2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture

RedEye: Analog ConvNet Image Sensor Architecture for Continuous Mobile Vision

Robert LiKamWa, Yunhui Hou, Julian Gao, Mia Polansky, Lin Zhong
Department of Electrical and Computer Engineering
Rice University
{roblkw, houyh, yg18, mia.polansky, lzhong}@rice.edu

***Abstract*—Continuous mobile vision is limited by the inability to efficiently capture image frames and process vision features. This is largely due to the energy burden of analog readout circuitry, data traffic, and intensive computation. To promote efficiency, we shift early vision processing into the analog domain. This results in RedEye, an analog convolutional image sensor that performs layers of a convolutional neural network in the analog domain before quantization. We design RedEye to mitigate analog design complexity, using a modular column-parallel design to promote physical design reuse and algorithmic cyclic reuse. RedEye uses programmable mechanisms to admit noise for tunable energy reduction. Compared to conventional systems, RedEye reports an 85% reduction in sensor energy, 73% reduction in cloudlet-based system energy, and a 45% reduction in computation-based system energy.**

***Keywords*-continuous mobile vision; programmable analog computing; computer vision; pre-quantization processing;**

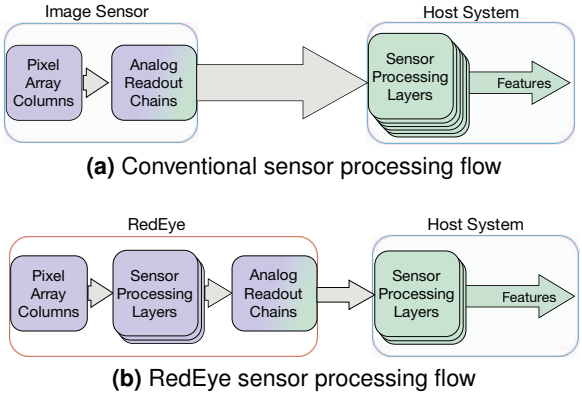

Figure 1: Conventional sensor processing incurs significant workload on the analog readout (top), whereas early processing alleviates the quantization in the analog readout (bottom).

I. Introduction

The recent emergence of wearable devices have led many to entertain the concept of "showing computers what you see" [1], [2], or *continuous mobile vision*. While devices are increasingly capable of interpreting visual data, they face a daunting barrier: energy efficiency. For example, continuous vision tasks drain the battery of Google Glass in 40 minutes [3]. While process technology and system-level optimization may continue to improve the efficiency of digital circuits, recent measurements have pointed to a fundamental bottleneck to energy efficiency: the image sensor, especially its analog readout circuitry [4].

The analog readout bottleneck, illustrated in Figure 1a, is fundamental for two reasons. First, compared to digital circuits, the energy efficiency of analog readout improves much slower over technological generations. Namely, smaller transistors and lower supply voltage do not automatically make analog circuits more efficient. Second, while the trend in the solid-state circuit community is to move functionality from the analog domain to the digital, the analog readout will necessarily exist as a bridge from the analog physical world to the digital realm. That is, conventional systems force imaging data to traverse a costly analog-to-digital converter to be usable by digital systems. Only then can the system apply algorithms for vision, such as processing convolutional neural network (ConvNet) features for image classification. We elaborate upon the bottleneck in §II-A.

Toward addressing this bottleneck, our key idea is to push early processing into the analog domain to reduce the workload of the analog readout, as illustrated in Figure 1b. This results in our design of *RedEye*, an analog convolutional image sensor for continuous mobile vision. RedEye discards raw data, exporting features generated by processing a ConvNet in the analog domain. This saves energy not only in the sensor's analog readout but also throughout the system.

In designing RedEye, we address the following important challenges. (*i*) Analog circuits suffer from high design complexity, discouraging portability between process technology nodes. RedEye constrains the complexity with a modular design that facilitates physical reuse within a circuit design and algorithmic reuse through a cyclic dataflow. (*ii*) Analog processing is subject to accumulating noise, impairing task accuracy. We identify key noise-energy tradeoffs to sacrifice energy to temper noise during processing. RedEye provides mechanisms to tune noise parameters for efficiency and accuracy at run-time. (*iii*) Developers using RedEye require an estimation of a ConvNet's task accuracy and energy consumption under the influence of noise. We provide a simulation framework to help developers tune the balance of

1063-6897/16 $31.00 © 2016 IEEE
DOI 10.1109/ISCA.2016.31

accuracy and efficiency when running ConvNets on RedEye. §III presents the system design of RedEye.

We present the circuit design of RedEye in 0.18-µm process, and model circuit noise, power and timing characteristics with Cadence Spectre. We use these characteristics in the RedEye simulation to precisely estimate circuit behavior. §IV provides details of RedEye's circuit design.

We evaluate RedEye on early layers of the GoogLeNet ConvNet, the image classification champion of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2014. We use a simulation-based framework to estimate task accuracy and energy consumption. We find that RedEye can accurately process early layers of ConvNets, while reducing the energy consumption of an image sensor from 1.1 mJ to 0.2 mJ per frame. By processing convolutions in the analog domain, RedEye also reduces the computational workload of its host device, reducing the energy consumed by a Jetson TK1 from 406 mJ to 226 mJ per frame. RedEye positions a convergence of analog circuit design, systems architecture, and machine learning, which allows it to perform early operations in the image sensor's analog domain, moving toward continuous mobile vision at ultra low power.

While we demonstrate the tremendous efficiency benefit of RedEye, we note that this comes at the expense of analog circuit development effort and production cost. We do not provide a RedEye layout design, nor have we investigated the economics of the RedEye premise. Instead, we hope the efficiency benefit provides strong incentive that invites more research into this approach and analog/mixed-signal circuit experts to optimize the analog design cost.

We note the efficiency bottleneck in the analog readout is universal to all sensors, not just image sensors. Because the energy consumed by digital processing of sensor data is usually proportional to data size, the analog readout consumes a large proportion of overall system energy, even for low-rate sensors such as accelerometers. We project that the ideas of RedEye, especially those related to shifting processing into the analog domain for efficiency and optimizing noise configuration, will be effective in improving system efficiency for a wide range of continuous sensing modalities.

II. Background

A. Image sensing

Modern mobile devices employ CMOS image sensors for photography and videography. While more efficient than CCD, they still consume hundreds of milliwatts, due to be their power-hungry analog readout [4].

Modern image sensors consist of three major components: pixel array, analog readout, and digital logic. The pixel array converts light into electrons and then voltages, consuming little energy [5]. Analog readout amplifies analog signals from the pixel array and converts them into digital values. Digital logic maintains sensor operation, managing timing generation, frame scanner and drivers, and data interface.

Modern sensors employ a column-based architecture for analog readout; pixels in each column share dedicated circuitry. Column amplifiers sample and amplify raw pixels to harness available signal range and overcome noise; they consume static power to bias the transistors operating linearly. Column ADCs digitize amplified signals, consuming static and dynamic power. The analog readout consumes 50%-75% of overall energy in recent sensor designs [5]–[7]. This is because (*i*) the entire image signal must undergo analog readout and *(ii)* analog readout is designed for high-fidelity.

We emphasize that digital optimization, e.g., low-power hardware for data reduction [8], early discard, or computational offloading, do not affect energy consumed by the analog readout. This is the fundamental motivation behind our proposition to move processing before analog readout.

B. Analog computing efficiency

Analog computing is more efficient than digital computing; for basic arithmetic, such as addition and multiplication, the energy per operation of analog implementations is orders of magnitude lower [9]. Unlike digital computing, which uses multiple binary charges (bits) to represent circuit state, analog computing uses a single charge to represent a value. This reduces hardware count; analog addition only needs an interconnect to join two currents, whereas a 16-bit digital adder requires over 400 transistors.

More importantly, profitable tradeoffs between energy efficiency and signal fidelity can be made in the analog domain, relevant for noise-robust vision tasks. That is, *energy consumption of analog circuits can be reduced by allowing a higher noise level* [10], [11]. This is because maintaining state through capacitance C is susceptible to thermal noise $\overline{V_n^2} = kT/C$ [12]. While raising C deters noise, the energy required to fill the memory cell rises proportionally. Hence,

$$E \propto C \propto 1/\overline{V_n^2}$$

Therefore, the choice of C is a fundamental tradeoff between fidelity and efficiency of analog computing. Here we identify inherent energy-noise tradeoffs in analog computing.

Memory: As an analog pipeline must be constructed in stages to facilitate configurability and maintainability, analog memory is indispensable for inter-stage buffers. Memory cells use capacitors to maintain states, and thus exhibit energy-noise tradeoffs upon reading and writing values.

Arithmetic: Arithmetic in charge-based circuits, such as in RedEye, typically requires an operational amplifier (op amp), which induces thermal noise $\propto 1/C$ and energy $\propto C$.

In current-based arithmetic circuits, signal currents are shadowed by thermal noise currents, which cannot be indefinitely lowered due to process and design constraints. To improve signal fidelity, the effective approach is to elevate signal current itself, which increases power draw.

Readout: To traverse the analog/digital boundary consumes significant energy. We choose Successive Approximation Register (SAR) as the on-chip analog-to-digital converter (ADC), not only because of its unparalleled efficiency in low/mid-resolution applications [13]–[17], but also for its ability to trade accuracy for energy. SAR uses an array of capacitors to approximate an analog signal into digital bits.

These capacitors are the source of systematic and random inaccuracies. Systematic inaccuracy is determined by capacitor mismatch in the array, which impacts the approximate signal and thus the correctness of the output. Using a larger unit capacitor C_0 improves matching but consumes more energy, creating a tradeoff between efficiency and linearity.

Random errors in SAR are dominated by quantization noise. With ADC resolution n increased by each bit, quantization noise amplitude diminishes in half. However, this doubles energy consumption, as the total size of the capacitor array ($C_\Sigma = 2^n C_0$) increases exponentially with n. This is yet another energy-noise tradeoff that we elect to utilize.

Analog design complexity: While basic operations consist of few components, the complexity of large-scale analog systems can be prohibitively high. Routing interconnects is particularly daunting; unlike digital transactions on shared buses, analog data exchanges via pre-defined routes. Designers must preplan routes to avoid congestion and overlap.

In addition, without automatic verification, an analog designer must formulate and manually verify circuit specifications. Moreover, a high analog hardware count slows the simulation process and aggravates the verification cost.

Design complexity limits portability between fabrication processes, as specialized units requires redesign for performance standards. Complexity also limits generalizability; complex hardware is difficult to adapt and scale. More importantly, complexity constrains depth of operation; chaining multiple operations can become complex for verification.

C. ConvNets

We design RedEye to compute continuous mobile vision tasks using Convolutional Neural Networks, also called ConvNets, for their efficacy in state-of-the-art object detection, handwriting recognition, and face detection [18]. This prominence has resulted in active academic and industrial research on implementing ConvNets for vision tasks.

ConvNets operate by feeding 3-dimension data through sequential layers of processing to generate vision features. Each layer consists of a set of neurons; each neuron takes in multiple input values from a local receptive field of neurons from the previous layer. The neuron operates on these values to generate a single output. Layers of neurons come in a small variety of types. Here we cover the key types of layers:

Convolutional layer neurons multiply a three dimensional receptive field of inputs with a kernel of weights to generate an output. Neurons in a convolution layer share kernel weights to perform their processing. Multiple sets of weights per layer generate depth channels of data. A backpropagation process trains weights to alter the convolution operation.

Nonlinearity layer neurons use activation functions to introduce nonlinear saturation to the outputs of convolutional layers. Sigmoid, tanh, and rectification functions have all been used to create non-linearity.

Max pooling layer neurons receive features generated from a receptive field of processed output, and identify the feature with maximum response. This allows the network to be robust to minute changes in the feature space.

These layers, in addition to other layers, such as normalization layers to constrain the signal swing, can be stacked in various sequential order, invoking each layer multiple times to create deep structures. For example, the AlexNet [19] image recognition ConvNet uses 6 convolutional layers, 7 non-linearity layers, 3 pooling layers, 2 normalization layers. AlexNet also includes 7 "other" layers late in its operation.

Because trained ConvNets generalize across image inputs, they are naturally robust to noise [20]. We study and exploit robustness to leverage energy-noise tradeoffs in RedEye.

III. RedEye System Architecture

We design *RedEye*, an analog computational image sensor architecture that captures an image and passes it through a deep convolutional network (ConvNet) while in the analog domain, exporting digital vision features. This substantially reduces the workload of the analog-to-digital quantization and of the post-sensor digital host system, enabling efficient continuous mobile vision. We design RedEye, shown in Figure 2, to constrain design complexity, area, and energy consumption, while accurately executing vision tasks.

A. Overview

Our key insight toward addressing the analog readout bottleneck is that acceptable computer vision does not need high-fidelity images or high-fidelity computation. Leveraging this, we move early-stage ConvNet vision processing from the digital host system into the sensor's analog domain. This shift substantially reduces the workload of the analog readout and therefore reduces system energy consumption.

As analog circuits suffer from design complexity and accumulating signal noise, which limit the potential depth of operation in the analog domain, RedEye exploits the structure of ConvNet to effectively contain complexity and noise. The architecture avoids redundant design by sending data through *layers of processing modules*, cyclically reusing modules before analog readout, as shown in Figure 3. As modules process locally on patches of pixels or previous data, we can further reduce the complexity of the interconnects by structuring the operation layers into a *column-based topology*. This arranges the modules in each processing layer in a column pipeline, as illustrated in Figure 2. This grants each processing module a physical proximity to its input data. §III-B elaborates upon the RedEye hardware design.

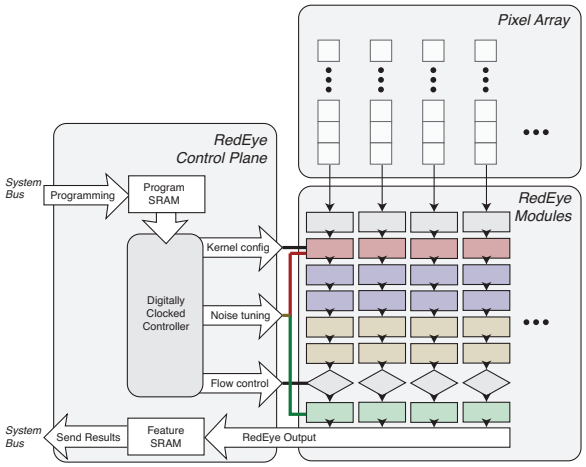

Figure 2: RedEye. The controller programs kernel weights, noise mechanisms, and flow control into RedEye modules. Output is stored in Feature SRAM for retrieval.

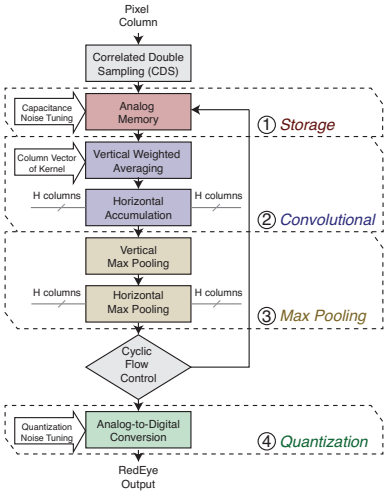

Figure 3: Convolutional column of the RedEye modules. A column of pixels is processed in a cyclic pipeline of convolutional and max pooling operations.

RedEye provides a *ConvNet programming interface* that allows developers to load ConvNet programs into the RedEye program SRAM. To allow for changing task requirements, we also provide programmable *noise admission mechanisms* that can be dynamically tuned to save energy by allowing noisy operation. §III-C elaborates upon this.

To assist developers in designing RedEye programs, we provide a *simulation framework* that estimates the task performance and energy consumption of running a ConvNet on the RedEye in its noisy analog environment. This allows developers to optimize their RedEye programs and noise parameters for minimal energy consumption at sufficient task accuracy. §III-D elaborates upon the simulation framework.

B. RedEye hardware architecture

While it is desirable to operate in the analog domain for efficiency, circuit complexity limits depth of operation. We focus on decisions that execute additional processing before quantization while limiting design complexity.

1) Cyclic module reuse for deep execution: RedEye features a novel design of cyclically-operated RedEye modules for deep ConvNet execution. The design exploits the stackable structure of ConvNet layers to reuse RedEye modules for ConvNet processing in the analog domain.

Of the four types of RedEye modules, enumerated in Figure 3, the *convolutional* and *max pooling modules* perform the operations of neurons in a ConvNet layer. Meanwhile, the *storage module* interfaces the captured or processed signal data with the processing flow. Finally, the *quantization module* exports the digital representation of the RedEye ConvNet result. A flow control mechanism routes dataflow through RedEye modules, skipping modules as necessary.

We next describe the design of each RedEye module, each itself designed for reusability and programmability.

① **Buffer module**: RedEye samples filtered color pixels and stores them to analog memory cells, prepared for the first computational layer. The buffer module also receives and stores intermediate results for further execution.

② **Convolutional module**: The 3-D convolutional module reads samples from the buffer module, performs multiplied accumulation along vertical y and channel (e.g., color) z axes with a weight kernel, and accumulates results horizontally to columns on the x axis. The module clips signals at maximum swing to perform nonlinear rectification.

③ **Max pooling module**: The max pooling module identifies and propagates the maximum value in a local vicinity. When local response normalization is required, the convolutional module uses this sample to adjust convolutional weights for the subsequent execution, equivalently realizing the functionality of normalization.

④ **Quantization module**: Used after the RedEye processing is complete, the quantization module converts the output features into a digital format and sends them into the digital host system for further processing.

By reusing modules through multiple cycles of ConvNet layer processing, RedEye can functionally perform deep ConvNet processing with its limited complexity design.

2) Flow Control: RedEye controls the signal flow using a synchronous digitally-clocked controller to simultaneously pipe signal flow through multiple modules, skipping modules as necessary, as shown in the control plane of Figure 2.

The controller uses a set of flow control mechanisms to route the signal for ConvNet processing. If the RedEye needs to execute additional processing, the *cyclic flow control* routes the output of the pooling module to the storage module for intermediate storage. If any layer is unneeded

in a ConvNet dataflow, the *bypass flow control* of each module (not pictured) provides an alternate signal route to circumvent the corresponding module. For example, if pooling is not required, the module can be skipped entirely. This procedural design enables RedEye to operate multiple layers of a ConvNet dataflow in the analog environment.

3) Column-based topology for data locality: To promote data locality for kernel window operations, we organize RedEye modules into a column-parallel design, as inspired by conventional image sensors with column-wise readout structure. As shown in Figure 2, this mitigates the complexity of unbounded interconnect structures.

As introduced in §II-A, image sensors use column structures to operate multiple ADCs in parallel, reading one image row at each timestep. Similarly, by adopting a column-based topology, we advance the processing window by one row at a time, controlled by a clocked timestep, allowing multiple modules to simultaneously operate in parallel.

However, more importantly, because multiple columns are synchronously processed, the input window of data surrounding an element will be synchronously prepared for execution. For vertical execution, data can be buffered as it is generated; vertical access can be pipelined across different timesteps. For horizontal access, RedEye needs only bridge interconnects across horizontally-proximate columns. Thus, the column-based topology allows for efficient data access for kernel processing in horizontal and vertical directions.

We design RedEye with columns of modules as homogeneous subcomponents. This design pattern allows RedEye to hierarchically reuse hardware design, thus enabling scalability to focal-plane resolution, and promoting portability and programmability.

C. ConvNet Programming Interface

A developer utilizes RedEye by writing a ConvNet program to the RedEye program SRAM of the control plane, and reading the results from the RedEye feature SRAM. The ConvNet program includes the layer ordering, layer dimensions, and convolutional kernel weights. RedEye uses the digitally-clocked controller to load the program from the SRAM into the cyclic signal flow, issuing the specified kernel weights and noise parameters.

Thus, the developer is responsible for partitioning ConvNets between RedEye operation and digital host system operation. The decision of the cut influences the energy consumption of the overall system. While a deeper cut reduces the workload of the analog readout and of the host system, it places more operation burden on the RedEye.

Programmable noise-admission: To allow developers to tune the RedEye processing for energy efficiency, we provide noise-admission mechanisms to trade signal fidelity for energy efficiency. To control the mechanisms, developers can specify the signal-to-noise ratio (SNR) for each layer.

Architecturally, RedEye uses the mechanisms to vary the capacitance of a damping circuit in the operation modules. This can be configured at runtime for each convolutional module. By using variable noise-damping, detailed in §IV-A, RedEye admits Gaussian noise in ConvNet module operations for energy savings. Similarly, RedEye uses a dynamic quantization mechanism to adjust the ADC resolution of the analog readout, thus scaling quantization noise.

As an alternative to noise damping, RedEye could use a boosted analog supply voltage to increase signal swing, and adjust signal gain accordingly to achieve higher SNR. This approach is theoretically more efficient than noise damping; however, in practice, this technique is sensitive to power supply variations. As foundries generally do not guarantee the transistor model to remain accurate when transistors operate outside recommended voltage regions, it is a risk that the actual circuit behavior may deviate from simulation.

Thus, we use capacitance-based noise-damping to allow the developer can also make noise parameter decisions, specifying the SNR of each convolutional module and quantization module. A developer can load these noise parameters into the RedEye SRAM alongside the ConvNet definition.

D. RedEye ConvNet simulation framework

Paramount to a developer's ConvNet programming decisions is a prediction of the accuracy and energy efficiency of running a given ConvNet on RedEye. We provide a simulation framework, taking in a developer's partitioned ConvNet and specified noise parameters and generating task accuracy and energy estimations. The ConvNet structure of the framework maps to a sequential execution of RedEye modules. Here, we describe the construction of our framework, using task simulation and circuit modeling.

Simulating noisy RedEye operation: Our simulation modifies a ConvNet framework to analyze the effect of noise admission on task accuracy and energy consumption. Starting with ConvNet models designed for execution on digital processors, we inject two types of noise layers into the processing flow, representative of Gaussian and quantization noise that invades the RedEye signal flow.

The *Gaussian Noise Layer* models noise inflicted by data transactions and computational operations. We insert a Gaussian Noise Layer to the output of each sampling layer, convolutional layer and normalization layer. RedEye uses the SNR to parametrize each Gaussian noise layer, allowing the developer to tune the noise in the simulation.

The *Quantization Noise Layer* represents error introduced at the circuit output by truncating to finite ADC resolution. We model quantization noise as uniformly distributed noise across the signal. We insert the quantization noise layer where RedEye outputs the signal's digital representation, and use the ADC resolution (q) as a tunable input parameter.

Then, by using the framework to run input data through the modified network, and applying an evaluation metric,

such as Top-N accuracy, we can simulate the effect of noise admission on the ConvNet accuracy.

Estimating RedEye operational energy: We use circuit models of RedEye modules to determine the energy consumption incurred with parameters of Gaussian noise layers $E_{Gauss.}(SNR)$ and quantization noise layers $E_{quant.}(q)$, estimating the energy consumed by the analog operations.

As introduced in §II-B, decreased noise admission increases the task accuracy, but also increases operational energy consumption. Developers should search for an optimal set of parameters that achieves task accuracy at minimal cost. In general, this is an intensive search over a parameter space of dimension $\Re^{n+1}$ for n Gaussian layers and 1 quantization layer. Such highly dimensional searches would typically require tools such as the canonical simplex search [21]. However, for GoogLeNet processing, our evaluation reveals that we can accept as much Gaussian noise as each analog operation can admit (SNR $>$ 40 dB). The problem, then, reduces to a single parameter selection, selecting an energy-optimal quantization q to balance accuracy and energy.

Framework implementation: To model convolutional vision tasks, we design our functional algorithmic simulation by adapting the Caffe neural network framework, written in Python and C++. Slight modifications to the Caffe source code allow us to add layers to simulate and optimize analog noise tradeoffs. We implement Gaussian and uniform noise layers by adding 191 lines of C++ code to the Caffe framework. Our Python framework consists of 2000 lines of code to set up ConvNets with injected noise layers. We use Cadence Spectre to generate noise-energy models of the operational layers of the ConvNet, based on the circuits in §IV. This provides a model of energy consumption under a given set of noise parameters.

IV. RedEye Circuit Design

We next present the RedEye circuit design. Our circuit not only validates the efficacy of our modular column-parallel design, but also structurally defines a behavioral circuit model to study signal fidelity and energy consumption. This model guides the simulation framework in §III-D.

A. Analog module design

We design transistor-level circuitry of analog RedEye modules in IBM 0.18-µm CMOS process. Given the efficiency and process constraints, we assume an SNR range of 40–60 dB, proven to be sufficient for our evaluation results.

Convolution: RedEye convolutional layers perform 3-D convolutions in two steps, as shown in Figure 3: *weighted averaging* on y-axis, and *accumulation* on x and z. We design a multiply-accumulate unit (MAC, Figure 4). As weights must be stored in digital form for long-term fidelity, they must cross the A/D boundary to interact with analog signals. In order to facilitate mixed-signal multiplication, we derive our design from switched-capacitor amplifiers.

For efficient digital-to-analog conversion, tunable capacitors are crucial for accuracy and efficiency in applying kernel weights to signal values. Through experimental validation, we find that our ConvNet tasks can use 8-bit fixed-point weights with accurate operation. To efficiently implement such weights, we design a charge-sharing based tunable capacitor, shown in Figure 5, to maximize energy savings.

The 8-bit tunable capacitor operates as follows:

1) Sample input signal onto C_j for which $b_j = 1$ ($8 \geq j \geq 1$), where b_j is the j-th bit of the weight. Ground other C_j capacitors;
2) Activate ϕ_D switches to share signal charge on each C_j with additional $(2^{8-j} - 1)$ C_0 capacitors;
3) Combine charge of all C_j to obtain weighted signal.

This design reduces the number of MAC sampling capacitors from $O(2^n)$ (the naïve design) to $O(n)$. This proportionally reduces input capacitance and energy consumption, as C_0 itself cannot shrink further due to process constraints. For the 8-bit MAC, this reduces energy by a factor of 32.

We enhance the versatility of RedEye by designing a dynamic noise damping mechanism, so that the fidelity of the convolutional layer can adapt to different tasks. At the output of MAC, a tunable capacitor adjusts the total load capacitance to trade thermal noise for energy efficiency.

Max pooling: The max pooling layer uses a straightforward algorithm to find the maximum local response max:

1) Convert all responses to absolute values;
2) Serially compare each absolute response with max, and preserve the larger value by assigning it to max;
3) Repeat (2) to traverse the entire localized window.

We design a fully dynamic comparator to eliminate static steering current, achieving zero idle power consumption. Dynamic comparators typically suffer from energy wastage when metastability occurs; the comparator requires long decision times and maximum current when the difference in inputs is small. We suppress this effect by forcing arbitrary decisions when the comparator fails to deliver a result in time. This mechanism works without introducing tangible error in pooling results, as confirmed by task simulation.

Quantization: As introduced in §II-B, variable ADC resolution trades bitdepth for energy efficiency. Our SAR ADC design achieves variable resolution by skipping bit cycles and cutting off corresponding capacitors. This mechanism is explained as follows. Significance of bit b_i is defined by the weight of the corresponding capacitor C_i with respect to the total active capacitance of the n-bit array:

$$w\{b_i\} = \frac{C_i}{C_\Sigma} = \frac{2^{i-1}C_0}{[\sum_{k=1}^{n} C_k] + C_0} = \frac{1}{2^{n-i+1}}$$

In our 10-bit SAR ADC design, when all bits are active, the weight of the most significant bit (MSB) b_{10} is $1/2$; the next bit b_9 has a weight of $1/4$. When ADC resolution

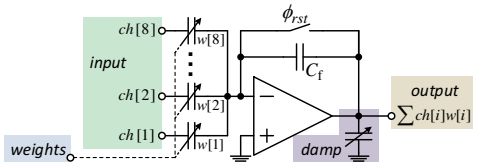

Figure 4: 8-input mixed-signal MAC (conceptual). Tunable capacitors apply digital weights $w[:]$ on analog inputs $ch[:]$. ϕ_{rst} clears C_f after each kernel window is processed.

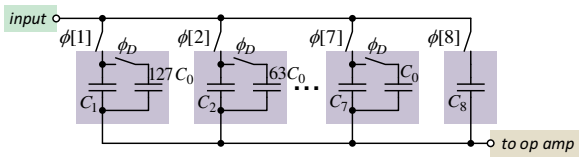

Figure 5: Tunable capacitor design. Digital weights $w[:]$ control the input-side switches $\phi[:]$ while sampling. C_1 to C_8 are identically sized to C_0.

decreases by 1, C_{10} is cut away from the capacitor array, halving the value of C_Σ; Meanwhile, b_9 becomes MSB, with its weight automatically promoted to $1/2$. As a result, this mechanism conserves signal range, and allows for straightforward bit-depth alignment with digital zero padding.

B. Behavioral circuit model

To assist the framework of §III-D by promoting simulation efficiency and preserving simulation accuracy, we derive a parameterized behavioral model from RedEye's circuit simulation. We extract model parameters of key building blocks from transistor-level simulation using Cadence Spectre.

While a full transistor-level mixed-signal simulation would provide robust precision, running signal flows through such a process would be prohibitively time-consuming. Hence, we use behavioral simulation to model fidelity characteristics and energy proportionality of the signal flow.

Aside from verification purposes, we use this behavioral model to describe the energy consumption as a function of noise in each circuit stage and network layer. These energy-noise functions guide the energy estimation function of the simulation framework of §III-D.

Structurally-guided circuit model: We use structural circuit models to design our behavioral model. Top-level elements of convolution, max pooling and quantization layers are composed of sub-layer circuit modules, such as analog memory, MAC and ADC. These modules are further dissected into internal units such as tunable capacitors, op amps and comparators. Each internal unit is simulated at transistor level so that circuit behavior, most importantly energy consumption and noise performance, can be precisely modeled. The energy and noise characteristics propagate upwards to assess the system-wide energy and noise statistics, allowing for fine-grained optimization in circuit design. Furthermore, as the model fully reproduces the signal flow with identical organization and operation of functional circuit elements, it confirms system functionality.

Model parameters: In the behavioral model, the energy consumption and noise contribution of each module are determined by a set of *noise*, *power* and *timing* parameters.

Noise parameters represent the noise performance of the circuit, including sampling noise from switches, op amp noise in signal buffers and amplifiers, comparator noise, and quantization noise. For sampling noise, as the thermal noise power density of transistor switches ($4\gamma kTR$) deviates from that of an ideal insulator by a factor of γ, we simulate the actual noise in a sample-and-hold circuit and use it as a noise parameter. For op amp, we measure the output noise power and refer it to the input, so that the noise parameter remains valid with variable gain settings (i.e., weight values). For ADC with n-bit resolution, we assume its noise contribution is identical to the quantization noise of an ideal m-bit ADC, where m equals the simulated $ENOB$ of the n-bit ADC.

Power parameters account for both static and dynamic power consumption. Static power includes bias current of op amps and leakage in digital circuits. Dynamic power is introduced by comparators when they make decisions or reset to initial state, capacitors when they are being charged, and digital elements when their state flips (e.g., programmable weight distribution). Both noise and power parameters are extracted from circuit simulation. For components with highly dynamic, input-dependent characteristics, we average the power consumption over a relatively long period (e.g. 200 cycles), using their means as power parameters.

Timing parameters refer to the time slots allocated to circuit operations. In switched-capacitor circuits (e.g. MAC), output signals require a certain amount of time to settle before they can be used by subsequent stages. As components consuming static current ought to be aggressively power-gated, it is essential to choose timing parameters properly. Timing parameters work with power parameters, which define the bandwidth of op amps, to report energy consumption as well as output signal inaccuracy from insufficient settling.

Model verification: Our approach of circuit simulation is similar to that of [9]. We build transistor-level schematics for key building blocks including MAC, comparator and ADC using IBM 0.18-µm process, and run Spectre simulations to verify circuit functionality and performance.

As we use circuit simulation to verify a major portion of parameters of the behavioral model, it is critical to reliably configure simulation settings. For time-domain simulations, we set accuracy defaults (errpreset) to conservative, which implies lower error tolerances, i.e., higher simulation accuracy (relative tolerance is 0.01%; absolute tolerances are 1 µV and 1 pA). For performance-critical components, e.g., op amp and comparator, we simulate over five process corners (TT 27°C, FF -20°C, SS 80°C, FS 27°C and SF 27°C) in order to ensure that variations of circuit characteristics

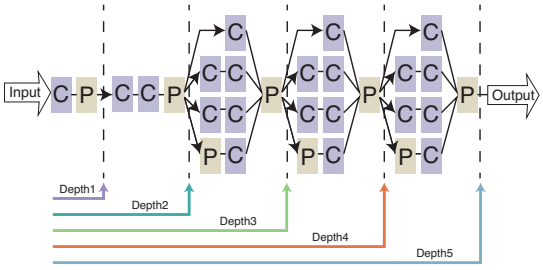

Figure 6: GoogLeNet partitions for evaluation, where C and P are RedEye convolutional and pooling operations.

remain acceptable in all reasonable fabrication scenarios and operating environments. This simulation strategy guarantees the provision for process and temperature variations, which affirms our confidence of the circuit design.

V. Evaluation

We investigate the utility of RedEye in performing analog convolutional processing for continuous mobile vision. We find that RedEye reduces sensing energy consumption by 84.5%. The reduction primarily comes from readout workload reduction. RedEye also assists mobile CPU/GPU systems by replacing the image sensor, nearly halving the system energy consumption by moving convolutional processing from the digital domain in the analog domain.

We also study the influence of noise on RedEye, evaluating our insight of using noise to minimize energy with sufficient task accuracy. Finally, we evaluate RedEye in the context of circuit design structures, studying the effect of our architectural decisions upon analog design complexity.

A. Evaluation methodology

To evaluate RedEye task accuracy and energy consumption, we use the simulation framework introduced in §III-D.

Task – Classification of ImageNet data: For our task, we study image classification, in which an image label is predicted with a degree of confidence. The ImageNet Large Scale Visual Recognition Challenge provides datasets of images, each image labeled with 1 of 1000 classes. These images include a labeled validation set of 50,000 images.

To simulate raw image sampling, we undo gamma correction to simulate raw pixel values. We emulate photodiode noise and other analog sampling effects by applying Poisson noise and fixed pattern noise in the input layer.

To measure classification performance, we run our framework on all validation images. We use the Top-5 metric to quantify task accuracy, counting the proportion of images for which the ground truth label is one of the top 5 predictions.

Evaluated ConvNets: We evaluate RedEye on the GoogLeNet ConvNet [22], with weights digitized to 8-bit values. We also evaluate RedEye on AlexNet [19] with similar findings, but for brevity, only present GoogLeNet results.

We partition GoogLeNet at 5 different depths, as shown in Figure 6, to evaluate deeper RedEye processing. Because GoogLeNet branches to a classifier layer after Depth5, our design is unable to execute further than the first 5 layers.

RedEye processes convolutional and pooling operations before the depth cut, leaving the remainder to be run on the host system. When configured, as in §V-C, we find Depth1 to consume the least RedEye energy per frame. While the readout cost decreases with deeper depth cuts, this is outpaced by processing costs, as shown in Figure 7a, leading to an increase in RedEye energy consumption with deeper operation. However, we find Depth5 to be the energy-optimal configuration when RedEye is combined with a host system, due to RedEye's substantial workload assistance in addition to a quantization energy reduction.

B. RedEye energy and timing performance

We evaluate RedEye against CMOS image sensors and in various system contexts to study its practicality for vision processing in mobile systems. We find that RedEye reduces analog sensor energy consumption by 84.5%, cloudlet-based system energy consumption by 73%, and computational-based system energy consumption by 45%. By pipelining RedEye with computations on the digital system, the Depth5 RedEye can operate at up to 30 frames per second (fps).

RedEye sensor vs. image sensor: Typical systems use an image sensor to capture frames. We find that RedEye performs favorably when comparing energy and timing performance against CMOS image sensors. Both sensors can use a low-power microcontroller for digital interface, consuming 0.4 mJ per frame. For comparison, we ignore the digital footprint, comparing analog performance alone.

To model quantization overhead, we model a 10-bit 227×227 color image sensor, sampling at 30 fps. Using a recent survey [13] to reference state-of-the-art ADC energy consumption, we conservatively estimate the analog portion of the image sensor to consume 1.1 mJ per frame.

By comparison, the processing and quantization of Depth1 on RedEye consumes 170 µJ per frame. This presents an 84.5% sensor energy reduction. RedEye achieves this reduction through noise admission, especially during analog readout; as shown in Figure 7c, 4-bit RedEye operation reduces output data size to nearly half of the image sensor's data size. We chart energy consumption, timing performance, and quantization workload of different depth configurations compared against the modeled image sensor in Figure 7.

RedEye with cloudlet-offload: Systems may offload computations to a nearby server to perform energy-expensive and computationally-intensive operations, i.e., cloudlet processing. While this negates the energy consumption of computations, the device energy consumption is then dominated by network transmission. As RedEye reduces the size of the output data representation, RedEye substantially reduces the energy expended during transmission by 74%.

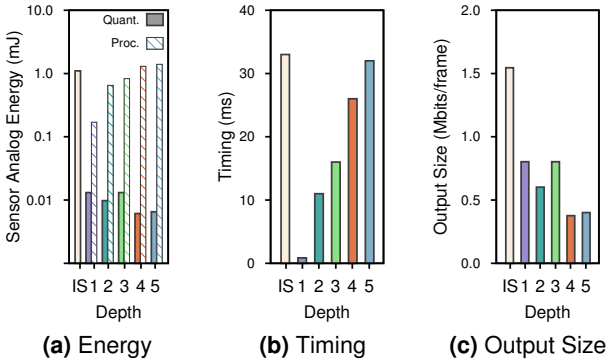

Figure 7: Performance metrics of image sensor (IS) and 4-bit, 40 dB RedEye at different depths. Energy on a log scale.

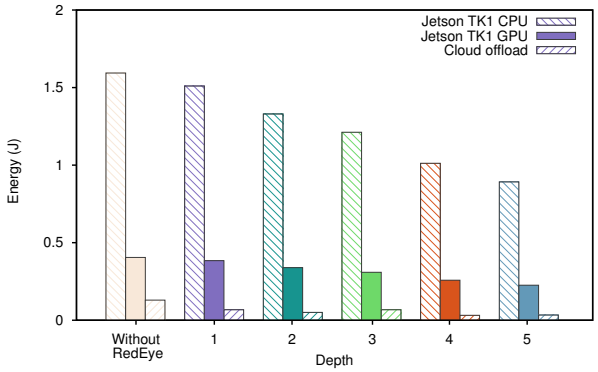

Figure 8: Per-Frame Energy Consumption on NVIDIA Jetson TK1 CPU, Jetson TK1 GPU, and cloud-offload with and without RedEye.

Using a characterization of Bluetooth Low-Energy power and latency [23], we find that conventionally exporting a 227x227 frame will consume 129.42 mJ over 1.54 seconds, in addition to the 1.1 mJ consumed by image sensor capture. Meanwhile, RedEye Depth4 output only consumes 33.7 mJ per frame, over 0.40 seconds. Including a RedEye overhead of 1.3 mJ per frame, RedEye saves 73.2% of system energy consumption for locally-offloaded execution.

RedEye with CPU/GPU execution: RedEye also demonstrates efficiency advantages when paired with a Jetson TK1 digital host system, as illustrated in Figure 8. While the Jetson provides best-in-class mobile ConvNet performance and efficiency, RedEye reduces its energy consumption by reducing its workload. RedEye also increases CPU classification speed, while keeping GPU classification speed.

Executing GoogLeNet with Caffe on the Jetson GPU paired with an image sensor consumes 12.2 W over 33 ms, for 406 mJ per frame, as measured through oscilloscope. Likewise, using the Jetson CPU, the Caffe framework requires 3.1 W over 545 ms, for 1.7 J per frame.

In comparison, using the Depth5 RedEye reduces the Jetson processing time for the GPU to 18.6 ms, and for the CPU to 297 ms. This results in a system energy reduction down to 226 mJ with the GPU, and 892 mJ with the CPU.

As the Depth5 RedEye can be pipelined with CPU and GPU operation, it reduces the processing time to the minimum of the RedEye and the digital operation. RedEye is not the limiting factor in either case, requiring only 32 ms. RedEye accelerates execution for the CPU from 1.83 fps to 3.36 fps and maintains GPU timing, i.e., "real-time" 30 fps.

Thus, paired with the GPU and CPU, using RedEye can save 44.3% and 45.6% of the energy per frame, respectively. System energy consumption is dominated by the energy-expensive GPU/CPU for executing ConvNet layers.

RedEye with hardware acceleration: To compare against specialized digital hardware acceleration, we compare the execution of the RedEye circuit against cited timing and energy statistics of the ShiDianNao neural network accelerator. We find that RedEye computing efficiency reduces the system overhead through analog readout reduction.

For comparison purposes, we consider the 7-layer ConvNets (3 convolution layers), implemented in the ShiDianNao work, and estimate performance on a 227×227 color frame. Specifically, we use 144 instances of the authors' 64×30 patch, with a stride of 16 pixels in the 227×227 region, for 2.18 mJ of energy consumption per frame. Including the image sensor, this consumes over 3.2 mJ per frame.

In comparison, when performing 7 layers of convolutions in a Depth4 configuration, RedEye consumes 1.3 mJ per frame. Thus, system energy consumption is reduced by 59%, due to the reduced readout energy consumption.

C. Noise vs. task accuracy, depth, and energy

We next evaluate the importance of controlling noise to minimize energy. We find we can admit a substantial amount of noise in the convolution operations, which is useful for trading signal fidelity for energy efficiency of quantization.

GoogLeNet task accuracy is robust against the 40–60 dB SNR range expected by RedEye. This is shown in Figure 9, wherein Gaussian noise is introduced to reduce the SNR and accumulated in all data layers, convolutional modules, and pooling modules. Increasing the Gaussian noise through the effective range of potential standard deviations does not effect significant task failure, reporting 89% Top-5 even at the lower SNR limit of 40 dB. Hence, the system should always choose 40 dB for energy efficiency; overprovisioning for low-noise incurs substantial energy consumption relative to the energy consumption corresponding to reduced SNR.

On the other hand, when scanning ADC resolution at a fixed Gaussian noise of 40 dB, we find a sizable accuracy-energy tradeoff in the effective region of quantization scaling, as shown in Figure 10. While using several bits allows robust task accuracy, the accuracy decreases as the RedEye

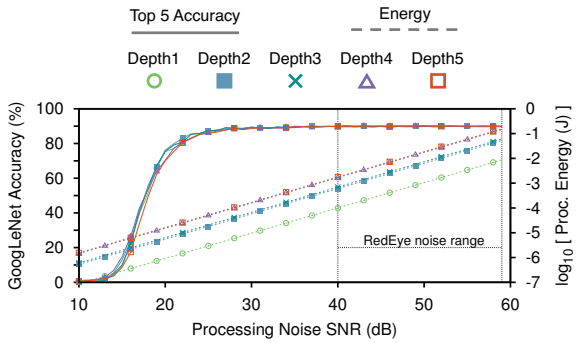

Figure 9: Accuracy (dashed) and Energy of ConvNet processing (solid) vs. Gaussian SNR of GoogLeNet running on RedEye at 4-bit quantization. Accuracy over $N = 2500$.

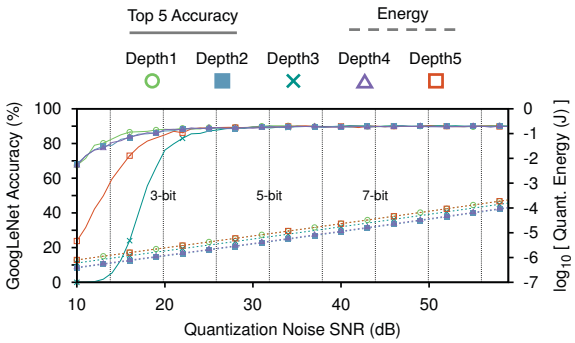

Figure 10: Accuracy (dashed) and Energy of Quantization (solid) vs. quantization SNR of GoogLeNet running on RedEye at Gaussian SNR $= 40$ dB. Accuracy over $N = 2500$.

Table I: RedEye operation modes and energy consumption for Depth5

Modes	SNR	Cap. size	Energy/frame
High-efficiency	40 dB	10 fF	1.4 mJ
Moderate	50 dB	100 fF	14 mJ
High-fidelity	60 dB	1 pF	140 mJ

uses fewer bits. However, from the range of 4–6 bits, all depth configurations operate with similarly high accuracy.

Choosing an optimal depth configuration depends on the energy consumption of the digital host system. For an energy-expensive host system, deeper depth configurations will reduce expensive digital processing; we find that when paired with a Jetson TK1, the most efficient configuration is Depth5. However, for an energy-inexpensive host, RedEye can operate shallower networks, such as Depth1 or Depth2 to alleviate analog readout with low processing energy.

D. Simulated RedEye circuit characteristics

We evaluate RedEye efficiency by simulating analog energy consumption and estimating the energy consumed by the on-chip microcontroller using state-of-the-art models.

RedEye can be configured for different fidelity by configuring noise-damping capacitance (Table I). In high-efficiency (40 dB SNR) mode, RedEye's analog circuitry uses 1.4 mJ to process a 227×227 color image through Depth5 at 30-fps, based on simulation results.

With a central clock frequency of 250 MHz for the 30-fps frame rate, we estimate that the Cortex-M0+ consumes an additional 12 mW, based on its power-frequency ratio (47.4 µW/MHz) fabricated in 0.18-µm process.

In order to understand RedEye's design footprint, we estimate the silicon area using the sizes of unit circuit components, multiplied by the number of components on chip. Each column slice is estimated to occupy 0.225 mm^2, with a low interconnect complexity of 23 per column. In addition, RedEye requires 100-kB memory to store features and 9-kB for kernels, which fit within the 128-kB on-chip SRAM. In total, RedEye components amount to a die size of 10.2×5.0 mm^2, including the 0.5×7 mm^2 customized on-chip microcontroller and the 4.5×4.5 mm^2 pixel array.

1) Opportunities from technology advancements: As semiconductor technology advances, RedEye will face new challenges and opportunities. In contrast with digital circuits, which possess performance and efficiency benefits from technology scaling, analog circuits only receive marginal improvements. This is mainly due to signal swing constraints, short-channel effects and well proximity effects. However, advanced analog processes allow RedEye to explore low-SNR regions for reduced energy cost-per-operation. Furthermore, RedEye is ideal for 3D stacking; pages of analog memory can be physically layered, reducing die size.

In addition, stacked RedEyes could be programmed with different tasks (e.g., face recognition, HOG, object classification, etc.), to coexist on the same module and operate in parallel. Finally, conventional image processing architecture could occupy a layer, allowing a device to acquire a full image through RedEye's optical focal plane when needed.

VI. Related Work

Image sensor energy optimization: Choi et al. survey power saving techniques, including voltage scaling, in-pixel A/D conversion, small pixel sizes and suppressing switching operations in [6]. The effectiveness of these techniques is limited; voltage scaling reduces signal swing, requiring higher power to maintain the SNR, as shown in [24]. In-pixel A/D and small-sized pixels reduce pixel fill factor due to process constraints. Finally, implementation-specific techniques, e.g., suppressing switching, subject their effectiveness to circuit functionality and behavior, and cannot generalize to the wide array of analog readout architectures.

Limited-precision computing: Works in approximate computing [9], [25]–[27] are especially related, utilizing efficiency-performance tradeoffs. These approaches target *computation* energy, while our work focuses on reducing

sensor readout and memory transaction overheads, especially when paired with specialized hardware or cloud offloading.

Many works share our strategy of analog efficiency in the design of limited-precision analog neural accelerators [9], [28], [29]. Towards assisting generic computing at low noise, each of these designs apply only a single layer of analog neurons per conversion, using a digital interface to support generic computing. Though readout energy is significant, limiting the benefit of analog computing, this is necessary overhead to support high-fidelity inter-neuron communications across the analog-digital barrier. ISAAC [29] uses in-situ crossbar pipelines and efficient "flipped" weight encodings, reducing the static and dynamic power consumption of analog-digital conversion.

As RedEye does not need to support generic computing, our contributions revolve around cyclic computing in the analog domain, despite accumulating noise and increasing complexity. This eliminates system data transactions by constraining inter-neuron signal flow to the analog domain; only one crossing at the analog-digital boundary is needed for the entirety of the pipeline.

Hardware ConvNet acceleration: Many convolutional architectures are focused on reconfigurable flexibility [30]–[32]. Improving upon these, some accelerators exploit data locality for performance and efficiency of ConvNet processing [33]–[35]. These accelerators reduce data accesses and data copies, increasing energy efficiency. To further reduce data traffic, the ShiDianNao co-processor [36] streams raw data directly from the sensor. This brings processing closer to the sensor; we similarly push computation *into* the sensor, before analog readout, to reduce data and readout overhead.

In-sensor processing: Focal-plane compression [37], in-imager convolution [38], and programmable in-imager operation [39] enable certain low-power functionality in the image sensor. Bounded by complexity and noise, these designs perform small chains of specific filtering. By contrast, as we approach the problem of deep ConvNet support, we employ reusable module design and noise tuning to efficiently operate many layers of operation before quantization.

VII. Concluding remarks

In this paper, we present the system and circuit designs of RedEye, moving ConvNet processing into an image sensor's analog domain to reduce analog readout and computational burden. Because RedEye allows developers to trade signal fidelity for energy, we report a simulation framework to assist them in balancing accuracy and efficiency. With a per-frame sensor energy reduction of 84.5%, cloudlet system energy reduction of 73%, and computational system energy reduction of 45%, RedEye advances towards overcoming the energy-efficiency barrier to continuous mobile vision.

While we design and simulate RedEye energy, noise, and timing performance, we do not yet provide a circuit layout of the RedEye architecture. Thus, we have not accounted for layout issues including parasitics, signal crosstalk, or component mismatch. We also plan to investigate the following:

Situational uses for noise scaling: As shown in §V-C, GoogLeNet is robust to noise, only susceptible to signal infidelity when SNR drops below 30 dB. However, sensors may experience noise not represented in the ImageNet dataset: using RedEye in a 1 lux environment would reduce the lower limit of the RedEye SNR range to 25 dB. Dynamically scaling RedEye noise enables operation in poorly lit environments, at the cost of higher energy consumption.

Privacy of continuous mobile vision: While we were motivated by the fundamental physical barrier to continuous mobile vision, i.e., energy efficiency, RedEye also provides a platform to explore a novel approach toward the fundamental social barrier: *privacy*. Using techniques such as [40] to generate a quantified reconstruction error, we can train a ConvNet to guarantee image irreversibility. Processing such a ConvNet in the analog domain and discarding the raw image would provide a strong privacy guarantee to the user.

RedEye-specific ConvNet: Current ConvNets have been designed for digital architectures, with plentiful memory and floating point operations. We plan to investigate the training of a ConvNet specific to the RedEye architecture, aware of the efficiency and infidelity tradeoffs of the analog domain.

Acknowledgments

The authors thank Prof. Gene Frantz for input throughout the project. Prof. Boris Murmann provided useful feedback on an early version of the draft. The authors are grateful for comments made by anonymous reviewers. This work was supported in part by NSF Awards CNS #1054693, CNS #1218041, and CNS #1422312. Robert LiKamWa was supported by a Texas Instruments Graduate Fellowship.

References

[1] P. Bahl, M. Philipose, and L. Zhong, "Vision: cloud-powered sight for all: showing the cloud what you see," in *Proc. ACM Wrkshp. Mobile Cloud Computing & Services (MCCS)*, 2012.

[2] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance," in *Proc. ACM Int. Conf. Mobile Systems, Applications, & Services (MobiSys)*, 2014.

[3] R. LiKamWa, Z. Wang, A. Carroll, X. Lin, and L. Zhong, "Draining our Glass: an energy and heat characterization of Google Glass," in *Proc. ACM Asia-Pacific Wrkshp. on Systems (APSys)*, 2014.

[4] R. LiKamWa, B. Priyantha, M. Philipose, L. Zhong, and P. Bahl, "Energy characterization and optimization of image sensing toward continuous mobile vision," in *Proc. ACM Int. Conf. Mobile Systems, Applications, & Services (MobiSys)*, 2013.

[5] K. Kitamura, T. Watabe, T. Sawamoto, T. Kosugi, T. Akahori, T. Iida, K. Isobe, T. Watanabe, H. Shimamoto, H. Ohtake *et al.*, "A 33-Megapixel 120-frames-per-second 2.5-Watt CMOS image sensor with column-parallel two-stage cyclic analog-to-digital converters," *IEEE Transactions on Electron Devices*, no. 12, pp. 3426–3433, 2012.

[6] J. Choi, S. Park, J. Cho, and E. Yoon, "An energy/illumination-adaptive CMOS image sensor with reconfigurable modes of operations," *IEEE Journal of Solid-State Circuits*, no. 6, 2015.

[7] I. Takayanagi, M. Shirakawa, K. Mitani, M. Sugawara, S. Iversen, J. Moholt, J. Nakamura, and E. R. Fossum, "A 1.25-inch 60-frames/s 8.3-M-pixel digital-output CMOS image sensor," *IEEE Journal of Solid-State Circuits*, no. 11, 2005.

[8] W. Hu, B. Amos, Z. Chen, K. Ha, W. Richter, P. Pillai, B. Gilbert, J. Harkes, and M. Satyanarayanan, "The case for offload shaping," in *Proc. ACM Wrkshp. Mobile Computing Systems & Applications (HotMobile)*, 2015.

[9] R. St Amant, A. Yazdanbakhsh, J. Park, B. Thwaites, H. Esmaeilzadeh, A. Hassibi, L. Ceze, and D. Burger, "General-purpose code acceleration with limited-precision analog computation," in *Proc. ACM/IEEE Int. Symp. Computer Architecture (ISCA)*, 2014.

[10] B. Murmann, "A/D converter trends: Power dissipation, scaling and digitally assisted architectures," in *Proc. IEEE Custom Integrated Circuits Conf. (CICC)*, 2008.

[11] J. Mahattanakul and J. Chutichatuporn, "Design procedure for two-stage CMOS Opamp with flexible noise-power balancing scheme," *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1508–1514, August 2005.

[12] R. Sarpeshkar, T. Delbruck, C. Mead *et al.*, "White noise in MOS transistors and resistors," *IEEE Circuits and Devices Magazine*, vol. 9, no. 6, 1993.

[13] B. Murmann. ADC performance survey 1997-2015. [Online]. Available: http://web.stanford.edu/\~murmann/adcsurvey.htm

[14] P. Harpe, E. Cantatore, and A. van Roermund, "A 2.2/2.7 fj/conversion-step 10/12b 40kS/s SAR ADC with data-driven noise reduction," in *Digest IEEE Int. Solid-State Circuits Conf. (ISSCC)*, 2013.

[15] B. Murmann, "Energy limits in A/D converters," in *IEEE Faible Tension Faible Consommation (FTFC)*, 2013, pp. 1–4.

[16] N. Verma and A. P. Chandrakasan, "An ultra low energy 12-bit rate-resolution scalable SAR ADC for wireless sensor nodes," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 6, 2007.

[17] M. Yip and A. P. Chandrakasan, "A resolution-reconfigurable 5-to-10b 0.4-to-1V power scalable SAR ADC," in *Digest IEEE Int. Solid-State Circuits Conf. (ISSCC)*, 2011.

[18] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*, 2010.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[20] A. B. Patel, T. Nguyen, and R. G. Baraniuk, "A probabilistic theory of deep learning," *arXiv preprint arXiv:1504.00641*, 2015.

[21] G. B. Dantzig, A. Orden, P. Wolfe *et al.*, "The generalized simplex method for minimizing a linear form under linear inequality restraints," *Pacific Journal of Mathematics*, no. 2, pp. 183–195, 1955.

[22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *arXiv preprint arXiv:1409.4842*, 2014.

[23] M. Siekkinen, M. Hiienkari, J. K. Nurminen, and J. Nieminen, "How low energy is bluetooth low energy? comparative measurements with zigbee/802.15.4," in *Proc. IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, 2012.

[24] H.-S. Wong, "Technology and device scaling considerations for CMOS imagers," *IEEE Transactions on Electron Devices*, vol. 43, no. 12, pp. 2131–2142, 1996.

[25] R. S. Amant, D. A. Jiménez, and D. Burger, "Mixed-signal approximate computation: A neural predictor case study," *IEEE Micro*, no. 1, 2009.

[26] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Neural acceleration for general-purpose approximate programs," in *Proc. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, 2012.

[27] Z. Du, A. Lingamneni, Y. Chen, K. Palem, O. Temam, and C. Wu, "Leveraging the error resilience of machine-learning applications for designing highly energy efficient accelerators," in *Proc. IEEE Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2014.

[28] B. Belhadj, A. Joubert, Z. Li, R. Héliot, and O. Temam, "Continuous real-world inputs can open up alternative accelerator designs," in *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3. ACM, 2013, pp. 1–12.

[29] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and T. . I. Y. . . Srikumar, Vivek Booktitle = Proceedings of ACM/IEEE Int. Symp. Computer Architecture (ISCA).

[30] P. H. Pham, D. Jelaca, C. Farabet, B. Martini, Y. LeCun, and E. Culurciello, "NeuFlow: Dataflow vision processing system-on-a-chip," in *Proc. IEEE Midwest Symp. Circuits and Systems*, 2012.

[31] L. Cavigelli, D. Gschwend, C. Mayer, S. Willi, B. Muheim, and L. Benini, "Origami: A convolutional network accelerator," in *Proc. ACM Great Lakes Symp. VLSI*, 2015.

[32] S. Chakradhar, M. Sankaradas, V. Jakkula, and S. Cadambi, "A dynamically configurable coprocessor for convolutional neural networks," in *ACM SIGARCH Computer Architecture News*, no. 3, 2010, pp. 247–257.

[33] W. Qadeer, R. Hameed, O. Shacham, P. Venkatesan, C. Kozyrakis, and M. A. Horowitz, "Convolution engine: balancing efficiency & flexibility in specialized computing," in *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3, 2013, pp. 24–35.

[34] M. Peemen, A. a. a. Setio, B. Mesman, and H. Corporaal, "Memory-centric accelerator design for convolutional neural networks," in *Proc. IEEE Int. Conf. Computer Design*, 2013.

[35] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam, "DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," in *Proc. ACM Int. Conf. Architectural Support for Programming Languages & Operating Systems (ASPLOS)*, 2014.

[36] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, X. Feng, Y. Chen, and O. Temam, "ShiDianNao: Shifting vision processing closer to the sensor," in *Proc. ACM/IEEE Int. Symp. Computer Architecture (ISCA)*, 2015.

[37] W. D. León-Salas, S. Balkir, K. Sayood, N. Schemm, and M. W. Hoffman, "A CMOS imager with focal plane compression using predictive coding," *IEEE Journal of Solid-State Circuits*, no. 11, 2007.

[38] A. Nilchi, J. Aziz, and R. Genov, "Focal-plane algorithmically-multiplying CMOS computational image sensor," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 6, 2009.

[39] P. Hasler, "Low-power programmable signal processing," in *Proc. IEEE Int. Wrkshp. System-on-Chip for Real-Time Applications*, 2005.

[40] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, June 2015.