# Reconfiguring the Imaging Pipeline for Computer Vision

Mark Buckler
Cornell University

Suren Jayasuriya
Carnegie Mellon University

Adrian Sampson
Cornell University

## Abstract

*Advancements in deep learning have ignited an explosion of research on efficient hardware for embedded computer vision. Hardware vision acceleration, however, does not address the cost of capturing and processing the image data that feeds these algorithms. We examine the role of the image signal processing (ISP) pipeline in computer vision to identify opportunities to reduce computation and save energy. The key insight is that imaging pipelines should be be configurable: to switch between a traditional* photography mode *and a low-power* vision mode *that produces lower-quality image data suitable only for computer vision. We use eight computer vision algorithms and a reversible pipeline simulation tool to study the imaging system's impact on vision performance. For both CNN-based and classical vision algorithms, we observe that only two ISP stages, demosaicing and gamma compression, are critical for task performance. We propose a new image sensor design that can compensate for these stages. The sensor design features an adjustable resolution and tunable analog-to-digital converters (ADCs). Our proposed imaging system's vision mode disables the ISP entirely and configures the sensor to produce subsampled, lower-precision image data. This vision mode can save ∼75% of the average energy of a baseline photography mode with only a small impact on vision task accuracy.*

## 1. Introduction

The deep learning revolution has accelerated progress in a plethora of computer vision tasks. To bring these vision capabilities within the battery budget of a smartphone, a wave of recent work has designed custom hardware for inference in deep neural networks [15, 19, 30]. This work, however, only addresses part of the whole cost: embedded vision involves the entire imaging pipeline, from photons to task result. As hardware acceleration reduces the energy cost of inference, the cost to capture and process images will consume a larger share of total system power [9, 31].

We study the potential for co-design between camera systems and vision algorithms to improve their end-to-end efficiency. Existing imaging pipelines are designed for photography: they produce high-quality images for human consumption. An imaging pipeline consists of the image sensor itself and an *image signal processor* (ISP) chip, both of which are hard-wired to produce high-resolution, low-noise, color-corrected photographs. Modern computer vision algorithms, however, do not require the same level of quality that humans do. Our key observation is that mainstream, photography-oriented imaging hardware wastes time and energy to provide quality that computer vision algorithms do not need.

We propose to make imaging pipelines configurable. The pipeline should support both a traditional *photography mode* and an additional, low-power *vision mode*. In vision mode, the sensor can save energy by producing lower-resolution, lower-precision image data, and the ISP can skip stages or disable itself altogether. We examine the potential for a vision mode in imaging systems by measuring its impact on the hardware efficiency and vision accuracy. We study vision algorithms' sensitivity to sensor parameters and to individual ISP stages, and we use the results to propose an end-to-end design for an imaging pipeline's vision mode.

**Contributions:** This paper proposes a set of modifications to a traditional camera sensor to support a vision mode. The design uses variable-accuracy analog-to-digital converters (ADCs) to reduce the cost of pixel capture and power-gated selective readout to adjust sensor resolution. The sensor's subsampling and quantization hardware approximates the effects of two traditional ISP stages, demosaicing and gamma compression. With this augmented sensor, we propose to disable the ISP altogether in vision mode.

We also describe a methodology for studying the imaging system's role in computer vision performance. We have developed a tool that simulates a configurable imaging pipeline and its inverse to convert plain images to approximate raw signals. This tool is critical for generating training data for learning-based vision algorithms that need examples of images produced by

a hypothetical imaging pipeline. Section 3.2 describes the open-source simulation infrastructure.

We use our methodology to examine eight vision applications, including classical algorithms for stereo, optical flow, and structure-from-motion; and convolutional neural networks (CNNs) for object recognition and detection. For these applications, we find that:

- Most traditional ISP stages are unnecessary when targeting computer vision. For all but one application we tested, only two stages had significant effects on vision accuracy: demosaicing and gamma compression.
- Our image sensor can approximate the effects of demosaicing and gamma compression in the mixed-signal domain. Using these in-sensor techniques eliminates the need for a separate ISP for most vision applications.
- Our image sensor can reduce its bitwidth from 12 to 5 by replacing linear ADC quantization with logarithmic quantization while maintaining the same level of task performance.

Altogether, the proposed vision mode can use roughly a quarter of the imaging-pipeline energy of a traditional photography mode without significantly affecting the performance of most vision algorithms we studied.

## 2. Related Work

**Energy-efficient Deep Learning:** Recent research has focused on dedicated ASICs for deep learning [10, 15, 19, 30, 38**?** ] to reduce the cost of forward inference compared to a GPU or CPU. Our work complements this agenda by focusing on energy efficiency in the rest of the system: we propose to pair low-power vision implementations with low-power sensing circuitry.

**ISPs for Vision:** While most ISPs are fixed-function designs, Vasilyev et al. [45] propose to use a programmable CGRA architecture to make them more flexible, and other work has synthesized custom ISPs onto FPGAs [22, 23]. Mainstream cameras, including smartphones [2], can bypass the ISP to produce RAW images, but the associated impact on vision is not known. Liu et al. [32] propose an ISP that selectively disables stages depending on application needs. We also explore sensitivity to ISP stages, and we propose changes to the image sensor hardware that subsume critical stages in a traditional ISP.

**Image Sensors for Vision:** In industry, some cameras are marketed with vision-specific designs. For example, Centeye [5] offers image sensors based on a logarithmic-response pixel circuit [16] for high dynamic range. Omid-Zohoor et al. [35] propose logarithmic, low-bitwidth ADCs and on-sensor processing for efficient featurization using the histogram of oriented gradients. *Focal-plane processing* can compute basic functions such as edge detection in analog on the sensor [11, 33]. RedEye [30] computes initial convolutions for a CNN using a custom sensor ADC, and Chen et al. [8] approximate the first layer *optically* using angle-sensitive pixels. Event-based vision sensors detect temporal motion with custom pixels [3, 25]. Chakrabarti [7] proposes to learn novel, non-Bayer sensor layouts using backpropagation. We focus instead on minimally invasive changes to existing camera pipelines. To our knowledge, this is the first work to measure vision applications' sensitivity to design decisions in a traditional ISP pipeline. Our proposed pipeline can support both computer vision and traditional photography.

Other work has measured the energy of image sensing: there are potential energy savings when adjusting a sensor's frame rate and resolution [31]. Lower-powered image sensors have been used to decide when to activate traditional cameras and full vision computations [20].

Compressive sensing shares our goal of reducing sensing cost, but it relies on complex computations to recover images [14]. In contrast, our proposed pipeline lets vision algorithms work directly on sensor data without additional image reconstruction.

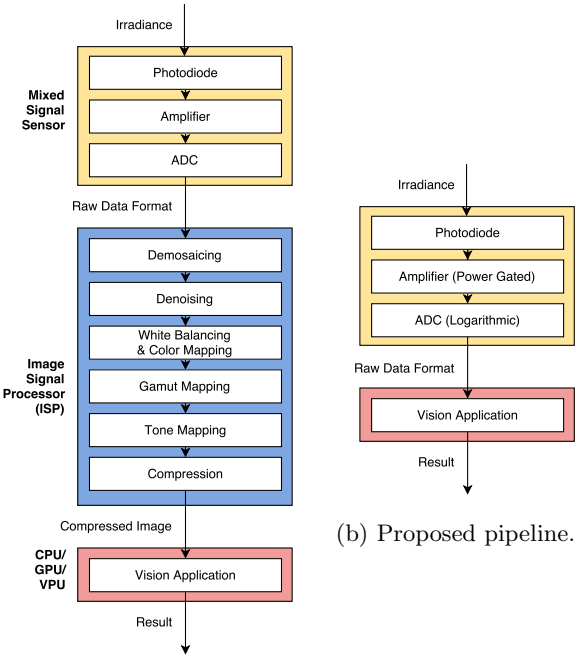
**Error Tolerance in CNNs:** Recent work by Diamond et al. [13] studies the impact of sensor noise and blurring on CNN accuracy and develops strategies to tolerate it. Our focus is broader: we consider a range of sensor and ISP stages, and we measure both CNN-based and "classical" computer vision algorithms.

## 3. Background & Experimental Setup

### 3.1. The Imaging Pipeline

Figure 1a depicts a traditional imaging pipeline that feeds a vision application. The main components are an image sensor, which reacts to light and produces a RAW image; an image signal processor (ISP) unit, which transforms, enhances, and compresses the signal to produce a complete image, usually in JPEG format; and the vision application itself.

ISPs consist of a series of signal processing stages. While the precise makeup of an ISP pipeline varies, we consider a typical set of stages common to all ISP pipelines: denoising, demosaicing, color transformations, gamut mapping, tone mapping, and image compression. This simple pipeline is idealized: modern ISPs can comprise hundreds of proprietary stages. For example, tone mapping and denoising can use complex, adaptive operations that are customized for specific camera hardware. In this paper, we consider a simple form of global tone mapping that performs *gamma com-*




(a) Standard pipeline.

(b) Proposed pipeline.

Figure 1: The standard imaging pipeline (a) and our proposed pipeline (b) for our design's *vision mode*.

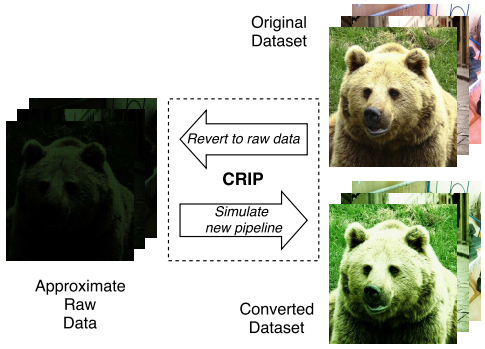



Figure 2: Configurable & Reversible Imaging Pipeline.

*pression*. We also omit analyses that control the sensor, such as autoexposure and autofocus, and specialized stages such as burst photography or high dynamic range (HDR) modes. We select these simple, essential ISP stages because we believe they represent the common functionality that may impact computer vision.

### 3.2. Pipeline Simulation Tool

Many computer vision algorithms rely on machine learning. Deep learning techniques in particular require vast bodies of training images. To make learning-based vision work on our proposed imaging pipelines, we need a way to generate labeled images that look as if they were captured by the hypothetical hardware. Instead of capturing this data from scratch, we develop a toolchain that can *convert* existing image datasets.

The tool, called the Configurable & Reversible Imaging Pipeline (CRIP), simulates an imaging pipeline in "forward" operation and inverts the function in "reverse" mode. CRIP takes as input a standard image file, runs the inverse conversion to approximate a RAW image, and then simulates a specific sensor/ISP configuration to produce a final RGB image. The result recreates the image's color, resolution and quantization as if it had been captured and processed by a specific image sensor and ISP design. Figure 2 depicts the workflow and shows the result of simulating a pipeline with only gamma compression and demosaicing. Skipping color transformations leads to a green hue in the output.

The inverse conversion uses an implementation of Kim et al.'s reversible ISP model [27] augmented with new stages for reverse denoising and demosaicing as well as re-quantization. To restore noise to a denoised image, we use Chehdi et al.'s sensor noise model [43]. To reverse the demosaicing process, we remove channel data from the image according to the Bayer filter. The resulting RAW image approximates the unprocessed output of a camera sensor, but some aspects cannot be reversed: namely, sensors typically digitize 12 bits per pixel, but ordinary 8-bit images have lost this detail after compression. For this reason, we only report results for quantization levels with 8 bits or fewer.

CRIP implements the reverse stages from Kim et al. [27], so its model linearization error is the same as in that work: namely, less than 1%. To quantify CRIP's error when reconstructing RAW images, we used it to convert a Macbeth color chart photograph and compared the result with its original RAW version. The average pixel error was 1.064% and the PSNR was 28.81 dB. Qualitatively, our tool produces simulated RAW images that are visually indistinguishable from their real RAW counterparts.

CRIP's reverse pipeline implementation can use any camera model specified by Kim et al. [27], but for consistency, this paper uses the Nikon D7000 pipeline. We have implemented the entire tool in the domain-specific language Halide [37] for speed. For example, CRIP can convert the entire CIFAR-10 dataset [28] in one hour on an 8-core machine. CRIP is available as open source: `https://github.com/cucapra/approx-vision`

### 3.3. Benchmarks

Table 1 lists the computer vision algorithms we study. It also shows the data sets used for evaluation and, where applicable, training. Our suite consists of 5 CNN-based algorithms and 3 "classical," non-learning

| Algorithm | Dataset | Vision Task |
|---|---|---|
| 3 Deep LeNet [29] | CIFAR-10 [28] | Obj. Classification |
| 20 Deep ResNet [21] | CIFAR-10 | Obj. Classification |
| 44 Deep ResNet [21] | CIFAR-10 | Obj. Classification |
| Faster R-CNN [39] | VOC-2007 [18] | Object Detection |
| OpenFace [1] | CASIA [46] and LFW [24] | Face Identification |
| OpenCV Farneback [26] | Middlebury [41] | Optical Flow |
| OpenCV SGBM [26] | Middlebury | Stereo Matching |
| OpenMVG SfM [34] | Strecha [42] | Structure from Motion |

Table 1: Vision applications used in our evaluation.

implementations covering a range of vision tasks: object classification, object detection, face identification, optical flow, and structure from motion. For object classification, we test 3 different implementations of varying sophistication to examine the impact of neural network depth on error tolerance.

For each experiment, we configure CRIP to apply a chosen set of ISP stages and to simulate a given sensor resolution and ADC quantization. For the CNNs, we convert a training set and train the network starting with pre-trained weights using the same learning rates and hyperparameters specified in the original paper. For all applications, we convert a test set and evaluate performance using an algorithm-specific metric.

## 4. Sensitivity to ISP Stages

We next present an empirical analysis of our benchmark suite's sensitivity to stages in the ISP. The goal is to measure, for each algorithm, the relative difference in task performance between running on the original image data and running on data converted by CRIP.

**Individual Stages:** First, we examine the sensitivity to each ISP stage in isolation. Testing the exponential space of all possible stage combinations is intractable, so we start with two sets of experiments: one that *disables* a single ISP stage and leaves the rest of the pipeline intact (Figure 3a); and one that *enables* a single ISP stage and disables the rest (Figure 3b).

In these experiments, gamut mapping and color transformations have a minimal effect on all benchmarks. The largest effects are on ResNet44, where classification error increases from 6.3% in the baseline to 6.6% without gamut mapping, and OpenMVG, where removing the color transform stage increases RMSE from 0.408 to 0.445. This finding confirms that features for vision are not highly sensitive to color.

There is a strong sensitivity, in contrast, to gamma compression and demosaicing. The OpenMVG Structure from Motion (SfM) implementation fails entirely when gamma compression is disabled: it was unable to find sufficient features using either of its feature extractors, SIFT and AKAZE. Meanwhile, removing demosaicing worsens the error for Farneback optical flow by nearly half, from 0.227 to 0.448. Both of these classical (non-CNN) algorithms use hand-tuned feature extractors, which do not take the Bayer pattern into account. The CIFAR-10 benchmarks (LeNet3, ResNet20, ResNet44) use low-resolution data (32×32), which is disproportionately affected by the removal of color channels in mosaiced data. While gamma-compressed data follows a normal distribution, removing gamma compression reverts the intensity scale to its natural log-normal distribution, which makes features more difficult to detect for both classical algorithms and CNNs.

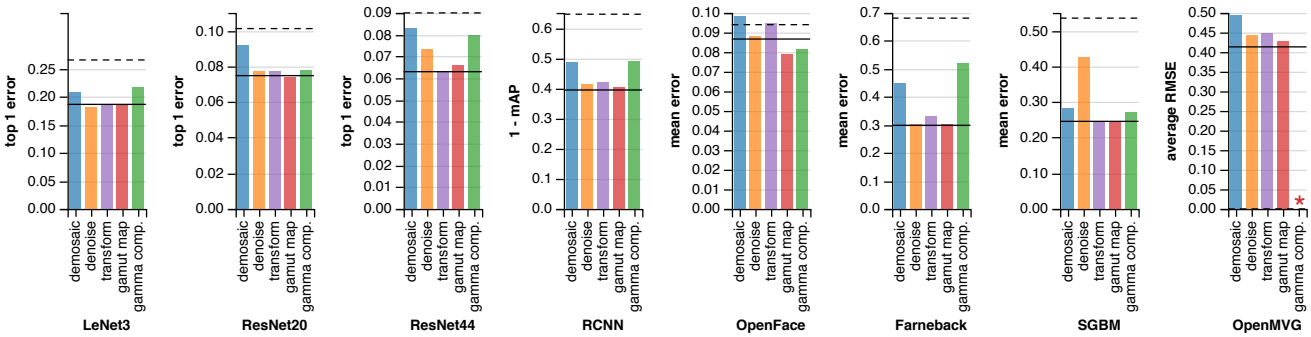
Unlike the other applications, Stereo SGBM is sensitive to noise. Adding sensor noise increases its mean error from 0.245 to 0.425, an increase of over 70%. Also unlike other applications, OpenFace counter-intuitively performs *better* than the baseline when the simulated pipeline omits gamut mapping or gamma compression. OpenFace's error is 8.65% on the original data and 7.9% and 8.13%, respectively, when skipping those stages. We attribute the difference to randomness inherent in the training process. Across 10 training runs, OpenFace's baseline error rate varied from 8.22% to 10.35% with a standard deviation of 0.57%.

**Minimal Pipelines:** Based on these results, we study the effect of combining the most important stages: demosaicing, gamma compression, and denoising. Figure 4 shows two configurations that enable only the first two and all three of these stages. Accuracy for the minimal pipeline with only demosaicing and gamma compression is similar to accuracy on the original data. The largest impact, excluding SGBM, is ResNet44, whose top-1 error increases only from 6.3% to 7.2%. Stereo SGBM, however, is noise sensitive: without denoising, its mean error is 0.33; with denoising, its error returns to its baseline of 0.25. Overall, the CNNs are able to rely on retraining themselves to adapt to changes in the capture pipeline, while classical benchmarks are less flexible and can depend on specific ISP stages.

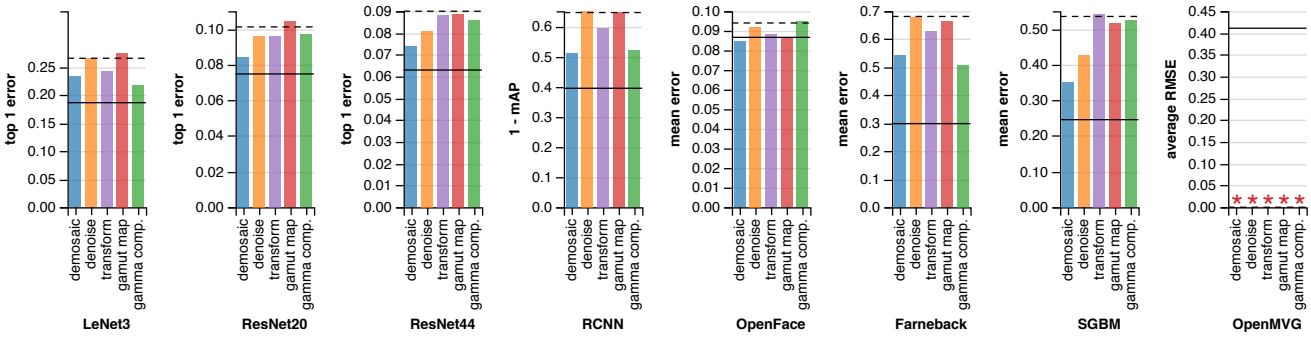
We conclude that demosaicing and gamma compression are the only important stages for all applications except for one, which also benefits from denoising. Our goal in the next section is to show how to remove the need for these two stages to allow vision mode to disable the ISP entirely. For outliers like SGBM, selectively enabling the ISP may still be worthwhile.

## 5. Image Sensor Design

Based on our experiments with limited ISP processing, we propose a new image sensor design that can




(a) Disabling a single ISP stage.




(b) Enabling a single ISP stage and disabling the rest.

Figure 3: The impact on vision accuracy when adding and removing stages from the standard ISP pipeline. The solid line shows the baseline error with all ISP stages enabled, and the dotted line shows the error when all ISP stages are disabled. Asterisks denote aborted runs where no useful output was produced.

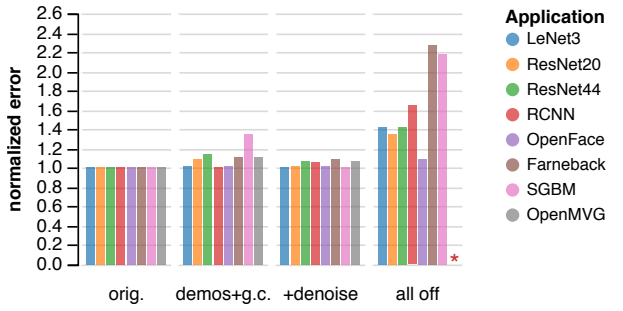
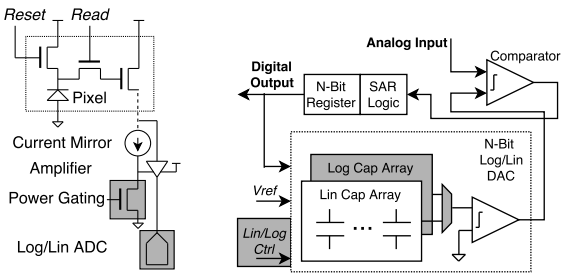



Figure 4: Each algorithm's vision error, normalized to the original error on plain images, for two minimal ISP pipelines. The *demos+g.c.* pipeline only enables demosaicing and gamma compression; the *+denoise* bars also add denoising. The *all off* column shows a configuration with all stages disabled for reference.

operate in a low-power vision mode. We propose three key features: adjustable resolution via selective pixel readout and power gating; subsampling to approximate ISP-based demosaicing; and nonlinear ADC quantization to perform gamma compression. All three are well-known sensor design techniques; we propose to use them in an optional camera mode to replace the ISP's role in embedded vision applications.

**Resolution:** A primary factor in a sensor's energy consumption is the resolution. Frames are typically read out in column-parallel fashion, where each column of pixels passes through amplification and an ADC. Our design can selectively read out a region of interest (ROI) or subset of pixels, and save energy, by power-gating column amplifiers and ADCs. Figure 5a depicts the power-gating circuitry. The image sensor's controller unit can turn the additional transistor on or off to control power for the amplifier and ADC in each column.

**Subsampling:** Section 4 finds that most vision tasks depend on demosaicing for good accuracy. There are many possible demosaicing techniques, but they are typically costly algorithms optimized for perceived image quality. We hypothesize that, for vision algorithms, the nuances of advanced demosaicing techniques are less important than the image format: raw images exhibit the Bayer pattern, while demosaiced images use a standard RGB format.




(a) Column circuitry. (b) Log/lin SAR ADC.

Figure 5: Our proposed camera sensor circuitry, including power gating at the column level (a) and our configurable logarithmic/linear SAR ADC (b).

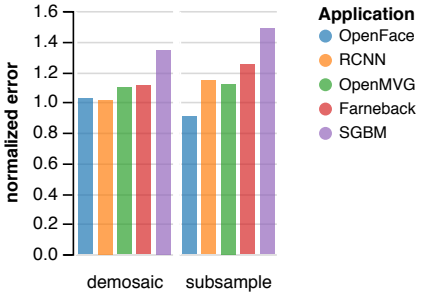



Figure 6: Demosaicing on the ISP vs. subsampling in the sensor. Error values are normalized to performance on unmodified image data.

We propose to modify the image sensor to achieve the same format-change effect as demosaicing without any signal processing. Specifically, our camera's vision mode *subsamples* the raw image to collapse each 2×2 block of Bayer-pattern pixels into a single RGB pixel. Each such block contains one red pixel, two green pixels, and one blue pixel; our technique drops one green pixel and combines it with the remaining values to form the three output channels. The design power-gates one of the two green pixels interprets resulting red, green, and blue values as a single pixel.

**Nonlinear Quantization:** In each sensor column, an analog-to-digital (ADC) converter is responsible for quantizing the analog output of the amplifier to a digital representation. A typical linear ADC's energy cost is exponential in the number of bits in its output: an 12-bit ADC costs roughly twice as much energy as a 11-bit ADC. There is an opportunity to drastically reduce the cost of image capture by reducing the number of bits.

As with resolution, ADC quantization is typically fixed at design time. We propose to make the number of bits configurable for a given imaging mode. Our proposed image sensor uses *successive-approximation* (SAR) ADCs, which support a variable bit depth controlled by a clock and control signal [44].

ADC design can also provide a second opportunity: to change the *distribution* of quantization levels. Nonlinear quantization can be better for representing images because their light intensities are not uniformly distributed: the probability distribution function for intensities in natural images is log-normal [40]. To preserve more information about the analog signal, SAR ADCs can use quantization levels that map the intensities uniformly among *digital* values. (See the supplementary material for a more complete discussion of intensity distributions.) We propose an ADC that uses logarithmic quantization in vision mode. Figure 5b shows the ADC design, which can switch between linear quantization levels for photography mode and logarithmic quantization for vision mode. The design uses a separate capacitor bank for each quantization scheme.

Logarithmic quantization lets the camera capture the same amount of image information using fewer bits, which is the same goal usually accomplished by the gamma compression stage in the ISP. Therefore, we eliminate the need for a separate ISP block to perform gamma compression.

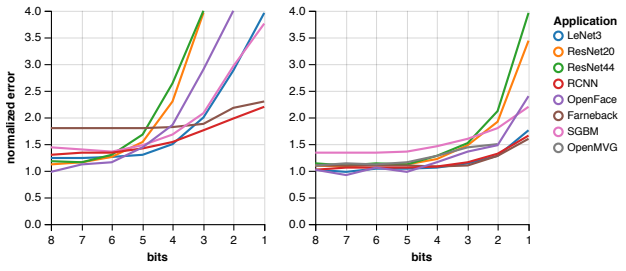
**System Considerations:** Our proposed vision mode controls three sensor parameters: it enables subsampling to produce RGB images; it allows reduced-resolution readout; and it enables a lower-precision logarithmic ADC configuration. The data is sent off-chip directly to the application on the CPU, the GPU, or dedicated vision hardware without being compressed. This mode assumes that the vision task is running in real time, so the image does not need to be saved.

In the traditional photography mode, we configure the ADCs to be at high precision with linear quantization levels. Then the image is sent to the separate ISP chip to perform all the processing needed to generate high quality images. These images are compressed using the JPEG codec on-chip and stored in memory for access by the application processor.

## 6. Sensitivity to Sensor Parameters

We empirically measure the vision performance impact of the design decisions in our camera's vision mode. We again use the CRIP tool to simulate specific sensor configurations by converting image datasets and evaluate the effects on the benchmarks in Table 1.

**Approximate Demosaicing with Subsampling:** We first study subsampling as an alternative to true demosaicing in the ISP. In this study, we omit the benchmarks that work on CIFAR-10 images [28] because their resolution, 32×32, is unrealistically small for a sensor, so subsampling beyond this size is not




(a) Linear quantization. (b) Logarithmic quantization.

Figure 7: Effect of quantization on vision accuracy in a pipeline with only demosaicing enabled.

meaningful. Figure 6 compares data for "true" demosaicing, where CRIP has not reversed that stage, to a version that simulates our subsampling instead. Replacing demosaicing with subsampling leads to a small increase in vision error. Farneback optical flow sees the largest error increase, from 0.332 to 0.375.

**Quantization:** Next, we study the impact of signal quantization in the sensor's ADCs. There are two parameters: the number of bits and the level distribution (linear or logarithmic). Figure 7 shows our vision applications' sensitivity to both bitwidth and distribution. Both sweeps use an ISP pipeline with demosiacing but without gamma compression to demonstrate that the logarithmic ADC, like gamma compression, compresses the data distribution.

The logarithmic ADC yields higher accuracy on all benchmarks than the linear ADC with the same bitwidth. Farneback optical flow's sensitivity is particularly dramatic: using a linear ADC, its mean error is 0.54 and 0.65 for 8 and 2 bits, respectively; while with a logarithmic ADC, the error drops to 0.33 and 0.38.

Switching to a logarithmic ADC also increases the applications' tolerance to smaller bitwidths. All applications exhibit minimal error increases down to 5 bits, and some can even tolerate 4- or 3-bit quantization. OpenMVG's average RMSE only increases from 0.454 to 0.474 when reducing 8 bit logarithmic sampling to 5 bits, and ResNet20's top-1 error increases from 8.2% to 8.42%. To fit all of these applications, we propose a 5-bit logarithmic ADC design in vision mode.

**Resolution:** We next measure the impact of resolution adjustment using column power gating. Modern image sensors use multi-megapixel resolutions, while the input dimensions for most convolutional neural networks are often 256×256 or smaller. While changing the input dimensions of the network itself may also be an option, we focus here on downsampling images to match the network's published input size.

To test the downsampling technique, we concocted a new higher-resolution dataset by selecting a subset of ImageNet [12] which contains the CIFAR-10 [28] object classes (∼15,000 images). These images are higher resolution than the input resolution of networks trained on CIFAR-10, so they let us experiment with image downsampling.

We divide the new dataset into training and testing datasets using an 80–20 balance and train the LeNet, ResNet20, and ResNet44 networks from pre-trained weights. For each experiment, we first downsample the images to simulate sensor power gating. Then, after demosaicing, we scale down the images the rest of the way to 32×32 using OpenCV's edge-aware scaling [26]. Without any subsampling, LeNet achieves 39.6% error, ResNet20 26.34%, and ResNet44 24.50%. We then simulated downsampling at ratios of ¼, 1⁄16, and 1⁄64 resolution. Downsampling does increase vision error, but the effect is small: the drop in accuracy from full resolution to ¼ resolution is approximately 1% (LeNet 40.9%, ResNet20 27.71%, ResNet44 26.5%). Full results are included in this paper's supplemental material.

## 7. Quantifying Power Savings

Here we estimate the potential power efficiency benefits of our proposed vision mode as compared to a traditional photography-oriented imaging pipeline. Our analysis covers the sensor's analog-to-digital conversion, the sensor resolution, and the ISP chip.

**Image Sensor ADCs:** Roughly half of a camera sensor's power budget goes to readout, which is dominated by the cost of analog-to-digital converters (ADCs) [6]. While traditional sensors use 12-bit linear ADCs, our proposal uses a 5-bit logarithmic ADC.

To compute the expected value of the energy required for each ADC's readout, we quantify the probability and energy cost of each digital level that the ADC can detect. The expected value for a single readout is:

$$\mathrm{E}\left[\mathrm{ADC\_energy}\right] = \sum_{m=1}^{2^n} p_m e_m$$

where $n$ is the number of bits, $2^n$ is the total number of levels, $m$ is the level index, $p_m$ is the probability of level $m$ occuring, and $e_m$ is the energy cost of running the ADC at level $m$.

To find $p_m$ for each level, we measure the distribution of values from images in the CIFAR-10 dataset [28] in raw data form converted by CRIP (Section 3.2). To find a relative measure for $e_m$, we simulate the operation of the successive approximation register (SAR) ADC in charging and discharging the capacitors in its bank. This capacitor simulation is a simple first-order model

of a SAR ADC's power that ignores fixed overheads such as control logic.

In our simulations, the 5-bit logarithmic ADC uses 99.95% less energy than the baseline 12-bit linear ADC. As the ADCs in an image sensor account for 50% of the energy budget [6], this means that the cheaper ADCs save approximately half of the sensor's energy cost.

**Image Sensor Resolution:** An image sensor's readout, I/O, and pixel array together make up roughly 95% of its power cost [6]. These costs are linearly related to the sensor's total resolution. As Section 5 describes, our proposed image sensor uses selective readout circuitry to power off the pixel, amplifier, and ADC for subsets of the sensor array. The lower-resolution results can be appropriate for vision algorithms that have low-resolution inputs (Section 6). Adjusting the proposed sensor's resolution parameter therefore reduces the bulk of its power linearly with the pixel count.

**ISP:** While total power consumption numbers are available for commercial and research ISP designs, we are unaware of a published breakdown of power consumed per stage. To approximate the relative cost for each stage, we measured software implementations of each using OpenCV 2.4.8 [26] and profile them when processing a 4288×2848 image on an Intel Ivy Bridge i7-3770K CPU. We report the number of dynamic instructions executed, the CPU cycle count, the number of floating-point operations, and the L1 data cache references in a table in our supplementary material.

While this software implementation does not directly reflect hardware costs in a real ISP, we can draw general conclusions about relative costs. The denoising stage is by far the most expensive, requiring more than two orders of magnitude more dynamic instructions. Denoising—here, non-local means [4]—involves irregular and non-local references to surrounding pixels. JPEG compression is also expensive; it uses a costly discrete cosine transform for each macroblock.

Section 4 finds that most stages of the ISP are unnecessary in vision mode, and Section 5 demonstrates how two remaining stages—gamma compression and demosaicing—can be approximated using in-sensor techniques. The JPEG compression stage is also unnecessary in computer vision mode: because images do not need to be stored, they do not need to be compressed for efficiency. Therefore, the pipeline can fully bypass the ISP when in vision mode. Power-gating the integrated circuit would save all of the energy needed to run it.

**Total Power Savings:** The two components of an imaging pipeline, the sensor and the ISP, have comparable total power costs. For sensors, typical power costs range from 137.1 mW for a security camera to 338.6 mW for a mobile device camera [31]. Industry ISPs can range from 130 mW to 185 mW when processing 1.2 MP at 45 fps [36], while Hegarty et al. [22] simulated an automatically synthesized ISP which consumes 250 mW when processing 1080p video at 60 fps. This power consumption is comparable to recent CNN ASICs such as TrueNorth at 204 mW [17] and EIE at 590 mW [19].

In vision mode, the proposed image sensor uses half as much energy as a traditional sensor by switching to a 5-bit logarithmic ADC. The ISP can be disabled entirely. Because the two components contribute roughly equal parts to the pipeline's power, the entire vision mode saves around 75% of a traditional pipeline's energy. If resolution can be reduced, energy savings can be higher.

This first-order energy analysis does not include overheads for power gating, additional muxing, or off-chip communication. We plan to measure complete implementations in future work.

## 8. Discussion

We advocate for adding a *vision mode* to the imaging pipelines in mobile devices. We show that design choices in the sensor's circuitry can obviate the need for an ISP when supplying a computer vision algorithm.

This paper uses an empirical approach to validate our design for a vision-mode imaging pipeline. This limits our conclusions to pertain to specific algorithms and specific datasets. Follow-on work should take a theoretical approach to model the statistical effect of each ISP stage. Future work should also complete a detailed hardware design for the proposed sensor modifications. This paper uses a first-order energy evaluation that does not quantify overheads; a full design would contend with the area costs of additional components and the need to preserve pixel pitch in the column architecture. Finally, the proposed vision mode consists of conservative changes to a traditional camera design and no changes to vision algorithms themselves. This basic framework suggests future work on deeper co-design between camera systems and computer vision algorithms. By modifying the abstraction boundary between hardware and software, co-designed systems can make sophisticated vision feasible in energy-limited mobile devices.

## 9. Acknowledgements

Many thanks to Alyosha Molnar and Christopher Batten for their feedback and to Taehoon Lee and Omar Abdelaziz for hacking. This work was supported by a 2016 Google Faculty Research Award. Suren Jayasuriya was supported by an NSF Graduate Research Fellowship and a Qualcomm Innovation Fellowship.

## References


[1] B. Amos, B. Ludwiczuk, and M. Satyanarayanan. OpenFace: A general-purpose face recognition library with mobile applications. Technical Report CMU-CS-16-118, CMU School of Computer Science, 2016.

[2] Apple Inc. AVCapturePhotoOutput. `https://developer.apple.com/reference/avfoundation/avcapturephotooutput`.

[3] D. Borer and T. Rösgen. Large-scale particle tracking with dynamic vision sensors. In *International Symposium on Flow Visualization (ISFV)*, 2014.

[4] A. Buades, B. Coll, and J. M. Morel. A review of image denoising algorithms, with a new one. *Multiscale Model Simulation*, 4(2):490–530, 2005.

[5] Centeye, Inc. Logarithmic pixels. `http://www.centeye.com/technology/47-2/`.

[6] Y. Chae, J. Cheon, S. Lim, M. Kwon, K. Yoo, W. Jung, D. H. Lee, S. Ham, and G. Han. A 2.1 M pixels, 120 frame/s CMOS image sensor with column-parallel $\delta\sigma$ ADC architecture. *IEEE Journal of Solid-State Circuits*, 46(1):236–247, Jan. 2011.

[7] A. Chakrabarti. Learning sensor multiplexing design through backpropagation. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[8] H. G. Chen, S. Jayasuriya, J. Yang, J. Stephen, S. Sivaramakrishnan, A. Veeraraghavan, and A. C. Molnar. ASP vision: Optically computing the first layer of convolutional neural networks using angle sensitive pixels. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[9] X. Chen, Y. Chen, Z. Ma, and F. C. A. Fernandes. How is energy consumed in smartphone display applications? In *HotMobile*, 2013.

[10] Y.-H. Chen, T. Krishna, J. Emer, and V. Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. In *IEEE International Solid-State Circuits Conference (ISSCC)*, 2015.

[11] M. A. Clapp, V. Gruev, and R. Etienne-Cummings. *Focal-Plane Analog Image Processing*, pages 141–202. Springer US, Boston, MA, 2004.

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[13] S. Diamond, V. Sitzmann, S. Boyd, G. Wetzstein, and F. Heide. Dirty pixels: Optimizing image classification architectures for raw sensor data, 2017. `https://arxiv.org/abs/1701.06487`.

[14] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, Apr. 2006.

[15] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, and O. Temam. ShiDianNao: Shifting vision processing closer to the sensor. In *International Symposium on Computer Architecture (ISCA)*, 2015.

[16] P. E. J. Duhamel, C. O. Perez-Arancibia, G. L. Barrows, and R. J. Wood. Biologically inspired optical-flow sensing for altitude control of flapping-wing microrobots. *IEEE/ASME Transactions on Mechatronics*, 18(2):556–568, Apr. 2013.

[17] S. K. Esser, P. A. Merolla, J. V. Arthur, A. S. Cassidy, R. Appuswamy, A. Andreopoulos, D. J. Berg, J. L. McKinstry, T. Melano, D. R. Barch, C. di Nolfo, P. Datta, A. Amir, B. Taba, M. D. Flickner, and D. S. Modha. Convolutional networks for fast, energy-efficient neuromorphic computing. *Proceedings of the National Academy of Sciences*, 113(41):11441–11446, 2016.

[18] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes challenge 2007 (VOC2007) results.

[19] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally. EIE: Efficient inference engine on compressed deep neural network. In *International Symposium on Computer Architecture (ISCA)*, 2016.

[20] S. Han, R. Nandakumar, M. Philipose, A. Krishnamurthy, and D. Wetherall. GlimpseData: Towards continuous vision-based personal analytics. In *Workshop on Physical Analytics (WPA)*, 2014.

[21] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015. `https://arxiv.org/abs/1512.03385`.

[22] J. Hegarty, J. Brunhaver, Z. DeVito, J. Ragan-Kelley, N. Cohen, S. Bell, A. Vasilyev, M. Horowitz, and P. Hanrahan. Darkroom: Compiling high-level image processing code into hardware pipelines. In *SIGGRAPH*, 2014.

[23] J. Hegarty, R. Daly, Z. DeVito, J. Ragan-Kelley, M. Horowitz, and P. Hanrahan. Rigel: Flexible multi-rate image processing hardware. In *SIGGRAPH*, 2016.

[24] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, Oct. 2007.

[25] iniLabs Ltd. Dynamic vision sensor. `https://inilabs.com/products/dynamic-vision-sensors/`.

[26] Itseez. OpenCV. `http://opencv.org`.

[27] S. J. Kim, H. T. Lin, Z. Lu, S. Süsstrunk, S. Lin, and M. S. Brown. A new in-camera imaging model

for color computer vision and its application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(12):2289–2302, Dec. 2012.

[28] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[29] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[30] R. LiKamWa, Y. Hou, J. Gao, M. Polansky, and L. Zhong. RedEye: Analog ConvNet image sensor architecture for continuous mobile vision. In *International Symposium on Computer Architecture (ISCA)*, 2016.

[31] R. LiKamWa, B. Priyantha, M. Philipose, L. Zhong, and P. Bahl. Energy characterization and optimization of image sensing toward continuous mobile vision. In *Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2013.

[32] Z. Liu, T. Park, H. S. Park, and N. S. Kim. Ultra-low-power image signal processor for smart camera applications. *Electronics Letters*, 51(22):1778–1780, 2015.

[33] N. Massari, M. Gottardi, L. Gonzo, D. Stoppa, and A. Simoni. A CMOS image sensor with programmable pixel-level analog processing. *IEEE Transactions on Neural Networks*, 16(6):1673–1684, Nov. 2005.

[34] P. Moulon, P. Monasse, R. Marlet, and Others. OpenMVG: An open multiple view geometry library. `https://github.com/openMVG/openMVG`.

[35] A. Omid-Zohoor, C. Young, D. Ta, and B. Murmann. Towards always-on mobile object detection: Energy vs. performance tradeoffs for embedded HOG feature extraction. *IEEE Transactions on Circuits and Systems for Video Technology*, PP(99):1–1, 2017.

[36] ON Semiconductor. Image processors. Commercial Website, `http://www.onsemi.com/PowerSolutions/parametrics.do?id=16880`.

[37] J. Ragan-Kelley, C. Barnes, A. Adams, S. Paris, F. Durand, and S. Amarasinghe. Halide: A language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines. In *ACM Conference on Programming Language Design and Implementation (PLDI)*, 2013.

[38] B. Reagen, P. N. Whatmough, R. Adolf, S. Rama, H. Lee, S. K. Lee, J. M. Hernández-Lobato, G.-Y. Wei, and D. M. Brooks. Minerva: Enabling low-power, highly-accurate deep neural network accelerators. In *International Symposium on Computer Architecture (ISCA)*, 2016.

[39] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[40] W. A. Richards. Lightness scale from image intensity distributions. *Appl Opt*, 21(14):2569–2582, July 1982.

[41] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nesic, X. Wang, and P. Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *German Conference on Pattern Recognition (GCPR)*, 2014.

[42] C. Strecha, W. von Hansen, L. V. Gool, P. Fua, and U. Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[43] M. L. Uss, B. Vozel, V. V. Lukin, and K. Chehdi. Image informative maps for component-wise estimating parameters of signal-dependent noise. *Journal of Electronic Imaging*, 22(1):013019–013019, 2013.

[44] R. J. Van de Plassche. *CMOS integrated analog-to-digital and digital-to-analog converters*, volume 742. Springer Science & Business Media, 2013.

[45] A. Vasilyev, N. Bhagdikar, A. Pedram, S. Richardson, S. Kvatinsky, and M. Horowitz. Evaluating programmable architectures for imaging and vision applications. In *MICRO*, 2016.

[46] D. Yi, Z. Lei, S. Liao, and S. Z. Li. Learning face representation from scratch, 2014. `http://arxiv.org/abs/1411.7923`.