isozaki, I. Takayanagi, J. Nakamura, S.-G. Wuu, L. Bainbridge, A. Berkovich, *et al.*, "A 4.6-μm, 127-db dynamic range, ultra-low power stacked digital pixel sensor with overlapped triple quantization," *T-ED*, vol. 69, no. 6, pp. 2943–2950, 2022.

[33] M.-W. Seo, M. Chu, H.-Y. Jung, S. Kim, J. Song, D. Bae, S. Lee, J. Lee, S.-Y. Kim, J. Lee, *et al.*, "2.45 e-rms low-random-noise, 598.5 mw low-power, and 1.2 kfps high-speed 2-mp global shutter cmos image sensor with pixel-level adc and memory," *JSSC*, vol. 57, no. 4, pp. 1125–1137, 2022.

[34] R. Goyal, S. Ebrahimi Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag, *et al.*, "The" something something" video database for learning and evaluating visual common sense," in *ICCV*, pp. 5842–5850, 2017.

[35] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, *et al.*, "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017.

[36] K. Soomro, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[37] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *ICCV*, pp. 4489–4497, 2015.

[38] B. Kellogg, V. Talla, S. Gollakota, and J. R. Smith, "Passive wi-fi: Bringing low power to wi-fi transmissions," in *NSDI*, pp. 151–164, 2016.

[39] "Nvidia reveals xavier soc details." https://www.forbes.com/sites/moorinsights/2018/08/24/nvidia-reveals-xavier-soc-details/amp/.

[40] G. K. Wallace, "The jpeg still picture compression standard," *Communications of the ACM*, vol. 34, no. 4, pp. 30–44, 1991.

[41] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Deep convolutional autoencoder-based lossy image compression," in *PCS*, pp. 253–257, IEEE, 2018.

[42] T. Polonelli, D. Battistini, M. Rusci, D. Brunelli, and L. Benini, "An energy optimized jpeg encoder for parallel ultra-low-power processing-platforms," in *APPLEPIES 2019 7.*, pp. 125–133, Springer, 2020.

[43] J. Zhang, J. P. Newman, X. Wang, C. S. Thakur, J. Rattray, R. Etienne-Cummings, and M. A. Wilson, "A closed-loop, all-electronic pixel-wise adaptive imaging system for high dynamic range videography," *TCAS-I*, vol. 67, no. 6, pp. 1803–1814, 2020.

[44] 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 *ISLPED*, pp. 1–6, 2022.

[45] D.-W. Jee, S.-M. Ko, K. Kasichainula, I. Yeo, Y. Cao, and J.-S. Seo, "A time-memory-based cmos vision sensor with in-pixel temporal derivative computing for multi-mode image processing," in *ESSCIRC*, pp. 109–112, IEEE, 2023.

[46] R. LiKamWa, Y. Hou, Y. Gao, M. Polansky, and L. Zhong, "Redeye: Analog convnet image sensor architecture for continuous mobile vision," in *ISCA*, pp. 255–266, 2016.