sensors

MDPI

Article

Dynamic Temperature Management of Near-Sensor Processing for Energy-Efficient High-Fidelity Imaging

Venkatesh Kodukula *, Saad Katrawala, Britton Jones, Carole-Jean Wu and Robert LiKamWa

School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85281, USA; skatrawa@asu.edu (S.K.); bsjones7@asu.edu (B.J.); carole-jean.wu@asu.edu (C.-J.W.); likamwa@asu.edu (R.L.)
* Correspondence: vkoduku1@asu.edu

Abstract: Vision processing on traditional architectures is inefficient due to energy-expensive off-chip data movement. Many researchers advocate pushing processing close to the sensor to substantially reduce data movement. However, continuous near-sensor processing raises sensor temperature, impairing imaging/vision fidelity. We characterize the thermal implications of using 3D stacked image sensors with near-sensor vision processing units. Our characterization reveals that near-sensor processing reduces system power but degrades image quality. For reasonable image fidelity, the sensor temperature needs to stay below a threshold, situationally determined by application needs. Fortunately, our characterization also identifies opportunities—unique to the needs of near-sensor processing—to regulate temperature based on dynamic visual task requirements and rapidly increase capture quality on demand. Based on our characterization, we propose and investigate two thermal management strategies—stop-capture-go and seasonal migration—for imaging-aware thermal management. For our evaluated tasks, our policies save up to 53% of system power with negligible performance impact and sustained image fidelity.

Keywords: thermal management; image sensors; fidelity; continuous mobile vision

Citation: Kodukula, V.; Katrawala, S.; Jones, B.; Wu, C.-J.; LiKamWa, R. Dynamic Temperature Management of Near-Sensor Processing for Energy-Efficient High-Fidelity Imaging. *Sensors* **2021**, *21*, 926. https://doi.org/10.3390/s21030926

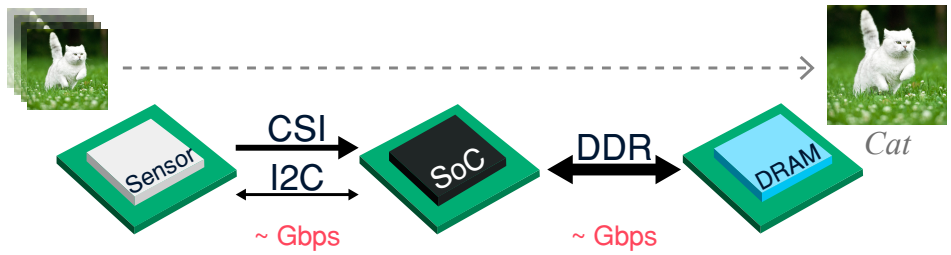
Received: 7 December 2020
Accepted: 27 January 2021
Published: 30 January 2021

1. Introduction

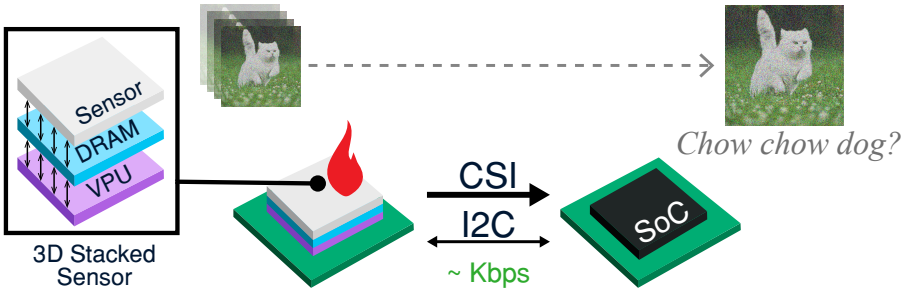
Imaging and vision systems allow computing systems to sense real-world visual situations and to capture images for human consumption. Camera-enabled devices can now perform a wide range of visual tasks such as detecting and tracking objects [1], constructing spatial maps for augmented reality [2], and providing driverless navigation assistance [3]. Unfortunately, imaging requires high data rates, e.g., 2 Gbps for 4 K@30 fps, to transfer pixel data from the image sensor to computational units. In traditional systems (Figure 1a), where the computational units are separated from the sensor via long interconnects, e.g., ribbon cables, these data rates create bottlenecks to energy efficiency and processing. Thus, current vision systems result in power profiles on the order of multiple watts. It has been shown that state-of-the-art convolutional neural networks (ConvNets) consume over 1 W of processing power to achieve a desirable performance of 30 frames per second (fps) with low-resolution QVGA frames on ASICs [4,5]. The power consumption increases significantly with higher resolution inputs and higher frame rates. To enable more exciting use cases, imaging systems need order-of-magnitude energy efficiency improvements to be able to analyze higher resolution image inputs while achieving higher frame rates.

This need for energy-efficiency has motivated recent investigations into 3-dimensional *stacked* integrated circuit architectures (Figure 1b) to enable *near sensor processing*. In 2012, the first prototype of stacked sensors became available [6], enabling rudimentary image processing, such as demosiacing. Other architectural trends propose processing near sensors; RedEye [7] and ShiDianNao [8] perform ConvNet inference near the sensor, substantially improving energy efficiency of vision systems by 45%. In combination of the two trends, i.e., integrating light-weight and heavy-weight processing into the sensor,

future 3D-stacked sensors can layer the sensor, vision processor unit (VPU), and memory in the same package.



(**a**) Traditional vision pipeline.



(**b**) Near-sensor vision pipeline.

Figure 1. Due to energy-expensive interface data movements, traditional pipelines are inefficient. Near-sensor processing helps greatly reduce data traffic promoting energy-efficiency. However, it generates heat increasing sensor temperature, thereby resulting in noisy images potentially degrading task accuracy.

Unfortunately, sensor temperature sensitivity prevents a full adoption of near-sensor processing, creating thermal noise, e.g., dark current and read noise, in captured images. In addition to generating less aesthetically pleasing images for human viewing, these noisy images reduce the visual task accuracy of computer vision applications [9]. Furthermore, poor lighting environments force the sensor to operate at high exposure and ISO (ISO controls the sensitivity of an image sensor to light.) for better scene capturing, which increases a sensor's vulnerability to noise. Despite a plethora of CPU dynamic thermal management (DTM) mechanisms [10–14], existing DTM techniques do not account for imaging requirements, turning a blind eye to the transient imaging needs of near-sensor processing. Thus, despite performance and energy benefits of near-sensor processing, the temperature profile of visual computing limits stacked architectures in many situations.

While others have reported the thermal and noise implications of stacked-sensor processing [15], there lacks a comprehensive end-to-end modeling framework to allow characterization studies considering energy, thermal, and noise implications of near-sensor processing workloads together. Thus, in Section 3, we design and validate one such framework, aimed to assess thermal behavior of stacked-sensor architectures. We validate our thermal models against image sensor measurements and find that our RC models closely match with real measurements with an error margin of 0.1%. In addition to confirming and characterizing relationships between near-sensor processing power and sensor temperature, our modeling reveals a consequential insight: despite the coarse thermal time constant for the sensor to settle to steady-state temperatures, *removing near-sensor power results in an immediate and significant reduction in transient junction temperature* (Junction temperature typically means the temperature of the hottest location in the die). For example, for a 2.5 W system, the sensor temperature drops by roughly 13 °C within 20 ms when the processing is deactivated.

In Section 4, we build on characterized challenges to provision for imaging-specific temperature management. We designed the *Stagioni* runtime to orchestrate temperature

management for near-sensor processing, including two temperature-aware scheduling policies—*stop-capture-go* and *seasonal migration*—for effective near-sensor vision processing. The policies aim to minimize system energy consumption and afford high-performance computation and high fidelity capture. Stop-capture-go briefly suspends processing to allow for on-demand high fidelity captures and resumes the processing after the capture. Seasonal migration occasionally shifts processing to a thermally isolated far-sensor processing unit for high fidelity capture.

In Section 6, we then use an emulation framework to evaluate the effectiveness of Stagioni's mechanisms to manage a sensor temperature to suit imaging needs. We make the following contributions:

- We develop and validate an end-to-end modeling framework for studying energy, thermal, and noise implications of near-sensor processing. We use this framework to characterize different implications of a typical 3D stacked near-sensor architecture.
- Motivated by the characterization findings, we design principles and propose novel fidelity-driven runtime mechanisms for effective sensor thermal management.
- Through emulation-based evaluation, we show that Stagioni determines duty cycling of near-sensor task processing to avoid fidelity issues that stem from thermal problems. We find that Stagioni reduces average system power by 22–53%; actual savings depend on specific power profiles and image fidelity needs.

Enabling high performance and high efficiency near-sensor processing would unlock the potential for several vision/imaging applications, including sophisticated dashboard cameras, continuous augmented reality tracking, and other futuristic use cases. Throughout this work, we study the implications of near-sensor processing and evaluate the policies around a life-logger case study. A wearable life-logger device chronicles important events and objects in a person's life. The device runs object detection and tracking algorithms to continuously identify and locate objects in the scene. Meanwhile, the device performs occasional captures upon detecting any important event, e.g., a person entering the scene. This can form the basis for personalized real-world search engines, and assist those with memory impairments or visual impairments.

2. Background and Related Work

The paradigm of near-sensor processing emerged in the 1990s to reduce the communication and storage overhead of off-sensor processing. Early works [16] leveraged physical properties to perform low-level image processing tasks, e.g., median filtering. Later, researchers integrated image processing units [17] after the read-out circuits in the imaging plane, outputting extracted image features. With advancement in 3D circuit integration, recent works [15] design 3D stacked image sensors, some of which include a system-on-a-chip (SoC). Inside the SoC, sensor, processor, and memory are stacked into the same package. This architecture performs high-level image processing tasks, such as ConvNet-based classification. 3D stacked architectures have seen commercial advances. These imagers still use the traditional CSI interface to communicate with the SoC. For slow-motion capture, Sony [18] stacks a DRAM beneath the sensor layers. With local memory, the sensor captures and buffers frames at 1000 fps, sending them across the slower camera interface to the host. Samsung [19] uses a similar sensor for their recent Galaxy phone. For surveillance, Sony [20] integrated a motion estimation block, microcontroller, and DRAM in the 3D stacked sensor. More recently, Sony [21] integrated an AI vision processor inside their recent sensor for performing in-sensor computer vision.

Though vision can be done through handcrafted feature analysis [22], the current trend uses ConvNets for visual tasks on a wide range of architectures. High programmability, performance, and energy-efficiency are desired to meet the rapid pace at which ConvNets are evolving. General-purpose platforms built around GPUs provide programmable high performance software libraries [23,24] to implement ConvNets at the expense of more power, e.g., 60 fps at 10 s to 100 s of watts [25–27]. FPGAs provide performance and scalability at reduced power. The state-of-the-art FPGA implementations [26,28,29] typically

consume several watts of power. In recent years, we see the rise of domain specific processors such as Myriad2 [5] that provide programmable SIMD capabilities on a RISC processor. This brings down the power to a few watts [5], but at the cost of performance, e.g., 3 fps. Meanwhile, academic ASICs [8,30,31] provide energy-efficiency and performance for ConvNets. However, benefits are bottlenecked by DRAM accesses. For example, Eyeriss [30] achieves 278 mW@35 fps for AlexNet. When scaled for VGG16 [32], performance drops to about 10 fps within the same power budget. For reasonable performance, scalability, and mobility, the system power profile ranges from 1 to 15 W. Placing these VPUs near the sensor and solving temperature challenges would unlock substantial improvements in performance and energy-efficiency through near-sensor processing.

Image sensors are susceptible to different types of noise due to imperfections in lighting, sensing elements, and imaging circuitry. Sources of noise can be grouped into fixed-pattern noise and temporal noise. Fixed-pattern noise arises due to non-uniform sensitivities of photodiodes to light. As it remains constant over time, conventional strategies read it once and subtract it later to eliminate its effect. Notably, there is also dark current (leakage from transistors) that exponentially increases with temperature. In contrast to fixed-pattern noise, temporal noise sources vary with each capture. Temporal noise sources include read noise and dark current shot noise, which exhibit strong dependence on temperature. All electronic noise sources, e.g., readout elements, amplifiers, are grouped together as read noise, which has a variance of kT/C. This noise is due to random thermal activity of electronic charge carriers. Dark current shot noise also stems from similar phenomenon happening in photodiodes; high temperatures trigger randomness in the photodiode charge carriers, thereby inducing more noise in images. Unfortunately, thermal noise cannot be fully suppressed using signal processing techniques without generating imaging artifacts [33]. The only solution is to manage sensor temperature.

For efficient thermal management, different techniques have been explored for multi-core processors. Stop-and-go [34] suspends the execution of a thread, for a while, when a core on which it is running gets overheated and resumes its operation once the core cools down. Heat-and-run [14] technique migrates the thread from a hotter core to a cooler one to allow the hotter core to cool down. Traditional DTM techniques are designed to keep the processor power within a thermal design power (TDP). We are inspired by the same core mechanisms—stop-go and seasonal migration—for power and temperature reduction. In contrast to the existing works, we redesign these mechanisms to fulfill dynamic imaging needs.

Recent works report temperature issues in 3D stacked image sensors. Amir et al. [15] stack a DRAM and a deep neural network (DNN) processor beneath the sensor layer. They report that sensor temperature can increase due to DNN computation, resulting in higher noise and lower ConvNet accuracy. Lie et al. [35] report similar issues for their 3-layer stack architecture with a image compression unit integrated inside the stack. Similar to earlier works, we report similar issues for our characterized 3D stacked image sensor. However, previous works provide *design time* solutions, e.g., statically partitioning computation to execute partial ConvNets on the sensor and the rest on the host. Our work is complementary, providing *runtime solutions* for thermal management.

3. Modeling Energy, Thermal, and Fidelity Implications of Near-Sensor Processing

In this section, we construct a modeling framework to examine the implications of using stacked integration to place a VPU layered underneath the sensor for near-sensor processing. Our estimates are based on a suite of parameterizable energy, temperature, and noise models of different hardware structures of a 3D stacked system. To develop our models, we leverage datasheets, ITRS roadmaps, and commercial simulation software to produce accurate estimation of different thermal characteristics of 3D stacked sensors. We also validate these models through sensor hardware measurements.

To better appreciate the insights offered by near-sensor processing, we study various system implications around our life-logger case study. Our studies confirm that near-

sensor processing minimizes off-chip data movement, thereby substantially reducing interface power and overall system energy consumption. With near-sensor processing in our case study, we can reduce the system power of ResNet-based image classification by 52%. We also relate near-sensor processing power to image fidelity through temperature simulation, confirming that image fidelity degrades over time with additional near-sensor processing power. We observe that removal of near-sensor processing power via throttling or computation offloading leads to rapid drops in sensor temperature, e.g., reducing temperature by 13 °C in 20 ms. We can exploit this observation to allow the sensor to operate at higher temperatures and lower image fidelities for energy-efficient vision, e.g., continuous object detection, while switching to low temperature operation for high-fidelity image capture when an application needs high quality photographs of a particular object.

3.1. Energy Analysis of Near-Sensor Processing

Near-sensor processing reduces energy-expensive data movement across the interconnects between different chips. Here we examine energy profiles of vision pipelines, comparing traditional and near-sensor pipelines. Our energy models provide coarse estimation; actual numbers will depend on factors such as architectural decisions and patterns of execution.

Energy of Vision Pipeline Components

Traditional pipelines operate across chips to connect a variety of subsystems: camera, processing unit, memory. The camera chip connects to processing units on the system-on-chip (SoC) through a standard camera serial interface (CSI) for data transfer and an I^2C interface for control and configuration. Meanwhile, the SoC buffers image frames with DRAM through an external DDR interface.

Using regression models on measurements and reported values, we construct a coarse energy profile model to motivate the need for near-sensor processing. As shown in Table 1, we find that sensing, processing, and storage consume 100 s of pJ per pixel. On the other hand, communication interfaces consume over 3 nJ per pixel.

Table 1. Energy-per-pixel of various components.

Component	Energy (pJ/pixel)
Sensing	595
Communication (Sensor-SoC)	900
Communication (SoC-DRAM)	2800
Storage (Read)	283
Storage (Write)	394

Sensing requires an energy of 595 pJ/pixel [36,37], mostly drawn from three components: pixel array, read-out circuits, and analog signal chain, which consume 25 pJ/pixel, 43 pJ/pixel, and 527 pJ/pixel, respectively. DRAM storage on standard mobile-class memory chips (8 Gb, 32-bit LPDDR4) draws 677 pJ/pixel for writing and reading a pixel value [38]. This roughly divides into 283 pJ/pixel for reading and 394 pJ/pixel for writing. Communication over CSI and DDR interfaces incur 3.7 nJ/pixel, mostly due to operational amplifiers on transmitters and receivers. We measure the interface power dissipation [39] on 4-lane CSI interfaces and LPDDR4 interfaces by inputting several data rates. From this information, we construct a linear-regression model to estimate the energy per pixel to be 0.9 nJ/pixel over CSI and 2.8 nJ/pixel over DDR. For computation, we gather reported power dissipations of various ConvNet architectures from the literature.

To illustrate the efficacy of near-sensor processing, in Table 2, we use our energy models to estimate the system power numbers of traditional and stacked-sensor processing for state-of-the-art ConvNet models. We combine reported computation values with

modeled sensing, storage, and communication energy to estimate system power dissipation. At 1920 × 1080 and 34 fps, and using ResNet for inference on the SoC VPU, the modeled system consumes 2.7 W.

Table 2. Deep neural network (DNN) models and corresponding power profiles.

DNN Model (VPU Arch)	Frame Rate (fps)	Trad. Sys Power (W)	NSP Sys Power (W)
AlexNet (Myriad2)	12	3	1.86
mobileNetSSD (Myriad2)	11.8	1.92	0.9
GoogLeNet (Neurostream)	83	3.13	1.81
ResNet50 (Neurostream)	34	2.67	1.34

On-chip data movement is known to be significantly more power efficient than off-chip data movement by six orders of magnitude [40]. Advances in near-sensor processing leverage this for energy-efficiency gains, as shown in Figure 1b. Near-sensor processing moves the DRAM into the sensor to eliminate off-chip DDR movement, and moves the VPU into the sensor to reduce the CSI interface data rate. Thus, the output of the sensor can be reduced from a few MB to a few bytes. This information can be sent across efficient low data rate interfaces, e.g., I^2C. Altogether, when applying our energy profile models to the near sensor processing pipeline, we find that the VPU near sensor system consumes 1.3 W, thereby yielding 52% savings over traditional architectures.

3.2. Thermal Analysis of Sensor Processing

Though tight integration yields energy efficiency and performance benefits, near-sensor processing generates heat at the sensor through thermal coupling between tightly integrated components. While dynamic thermal management for CPU is only concerned with keeping peak power draw below a TDP, we pay close attention to temperature patterns, as transient temperature behavior affects image fidelity. Conduction is the dominant heat transfer mechanism in integrated circuits. To model temperature dynamics, we use simple thermal resistance-capacitance (RC) modeling [41] techniques to determine stacked sensor characteristics.

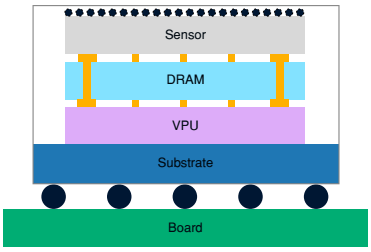
3.2.1. Deriving the Component Values in the RC Model

Figure 2 shows a typical structure of a 3D stacked sensor package and its RC model. The sensor, DRAM, and VPU layers are stacked on top of each others, connected to each other, e.g., using through-silicon-vias. The top of the stack opens to the surroundings through microlenses, while the bottom sits on a substrate that opens to the printed circuit board. Mobile-class image sensors omit heat sinks or cooling fans, due to their size, weight, and placement challenges. The layers consume power when active, which dissipates as heat. We primarily consider vertical heat transfer; vertical resistances are several orders of magnitude smaller than the lateral resistances of convective heat transfer. We obtain component values of the layers through analytical and empirical approaches.

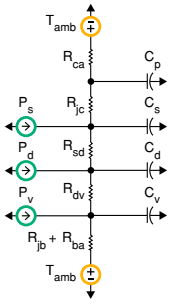
Figure 2c shows different RC component values derived for our model. Previous works report layer dimension values of typical 3D stacked image sensors [15]. In these works, the layer thickness ranges in the order of 1–10 s of microns, while the layer's area ranges from 10–100 s of mm^2. The ITRS roadmap provides layer dimensions and material property constants ρ and c to define the guidelines for semiconductor fabrication. From these, we derive the thermal resistance $R = \rho t / A$ and thermal capacitance as $C = ctA$ where A is the layer's cross sectional area and t the thickness.

Package capacitance can be deduced empirically by observing the temperature trace of an image sensor chip while subjecting the sensor to thermal stress. We construct regression models from the temperature trace of an OnSemi AR0330 smartphone-class image sensor to derive package capacitance. Finally, termination thermal resistance depends on the

properties of the casing and board. Sensor companies make these values available through datasheets. We use such provided values for typical packages directly in our model.



(**a**) Cross-section of the 3D stack.



(**b**) RC model of stack.

Component	R (K/W)
R_{ca}: Case-to-Ambient	56
R_{jc}: Junction-to-Case	6
R_{sd}: Sensor-to-DRAM	0.6
R_{dv}: DRAM-to-VPU	0.6
R_{jb}: Junction-to-Board	40
R_{ba}: Board-to-Ambient	14

Layer	C (J/K)
C_p: Package	1
C_s: Sensor	0.65m
C_d: DRAM	0.65m
C_v: VPU	0.65m

(**c**) RC values of different components.

Figure 2. Using the well-known duality between thermal and electrical phenomena, thermal modeling of stacked sensors can be performed by analyzing an equivalent RC circuit.

Off-sensor power does not affect sensor temperature. While processing far from the sensor, the off-sensor VPU and SoC components do not influence the sensor temperature. Even in tightly integrated systems, e.g., smartphones, the sensor and SoC reside on two different boards and communicate over a ribbon cable. As a result, the sensor and SoC are nearly in thermal isolation. That is, any increase in temperature of one component will not cause appreciable change in temperature of the other. We verify this effect by running a CPU-bound workload on SoC on a Google Nexus smartphone while keeping the camera idle. Our thermal camera instruments do not report any associated rise in camera temperature with an induced rise in SoC temperature. Thus, in our study, we do not consider off-sensor thermal coupling effects.

3.2.2. Simulation-Based Thermal Analysis

Through LTSpice simulation on our RC models, we estimate the thermal behavior of near-sensor processing architectures. We evaluate temperature profiles as the sensor operates in two different modes: *NSP* mode, in which power dissipation is representative of capturing image frames and processing vision workloads near the sensor, and *CAP* mode, in which power dissipations are representative of capturing image frames and either dropping frames or transmitting them to the SoC. With various execution patterns, we can simulate the thermal behavior of the sensor as the system operates among different sensor modes. Previous analysis has reported that we can safely ignore spatial variations in temperature if the chip power density is within 20 W cm^{-2} [42], as is the case in NSP mode. Power density, which is the power dissipated over the chip area, measures the degree of spatial non-uniformities in temperature. The physical dimensions of our 3D stacked image sensor combined with the power profile of our case study results in a power density of 16 W cm^{-2}. Therefore, we do not consider the spatial variations of temperature inside the stack for our modeling near-sensor processing architectures.

We model static power as a voltage dependent current source as shown in the following equation, which is the popular model used in computer architecture literature [43].

$$P_{leakage} = P_{ref} * e^{\beta * (T - T_{ref})} \quad (1)$$

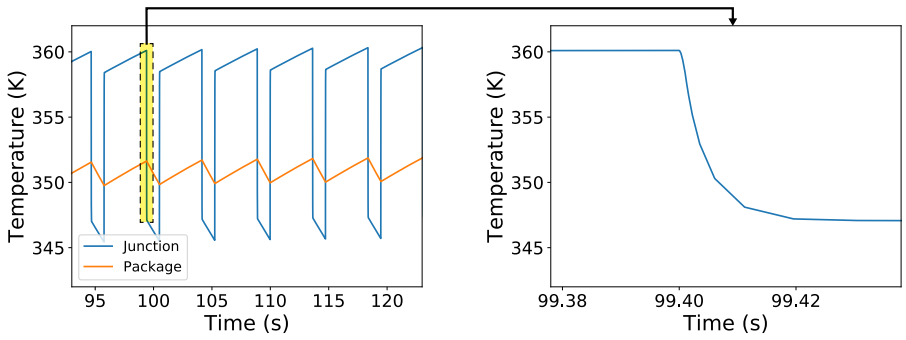
We choose reference temperature as 300 K and obtain beta value of 0.0075 from the ITRS roadmap.

Inter-layer resistances are at least two orders of magnitude smaller than termination resistances. This results in negligible drop across the resistor, leading to minuscule temperature gradients between layers. For example, for 1 W of VPU power, the sensor, DRAM, and VPU will be at 60.7 °C, 60.9 °C, and 61.0 °C, respectively. Thus, we combine the layers and model the sensor temperature as a single RC point. Generally, reducing VPU power

dissipation corresponds directly to temperature decrease. The RC-based model shows that reducing near-sensor power from 1 W to 100 mW results in a temperature drop of 5 °C. In addition, a higher ambient temperature leads to raised steady-state temperatures. With static power, we find that the steady-state temperatures rise even further.

Thermal time constants govern the transient temperature of the stacked image sensor. As the thermal capacitance of a chip package is often several orders of magnitude greater than that of a die, the thermal time constant of the package predominantly guides the trajectory of temperature to steady-state, taking 10 s of seconds to reach a steady-state temperature. With static power, we find that the temperature rise is steeper. The coarser thermal time constant allows dynamic temperature management policies ample time to form decisions, e.g., altering temperature by changing near-sensor power draw.

Notably, near-sensor power dissipation raises the transient temperature of the sensor die above the package temperature. This is because the heat source is on the sensor die itself, dissipating heat through the package into the ambient environment. Consequently, reducing power dissipation rapidly reduces the gap between sensor die transient temperature and package temperature, as shown in Figure 3. The speed of the temperature drop is governed by the sensor junction die time constant, which is on the order of milliseconds. Prior works, e.g., [10–14], have found similar temperature characteristics of the large, sudden drop for CPUs. However, CPU thermal management can neglect fine-grained temperature variations because its goal is to govern the processor junction temperature below a threshold. As a result that transient temperature affects image fidelity, *these rapid temperature drops—such as the charted 13 °C drop in 20 ms—provide unique opportunities for dynamic thermal management for on-demand image fidelity*. We further discuss this in Section 4.



(a) Junction temperature trace with downward vertical lines indicating the immediate jump.

(b) Zoomed-in version of the jump: the junction temperature drops by 13°C within 20 ms

Figure 3. When disabling NSP, a rapid drop in junction temperature occurs within 20 ms, due to junction time constants.

Due to the lack of configurable stacked sensors, we use an OnSemi mobile class camera [44] with an on-chip temperature sensor to validate our thermal insights. We find that the validation results follow our RC model prediction. In particular, we collect the real-time temperature trace from the on-chip thermal sensor while configuring the sensor in preview and low-power modes. We validate the relationship of temperature to power dissipation. The sensor reaches a temperature of 35.0 °C under a dynamic power dissipation of 250 mW.

When we use the power dissipation as input to our model, the steady-state temperature is estimated to be 34.8 °C, within 0.06% of real measurement. When we switch from the preview mode to the low-power mode, reducing the power dissipation to 150 mW, we observe a steady-state temperature of 31.6 °C. The model prediction is 31.4 °C, within 0.06% of real measurement.

We also observe the sudden temperature drop due to removal of near-sensor power dissipation. Upon transition to the lower power state, we see 30% of the temperature reduction occurring within 30 ms. Our RC model predicts the junction time constant to be 20 ms, which is close to what we observe through hardware measurement. We additionally validate the sudden temperature drop characteristic for higher power dissipation differences, leveraging a mobile SoC [45] and SnapDragon profiler [46], which also profiles battery power draw. We notice substantial drop of 15 °C in when there is a 3 W power removal by turning off a neural network-based object detection application.

3.3. Image Fidelity Implications of Temperature

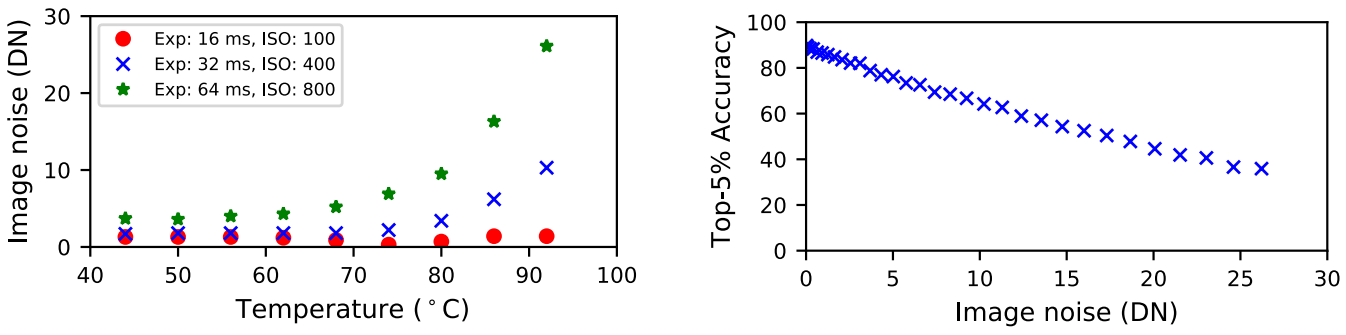
While raised temperatures cause reliability and packaging issues for integrated circuits, they introduce another problem for image sensors: noise. The influence of noise on vision tasks has been widely reported. Dodge et al. [9] find that neural networks have difficulty predicting semantics of an image when challenged by image noise. Similar findings from Amir et al. [15] find that image classification accuracy degrades with increase in temperature. Thus, reliable vision demands images of reasonable fidelity.

The fidelity bar is often high for images captured for human consumption. For example, if a set of dashcam images is to be used in an auto insurance claim, the images need to have superior quality to obtain maximal information for decision-making. While denoising can help mitigate fidelity issues, denoising algorithms often create imaging artifacts which also impair perceived image quality. Thus, as images are required to fiducially represent the real physical world, imaging fidelity needs are more stringent than vision-based needs.

The sources of image noise are theoretically well-understood (Section 2). However, to understand the practical relationship between temperature and image quality on commercial sensors, we perform thermal characterization on a 3 Mp OnSemi AR0330 sensor [47] connected to a Microsemi SmartFusion2 FPGA [48]. The AR0330 sensor includes noise correction stages inside the sensor, as is common in commercial sensors. We use a heat gun to raise sensor temperature and capture raw images in a dark room setting while monitoring sensor temperature with an FLIR One thermal camera.

3.3.1. Noise Is more Prevalent at High Temperatures

Figure 4 charts a trend: sensors are particularly susceptible to noise above a particular temperature value. This is despite the presence of noise correction stages inside the sensor. We observe that the correction blocks bring the noise under control but only for lower temperature settings. However, for high temperatures, the denoising fails to exercise control on noise minimization. Notably, this knee shifts with exposure and analog gain settings, presumably due to noise amplification. For instance, at high exposure and analog gain, which correspond to low light situations, sensors start to become thermally sensitive even at low temperatures, e.g., 52 °C. To adapt to experienced conditions, the sensor's thermal management should be adaptive to varying lighting conditions.



(**a**) Image noise (**b**) Task accuracy

Figure 4. Variance of noise, expressed in pixel intensities, and task accuracy showing sensitivity to temperature, exposure, and ISO.

3.3.2. Noise Visibly and Substantially Impairs Quality

Thermal noise is visibly apparent on images, whether in low light or bright light conditions. For example, Figure 5 shows images captured under daylight conditions at different temperatures. We can observe the graininess in the hotter image due to the strong influence of noise. Paired with the noisy images, the histograms represent the pixel intensity distribution of an image. The wider peaks in the distribution signify the variance of pixel intensity, while the mean of the peaks represent average intensity. We can observe that the histogram of the hotter image shifts to the right, increasing pixel intensity due to dark current. We also observe that the variance of the pixel intensity increases, due to increased thermal noise.

(**a**) Image at 44 °C (**b**) Image at 92 °C (**c**) Image histograms

Figure 5. Two images captured at different temperatures and their histograms. The hotter image is brighter and grainier, due to the influence of thermal noise. This is also reflected in the shift in mean and variance.

Thermal noise also substantially impairs the vision task accuracy, which we study around te image classification task. As shown in Figure 4b, we find that the task accuracy degrades with more image noise. For our experimentation, we feed noisy images into the MobileNet neural network and evaluate the accuracy of the inference by comparing the results against the ground truth labels. To generate noisy images, we inject Gaussian thermal noise to the validation dataset of ImageNet. We repeat this experiment for other classification networks such as SqueezeNet and tiny-ResNet and notice similar trends.

3.4. Motivational Observations

To summarize, we have the following insights for NSP:

- Near-sensor processing promotes system energy-efficiency, but also increases sensor temperature.
- Raised sensor temperatures aggravate thermal noise.
- Smaller (ms) sensor junction time constants facilitate an immediate sensor temperature drop.
- Fidelity needs are highly dynamic and depend on environment, e.g., lighting and ambient temperature.
- Imaging demands more fidelity, but vision tasks are also susceptible to noise, especially in low light.

These observations motivate the need for novel dynamic thermal management for near-sensor processing.

4. Thermal Management for Near-Sensor Processing

Our characterization shows that near-sensor processing increases system energy efficiency, but sacrifices image fidelity due to increased sensor temperatures. This raises a natural question: *Can we leverage near-sensor processing to create efficiency benefits while maintaining sufficient image fidelity for vision and imaging tasks?*Driven by this, we develop novel mechanisms that can efficiently regulate sensor temperature for continuous and on-demand image fidelity needs. In our design, these mechanisms are governed by a runtime controller, which we call Stagioni. Stagioni could be designed in a multitude of ways, e.g., a dynamically linked library, a runtime OS service, or dedicated hardware.

In our implementation and evaluation, Stagioni is a runtime OS service that sits on the near-sensor processor, allowing the SoC to sleep.

Dynamic temperature management for microprocessors is a mature research area, as we summarize in Section 2. However, traditional processor DTM mechanisms are not designed to suit imaging needs. Rather than simply being limited by TDP, fidelity is impaired by the immediate transient sensor temperature while capturing. Furthermore, thermal management for near-sensor processing should adapt to the situational needs of the vision/imaging application, e.g., allowing higher temperatures when in brighter environments and rapidly dropping temperature when high fidelity is required.

To account for near-sensor processing temperature management, we modify traditional DTM techniques to introduce two mechanisms that quell image quality concerns, while striving to optimize for system power and performance. (1) Stop-capture-go: Temporarily *halt* near-sensor processing for thermal regulation and on-demand high fidelity capture. (2) Seasonal migration: Occasionally *migrate* processing to a thermally isolated far-sensor VPU for thermal regulation and on-demand high fidelity capture.

4.1. Design Principles for Sensor Thermal Management

To design thermal management mechanisms that are effective for near-sensor processing, we introduce three core principles: (1) Situational temperature regulation: The mechanism should confine sensor temperature within a threshold that suffices for imaging fidelity needs. (2) On-demand fidelity: Upon application request, the mechanism should quickly drop the temperature to desired capture temperature for high fidelity imaging. (3) Duty cycle governs system efficiency. Here, we discuss these in detail.

4.1.1. Situational Temperature Regulation

As we discuss in Section 3, vision tasks have varying fidelity needs, which are sensitive to camera settings, e.g., ISO and exposure, and lighting situation, e.g., bright conditions. This translates directly to a simple upper bound for temperature:

$$T_{sensor} < T_{vision} \tag{2}$$

Thus, temperature management must be cognizant and respectful of immediate vision task requirements in situational conditions to provision for effective vision accuracy.

4.1.2. On-Demand Fidelity

While vision processing can operate on low fidelity images, certain applications may require high fidelity images on demand, e.g., life logging capture after object detection. Such capture must be immediate, before the object leaves the view of the camera. Fortunately, as we characterized, sensor temperature rapidly drops with the removal of near-sensor power, i.e., by entering CAP mode. For example, when the sensor drops its near-sensor power consumption from 2.5 W to 100 mW, the sensor drops in temperature by 13.2 °C. We experimentally observe that sufficient temperature drop (98.2%) can be achieved within a time of four time constants, which we define as $t_{jump} = 4 \times RC_{die}$. In our simulation, this amounts to 20 ms. Temperature management can leverage this drop to provision for on-demand high fidelity.

The temperature drop is directly proportional to the disparity between the near-sensor power before and after power reduction: $T_{jump} = \alpha(P_{NSP} - P_{CAP})$. We find that for our modeled sensor, every 1 W causes a 5.5 °C temperature jump, i.e., $\alpha = 5.5\,°C\,W^{-1}$. When constrained by a latency deadline, e.g., to immediately capture a moving object or to meet a synchronization deadline, the achievable jump within the latency deadline is a fraction of the time it takes to drop: $T_{jump}^{latency} = T_{jump} \times (e^{-t_{latency}/RC_{die}})$. Thus, to provision for

predicted fidelity needs and latency needs of an application, the temperature management mechanism can set reduced bounds:

$$T_{sensor} < T_{imaging} + T_{jump}^{latency} \tag{3}$$

4.1.3. System Power Minimization through Duty Cycle

While removal of processing power can regulate temperature and provide on-demand high fidelity captures, the scheduling of operation should also strive to optimize for average system power. We can characterize this through the duty cycle and frequency of switches between *NSP* and *CAP* modes. For duty cycle d, switching frequency f_{switch} and energy per switch E_{switch}, average system power can be modeled as:

$$P_{avg} = d \times P_{NSP}^{system} + (1 - d) \times P_{CAP}^{system} + f_{switch} \times E_{switch} \tag{4}$$

In minimizing average power, there is a notable tradeoff between the duty cycle and the frequency of switches. Spending more time in CAP mode allows the sensor to cool down, increasing the length of time spent in NSP mode as well. On the other hand, spending less time in CAP mode allows the sensor to spend a greater proportion of time in NSP mode, promoting energy savings through the duty cycle, at the expense of number of switches. Notably, time spent in each mode must be a multiple of time spent capturing an image. It is not possible to switch to CAP mode for a partial frame duration while an image is being captured. For our implementation, which has minimal switching overhead, higher duty-cycles tend to provide favorable average system power profiles.

4.2. Stop-Capture-Go

The traditional stop-go DTM technique regulates processor temperature by halting execution through clock gating. For near-sensor processing, we can similarly put the sensor in *CAP* mode, gating near-sensor units for some time before resuming *NSP* mode. The resulting "temporal slack" allows the sensor to regulate capture fidelity at the expense of task performance. Stop-go is architecturally simple, requiring only the ability to gate the clock or power of components.

Unlike traditional stop-go, our proposed stop-capture-go requires unique modifications for near-sensor processing. First, frequently clock gating the entire sensor is not advisable; interruptions to the camera pipeline create capture delays on the order of multiples of frames. Instead, the system will clock gate the near-sensor VPU and DRAM, putting the sensor into CAP mode. Second, rather than being governed by TDP, the temperature regulation will trigger as the sensor reaches a situational upper bound specified by the design principles, such that $T_{sensor} < T_{vision}$ and $T_{sensor} < T_{imaging} + T_{jump}^{latency}$. Third, the application can request an execution halt to achieve on-demand fidelity. For this, the sensor enters CAP mode to retrieve the frame.

The amount of "stop" time—the amount of time the processor is halted—is an important policy parameter. During the stop time, the system will "drop" frames, failing to process them, although they may be captured. Elongated stop times allow a sensor to cool down, reducing the number of switches. Stop times can be detrimental, as contiguously dropped frames may contain important ephemeral visual information. Thus, if a system wishes to prioritize a continuity of visual information, stop time should be reduced. In our simulated study, we find that the minimal stop time of 33 ms (one frame time) is sufficient to cool the sensor from 87 to 74 °C, enabling sufficient temperature regulation and on-demand fidelity.

Due to the architectural simplicity of stop-capture-go, system overhead is minimal, promoting continuously low system power. However, frequent frame drops impair visual task performance. Thus, stop-capture-go is suitable for systems that demand low power but are not performance-critical and/or systems that require minimal architecture modifications.

4.3. Seasonal Migration

While stop-capture-go is a simple policy for temperature regulation and high-fidelity captures, it degrades application performance by halting execution. Towards minimizing performance loss, we investigate seasonal migration for near-sensor processing. Seasonal migration shifts the processing to a thermally isolated computational unit, allowing continuous computing. As we model in Section 3, spatial thermal isolation between the sensor and SoC allows thermal relief. Enabling seasonal migration comes at the expense of duplicated computational units near and far from the sensor, but effectively regulates temperature without sacrificing performance.

As shown in Figure 6, seasonal migration is governed by two temperature boundaries: T_{high} and T_{low}. In the efficiency phase, triggered when the sensor reaches a temperature below T_{low}, it will enter NSP mode, performing near-sensor processing for system efficiency. In the cooling phase, triggered when the sensor reaches a temperature above T_{high}, it will enter CAP mode, performing off-sensor processing on the SoC, allowing the sensor to cool down. The alternation between phases allows the system to balance efficiency with temperature. For on-demand fidelity, the system enters the cooling phase regardless of sensor temperature.

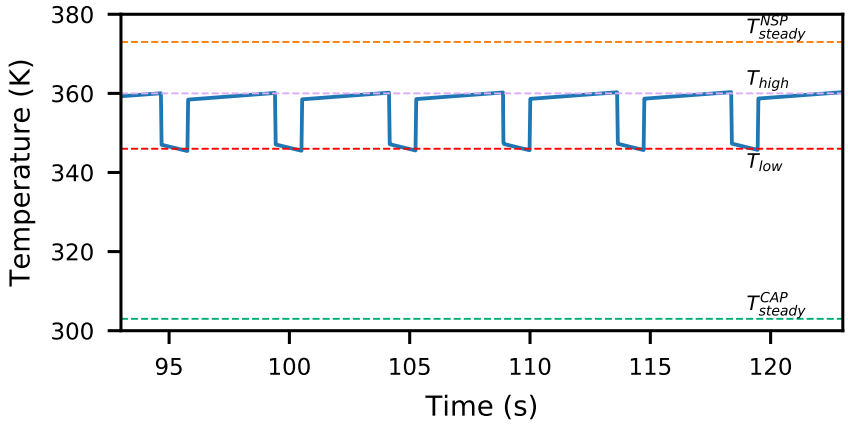


Figure 6. Transient response of seasonal migration mechanism with 77% duty cycle to confine sensor temperature within thermal boundaries (T_{high} and T_{low}).

T_{high} and T_{low} are important policy parameters, controlling the balance of efficiency and temperature. T_{high} forces sensor temperature regulation, and thus should be set to shift to situational needs:

$$T_{high} = min(T_{vision}, T_{imaging} + T^{latency}_{jump})$$

Meanwhile, the gap between T_{high} and T_{low} controls the system efficiency implications of the policy. As a result that it takes more time for the sensor temperature to bridge a larger gap, larger gaps decrease the frequency of switches, while smaller gaps increase the frequency of switches. The $T_{high} - T_{low}$ gap also controls the duty cycle of the system. When the desired sensor temperature range is closer to steady-state NSP temperature than steady-state CAP temperature, smaller gaps produce favorable duty cycles, spending more time in NSP mode. As shown in Equation (4), the average system power is a function of this duty cycle, balanced against the energy overhead and frequency of switches. Thus, T_{low} should be chosen to create a gap that optimizes average system power.

As we defined earlier, the duty cycle is the proportion of time spent in NSP mode. For seasonal migration, the relationships can be derived from standard charging models. After the rapid drop or rise in temperature T_{jump}, which takes approximately t_{jump} amount of time, the sensor follows an RC charging curve towards the steady-state temperature of the

NSP or CAP mode. Altogether, this can be used to model duty cycle d and frequency of migration $f_{migration}$.

$$t_{warming} = RC \times \ln\left(\frac{T^{NSP}_{steady} - (T_{low} + T_{jump})}{T^{NSP}_{steady} - T_{high}}\right) + t_{jump}$$

$$t_{cooling} = RC \times \ln\left(\frac{(T_{high} - T_{jump}) - T^{CAP}_{steady}}{T_{low} - T^{CAP}_{steady} - T_{jump}}\right) + t_{jump}$$

$$d = t_{warming}/(t_{warming} + t_{cooling})$$

$$f_{migration} = 2/(t_{warming} + t_{cooling})$$

Depending on implementation, seasonal migration could suffer from the switching latency and energy overhead resulting from state transfer and synchronization in shifting processing among computational units. However, reducing this migration overhead is a well-studied problem in distributed systems [49]. Several reported techniques mitigate migration latency, e.g., pre-copy-based migration [50], which promote smooth execution performance while incurring energy overhead by keeping both computational units on while preparing for migration. Similarly, in our implementation, prior to migration, we prepare the system by preemptively starting up the target computational unit and initiating its context so it is prepared for execution. Consequentially, there is only a minimal switching overhead of 100 µs, which is negligible in comparison to ms-scale image capture times.

5. Experimental Methodology

Since there are no readily available programmable 3D stacked image sensors, we use emulation techniques to implement Stagioni mechanisms. Our emulation framework operates on characterized energy, noise, and thermal models and reports system metrics such as system power and performance. We implement Stagioni as a runtime controller and integrate it into the emulation setup to study execution patterns of different policies.

5.1. Emulated Architecture

We model a 3D stacked sensor architecture in our emulation framework. For its sensing element, we emulate the fidelity characteristics of an AR0330 [47] which is a typical mobile-class image sensor with sufficient number of pixels for providing high-quality images. For its storage element, we emulate the power profile of a 4 Gb LPDDR4 DRAM, which is commonly seen in commercial 3D stacked sensors [18] for slow-motion video capture. Finally, for its processing element, we emulate the power characteristics of a Myriad2, a vision co-processor found in mobile devices [51,52], capable of neural network processing and feature-based processing, and also Neurostream [4], another recent candidate architecture for energy-efficient vision processing. In our emulation, the resulting stacked sensor connects to an ARM-based mobile-class SoC through a standard 5 Gbps CSI interface. We assume that the SoC also contains a vision co-processor, i.e., Myriad2/Neurostream, to which it can offload the tasks.

5.2. Emulation Setup

While we can use our modeling to evaluate thermal and energy behavior, the runtime behavior of Stagioni and its adaptiveness to different ambient conditions can only be assessed through hardware and software implementation. While we could study the mechanisms on any mobile platform such as SnapDragon and TX2, we choose an FPGA platform because it has programmable fabric where we can synthesize a neural processing unit to emulate a vision co-processor. To this end, we build an FPGA-based emulation platform based off two ZCU102 boards. One of them emulates the stacked sensor, while the other emulates the SoC. Both employ hardware-accelerated vision processing through the

CHaiDNN library [53]. We use 1 Gbps Ethernet for communication, simulating a standard CSI interface that has similar bandwidth characteristics.

The Stagioni controller takes the type of policy and associated model parameters as inputs. The parameters generate a temperature-dependent mode schedule that governs task execution at runtime. The controller also handles high fidelity requests and services them to deliver high quality images through appropriate mechanisms. During CAP mode for stop-capture-go, the controller gates the execution of the neural network invocation. For seasonal migration, the controller performs message passing over Ethernet for state transfer and implements producer-consumer queues for synchronization. During NSP mode, the controller gates the SoC FPGA.

5.3. Workloads

We evaluate the life-logger use case which performs continuous vision with occasional imaging upon detection of interesting events. For our vision tasks, we study two forms of vision: (i) image classification, identifying scenes, and (ii) object detection, locating objects in a scene. For each of these tasks, as shown in Table 2, we choose a variety of the state-of-the-art DNN models with different input, memory, and computational requirements to stress different elements in our stacked architecture.

We use SNR to gauge image quality and frame drops for performance overhead. In addition to stop-capture-go and seasonal migration, we consider full-far sensor processing (status quo) for comparison.

In imaging, an SNR of 20 dB is considered as acceptable quality under well-lit conditions. However, the bar is higher for more challenging conditions, including environments where fiducial detail is important. This can be seen in sensor data sheets [47] where manufacturers design cameras to deliver higher SNR values, e.g., 35 dB for excellent performance under low-light conditions. Therefore, to capture all real-world scenarios, we use range of fidelity choices {35 dB, 26 dB, 20 dB}, and a "don't care" scenario in which the application continuously performs vision without any on-demand high fidelity imaging.

We evaluate a wide range of temperature and lighting conditions. For evaluating ambient temperature effects, we use values from 20 °C to 40 °C. Meanwhile, lighting translates into different camera settings, i.e., exposure and ISO. We use the flexible CapraRawCamera [54] camera app to automatically determine appropriate camera settings based on the scene lighting. We use the following camera settings for three sensor illuminations. (a) Outdoor daylight (32,000 lux): Exp.: 16 ms, ISO: 100. (b) Indoor office light (320 lux): Exp.: 32 ms, ISO: 400. (c) Dimly lit office light (3.2 lux): Exp.: 64 ms, ISO: 800.

6. Evaluation Results and Analysis

In this section, we investigate the effectiveness of our proposed policies in meeting fidelity demands of various vision tasks around the life-logger use case.

6.1. Duty Cycle

Stagioni determines optimal duty cycles, based on power profile and imaging fidelity requirements. Figure 7 shows a sensitivity analysis of duty cycle for a range of power dissipations across different fidelity needs. This range of power profile can represent different executions on Myriad2-like architecture or on ASIC, GPU, and FPGA architectures with different performance expectations. We can see that the duty cycle varies widely, due to the strong interplay between the fidelity and the power profile. While the application's power profile determines the steady-state temperature, the fidelity requirement determines the placement of thermal boundaries in the temperature trace. This can result in a broad range of duty cycle based on where the thermal boundaries are situated in the temperature response, which we explain below.

If thermal boundaries are placed above the temperature response, Stagioni operates at 100% duty cycle, i.e., in NSP mode all the time. This is relevant as the steady-state temperatures of an application power profile are below thermal limits, e.g., $<$1 W On the

other hand, the boundary placement within the gradual rise and steeper fall region of the temperature trace means that the system spends more time in *NSP* mode than *CAP* mode, resulting in duty cycles greater than 50%. If the boundaries lie in the steeper rise and gradual fall region, this time, the system spends more time in *CAP* mode than *NSP* mode, thereby leading to duty cycles <50%.

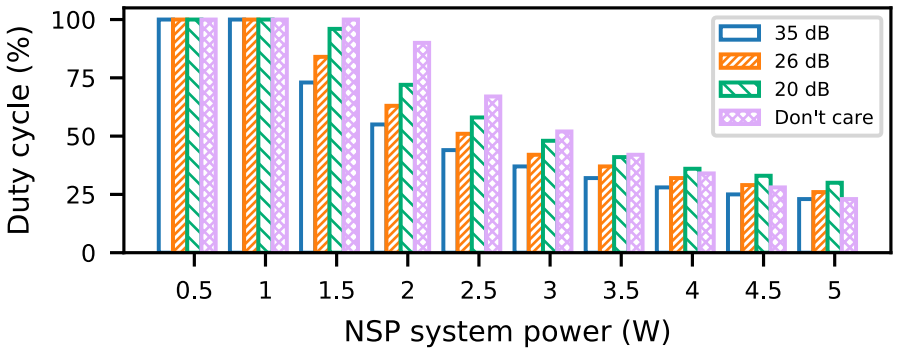
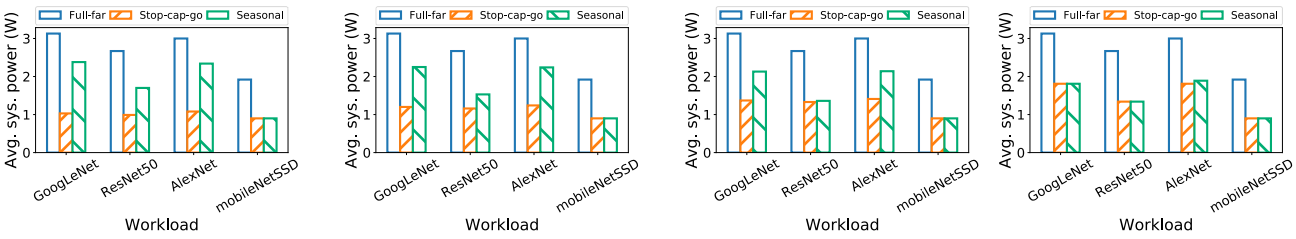


Figure 7. Influence of fidelity and NSP power on duty cycle.

6.2. System Power Consumption

Here, we examine system power during emulated workloads. We find that stop-capture-go and seasonal migration substantially reduce system power compared to the status quo. Figure 8 shows the system power for different applications for different policies, across different fidelity needs. We see that stop-capture-go consumes the lowest amount of power among all the policies. This is because stop-capture-go operates entirely on the near-sensor VPU for whole program execution in both *NSP* and *CAP* modes. In contrast, seasonal migration operates on far-sensor VPU during *CAP* mode and on near-sensor VPU during *NSP* mode. Thus, it consumes more power than stop-capture-go but less than full-far policy.



(**a**) 35 dB (**b**) 26 dB (**c**) 20 dB (**d**) Don't care

Figure 8. For seasonal migration, raised duty cycles decrease system power, while it increases the system power for stop-capture-go.

System power changes with fidelity demands due to change in duty cycle; high fidelity pulls down the duty cycle, reducing efficiency. This is evident in seasonal migration; we see higher power for higher app fidelities (higher SNR). For stop-capture-go, a lower duty cycle increases VPU sleep time, while dropping frames from processing. Therefore, we see power decrease as we go from low to high app fidelity. For full-far policy, there is no change in system power, as it does not create fidelity issues.

6.3. Overhead

We discuss policy execution overhead for seasonal migration and stop-capture-go policies. As the system executes seasonal migration, it switches between near-sensor and far-sensor VPUs. However, through the use of practically available techniques, task offload overhead can be kept to a minimum. We use one such technique called pre-copy migration

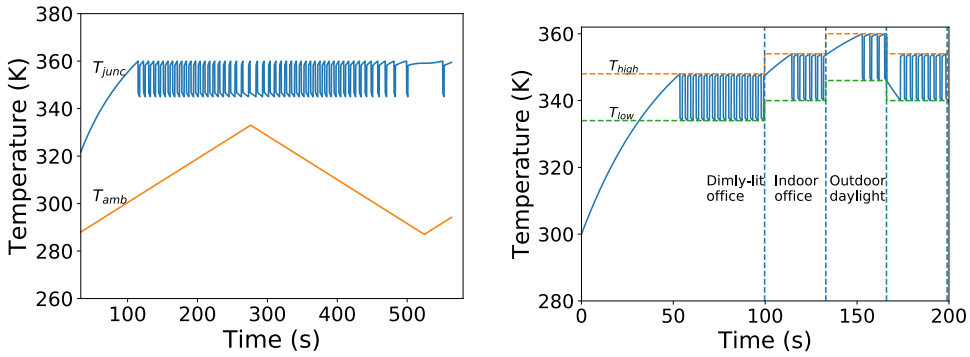
which preemptively transfers the state before the migration deadline. Consequentially, one needs to only take care of synchronization between the VPU and SoC, which involves only basic handshaking operations incurring minimal overhead. We measure this switching overhead on our emulation setup to be 100 µs, which is negligible in the context of the frame capture time, i.e., 33 ms. Ergo, seasonal migration has no effect on the performance of the vision application.

For stop-capture-go, stop time determines the number of frame drops which could potentially lead to performance hit. The actual performance loss depends on the duty cycle and effective frame rate when the system executes stop-capture-go will be scaled by a factor of the duty cycle which has interesting implications. If the duty cycle is very high, then the performance loss would be minimal, e.g., 30 fps with 98% duty cycle leads to an effective 29.4 fps. On the other hand, a lower duty cycle can lead to a substantial performance loss, e.g., 30 fps with 40% duty cycle leads to 12 fps, which is a reduction by more than 50%. Therefore, even though stop-capture-go consumes the lowest system power, it can hit the performance of the vision application when near sensor power and/or fidelity requirements are high.

6.4. Situational Awareness

One feature of Stagioni that differs from traditional DTM techniques is situational awareness to dynamic settings. Stagioni smoothly adapts thermal boundaries to match ambient temperature and lighting situations.

Ambient temperature determines steady-state temperatures, which determine warming and cooling times. Higher ambient temperatures push T^{NSP}_{steady} far from T_{low} and push T^{CAP}_{steady} close to T_{high}. This forces the warming phase to take a steeper rise and the cooling phase to take a gradual fall in the exponential curve. Thus, increasing ambient temperature decreases duty cycle and vice-versa. In simulation, decreasing ambient temperature increases rise times and reduces fall times in the simulated temperature trace, as shown in Figure 9a.



(**a**) Adaptive to temperature (**b**) Adaptive to lighting

Figure 9. Increasing ambient temperature (**a**) and/or decreasing ambient illumination (**b**) pulls T^{NSP}_{steady} away from T_{low} and pushes T^{CAP}_{steady} close to T_{high}. Stagioni shifts thermal boundaries to smoothly adapt to different ambient conditions.

Lighting dictates fidelity requirements, changing T_{high} and T_{low}. Stagioni adapts to these changes. We simulate change in illumination to generate a trace with random juggling between lighting scenarios. We provide this trace as input to our runtime and collect the temperature trace. Figure 9b shows the temperature trace overlaid with T_{high} and T_{low}. We can observe the smooth variation of temperature with light intensity.

7. Conclusions

Near-sensor processing can unlock energy-efficient imaging and vision, as demonstrated by recent academic and industrial efforts. However, we show that doing so hampers

sensor fidelity due to thermal noise, thereby limiting the adoption of near-sensor processing. Our characterization reveals that immediate drop in temperature can be realized within a short duration. We use this observation to design principles for managing sensor temperature for efficient temperature regulation and high fidelity temperatures, while optimizing for system power. To implement the policies, we design and implement the Stagioni runtime to manage sensor temperature, while fulfilling imaging needs. For our evaluated tasks, the policies deliver system power savings ranging from 22–53% with little performance loss. Stagioni also quickly and smoothly adapts the thermal boundaries based on ambient conditions. Our work is the first runtime solution for stacked sensor thermal management. We foresee our work as early steps to imaging-aware DTM techniques.

Author Contributions: Conceptualization, V.K.; Data curation, B.J.; Investigation, V.K. and R.L.; Project administration, V.K.; Software, S.K.; Validation, V.K.; Writing—original draft, V.K.; Writing—review and editing, C.-J.W. and R.L. All authors have read and agreed to the published version of the manuscript.

Funding: This material is based upon work supported by the National Science Foundation under Grant No. 1657602.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors thank Linda Nguyen for her help with the figures and Microsemi for their generous support through SmartFusion2 hardware kit and software licenses.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Enhancing Facebook's AR Platform with Target Tracking Capabilities. Available online: https://research.fb.com/enhancing-facebooks-ar-platform-with-target-tracking-capabilities/ (accessed on 29 January 2021).
2. Microsoft Hololens. Available online: https://www.microsoft.com/en-us/research/blog/ideas-blossom-for-using-microsoft-hololens/ (accessed on 29 January 2021).
3. End-to-End Deep Learning for Self-Driving Cars. Available online: https://devblogs.nvidia.com/deep-learning-self-driving-cars/ (accessed on 29 January 2021).
4. Azarkhish, E.; Rossi, D.; Loi, I.; Benini, L. Neurostream: Scalable and energy efficient deep learning with smart memory cubes. *IEEE Trans. Parallel Distrib. Syst.* **2018**, *29*, 420–434. [CrossRef]
5. Pena, D.; Forembski, A.; Xu, X.; Moloney, D. Benchmarking of CNNs for low-cost, low-power robotics applications. In Proceedings of the RSS 2017 Workshop: New Frontier for Deep Learning in Robotics, Boston, MA, USA, 15 July 2017.
6. History of 3D Stacked Image Sensors. Available online: http://www.3dic.org/3D_stacked_image_sensor (accessed on 7 August 2018).
7. LiKamWa, R.; Hou, Y.; Gao, J.; Polansky, M.; Zhong, L. RedEye: Analog ConvNet image sensor architecture for continuous mobile vision. *ACM SIGARCH Comput. Archit. News* **2016**, *44*, 255–266. [CrossRef]
8. Du, Z.; Fasthuber, R.; Chen, T.; Ienne, P.; Li, L.; Luo, T.; Feng, X.; Chen, Y.; Temam, O. ShiDianNao: Shifting vision processing closer to the sensor. In Proceedings of the 42nd Annual International Symposium on Computer Architecture, Portland, OR, USA, 13–17 June 2015.
9. Dodge, S.; Karam, L. Understanding how image quality affects deep neural networks. In Proceedings of the 2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX), Lisbon, Portugal, 6–8 June 2016.
10. Skadron, K.; Stan, M.R.; Sankaranarayanan, K.; Huang, W.; Velusamy, S.; Tarjan, D. Temperature-aware microarchitecture: Modeling and implementation. *ACM Trans. Archit. Code Optim.* **2004**, *1*, 94–125. [CrossRef]
11. Kumar, A.; Shang, L.; Peh, L.S.; Jha, N.K. System-level dynamic thermal management for high-performance microprocessors. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2008**, *27*, 96–108. [CrossRef]
12. Donald, J.; Martonosi, M. Techniques for multicore thermal management: Classification and new exploration. *ACM SIGARCH Comput. Archit. News* **2006**, *34*, 78–88. [CrossRef]
13. Isci, C.; Buyuktosunoglu, A.; Cher, C.Y.; Bose, P.; Martonosi, M. An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget. In Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture, Orlando, FL, USA, 9–13 December 2006.
14. Gomaa, M.; Powell, M.D.; Vijaykumar, T. Heat-and-run: Leveraging SMT and CMP to manage power density through the operating system. *ACM Sigplan Not.* **2004**, *39*, 260–270 [CrossRef]
15. Amir, M.F.; Ko, J.H.; Na, T.; Kim, D.; Mukhopadhyay, S. 3-D Stacked Image Sensor With Deep Neural Network Computation. *IEEE Sens. J.* **2018**, *18*, 4187–4199. [CrossRef]

16. Forchheimer, R.; Astrom, A. Near-sensor image processing: A new paradigm. *IEEE Trans. Image Process.* **1994**, *3*, 736–746. [CrossRef] [PubMed]
17. Shi, Y.; Lichman, S. *Smart Cameras: A Review*; Citeseer Survey: Gothenburg, Sweden, 2005.
18. Haruta, T.; Nakajima, T.; Hashizume, J.; Umebayashi, T.; Takahashi, H.; Taniguchi, K.; Kuroda, M.; Sumihiro, H.; Enoki, K.; Yamasaki, T.; et al. A 1/2.3 inch 20Mpixel 3-layer stacked CMOS Image Sensor with DRAM. In Proceedings of the 2017 IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, CA, USA, 5–9 February 2017.
19. Tech Insights. Samsung Galaxy S9 Camera Teardown. Available online: http://techinsights.com/about-techinsights/overview/blog/samsung-galaxy-s9-camera-teardown (accessed on 29 January 2021).
20. Kumagai, O.; Niwa, A.; Hanzawa, K.; Kato, H.; Futami, S.; Ohyama, T.; Imoto, T.; Nakamizo, M.; Murakami, H.; Nishino, T.; et al. A 1/4-inch 3.9 Mpixel low-power event-driven back-illuminated stacked CMOS image sensor. In Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, CA, USA, 11–15 February 2018.
21. Sony. CIS with Embedded AI Processor. Available online: https://www.sony.net/SonyInfo/News/Press/202005/20-037E/ (accessed on 29 January 2021).
22. Lowe, D.G. *Object Recognition from Local Scale-Invariant Features*; ICCV: Vienna, Austria, 1999.
23. Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; Darrell, T. Caffe: Convolutional architecture for fast feature embedding. In Proceedings of the 22nd ACM international conference on Multimedia, Mountain View, CA, USA, 18–19 June 2014.
24. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. *Tensorflow: A System for Large-Scale Machine Learning*; OSDI: Boulder, CO, USA, 2016.
25. BVLC. Caffe Performance Measurements on NVIDIA GPUs. Available online: http://tutorial.caffe.berkeleyvision.org/performance_hardware.html (accessed on 29 January 2021).
26. Pham, P.H.; Jelaca, D.; Farabet, C.; Martini, B.; LeCun, Y.; Culurciello, E. NeuFlow: Dataflow vision processing system-on-a-chip. In Proceedings of the 2012 IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS), Boise, Idaho, 5–8 August 2012.
27. Cavigelli, L.; Magno, M.; Benini, L. Accelerating real-time embedded scene labeling with convolutional networks. In Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, 8–12 June 2015.
28. Zhang, C.; Li, P.; Sun, G.; Guan, Y.; Xiao, B.; Cong, J. Optimizing fpga-based accelerator design for deep convolutional neural networks. In Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, 22–24 February 2015.
29. Zhang, C.; Fang, Z.; Zhou, P.; Pan, P.; Cong, J. Caffeine: towards uniformed representation and acceleration for deep convolutional neural networks. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2018**, *38*, 2072–2085. [CrossRef]
30. Chen, Y.H.; Emer, J.; Sze, V. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. *ACM SIGARCH Comput. Archit. News* **2016**, *44*, 367–379. [CrossRef]
31. Han, S.; Liu, X.; Mao, H.; Pu, J.; Pedram, A.; Horowitz, M.A.; Dally, W.J. EIE: Efficient inference engine on compressed deep neural network. *ACM SIGARCH Comput. Archit. News* **2016**, *44*, 243–254. [CrossRef]
32. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
33. Levoy, M. Noise and ISO. Available online: https://graphics.stanford.edu/courses/cs178/lectures/noise-29apr14.pdf (accessed on 29 January 2021).
34. Brooks, D.; Martonosi, M. Dynamic thermal management for high-performance microprocessors. In Proceedings of the Seventh International Symposium on High-Performance Computer Architecture, Nuevo Leone, Mexico, 20–24 January 2001.
35. Lie, D.; Chae, K.; Mukhopadhyay, S. Analysis of the performance, power, and noise characteristics of a cmos image sensor with 3-d integrated image compression unit. *IEEE Trans. Compon. Packag. Manuf. Technol.* **2014**, *4*, 198–208. [CrossRef]
36. LiKamWa, R.; Priyantha, B.; Philipose, M.; Zhong, L.; Bahl, P. Energy characterization and optimization of image sensing toward continuous mobile vision. In Proceedings of the 11th Annual International Conference on Mobile Systems, Applications, and Services, Taipei, Taiwan, 25–28 June 2013.
37. Choi, J.; Park, S.; Cho, J.; Yoon, E. An energy/illumination-adaptive CMOS image sensor with reconfigurable modes of operations. *IEEE J. Solid State Circuits* **2015**, *50*, 1438–1450. [CrossRef]
38. Micron Technologies. Power Calculators. Available online: https://www.micron.com/support/tools-and-utilities/power-calc (accessed on 29 January 2021).
39. Xilinx. Xilinx Power Estimator. Available online: https://www.xilinx.com/products/technology/power/xpe.html (accessed on 29 January 2021).
40. Borkar, S. Design challenges of technology scaling. *IEEE Micro* **1999**, *19*, 23–29. [CrossRef]
41. Skadron, K.; Abdelzaher, T.; Stan, M.R. Control-theoretic techniques and thermal-RC modeling for accurate and localized dynamic thermal management. In Proceedings of the Eighth International Symposium on High Performance Computer Architecture, Boston, MA, USA, 2–6 February 2002.
42. Yu, Y.J.; Wu, C.J. Designing a Temperature Model to Understand the Thermal Challenges of Portable Computing Platforms. In Proceedings of the 2018 17th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm), San Diego, CA, USA, 29 May–1 June 2018.

43. Heo, S.; Barr, K.; Asanović, K. Reducing power density through activity migration. In Proceedings of the 2003 International Symposium on Low Power Electronics and Design, Seoul, Korea, 25–27 August 2003.
44. OnSemi. Python 1300 Datasheet. Available online: https://www.onsemi.com/pub/Collateral/NOIP1SN1300A-D.PDF (accessed on 29 January 2021).
45. Qualcomm. Snapdragon 636 Mobile Platform. Available online: https://www.qualcomm.com/products/snapdragon-636-mobile-platform (accessed on 29 January 2021).
46. Qualcomm. Snapdragon Profiler. Available online: https://developer.qualcomm.com/software/snapdragon-profiler (accessed on 29 January 2021).
47. OnSemi. AR0330 Image Sensor Datasheet. Available online: https://www.onsemi.com/pub/Collateral/AR0330CM-D.PDF (accessed on 10 August 2018).
48. Microsemi. Imaging and Video Solution. Available online: https://www.microsemi.com/products/fpga-soc/imaging (accessed on 29 January 2021).
49. Milojičić, D.S.; Douglis, F.; Paindaveine, Y.; Wheeler, R.; Zhou, S. Process migration. *ACM Comput. Surv.* **2000**, *32*, 241–299. [CrossRef]
50. Richmond, M.; Hitchens, M. A new process migration algorithm. *ACM SIGOPS Oper. Syst. Rev.* **1997**, *31*, 31–42. [CrossRef]
51. Google. AIY Vision Kit. Available online: https://aiyprojects.withgoogle.com/vision/ (accessed on 29 January 2021).
52. Intel. Intel Neural Compute Stick. Available online: https://software.intel.com/en-us/neural-compute-stick/get-started (accessed on 29 January 2021).
53. Xilinx. HLS Based Deep Neural Network Accelerator Library for Xilinx Ultrascale+ MPSoCs. Available online: https://github.com/Xilinx/CHaiDNN (accessed on 29 January 2021).
54. Po-Hsun, S. CapraRawCamera: Android App for Collecting Raw Photos for Computer Vision. Available online: https://github.com/cucapra/CapraRawCamera (accessed on 29 January 2021).