# P2M: A Processing-in-Pixel-in-Memory Paradigm for Resource-Constrained TinyML Applications

**Gourav Datta** ( gdatta@usc.edu )
University of Southern California
**Souvik Kundu**
University of Southern California
**Zihan Yin**
University of Southern California
**Ravi Teja Lakkireddy**
University of Southern California
**Joe Mathai**
Information Sciences Institute
**Ajey Jacob**
Information Sciences Institute
**Peter Beerel**
University of Southern California
**Akhilesh Jaiswal**
Information Sciences Institute

**Article**

**Keywords:**

**Posted Date:** March 17th, 2022

**DOI:** https://doi.org/10.21203/rs.3.rs-1459821/v1

# P$^2$M: A Processing-in-Pixel-in-Memory Paradigm for Resource-Constrained TinyML Applications

**Gourav Datta[1,*,+], Souvik Kundu[1,*], Zihan Yin[1,*], Ravi Teja Lakkireddy[1], Joe Mathai[2], Ajey P. Jacob[2], Peter A. Beerel[1], and Akhilesh R. Jaiswal[1,2]**

[1]Ming Hsieh Department of Electrical and Computer Engineering, University of Southern California, USA
[2]Information Sciences Institute, University of Southern California, USA
[*]these authors contributed equally to this work
[+]Corresponding author: gdatta@usc.edu

## ABSTRACT

The demand to process vast amounts of data generated from state-of-the-art high resolution cameras has motivated novel energy-efficient on-device AI solutions. Visual data in such cameras are usually captured in analog voltages by a sensor pixel array, and then converted to the digital domain for subsequent AI processing using analog-to-digital converters (ADC). Recent research has tried to take advantage of massively parallel low-power analog/digital computing in the form of near- and in-sensor processing, in which the AI computation is performed partly in the periphery of the pixel array and partly in a separate on-board CPU/accelerator. Unfortunately, high-resolution input images still need to be streamed between the camera and the AI processing unit, frame by frame, causing energy, bandwidth, and security bottlenecks. To mitigate this problem, we propose a novel Processing-in-Pixel-in-memory (P$^2$M) paradigm, that customizes the pixel array by adding support for analog multi-channel, multi-bit convolution, batch normalization, and ReLU (Rectified Linear Units). Our solution includes a holistic algorithm-circuit co-design approach and the resulting P$^2$M paradigm can be used as a drop-in replacement for embedding memory-intensive first few layers of convolutional neural network (CNN) models within foundry-manufacturable CMOS image sensor platforms. Our experimental results indicate that P$^2$M reduces data transfer bandwidth from sensors and analog to digital conversions by $\sim 21\times$, and the energy-delay product (EDP) incurred in processing a MobileNetV2 model on a TinyML use case for visual wake words dataset (VWW) by up to $\sim 11\times$ compared to standard near-processing or in-sensor implementations, without any significant drop in test accuracy.

## 1 Introduction

Today's widespread applications of computer vision spanning surveillance[1], disaster management[2], camera traps for wildlife monitoring[3], autonomous driving, smartphones, etc., are fueled by the remarkable technological advances in image sensing platforms[4] and the ever-improving field of deep learning algorithms[5]. However, hardware implementations of vision sensing and vision processing platforms have traditionally been physically segregated. For example, current vision sensor platforms based on CMOS technology act as transduction entities that convert incident light intensities into digitized pixel values, through a two-dimensional array of photodiodes[6]. The vision data generated from such CMOS Image Sensors (CIS) are often processed elsewhere in a cloud environment consisting of CPUs and GPUs[7]. The physical segregation of vision sensing and computing platforms leads to multiple bottlenecks concerning throughput, bandwidth, and energy-efficiency.

To address these bottlenecks, many researchers are trying to bring intelligent data processing closer to the source of the vision data, *i.e.,* closer to the CIS, taking one of three broad approaches - near-sensor processing[8,9], in-sensor processing[10], and in-pixel processing[11–13]. Near-sensor processing aims to incorporate a dedicated machine learning accelerator chip on the same printed circuit board[8], or even 3D-stacked with the CIS chip[9]. Although this enables processing of the CIS data closer to the sensor rather than in the cloud, it still suffers from the data transfer costs between the CIS and processing chip. On the other hand, in-sensor processing solutions[10] integrate digital or analog circuits within the periphery of the CIS sensor chip, reducing the data transfer between the CIS sensor and processing chips. Nevertheless, these approaches still often require data to be streamed (or read in parallel) through a bus from CIS photo-diode arrays into the peripheral processing circuits[10]. In contrast, in-pixel processing solutions, such as[11–15], aim to embed processing capabilities within the individual CIS pixels. Initial efforts have focused on in-pixel analog convolution operation[14,15] but many[11,14–16] require the use of emerging non-volatile memories or 2D materials. Unfortunately, these technologies are not yet mature and thus not amenable to the existing foundry-manufacturing of CIS. Moreover, these works fail to support multi-bit, multi-channel convolution operations, batch normalization (BN), and Rectified Linear Units (ReLU) needed for most practical deep learning applications.

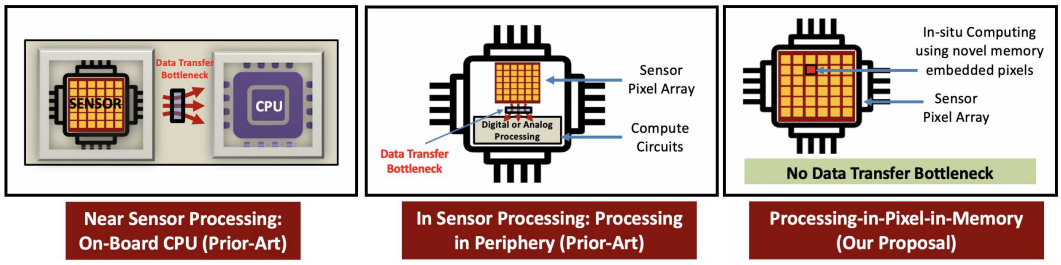


**Figure 1.** Existing and Proposed Solutions to alleviate the energy, throughput, and bandwidth bottleneck caused by the segregation of *Sensing* and *Compute*.

Furthermore, works that target digital CMOS-based in-pixel hardware, organized as pixel-parallel single instruction multiple data (SIMD) processor arrays[12], do not support convolution operation, and are thus limited to toy workloads, such as digit recognition. Many of these works rely on digital processing which typically yields lower levels of parallelism compared to their analog in-pixel alternatives. In contrast, the work in[13], leverages in-pixel parallel analog computing, wherein the weights of a neural network are represented as the exposure time of individual pixels. Their approach requires weights to be made available for manipulating pixel-exposure time through control pulses, leading to a data transfer bottleneck between the weight memories and the sensor array. Thus, an in-situ CIS processing solution where both the weights and input activations are available within individual pixels that efficiently implements critical deep learning operations such as multi-bit, multi-channel convolution, BN, and ReLU operations has remained elusive. Furthermore, all existing in-pixel computing solutions have targeted datasets that do not represent realistic applications of machine intelligence mapped onto state-of-the-art CIS. Specifically, most of the existing works are focused on simplistic datasets like MNIST[12], while few[13] use the CIFAR-10 dataset which has input images with a significantly low resolution ($32\times32$), that does not represent images captured by state-of-the-art high resolution CIS.

Towards that end, we propose a novel in-situ computing paradigm at the sensor nodes called *Processing-in-Pixel-in-Memory* ($P^2M$), that incorporates both the network weights and activations to enable massively parallel, high-throughput intelligent computing inside CISs. In particular, our circuit architecture not only enables in-situ multi-bit, multi-channel, dot product analog acceleration needed for convolution, but re-purposes the on-chip digital *correlated double sampling* (CDS) circuit and single slope ADC (SS-ADC) typically available in conventional CIS to implement *all the required computational aspects for the first few layers* of a state-of-the-art deep learning network. Furthermore, the proposed architecture is coupled with a circuit-algorithm co-design paradigm that captures the circuit non-linearities, limitations, and bandwidth reduction goals for improved latency and energy-efficiency. The resulting paradigm is the first to demonstrate feasibility for enabling complex, intelligent image processing applications (beyond toy datasets), on high resolution images of Visual Wake Words (VWW) dataset, catering to a real-life TinyML application. We choose to evaluate the efficacy of $P^2M$ on TinyML applications, as they impose tight compute and memory budgets, that are otherwise difficult to meet with current in- and near-sensor processing solutions, particularly for high-resolution input images. Key highlights of the presented work are as follows:

1. We propose a novel processing-in-pixel-in-memory ($P^2M$) paradigm for resource-constrained sensor intelligence applications, wherein novel memory-embedded pixels enable massively parallel dot product acceleration using in-situ input activations (photodiode currents) and in-situ weights all available within individual pixels.

2. We propose re-purposing of on-chip memory-embedded pixels, CDS circuits and SS-ADCs to implement positive and negative weights, BN, and digital ReLU functionality within the CIS chip, thereby mapping all the computational aspects for the first few layers of a complex state-of-the-art deep learning network within CIS.

3. We further develop a compact MobileNet-V2 based model optimized specifically for $P^2M$-implemented hardware constraints, and benchmark its accuracy and energy-delay product (EDP) on the VWW dataset, which represents a common use case of visual TinyML.

The remainder of the paper is organized as follows. Section 2 discusses the challenges and opportunities for $P^2M$. Section 3 explains our proposed $P^2M$ circuit implementation using manufacturable memory technologies. Then, Section 4 discusses our approach for $P^2M$-constrained algorithm-circuit co-design. Section 5 presents our TinyML benchmarking dataset, model architectures, test accuracy and EDP results. Finally, some conclusions are provided in Section 6.

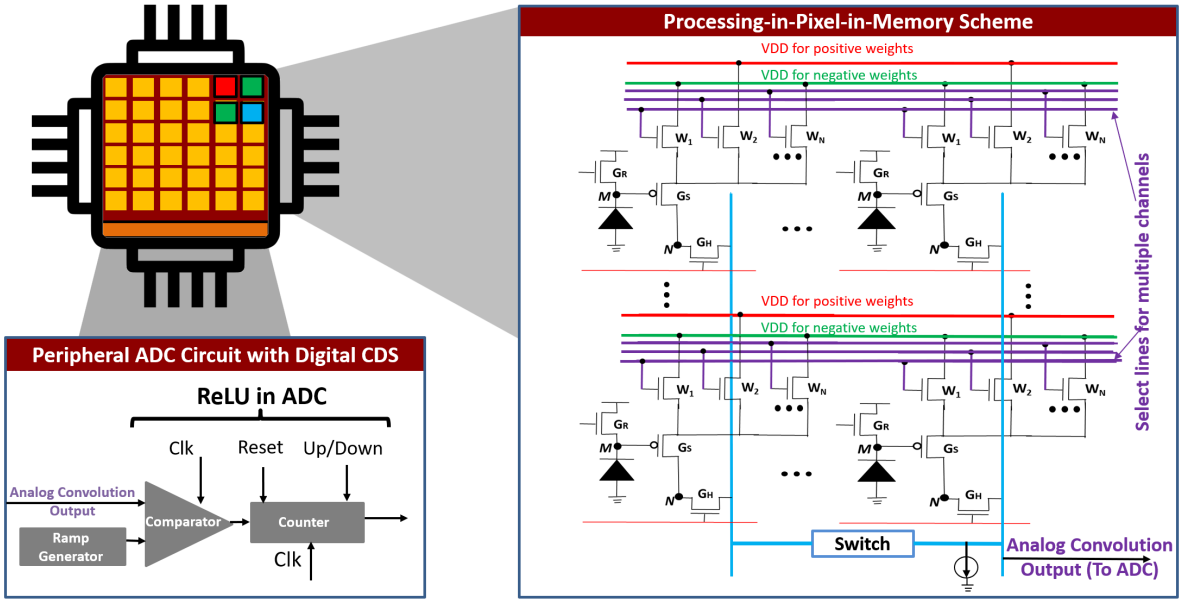


**Figure 2.** Proposed circuit techniques based on presented P$^2$M scheme capable of mapping all computational aspects for the first few layers of a modern CNN layer within CIS pixel arrays.

## 2 Challenges & Opportunities in P$^2$M

The ubiquitous presence of CIS-based vision sensors has driven the need to enable machine learning computations closer to the sensor nodes. However, given the computing complexity of modern CNNs, such as Resnet-18[17] and SqueezeNet[18], it is not feasible to execute the entire deep-learning network, including all the layers within the CIS chip. As a result, recent intelligent vision sensors, for example, from Sony[9], which is equipped with basic AI processing functionality (e.g., computing image metadata), features a multi-stacked configuration consisting of separate pixel and logic chips that must rely on high and relatively energy-expensive inter-chip communication bandwidth.

Alternatively, we assert that embedding part of the deep learning network within pixel arrays in an in-situ manner can lead to a significant reduction in data bandwidth (and hence energy consumption) between sensor chip and downstream processing for the rest of the convolutional layers. This is because the first few layers of carefully designed CNNs, as explained in Section 4, can have a significant compressing property, i.e., the output feature maps have reduced bandwidth/dimensionality compared to the input image frames. In particular, our proposed P$^2$M paradigm enables us to map all the computations of the first few layers of a CNN into the pixel array. The paradigm includes a holistic hardware-algorithm co-design framework that captures the specific circuit behavior, including circuit non-idealities, and hardware limitations, during the design, optimization, and training of the proposed machine learning networks. The trained weights for the first few network layers are then mapped to specific transistor sizes in the pixel-array. Because the transistor widths are fixed during manufacturing, the corresponding CNN weights lack programmability. Fortunately, it is common to use the pre-trained versions of the first few layers of modern CNNs as high-level feature extractors are common across many vision tasks[19]. Hence, the fixed weights in the first few CNN layers do not limit the use of our proposed scheme for a wide class of vision applications. Moreover, we would like to emphasize that the memory-embedded pixel also work seamlessly well by replacing fixed transistors with emerging non-volatile memories, as described in Section 3.4. Finally, the presented P$^2$M paradigm can be used in conjunction with existing near-sensor processing approaches for added benefits, such as, improving the energy-efficiency of the remaining convolutional layers.

## 3 P$^2$M Circuit Implementation

This section describes key circuit innovations that enable us to embed all the computational aspects for the first few layers of a complex CNN architecture within the CIS. An overview of our proposed pixel array that enables the availability of weights and activations within individual pixels with appropriate peripheral circuits is shown in Fig. 2.

### 3.1 Multi-Channel, Multi-Bit Weight Embedded Pixels

Our modified pixel circuit builds upon the standard three transistor pixel by embedding additional transistors $W_i$s that represent weights of the CNN layer, as shown in Fig. 2. Each weight transistor $W_i$ is connected in series with the source-follower transistor

$G_s$. When a particular weight transistor $W_i$ is activated (by pulling its gate voltage to $V_{DD}$), the pixel output is modulated both by the driving strength of the transistor $W_i$ and the voltage at the gate of the source-follower transistor $G_s$. Thus, the pixel output performs an approximate multiplication operation between the input light intensity (voltage at the gate of transistor $G_s$) and the weight (or driving strength) of the transistor $W_i$. Multiple weight transistors $W_i$s are incorporated within the same pixel and are controlled by independent gate control signals. These weight transistors can be used to implement different channels in the output feature map of the layer. Thus, the gate signals represent select lines for specific channels in the output feature map.

The presented circuit can support both overlapping and non-overlapping strides depending on the number of weight transistors $W_i$s per pixel. Specifically, each stride for a particular kernel can be mapped to a different set of weight transistors over the pixels (input activations). The transistors $W_i$s represent multi-bit weights as the driving strength of the transistors can be controlled over a wide range based on transistor width, length, and threshold voltage.

### 3.2 In-situ Multi-pixel Convolution Operation

To achieve the convolution operation, we simultaneously activate multiple pixels. In the specific case of VWW, we activate $X \times Y \times 3$ pixels at the same time, where $X$ and $Y$ denote the spatial dimensions and 3 corresponds to the RGB (red, blue, green) channels in the input activation layer. For each activated pixels, the pixel output is modulated by the photo-diode current and the weight of the activated $W_i$ transistor associated with the pixel. The weight transistors are activated by respective select lines connected to their gates. The weight transistors $W_i$ represent multi-bit weight through its driving strength. Outputs from multiple pixel combine with each other, thereby, mimicking an accumulation operation. Thus, the voltage at the output of the select lines, shown as vertical blue lines in Fig. 2, represent the convolution operation between input activations and the stored weight inside the pixel. Note, in order to generate multiple output feature maps, the convolution operation has to be repeated for each channel in the output feature map. The corresponding weight for each channel is stored in a separate weight transistor embedded inside each pixel. Thus, there are as many weight transistors embedded within a pixel as there are number of channels in the output feature map.

In summary, the presented scheme can perform in-situ multi-bit, multi-channel analog convolution operation inside the pixel array, wherein both input activations and network weights are present within individual pixels.

### 3.3 Re-purposing Digital Correlated Double Sampling Circuit and Single-Slope ADCs as ReLU Neurons

Weights in a CNN layer span positive and negative values. As discussed in the previous sub-section, weights are mapped by the driving strength (or width) of transistors $W_i$s. As the width of transistors cannot be negative, the $W_i$ transistors themselves cannot represent negative weights. Interestingly, we circumvent this issue by re-purposing on-chip digital CDS circuit present in many state-of-the-art commercial CIS[20,21]. A digital CDS is usually implemented in conjunction to column parallel Single Slope ADCs (SS-ADCs). A single slope ADC consists of a ramp-generator, a comparator, and a counter (see Fig. 2). An input

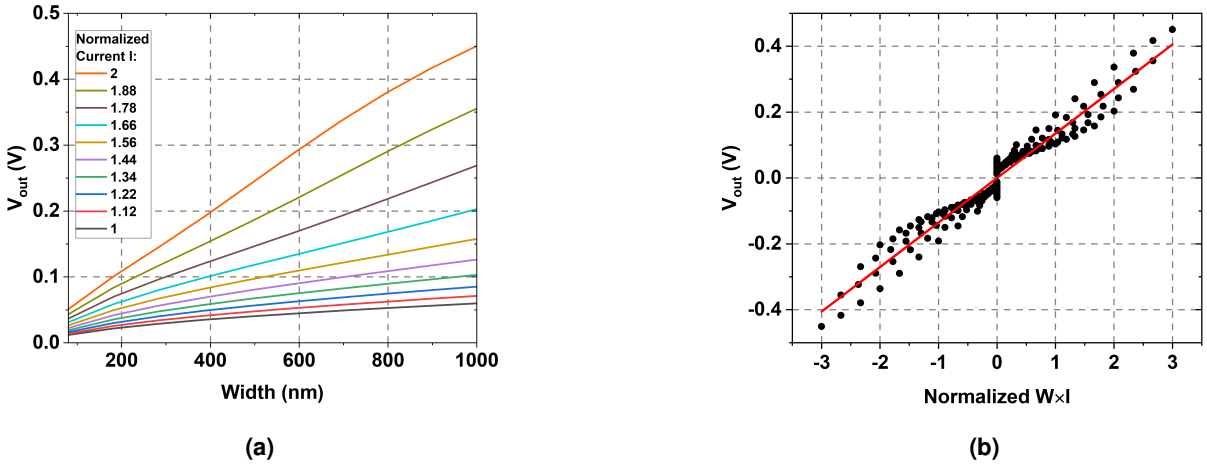


**Figure 3.** (a) Pixel output voltage as a function of weight (transistor width) and input activation (Normalized photo-diode current) simulated on Globalfoundries 22nm FD-SOI node. As expected pixel output increases both as a function of weights and input activation. (b) A scatter plot comparing pixel output voltage to ideal multiplication value of Weights×Input activation (Normalized $W \times I$). The plot confirms that the output pixel voltage from each pixel represents approximate product of Weights×Input activation.

analog voltage is compared through the comparator to a ramping voltage with a fixed slope, generated by the ramp generator. A counter which is initially reset, and supplied with an appropriate clock, keeps counting until the ramp voltage crosses the analog input voltage. At this point, the output of counter is latched and represents the converted digital value for input analog voltage. A traditional CIS digital CDS circuit takes as input two correlated samples at two different time instances. The first sample corresponds to the reset noise of the pixel and the second sample to the actual signal superimposed with the reset noise. A digital CIS CDS circuit then takes the difference between the two samples, thereby, eliminating reset noise during ADC conversion. In an SS-ADC the difference is taken by simply making the counter 'up' count for one sample and 'down' count for the second.

We utilize the noise cancelling, differencing behavior of the CIS digital CDS circuit already available on commercial CIS chips to implement positive and negative weights and implement ReLU. First, each weight transistor embedded inside a pixel is 'tagged' as a positive or a 'negative weight' by connecting it to 'red lines' (marked as VDD for positive weights in Fig. 2) and 'green lines' (marked as VDD for negative weights in Fig. 2). For each channel, we activate multiple pixels to perform an inner-product and read out two samples. The first sample corresponds to a high VDD voltage applied on the 'red lines' (marked as VDD for positive weights in Fig. 2) while the 'green lines' (marked as VDD for negative weights in Fig. 2) are kept at ground. The accumulated multi-bit dot product result is digitized by the SS-ADC, while the counter is 'up' counting. The second sample, on the other hand, corresponds to a high VDD voltage applied on the 'green lines' (marked as VDD for negative weights in Fig. 2) while the 'red lines' (marked as VDD for positive weights in Fig. 2) are kept at ground. The accumulated multi-bit dot product result is again digitized and also subtracted from the first sample by the SS-ADC, while the counter is 'down' counting. Thus, the digital CDS circuit first accumulates the convolution output for all positive weights and then subtracts the convolution output for all negative weights for each channel, controlled by respective select lines for individual channels. Note, possible sneak currents flowing between weight transistors representing positive and negative weights can be obviated by integrating a diode in series with weight transistors or by simply splitting each weight transistor into two series connected transistors, where the channel select lines control one of the series connected transistor, while the other transistor is controlled by a select line representing positive/negative weights.

Interestingly, re-purposing the on-chip CDS for implementing positive and negative weights also allows us to easily implement a quantized ReLU operation inside the SS-ADC. ReLU clips negative values to zero. This can be achieved by ensuring that the final count value latched from the counter (after the CDS operation consisting of 'up' counting and then 'down' counting') is either positive or zero. Interestingly, before performing the dot product operation, the counter can be reset to a non-zero value representing the scale factor of the BN layer as described in Section 4. Thus, by embedding multi-pixel convolution operation and re-purposing on-chip CDS and SS-ADC circuit for implementing positive/negative weights, batch-normalization and ReLU operation, our proposed P$^2$M scheme can implement all the computational aspect for the first few layers of a complex CNN within the pixel array enabling massively parallel in-situ computations.

Putting these features together, our proposed P$^2$M circuit computes one channel at a time and has three phases of operation:

1. Reset Phase: First, the voltage on the photodiode node $M$ (see Fig. 2) is pre-charged or reset by activating the

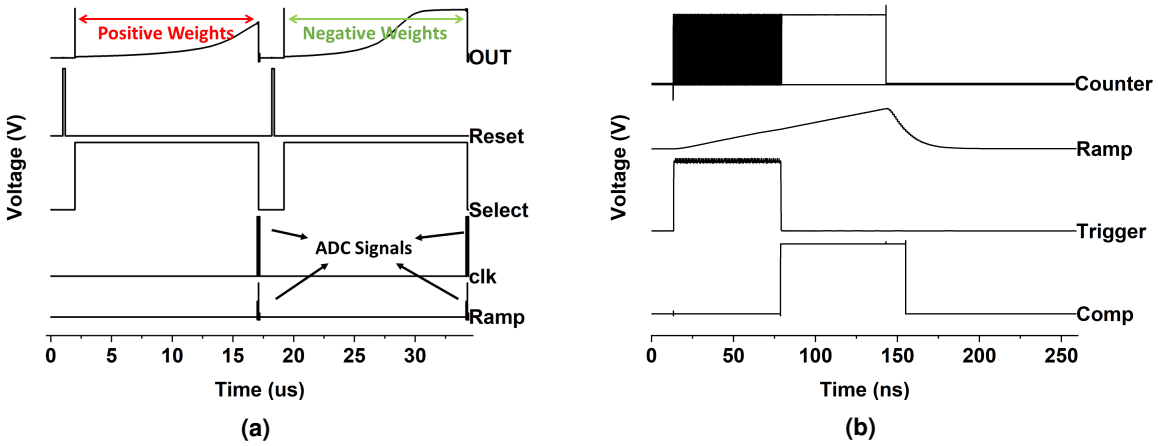


**Figure 4.** (a) A typical timing waveform, showing double sampling (one for positive and other for negative) weights. (b) Typical timing waveform for the SS-ADC showing comparator output (Comp), counter enable (trigger), ramp generator output, and counter clock (Counter).

reset transistor $G_r$. Note, since we aim at performing multi-pixel convolution, the set of pixels $X \times Y \times 3$ are reset, simultaneosuly.

2. Multi-pixel Convolution Phase: Next, we discharge the gate of the reset transistor $G_r$ which deactivates $G_r$. Subsequently, $X \times Y \times 3$ pixels are activated by pulling the gate of respective $G_H$ transistors to VDD. Within the activated set of pixels, a single weight transistor corresponding to a particular channel in the output feature map is activated, by pulling high its gate voltage through the select lines (labeled as select lines for multiple channels in Fig. 2). As the photodiode is sensitive to the incident light, photo-current is generated as light shines upon the diode (for a duration equal to exposure time), and voltage on the gate of $G_s$ is modulated in accordance to the photodiode current that is proportional to the intensity of incident light. The pixel output voltage is a function of the incident light (voltage on node $M$) and the driving strength of the activated weight transistor within each pixel. Pixel output from multiple pixels are accumulated on the column-lines and represent the multi-pixel analog convolution output. The SS-ADC in the periphery converts analog output to a digital value. Note, the entire operation is repeated twice, one for positive weights ('up' counting) and another for negative weights ('down counting').

3. ReLU Operation: Finally, the output of the counter is latched and represents a quantized ReLU output. It is ensured that the latched output is either positive or zero, thereby mimicking the ReLU functionality within the SS-ADC.

The entire P$^2$M circuit is simulated using commercial 22nm Globalfoundries FD-SOI (fully depleted silicon-on-insulator) technology, the SS-ADCs are implemented using a using a bootstrap ramp generator and dynamic comparators. Assuming the counter output which represents the ReLU function is an $N$-bit integer, it needs $2^N$ cycles for a single conversion. The ADC is supplied with a 2GHz clock for the counter circuit. SPICE simulations exhibiting the multiplicative nature of weight transistor embedded pixels with respect to photodiode current is shown in Fig. 3 (a)-(b). A typical timing waveform showing pixel operation along with SS-ADC operation simulated on 22nm Globalfoundries technology node is shown in Fig. 4a.

### 3.4 CIS Process Integration and Area Considerations

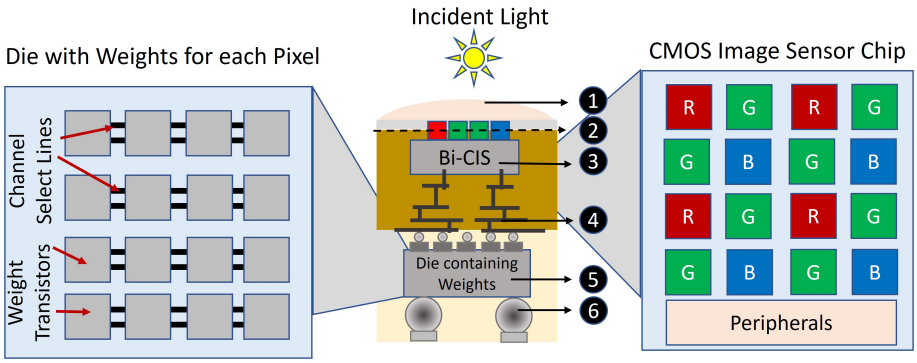


**Figure 5.** representative illustration of heterogeneously integrated system featuring P$^2$M paradigm, built on backside illuminated CMOS image sensor (Bi-CIS). ① Micro lens, ② Light shield, ③ Backside illuminated CMOS Image Sensor (Bi-CIS), ④ Backend of line of the Bi-CIS, ⑤ Die consisting of weight transistors, ⑥ solder bumps for input/output bus (I/O).

In this section, we would like to highlight the viability of the proposed P$^2$M paradigm featuring memory-embedded pixels with respect to its manufacturability using existing foundry processes. A representative illustration of a heterogeneously integrated system catering to the needs of the proposed P$^2$M paradigm is shown in Fig. 5. The figure consists of two key elements, i) backside illuminated CMOS image sensor (Bi-CIS), consisting of photo-diodes, read-out circuits and pixel transistors (reset, source follower and select transistors), and ii) a die consisting of multiple weight transistors per pixel (refer Fig 2). From Fig. 2, it can be seen that each pixel consists of multiple weight transistors that would lead to exceptionally high area overhead. However, with the presented heterogeneous integration scheme of Fig. 5, the weight transistors are vertically aligned below a standard pixel, thereby incurring no (or minimal) increase in footprint. Specifically, each Bi-CIS chip can be implemented in a leading or lagging technology node. The die consisting of weight transistors can be built on an advanced planar or non-planar technology node such that the multiple weight transistors can be accommodated in the same footprint occupied by a single pixel (assuming pixel sizes are larger than the weight transistor embedded memory circuit configuration). The Bi-CIS image sensor chip/die is heterogeneously integrated through a bonding process (die-to-die or die-to-wafer) integrating it onto the die consisting of weight transistors. Preferably, a die-to-wafer low-temperature metal-to-metal fusion with a

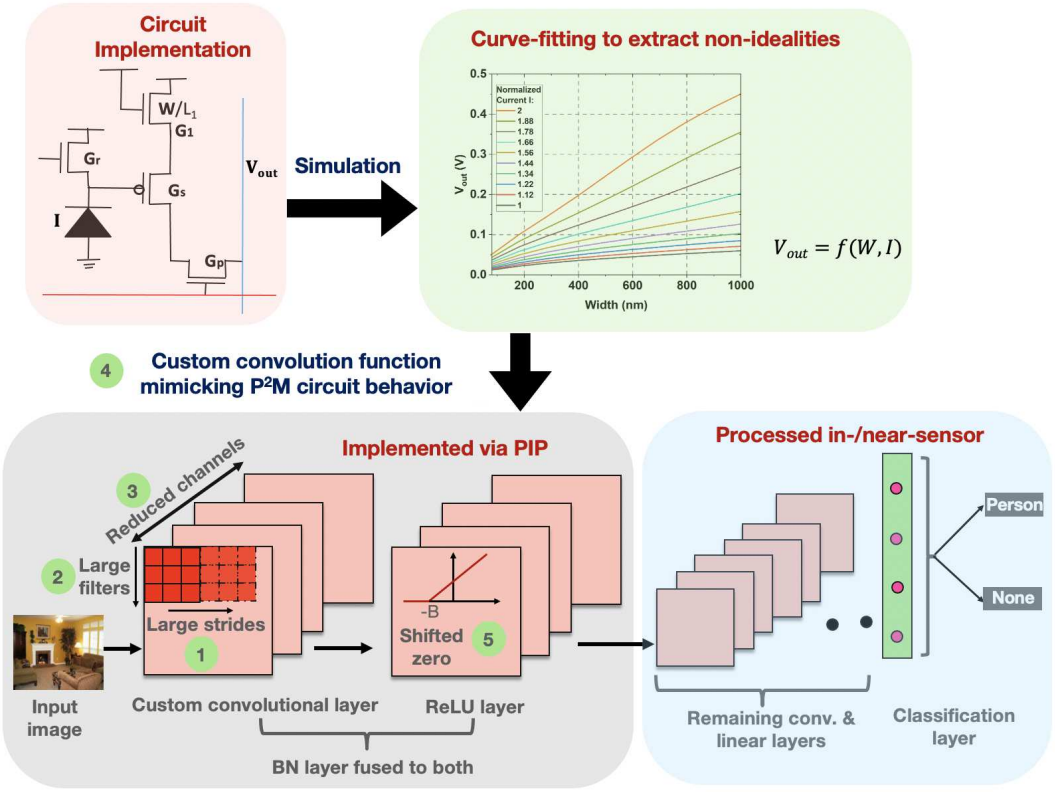


**Figure 6.** Algorithm-circuit co-design framework to enable our proposed P$^2$M approach optimize both the performance and energy-efficiency of vision workloads. We propose the use of ① large strides, ② large kernel sizes, ③reduced number of channels, ④ P$^2$M custom convolution, and ⑤ shifted ReLU operation to incorporate the shift term of the batch normalization layer, for emulating accurate P$^2$M circuit behaviour.

dielectric-to-dielectric direct bonding hybrid process can achieve high-throughput sub-micron pitch scaling with precise vertical alignment[22] . One of the advantages of adapting this heterogeneous integration technology is that chips of different sizes can be fabricated at distinct foundry sources, technology nodes, and functions and then integrated together. In case there are any limitations due to the increased number of transistors in the die consisting of the weights, a conventional pixel-level integration scheme, such as Stacked Pixel Level Connections (SPLC), which shields the logic CMOS layer from the incident light through the Bi-CIS chip region, would also provide a high pixel density and a large dynamic range[23]. Alternatively, one could also adopt the *through silicon via* (TSV) integration technique for front-side illuminated CMOS image sensor (Fi-CIS), wherein the CMOS image sensor is bonded onto the die consisting of memory elements through a TSV process. However, in the Bi-CIS, the wiring is moved away from the illuminated light path allowing more light to reach the sensor, giving better low-light performance[24].

Advantageously, the heterogeneous integration scheme can be used to manufacture P$^2$M sensor systems on existing as well as emerging technologies. Specifically, the die consisting of weight transistors could use a ROM-based structure as shown in Section 3 or other emerging programmable non-volatile memory technologies like PCM[25], RRAM[26], MRAM[27], ferroelectric field effect transistors (FeFETs)[28] etc., manufactured in distinct foundries and subsequently heterogeneously integrated with the CIS die. Thus, the proposed heterogeneous integration allows us to achieve lower area-overhead, while simultaneously enabling seamless, massively parallel convolution. As a result, individual pixels have in-situ access to both activation and weights as needed by the P$^2$M paradigm which obviates the need to transfer weights or activation from one physical location to another through a bandwidth constrained bus. Hence, unlike other multi-chip solutions[9], our approach does not incur energy bottlenecks.

## 4 P$^2$M-constrained Algorithm-Circuit Co-Design

In this section, we present our algorithmic optimizations to standard CNN backbones that are guided by 1) P$^2$M circuit constraints arising due to analog computing nature of the proposed pixel array and the limited conversion precision of on-chip SS-ADCs, 2) the need for achieving state-of-the-art test accuracy, and 3) maximizing desired hardware metrics of high

bandwidth reduction, energy-efficiency and low-latency of P$^2$M computing, and meeting the memory and compute budget of the VWW application. The reported improvement in hardware metrics (illustrated in Section 5.3), is thus a result of intricate circuit-algorithm co-optimization.

### 4.1 Custom Convolution for the First Layer Modeling Circuit Non-Idealities

From an algorithmic perspective, the first layer of a CNN is a linear convolution operation followed by BN, and non-linear (ReLU) activation. The P$^2$M circuit scheme, explained in Section 3, implements convolution operation in analog domain using modified memory-embedded pixels. The constituent entities of these pixels are transistors, which are inherently non-linear devices. As such, in general, any analog convolution circuit consisting of transistor devices will exhibit non-ideal non-linear behavior with respect to the convolution operation. Many existing works, specifically in the domain of memristive analog dot product operation, ignore non-idealities arising from non-linear transistor devices[29,30]. In contrast, to capture these non-linearities, we performed extensive simulations of the presented P$^2$M circuit spanning wide range of circuit parameters like the width of weight transistors, photodiode current *etc.* based on commercial 22nm Globafoundries transistor technology node. The resulting SPICE results, *i.e.* the pixel output voltages corresponding to a range of weights and photodiode currents, were modeled using a behavioral curve-fitting function. The generated function was then included in our algorithmic framework, replacing the convolution operation in the first layer of the network. In particular, we accumulate the output of the curve-fitting function, one for each pixel in the receptive field (we have 3 input channels, and a kernel size of $5\times5$, and hence, our receptive field size is 75), to model each inner-product generated by the in-pixel convolutional layer. This algorithmic framework was then used to optimize the CNN training for the VWW dataset.

### 4.2 Circuit-Algorithm Co-optimization of CNN Backbone subject to P$^2$M Constrains

As explained in Section 3.1, the P$^2$M circuit scheme maximizes parallelism and data bandwidth reduction by activating multiple pixels and reading multiple parallel analog convolution operations for a given channel in the output feature map. The analog convolution operation is repeated for each channel in the output feature map serially. Thus, parallel convolution in the circuit tends to improve parallelism, bandwidth reduction, energy-efficiency and speed. But, increasing the number of channels in the first layer increases the serial aspect of the convolution and degrades parallelism, bandwidth reduction, energy-efficiency, and speed. This creates an intricate circuit-algorithm trade-off, wherein the backbone CNN has to be optimized for having larger kernel sizes (that increases the concurrent activation of more pixels, helping parallelism) and non-overlapping strides (to reduce the dimensionality in the downstream CNN layers, thereby reducing the number of multiply-and-adds and peak memory usage), smaller number of channels (to reduce serial operation for each channel), while maintaining close to state-of-the-art classification accuracy and taking into account the non-idealities associated with analog convolution operation. Also, decreasing number of channels decreases the number of weight transistors embedded within each pixel (each pixel has weight transistors equal to the number of channels in the output feature map), improving area and power consumption. Furthermore, the resulting smaller output activation map (due to reduced number of channels, and larger kernel sizes with non-overlapping strides) reduces the energy incurred in transmission of data from the CIS to the downstream CNN processing unit and the number of floating point operations (and consequently, energy consumption) in downstream layers.

In addition, we propose to fuse the BN layer, partly in the preceding convolutional layer, and partly in the succeeding ReLU layer to enable its implementation via P$^2$M. Let us consider a BN layer with $\gamma$ and $\beta$ as the trainable parameters, which remain fixed during inference. During the training phase, the BN layer normalizes feature maps with a running mean $\mu$ and a running variance $\sigma$. However, during inference, $\mu$ and $\sigma$ are computed from the mini-batch statistics and kept fixed[31], and hence, the BN layer implements a linear function, as shown below.

$$Y = \gamma\frac{X-\mu}{\sqrt{\sigma^2+\varepsilon}} + \beta = \left(\frac{\gamma}{\sqrt{\sigma^2+\varepsilon}}\right)\cdot X + \left(\beta - \frac{\gamma\mu}{\sqrt{\sigma^2+\varepsilon}}\right) = A\cdot X + B \tag{1}$$

We propose to fuse the scale term $A$ into the weights (value of the pixel embedded weight tensor is $A\cdot\theta$, where $\theta$ is the final weight tensor obtained by our training) that are embedded as the transistor widths in the pixel array. Additionally, we propose to use a shifted ReLU activation function, following the covolutional layer, as shown in Fig. 6 to incorporate the shift term $B$. We use the counter-based ADC implementation illustrated in Section 3.3 to implement the shifted ReLU activation. This can be easily achieved by resetting the counter to a non-zero value corresponding the the term $B$ at the start of the convolution operation, as opposed to resetting the counter to zero.

Moreover, to minimize the energy cost of the analog-to-digital conversion in our P$^2$M approach, we must also quantize the layer output to as few bits as possible subject to achieving the desired accuracy. We train a floating-point model with close to state-of-the-accuracy, and then perform quantization in the first convolutional layer to obtain low-precision weights and activations during inference[32]. We also quantize the mean, variance, and the trainable parameters of the BN layer, as all these affect the shift term $B$ (please see Eq. 1), that should be quantized for the low-precision shifted ADC implementation. We avoid

| Hyperparameter | Value |
|---|---|
| kernel size of the convolutional layer ($k$) | 5 |
| padding of the convolutional layer ($p$) | 0 |
| stride of the convolutional layer ($s$) | 5 |
| number of output channels of the convolutional layer ($c_o$) | 8 |
| bit-precision of the P$^2$M-enabled convolutional layer output ($N_b$) | 8 |

**Table 1.** Model hyperparameters and their values to enable bandwidth reduction in the in-pixel layer.

quantization-aware training[33] because it significantly increases the training cost with negligible reduction in bit-precision for our model. With the bandwidth reduction obtained by all these approaches, the output feature map of the P$^2$M-implemented layers can more easily be implemented in micro-controllers with extremely low memory footprint, while P$^2$M itself greatly improves the energy-efficiency of the first layer. Our approach can thus enable TinyML applications that usually have a tight compute and memory budget, as illustrated in Section 5.1

### 4.3 Quantification of bandwidth reduction

To quantify the bandwidth reduction (BR) after the first layer obtained by P$^2$M (BN and ReLU layers do not yield any BR), let the number of elements in the RGB input image be $I$ and in the output activation map after the ReLU activation layer be $O$. Then, $BR$ can be estimated as

$$BR = \left(\frac{O}{I}\right)\left(\frac{4}{3}\right)\left(\frac{12}{N_b}\right) \tag{2}$$

Here, the factor $\left(\frac{4}{3}\right)$ represents the compression from Bayer's pattern of RGGB pixels to RGB pixels because we can either ignore the additional green pixel or design the circuit to effectively take the average of the photo-diode currents coming from the green pixels. The factor $\frac{12}{N_b}$ represents the ratio of the bit-precision between the image pixels captured by the sensor (pixels typically have a bit-depth of 12[34]) and the quantized output of our convolutional layer denoted as $N_b$. Let us now substitute

$$O = \left(\frac{i-k+2*p}{s}+1\right)^2 * c_o, \quad I = i^2 * 3 \tag{3}$$

into Eq. 2, where $i$ denotes the spatial dimension of the input image, and $k$, $p$, $s$ denote the kernel size, padding and stride of the in-pixel convolutional layer, respectively. These hyperparameters, along with $N_b$ are obtained via a thorough algorithmic design space exploration with the goal of achieving the best accuracy, subject to meeting the hardware constraints and the memory and compute budget of our TinyML benchmark. We show their values in Table 1, and substitute them in Eq. 2 to obtain a BR of 21×.

## 5 Experimental Results

### 5.1 Benchmarking Dataset & Model

This paper focuses on the potential of P$^2$M for TinyML applications, *i.e.*, with models that can be deployed on low-power IoT devices with only a few kilobytes of on-chip memory[35–37]. In particular, the Visual Wake Words (VWW) dataset[38] presents a relevant use case for visual TinyML. It consists of high resolution images that include visual cues to "wake-up" AI-powered home assistant devices, such as Amazon's Astro[39], that requires real-time inference in resource-constrained settings. The goal of the VWW challenge is to detect the presence of a human in the frame with very little resources - close to 250KB peak RAM usage and model size[38]. To meet these constraints, current solutions involve downsampling the input image to medium resolution (224×224) which costs some accuracy[32]. We choose MobileNetV2[40] as our baseline CNN architecture with 32 and 320 channels for the first and last convolutional layers respectively that supports full resolution (560×560) images. In order to avoid overfitting to only two classes in the VWW dataset, we decrease the number of channels in the last depthwise separable convolutional block by 3×. MobileNetV2, similar to other MobileNet class of models, is very compact[40] with size less than the maximum allowed in the VWW challenge. It performs well on complex datasets like ImageNet[41] and, as shown in Section 5, does very well on VWWs.

To evaluate P$^2$M on MobileNetV2, we create a custom model that replaces the first convolutional layer with our P$^2$M custom layer that captures the systematic non-idealities of the analog circuits, the reduced number of output channels, and limitation of non-overlapping strides, as discussed in Section 4.

| Image Resolution | Model | Test Accuracy (%) | Number of MAdds (G) | Peak memory usage (MB) |
|---|---|---|---|---|
| 560× 560 | baseline | 91.37 | 1.93 | 7.53 |
| | P$^2$M custom | 89.90 | 0.27 | 0.30 |
| 225× 225 | baseline | 90.56 | 0.31 | 1.2 |
| | P$^2$M custom | 84.30 | 0.05 | 0.049 |
| 115× 115 | baseline | 91.10 | 0.09 | 0.311 |
| | P$^2$M custom | 80.00 | 0.01 | 0.013 |

**Table 2.** Test accuracies, number of MAdds, and peak memory usage of baseline and P$^2$M custom compressed model while classifying on the VWW dataset for different input image resolutions.

| Authors | Description | Model architecture | Test Accuracy (%) |
|---|---|---|---|
| Saha et al. (2020)[32] | RNNPooling | MobileNetV2 | 89.65 |
| Han et al. (2019)[42] | ProxylessNAS | Non-standard architecture | 90.27 |
| Banbury et al. (2021)[37] | Differentiable NAS | MobileNet-V2 | 88.75 |
| Zhoue et al. (2021)[43] | Analog compute-in-memory | MobileNet-V2 | 85.7 |
| This work | P$^2$M | MobileNet-V2 | 89.90 |

**Table 3.** Performance comparison of the proposed P$^2$M-compatible models with state-of-the-art deep CNNs on VWW dataset.

We train both the baseline and P$^2$M custom models in PyTorch using the SGD optimizer with momentum equal to 0.9 for 100 epochs. The baseline model has an initial learning rate (LR) of 0.03, while the custom counterpart has an initial LR of 0.003. Both the learning rates decay by a factor of 0.2 at every 35 and 45 epochs. After training a floating-point model with the best validation accuracy, we perform quantization to obtain 8-bit integer weights, activations, and the parameters (including the mean and variance) of the BN layer. All experiments are performed on a Nvidia 2080Ti GPU with 11 GB memory.

### 5.2 Classification Accuracy

*Comparison between baseline and P$^2$M custom models*: We evaluated the performance of the baseline and P$^2$M custom MobileNet-V2 models on the VWW dataset in Table 2. Our baseline model currently yields the best test accuracy on the VWW dataset among the models available in literature that does not leverage any additional pre-training or augmentation. Note that our baseline model requires a significant amount of peak memory and MAdds (∼30× more than that allowed in the VWW challenge), however, serves a good benchmark for comparing accuracy. We observe that the P$^2$M-enabled custom model can reduce the number of MAdds by ∼7.15×, and peak memory usage by ∼25.1× with 1.47% drop in the test accuracy compared to the uncompressed baseline model for an image resolution of 560×560. With the memory reduction, our P$^2$M model can run on tiny micro-controllers with only 270 KB of on-chip SRAM. Note that peak memory usage is calculated using the same convention as[38]. Notice also that both the baseline and custom model accuracies drop (albeit the drop is significantly higher for the custom model) as we reduce the image resolution, which highlights the need for high-resolution images and the efficacy of P$^2$M in both alleviating the bandwidth bottleneck between sensing and processing, and reducing the number of MAdds for the downstream CNN processing.

*Comparison with SOTA models*: Table 3 provides a comparison of the performances of models generated through our algorithm-circuit co-simulation framework with SOTA TinyML models for VWW. Our P$^2$M custom models yield test accuracies within 0.37% of the best performing model in the literature[42]. Note that we have trained our models solely based on the training data provided, whereas ProxylessNAS[42], that won the 2019 VWW challenge leveraged additional pretraining with ImageNet. Hence, for consistency, we report the test accuracy of ProxylessNAS with identical training configurations on the final network provided by the authors, similar to[32]. Note that[43] leveraged massively parallel energy-efficient analog in-memory computing to implement MobileNet-V2 for VWW, but incurs an accuracy drop of 5.67% and 4.43% compared to our baseline and the previous state-of-the-art[42] models. This probably implies the need for intricate algorithm-hardware co-design and accurately modeling of the hardware non-idealities in the algorithmic framework, as shown in our work.

*Effect of quantization of the in-pixel layer*: As discussed in Section 4, we quantize the output of the first convolutional layer of our proposed model after training to reduce the power consumption due to the sensor ADCs and compress the output as outlined in Eq. 2. We sweep across output bit-precisions of {4,6,8,16,32} to explore the trade-off between accuracy and compression/efficiency as shown in Fig. 7(a). We choose a bit-width of 8 as it is the lowest precision that does not yield any accuracy drop compared to the full-precision models. As shown in Fig. 7, the weights in the in-pixel layer can also be quantized

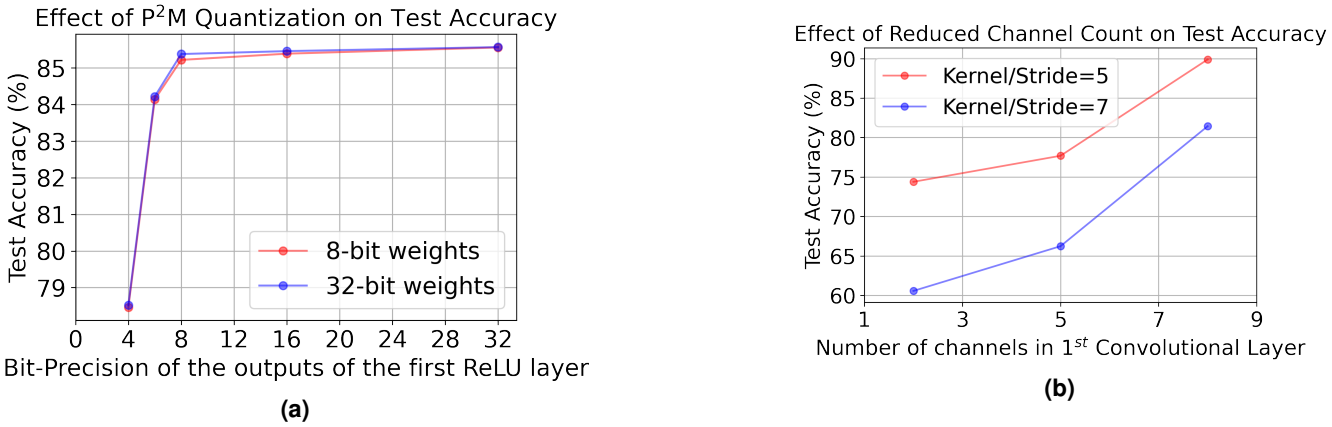


**Figure 7.** (a) Effect of quantization of the in-pixel output activations, and (b) Effect of the number of channels in the $1^{st}$ convolutional layer for different kernel sizes and strides, on the test accuracy of our P$^2$M custom model.

to 8 bits with an 8-bit output activation map, with less than 0.1% drop in accuracy.

*Ablation study*: We also study the accuracy drop incurred due to each of the three modifications (non-overlapping strides, reduced channels, and custom function) in the P$^2$M-enabled custom model. Incorporation of the non-overlapping strides (stride of 5 for 5×5 kernels from a stride of 2 for 3×3 in the baseline model) leads to an accuracy drop of 0.58%. Reducing the number of output channels of the in-pixel convolution by 4× (8 channels from 32 channels in the baseline model), on the top of non-overlapping striding, reduces the test accuracy by 0.33%. Additionally, replacing the element-wise multiplication with the custom P$^2$M function in the convolution operation reduces the test accuracy by a total of 0.56% compared to the baseline model. Note that we can further compress the in-pixel output by either increasing the stride value (changing the kernel size proportionately for non-overlapping strides) or decreasing the number of channels. But both of these approaches reduce the VWW test accuracy significantly, as shown in Fig. 7(b).

| Model type | Sensing (pJ) ($e_{pix}$) | ADC (pJ) ($e_{adc}$) | SoC comm. (pJ) ($e_{com}$) | MAdds (pJ) ($e_{mac}$) | Sensor output pixel ($N_{pix}$) |
|---|---|---|---|---|---|
| P$^2$M (ours) | 148 | 41.9 | 900 | 1.568 | 112×112×8 |
| Baseline (C) | 312 | 86.14 | | | 560×560×3 |
| Baseline (NC) | | | | | |

**Table 4.** Energy estimates for different hardware components. The energy values are measured for designs in 22nm CMOS technology. For the $e_{mac}$, we convert the corresponding value in 45nm to that of 22nm by following standard scaling strategy[44].

| Notation | Description | Value |
|---|---|---|
| $B_{IO}$ | I/O band-width | 64 |
| $B_W$ | Weight representation bit-width | 32 |
| $N_{bank}$ | Number of memory banks | 4 |
| $N_{mult}$ | Number of multiplication units | 175 |
| $T_{sens}$ | Sensor read delay | 35.84 ms (P$^2$M)<br>39.2 ms (baseline) |
| $T_{adc}$ | ADC operation delay | 0.229 ms (P$^2$M)<br>4.58 ms (baseline) |
| $t_{mult}$ | Time required to perform 1 mult. in SoC | 5.48 ns |
| $t_{read}$ | Time required to perform 1 read from SRAM in SoC | 5.48 ns |

**Table 5.** The description and values of the notations used for computation of delay. Note that we strategy and calculated the delay in 22nm technology for 32-bit read and MAdd operations by applying standard technology scaling rules initial values in 65nm technology[45]. We directly evaluated the $T_{read}$ and $T_{adc}$ through circuit simulations in 22nm technology node.

### 5.3 EDP Estimation

We develop a circuit-algorithm co-simulation framework to characterize the energy and delay of our baseline and P$^2$M-implemented VWW models. The total energy consumption for both these models can be partitioned into three major components: sensor ($E_{sens}$), sensor-to-SoC communication ($E_{com}$), and SoC energy ($E_{soc}$). Sensor energy can be further decomposed to pixel read-out ($E_{pix}$) and analog-to-digital conversion (ADC) cost ($E_{adc}$). $E_{soc}$, on the other hand, is primarily composed of the MAdd operations ($E_{mac}$) and parameter read ($E_{read}$) cost. Hence, the total energy can be approximated as:

$$E_{tot} \approx \underbrace{(e_{pix}+e_{adc})*N_{pix}}_{E_{sens}} + \underbrace{e_{com}*N_{pix}}_{E_{com}} + \underbrace{e_{mac}*N_{mac}}_{E_{mac}} + \underbrace{e_{read}*N_{read}}_{E_{read}}. \tag{4}$$

Here, $e_{sens}$ and $e_{com}$ represents per-pixel sensing and communication energy, respectively. $e_{mac}$ is the energy incurred in one MAC operation, $e_{read}$ represents a parameter's read energy, and $N_{pix}$ denotes the number of pixels communicated from sensor to SoC. For a convolutional layer that takes an input $\mathbf{I} \in R^{h_i \times w_i \times c_i}$ and weight tensor $\theta \in R^{k \times k \times c_i \times c_o}$ to produce output $\mathbf{O} \in R^{h_o \times w_o \times c_o}$, the $N_{mac}$[46] and $N_{read}$ can be computed as,

$$N_{mac} = h_o * w_o * k^2 * c_i * c_o \tag{5}$$

$$N_{read} = k^2 * c_i * c_o \tag{6}$$

The energy values we have used to evaluate $E_{tot}$ are presented in Table 4. While $e_{pix}$ is obtained from our circuit simulations, $e_{adc}$ and $e_{com}$ are obtained from[47] and[48] respectively. We ignore the value of $E_{read}$ as it corresponds to only a small fraction ($<10^{-4}$) of the total energy, similar to[49–52]. Fig. 8(a) shows the comparison of energy costs for standard vs P$^2$M-implemented models. In particular, P$^2$M can yield an energy reduction of up to 7.81×. Moreover, the energy savings is larger when the feature map needs to be transferred from an edge device to the cloud for further processing, due to the high communication costs. Note, here we assumed two baseline scenarios one with compression and one without compression. The first baseline is MobileNetV2 which aggressively down-samples the input similar to P$^2$M ($h_i/w_i : 560 \longrightarrow h_o/w_o : 112$). For the second baseline model, we assumed standard first layer convolution kernels causing standard feature down-sampling ($h_i/w_i : 560 \longrightarrow h_o/w_o : 279$).

To evaluate the delay of the models we assume sequential execution of the layer operations[45,53,54] and compute a single convolutional layer delay as[45]

$$t_{conv} \approx \lceil \frac{(k)^2 c_i c_o}{(B_{IO}/B_W) N_{bank}} \rceil * t_{read} + \lceil \frac{(k)^2 c_i c_o}{N_{Mult}} \rceil h_o * w_o * t_{mult}. \tag{7}$$

Based on this sequential assumption, the approximate compute delay for a single forward pass for our P$^2$M model can be given by

$$T_{delay} \approx T_{sens} + T_{adc} + T_{conv}. \tag{8}$$

Here, $T_{sens}$ and $T_{adc}$ correspond to the delay associated to the sensor read and ADC operation respectively. $T_{conv}$ corresponds to the delay associated with all the convolutional layers where each layer's delay is computed by Eq. 7. Fig. 8(b) shows the comparison of delay between P$^2$M and the corresponding baselines where the total delay is computed with the sequential sensing and SoC operation assumption. In particular, the proposed P$^2$M approach can yield an improved delay of up to 2.15×. Thus the total EDP advantage of P$^2$M can be up to 16.76×. On the other hand, even with the conservative assumption of total delay is estimated as $\max(T_{sens}+T_{adc}, T_{conv})$, the EDP advantage can be up to ∼11×.

## 6 Conclusions

With the increased availability of high-resolution image sensors, there has been a growing demand for energy-efficient on-device AI solutions. To mitigate the large amount of data transmission between the sensor and the on-device AI accelerator/processor, we propose a novel paradigm called *Processing-in-Pixel-in-Memory* (P$^2$M) which leverages advanced CMOS technologies to enable the pixel array to perform a wider range of complex operations, including many operations required by modern convolutional neural networks (CNN) pipelines, such as multi-channel, multi-bit convolution, BN and ReLU activation. Consequently, only the compressed meaningful data, for example after the first few layers of custom CNN processing, is transmitted downstream to the AI processor, significantly reducing the power consumption associated with the sensor ADC and required data transmission bandwidth. Our experimental results yield reduction of data rates after the sensor ADCs by up to ∼21× compared to standard near-sensor processing solutions, significantly reducing the complexity of downstream processing. This, in fact, enables the use of relatively low-cost micro-controllers for many low-power embedded vision applications and unlocks a wide range of visual TinyML applications that require high resolution images for accuracy, but are bounded by compute and memory usage. We can also leverage P$^2$M for even more complex applications, where downstream processing can be implemented using existing near-sensor computing techniques that leverage advanced 2.5 and 3D integration technologies[55].

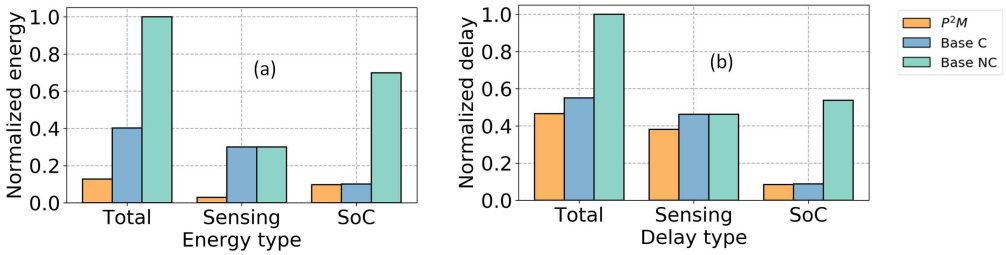


**Figure 8.** Comparison of normalized *total*, *sensing*, and *SoC* (a) energy cost and (b) delay between the P$^2$M, and baseline models architectures (compressed C, and non-compressed NC). Note, the normalization of each component was done by diving the corresponding energy (delay) value with the maximum total energy (delay) value of the three components.

## Acknowledgements

We would like to acknowledge the DARPA HR00112190120 award for supporting this work. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA.

## Author contributions statement

GD and SK proposed the use of P$^2$M for TinyML applications, developed the baseline and P$^2$M-constrained models, and analyzed their accuracies. GD and SK analyzed the EDP improvements over other standard implementations with the help of AJ$^2$ and ZY. AJ$^1$ and AJ$^2$ proposed the idea of P$^2$M and ZY and RL developed the corresponding circuit simulation framework. JM helped to incorporate the non-ideality in the P$^2$M layer in the ML framework. GD and AJ$^2$ wrote majority of the paper, while SK, AJ$^1$ and ZY wrote the remaining portions. AJ$^1$ helped in manufacturing feasibility analysis and proposed the use of heterogeneous integration scheme for P$^2$M. PB supervised the research and edited the manuscript extensively. All authors reviewed the manuscript. Note that AJ$^1$ and AJ$^2$ are Ajey P. Jacob and Akhilesh R. Jaiswal respectively.

## References

1. Xie, J. *et al.* Deep learning-based computer vision for surveillance in its: Evaluation of state-of-the-art methods. *IEEE Transactions on Veh. Technol.* **70**, 3027–3042 (2021).
2. Iqbal, U., Perez, P., Li, W. & Barthelemy, J. How computer vision can facilitate flood management: A systematic review. *Int. J. Disaster Risk Reduct.* **53**, 102030 (2021).
3. Gomez, A., Salazar, A. & Vargas, F. Towards automatic wild animal monitoring: Identification of animal species in camera-trap images using very deep convolutional neural networks. *arXiv preprint arXiv:1603.06169* (2016). 1603.06169.
4. Scaling CMOS Image Sensors. https://semiengineering.com/scaling-cmos-image-sensors/ (2020). Accessed: 04-20-2020.
5. Sejnowski, T. J. The unreasonable effectiveness of deep learning in artificial intelligence. *Proc. Natl. Acad. Sci.* **117**, 30033–30038 (2020).
6. Fossum, E. Cmos image sensors: electronic camera-on-a-chip. *IEEE Transactions on Electron Devices* **44**, 1689–1698, DOI: 10.1109/16.628824 (1997).
7. Buckler, M., Jayasuriya, S. & Sampson, A. Reconfiguring the imaging pipeline for computer vision. *2017 IEEE Int. Conf. on Comput. Vis. (ICCV)* 975–984 (2017).
8. Pinkham, R., Berkovich, A. & Zhang, Z. Near-sensor distributed dnn processing for augmented and virtual reality. *IEEE J. on Emerg. Sel. Top. Circuits Syst.* **11**, 663–676, DOI: 10.1109/JETCAS.2021.3121259 (2021).
9. Sony to Release World's First Intelligent Vision Sensors with AI Processing Functionality. https://www.sony.com/en/SonyInfo/News/Press/202005/20-037E/ (2020). Accessed: 12-01-2022.
10. Chen, Z. *et al.* Processing near sensor architecture in mixed-signal domain with cmos image sensor of convolutional-kernel-readout method. *IEEE Transactions on Circuits Syst. I: Regul. Pap.* **67**, 389–400 (2020).
11. Mennel, L. *et al.* Ultrafast machine vision with 2D material neural network image sensors. *Nature* **579**, 62–66 (2020).
12. Bose, L., Dudek, P., Chen, J., Carey, S. J. & Mayol-Cuevas, W. W. Fully embedding fast convolutional networks on pixel processor arrays. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXIX*, vol. 12374, 488–503 (Springer, 2020).

**13.** Song, R., Huang, K., Wang, Z. & Shen, H. A reconfigurable convolution-in-pixel cmos image sensor architecture. *arXiv preprint arXiv:2101.03308* (2021). 2101.03308.

**14.** Jaiswal, A. & Jacob, A. P. Integrated pixel and two-terminal non-volatile memory cell and an array of cells for deep in-sensor, in-memory computing (2021). US Patent 11,195,580.

**15.** Jaiswal, A. & Jacob, A. P. Integrated pixel and three-terminal non-volatile memory cell and an array of cells for deep in-sensor, in-memory computing (2021). US Patent 11,069,402.

**16.** Angizi, S., Tabrizchi, S. & Roohi, A. Pisa: A binary-weight processing-in-sensor accelerator for edge image processing. *arXiv preprint arXiv:2202.09035* (2022).

**17.** He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385* (2015). 1512.03385.

**18.** Iandola, F. N. *et al.* SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. *arXiv preprint arXiv:1602.07360* (2016). 1602.07360.

**19.** Jogin, M. *et al.* Feature extraction using convolution neural networks (CNN) and deep learning. In *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, vol. 1, 2319–2323 (2018).

**20.** Cho, K., Kim, D. & Song, M. A low power dual cds for a column-parallel cmos image sensor. *JSTS:Journal Semicond. Technol. Sci.* **12** (2012).

**21.** Ma, J., Masoodian, S., Starkey, D. A. & Fossum, E. R. Photon-number-resolving megapixel image sensor at room temperature without avalanche gain. *Optica* **4**, 1474–1481 (2017).

**22.** Gao, G. *et al.* Chip to wafer hybrid bonding with Cu interconnect: High volume manufacturing process compatibility study. In *2019 International Wafer Level Packaging Conference (IWLPC)*, vol. 1, 1–9 (2019).

**23.** Venezia, V. C. *et al.* 1.5$\mu$m dual conversion gain, backside illuminated image sensor using stacked pixel level connections with 13ke-full-well capacitance and 0.8e-noise. In *2018 IEEE International Electron Devices Meeting (IEDM)*, vol. 1, 10.1.1–10.1.4 (2018).

**24.** Sukegawa, S. *et al.* A 1/4-inch 8Mpixel back-illuminated stacked CMOS image sensor. In *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, vol. 1, 484–485 (2013).

**25.** Lee, B. C. *et al.* Phase-change technology and the future of main memory. *IEEE Micro* **30**, 143–143, DOI: 10.1109/MM.2010.24 (2010).

**26.** Guo, K. *et al.* RRAM based buffer design for energy efficient cnn accelerator. In *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, vol. 1, 435–440, DOI: 10.1109/ISVLSI.2018.00085 (2018).

**27.** Chih, Y.-D. *et al.* 13.3 a 22nm 32Mb embedded STT-MRAM with 10ns read speed, 1M cycle write endurance, 10 years retention at 150°c and high immunity to magnetic field interference. In *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, vol. 1, 222–224 (2020).

**28.** Khan, A., Keshavarzi, A. & Datta, S. The future of ferroelectric field-effect transistor technology. *Nat. Electron.* **3**, 588–597 (2020).

**29.** Jain, S., Sengupta, A., Roy, K. & Raghunathan, A. RxNN: A framework for evaluating deep neural networks on resistive crossbars. *Trans. Comp.-Aided Des. Integ. Cir. Sys.* **40**, 326–338 (2021).

**30.** Lammie, C. & Azghadi, M. R. Memtorch: A simulation framework for deep memristive cross-bar architectures. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 1, 1–5 (2020).

**31.** Bjorck, N., Gomes, C. P., Selman, B. & Weinberger, K. Q. Understanding batch normalization. In Bengio, S. *et al.* (eds.) *Advances in Neural Information Processing Systems*, vol. 31 (Curran Associates, Inc., 2018).

**32.** Saha, O., Kusupati, A., Simhadri, H. V., Varma, M. & Jain, P. RNNPool: Efficient non-linear pooling for RAM constrained inference. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F. & Lin, H. (eds.) *Advances in Neural Information Processing Systems*, vol. 33, 20473–20484 (Curran Associates, Inc., 2020).

**33.** Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R. & Bengio, Y. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. *arXiv preprint arXiv:1602.02830* (2016).

**34.** ON Semiconductor. *CMOS Image Sensor, 1.2 MP, Global Shutter* (220). Rev. 10.

**35.** Ray, P. P. A review on TinyML: State-of-the-art and prospects. *J. King Saud Univ. - Comput. Inf. Sci.* (2021).

**36.** Sudharsan, B. *et al.* TinyML benchmark: Executing fully connected neural networks on commodity microcontrollers. In *2021 IEEE 7th World Forum on Internet of Things (WF-IoT)*, vol. 1, 883–884 (2021).

**37.** Banbury, C. *et al.* Micronets: Neural network architectures for deploying TinyML applications on commodity microcontrollers. In Smola, A., Dimakis, A. & Stoica, I. (eds.) *Proceedings of Machine Learning and Systems*, vol. 3, 517–532 (2021).

**38.** Chowdhery, A., Warden, P., Shlens, J., Howard, A. & Rhodes, R. Visual wake words dataset. *arXiv preprint arXiv:1906.05721* (2019). 1906.05721.

**39.** Meet Astro, a home robot unlike any other. https://www.aboutamazon.com/news/devices/meet-astro-a-home-robot-unlike-any-other (2021). Accessed: 09-28-2021.

**40.** Howard, A. G. *et al.* Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017). 1704.04861.

**41.** Russakovsky, O. *et al.* Imagenet large scale visual recognition challenge. *arXiv preprint arXiv:1409.0575* (2015). 1409.0575.

**42.** Han, S., Lin, J., Wang, K., Wang, T. & Wu, Z. Solution to visual wakeup words challenge'19 (first place). https://github.com/mit-han-lab/VWW (2019).

**43.** Zhou, C. *et al.* Analognets: ML-HW co-design of noise-robust TinyML models and always-on analog compute-in-memory accelerator. *arXiv preprint arXiv:2111.06503* (2021). 2111.06503.

**44.** Stillmaker, A. & Baas, B. Scaling equations for the accurate prediction of CMOS device performance from 180 nm to 7 nm. *Integration* **58**, 74–81 (2017).

**45.** Ali, M. *et al.* IMAC: In-memory multi-bit multiplication and accumulation in 6T sram array. *IEEE Transactions on Circuits Syst. I: Regul. Pap.* **67**, 2521–2531 (2020).

**46.** Kundu, S., Nazemi, M., Pedram, M., Chugg, K. M. & Beerel, P. A. Pre-defined sparsity for low-complexity convolutional neural networks. *IEEE Transactions on Comput.* **69**, 1045–1058 (2020).

**47.** Gonugondla, S. K. *et al.* Fundamental limits on energy-delay-accuracy of in-memory architectures in inference applications. *IEEE Transactions on Comput. Des. Integr. Circuits Syst.* **1** (2021).

**48.** Kodukula, V. *et al.* Dynamic temperature management of near-sensor processing for energy-efficient high-fidelity imaging. *Sensors* **1** (2021).

**49.** Kundu, S., Datta, G., Pedram, M. & Beerel, P. A. Spike-thrift: Towards energy-efficient deep spiking neural networks by limiting spiking activity via attention-guided compression. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 3953–3962 (2021).

**50.** Datta, G., Kundu, S. & Beerel, P. A. Training energy-efficient deep spiking neural networks with single-spike hybrid input encoding. In *2021 International Joint Conference on Neural Networks (IJCNN)*, vol. 1, 1–8 (2021).

**51.** Datta, G. & Beerel, P. A. Can deep neural networks be converted to ultra low-latency spiking neural networks? *arXiv preprint arXiv:2112.12133* (2021). 2112.12133.

**52.** Kundu, S., Pedram, M. & Beerel, P. A. Hire-snn: Harnessing the inherent robustness of energy-efficient deep spiking neural networks by training with crafted input noise. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5209–5218 (2021).

**53.** Kang, M., Lim, S., Gonugondla, S. & Shanbhag, N. R. An in-memory vlsi architecture for convolutional neural networks. *IEEE J. on Emerg. Sel. Top. Circuits Syst.* **8**, 494–505 (2018).

**54.** Datta, G., Kundu, S., Jaiswal, A. & Beerel, P. A. HYPER-SNN: Towards Energy-efficient Quantized Deep Spiking Neural Networks for Hyperspectral Image Classification. *arXiv preprint arXiv:2107.11979* (2021). 2107.11979.

**55.** Amir, M. F. & Mukhopadhyay, S. 3D stacked high throughput pixel parallel image sensor with integrated ReRAM based neural accelerator. *2018 IEEE SOI-3D-Subthreshold Microelectron. Technol. Unified Conf. (S3S)* 1–3 (2018).