Intelligent Scene Caching to Improve Accuracy for Energy-Constrained Embedded Vision

Benjamin Simpson
University of Michigan
bssimps@umich.edu

Ekdeep Lubana
University of Michigan
eslubana@umich.edu

Yuchen Liu
University of Michigan
lyuchen@umich.edu

Robert Dick
University of Michigan
dickrp@umich.edu

Abstract

We describe an efficient method of improving the performance of vision algorithms operating on video streams by reducing the amount of data captured and transferred from image sensors to analysis servers in a data-aware manner. The key concept is to combine guided, highly heterogeneous sampling with an intelligent Scene Cache. This enables the system to adapt to spatial and temporal patterns in the scene, thus reducing redundant data capture and processing. A software prototype of our framework running on a general-purpose embedded processor enables superior object detection accuracy (by 56%) at similar energy consumption (slight improvement of 4%) compared to an H.264 hardware accelerator.

1. Introduction

Machine vision is a key component of many economically important technologies, with applications in the fields of smart infrastructure, autonomous vehicles, healthcare, surveillance, and more. However, its generally high energy demands limit its use in battery-powered, embedded systems. In contrast, human vision is energy efficient in similar scenarios [9].

Standard cameras sample data in a spatially and temporally constant manner, *i.e.*, with fixed frame resolutions and rates. However, the human vision system's (HVS) sampling and processing are spatially and temporally heterogeneous. The photoreceptive structures of the HVS react to changes in light intensity, producing output signal spikes when a threshold is exceeded. Since neuron energy consumption is correlated with signaling [2], this temporal method improves energy efficiency. In the HVS, photoreceptors are arranged heterogeneously, with their density varying from $50\,\mu m^{-2}$ at the center to only $1.6\,\mu m^{-2}$ on the periphery. This reduces energy consumption relative to uniform, high-resolution sampling, but requires guidance based on end goals and scene content.

Modern image sensors support operations for both subsampling and region-specific sampling, supporting heterogeneous spatial capture [14]. We describe a sensing method that combines heterogeneous sampling with an intelligent cache of prior samples. Comparing low-resolution captures with the cache identifies changes/moving regions, allowing the system to modulate spatial capture. This method allows standard machine vision systems to use guided sampling like the HVS, a process we name Scene Caching.

Scene Caching is compatible with any standard machine vision algorithm. Since its updates are change-based, it will be most efficient if few pixels change, making it well suited for use with stationary cameras. We thus evaluate the Scene Cache for object detection/surveillance tasks. We quantify accuracy using a variant of mean average precision (mAP) [5] that we call weighted mAP (WmAP) and compare energy consumption with industry-standard H.264 compression.

This paper makes the following contributions: 1) it demonstrates the design of a Scene Cache that uses heterogeneous spatial capture, temporal buffering, and multi-round capture to allow machine vision systems to use guided sampling to improve accuracy without energy penalties; 2) it presents a new video object detection dataset; and 3) it describes an energy model for embedded machine vision systems. The model is calibrated to a Raspberry Pi 3B+ and accounts for wireless transmission and H.264 video encoding energy, but can be easily adapted to another embedded system. The Scene Cache fits seamlessly into existing pipelines and permits dynamic trade-offs between data rate and image quality. In comparison to hardware-based H.264 compression, the software-based Scene Cache notably im-

proves object detection by 55.7% (as measured by WmAP), while reducing energy consumption by 3.6%.

2. Related Work

Reduced Camera Data Transfer: Lubana and Dick [14] previously described heterogeneous sampling mechanisms for single frame captures. Their work uses row/column decoders in image sensors to capture arbitrary rectangular regions of chosen resolution in a static scene. By using a multi-round heterogeneous capture–analysis process in single frame scenarios, the authors reduced energy consumption by 5× with only 0.65% loss in accuracy. However, their work only analyzes single-frame captures. Modeling a system capturing videos significantly differs from single frame captures because the idle time of the system and transfer time of the video dominate time and energy consumption. This change in energy-accuracy tradeoff and the opportunities opened by temporal sequences differentiate the single frame and video variations of this problem.

Event Cameras: Event cameras are specialized image sensors that sample a spatial signal in response to change in intensity. This method was inspired by the human visual system and naturally avoids temporally redundant sampling. Event cameras are rarely used and the tools and algorithms for processing their output stream are not directly compatible with regular video. Converting their output stream to work with standard machine vision algorithms reduces the energy savings offered by event-based sensing [1].

Reduced Application Algorithm Computation: Euphrates is a low-energy hardware/software co-designed system for object detection [15]. It uses custom hardware to estimate the motion of regions of interest, thus reducing the number of frames requiring expensive object detection. Euphrates reduces application-level analysis energy by 66% with 1% reduction in detection accuracy. In contrast, our work does not require hardware augmentation because the required operations are already supported by the camera pipeline.

Video Compression: Video compression algorithms exploit temporal correlation within a signal for efficient video encoding. Our work shares that philosophy. However, by focusing on inference, rather than video aesthetics and reconstruction, we are able to achieve higher application performance, while consuming similar amounts of energy. For example, our framework improves object detection by 55.7% and reduces energy consumption by 3.6% relative to a hardware-based H.264 compression baseline (see Section 6).

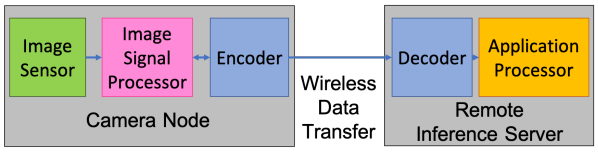

Figure 1: Scene Cache in image signal processing pipeline.

3. Scene Cache Design

Figure 1 shows a typical image processing pipeline for server-based object detection from wireless cameras. The pipeline consists of a camera that wirelessly transmits video to a server node, which performs object detection. The camera node captures raw data and converts it to a standard video format using the image signal processor. An on-device encoder compresses the video, which is then transmitted to the server. The server decoder uncompresses the video before running object detection algorithms on its application processor. The Scene Cache functions as an encoder-decoder pair. On the camera, it detects change at each frame and determines the regions to transmit to the remote inference server. On the server, the cache reconstructs the transmitted data into a high resolution image.

3.1. Scene Cache Implementation

The Scene Cache detects and transmits change using two cached images: a low-resolution image at the camera node for computing which regions to sample and transfer, and a high-resolution cache at the server, which is provided as input to the analysis algorithm. The update process has four steps which are described in detail below and visualized in Figure 2. Two parameters can be used to tune the cache's sampling behavior: the *downsample rate* and the *difference threshold*. The parameters themselves are described in the update process, while the tuning process is described in Section 6.1.

3.2. Cache Update Process

Low-resolution sampling: The Scene Cache uses a low-resolution image to estimate the amount of motion between frames. A low-resolution image can be acquired from the image sensor by skipping rows/columns or pixel binning, which is commonly availible. The amount of decimation is controlled by the *downsample rate*. A downsample rate of 2 means that each pixel of the low-resolution image represents one 2×2 square in the high-resolution image, a downsample rate of 4 means each low-resolution pixel corresponds to a 4×4 high-resolution region, etc.

Motion estimation: The difference image is the element-wise difference between the low-resolution cache from the previous frame and the low-resolution version of the current frame, taking the L^2 norm across the three color

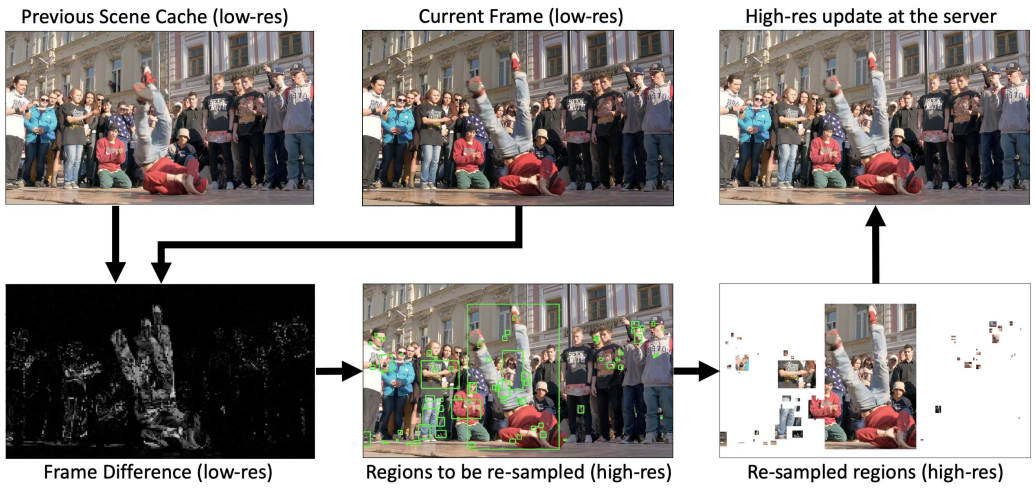

Figure 2: Scene Cache in action. Images taken from the DAVIS 2017 dataset [10, 11].

channels, and normalizing each pixel's value to be between 0 and 255. The resulting one-channel, 8-bit image has low values in the pixels where little change has occurred and high values where large change has occurred.

Difference Thresholding: The Scene Cache selects regions to be updated. All pixels of the difference image with a value above a specified *difference threshold* are re-sampled. The cache obtains the bounding box coordinates for the regions using OpenCV's `findContours()` and `boundingRect()` functions.

High-Resolution update: The regions to be updated are sampled at high resolution from the image sensor using the bounding box coordinates of the previous step. These high-resolution regions are then transmitted to the remote server to update the cache, and the low resolution version of the frame is updated at the camera node.

4. Energy Model

We evaluate the energy consumption of our model on an embedded machine vision platform. Specifically, we develop an energy model for a Raspberry Pi 3B+ augmented with a Sony IMX219 image sensor. The model presumes a pragmatic scenario where nodes collect and transmit video data wirelessly to a server for object detection as shown in Figure 1. Note that such a setup is standard for surveillance applications, which were estimated to generate more than 859 petabytes per day in 2017 [3]. While prior work had reported energy models for vision platforms, it focused on single frame capture [14]. In contrast, our model is designed for a continuous video stream and takes into account inter-frame energy consumption as well as time and energy consumption for data transmission from the platform to a server. As we show below, the particulars of video energy consumption are significantly different from single frame captures. Note that the model can be adapted to other embedded platforms by measuring or calculating the appropriate parameters (see Table 2).

The model uses a state-based methodology to estimate the energy consumed by components in the pipeline. Our baseline for comparison is the capture and transfer of a video encoded by the on-board H.264 accelerator. We also include results for uncompressed video transfer. All videos are captured at 15 fps. Figure 3 shows the pipeline stages for both a conventional camera network and one using Scene Caching.

Power States and Idle Energy Consumption: The system's idle power consumption ($P_{sys,idle}$) is measured with the device on, but not running any processes. We model all remaining power consumptions with respect to this base idle power because we can only measure the change in power as a component goes from one state (*e.g.*, idle) to another (*e.g.*, active). This allows for easy modeling of system energy as a function of state changes and durations. In net, if the system has a set of components C and takes time T_{frame} to completely process a single frame, the system energy can be written as follows:

$$E_{net} = P_{sys,idle} \cdot T_{frame} + \sum_{i \in C} P_{\Delta i,active} \cdot T_{i,active}, \quad (1)$$

where $P_{\Delta i,active}$ is the increase in system power consumption when the i^{th} component becomes active, while $T_{i,active}$ is the time it remains active for.

Image Sensor (a, b, c): Likamwa et al. show image sensor power consumption increases quadratically with respect to frame resolution [8] and is independent of sensor exposure time (T_{exp}). Since the time consumed by reading a frame is the inverse of the frame rate (f), this results in the

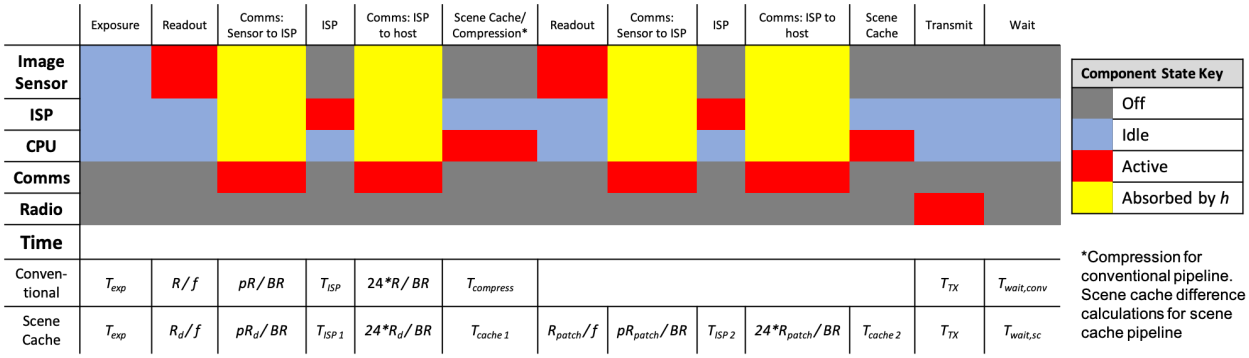

Figure 3: Pipeline timing for conventional vs. Scene Cache frameworks. For parameter and variable values, see Table 2 and Table 3, respectively.

Table 1: Energy consumed by conventional and Scene Cache frameworks. For symbol definitions, see Table 2 and Table 3.

Framework	Energy model
Conventional	$a\frac{R^2}{f} + b\frac{R}{f} + cT_{exp} + P_{sys,idle} \cdot (T_{frame}) + P_{\Delta GPU} \cdot (T_{ISP}) + (e_1 \cdot R_{compress} + e_2) + P_{\Delta TX} \cdot (T_{TX}) + P_{\Delta comm} \cdot (1+h)\left(\frac{pR}{BR} + \frac{24R}{BR}\right)$
Scene Cache	$a\left(\frac{R_d^2}{f} + \frac{R \cdot R_{patch}}{f}\right) + b\left(\frac{R_d}{f} + \frac{R_{patch}}{f}\right) + cT_{exp} + P_{sys,idle} \cdot (T_{frame}) + P_{\Delta GPU} \cdot (T_{ISP}) + P_{\Delta CPU} \cdot (T_{cache}) + P_{\Delta TX} \cdot (T_{TX}) + P_{\Delta comm} \cdot (1+h)\left(\frac{p(R_d+R_{patch})}{BR} + \frac{24(R_d+R_{patch})}{BR}\right)$

following energy model:

$$E_{sensor} = a\frac{R^2}{f} + b\frac{R}{f} + cT_{exp}. \quad (2)$$

Here a, b, and c are model coefficients specific to the embedded system used, and R is the captured frame's resolution. We use the Sony IMX219's datasheet to determine coefficients for a quadratic power model by relating the sensor's power consumption at different capture resolutions to the corresponding image sizes. The exposure time is set to a constant value.

The conventional pipeline captures frames at a fixed resolution (R). Scene Cache—being a multi-round framework—requires a frame at a downsampled resolution (R_d) to determine the region of interest and a frame of variable resolution (R_{patch}) corresponding to the patch that contains the region of interest.

Image Signal Processor (T_{ISP}, $P_{\Delta GPU}$): We use the Raspberry Pi's open source PiCamera framework to determine the amount of time spent by the image signal processor (ISP) to process the captured frame. This duration is linearly related to the frame resolution, as shown by prior work [14]. The increase in power when the ISP activates is determined by running an OpenGL program directly on the ISP. Note that the ISP, on a Raspberry Pi, is an embedded GPU and we therefore call its power consumption $P_{\Delta GPU}$. Therefore, the change in system energy if the ISP takes time T_{ISP} to process a frame is as follows:

$$E_{ISP} = P_{\Delta GPU} \cdot (T_{ISP}). \quad (3)$$

Scene Cache (T_{cache}, $P_{\Delta CPU}$): The Scene Cache runs on the CPU. The increase in CPU power consumption ($P_{\Delta CPU}$) is determined by measuring the increase in system power consumption when running a large random number generator. This power consumption is multiplied by the amount of time required by the Scene Cache operations (T_{cache}) to determine the regions of change (T_{cache1}) and update the cache (T_{cache2}). These time values are measured for each frame by running the framework on a Raspberry Pi, but can be modeled on the specific embedded platform of choice. The change in system energy if the CPU takes time $T_{cache} = T_{cache1} + T_{cache2}$ to run Scene Cache specific operations is as follows:

$$E_{cache} = P_{\Delta CPU} \cdot (T_{cache}). \quad (4)$$

Compression Energy (e_1, e_2): The energy consumption for compressing frames using H.264 is determined by evaluating the energy consumption of the on-chip encoder module in the Raspberry Pi ISP at a constant frame rate and variable resolution. We relate energy consumed per frame to resolution with a linear model. The estimated model has a

Pearson's correlation of 0.99 with the actual energy, indicating high confidence. We use this linear model to calculate energy consumption in our experiments:

$$E_{compress} = (e_1 \cdot R_{compress} + e_2)\,. \quad (5)$$

Transmission Energy ($P_{\Delta TX}$, T_{TX}): To model the transmission energy, we set up an 802.11n Wi-Fi network and transmit UDP packets from one Raspberry Pi to another. We find that the transmission power consumption ($P_{\Delta TX}$) is linearly related to the number of packets transmitted per second. This is consistent with previous observations on modeling power consumption for embedded Wi-Fi transmitters [6]. We keep the transmission time per packet (T_{TX}) constant by fixing the packet size. This results in the following energy model:

$$E_{TX} = P_{\Delta TX} \cdot (T_{TX})\,. \quad (6)$$

Communication Energy ($P_{\Delta comm}$, BR): Communicating an image involves transfer of 24 bits per pixel (8 bits each for 3 channels) which were extracted from a RAW image that had p bits per pixel, where p is usually 12 or 14. The communication time for system commands (*e.g.*, wake, sleep, etc.) usually imposes a 20–50% overhead on data transfer. A variable h is used to denote this overhead. Given the bit rate (BR) of the system, the energy for communicating a frame of resolution R across the system is as follows:

$$E_{comm} = P_{\Delta comm} \cdot (1+h)\left(\frac{p.R}{BR} + \frac{24R}{BR}\right). \quad (7)$$

Net Energy Model: Table 1 summarizes the net energy models for both a conventional vision framework and our Scene Cache framework using the variables described in this section. Note that the conventional model was modified to the Scene Cache pipeline as shown in Figure 3. Table 2 describes the exact values of the parameters modeled in our framework. Table 3 summarizes these parameters.

5. Experimental Design

We simulate the client portion of the Scene Cache running on a Raspberry Pi 3 microcontroller to evaluate its impact on data transferred, energy consumption, and accuracy. The simulator logs data transfer and processing time, which are used to evaluate energy consumption of a typical vision system running the conventional (H.264) vs. the cached pipeline on batches of videos. This section describes our evaluation methodology and Section 6 reports the results.

Application Performance: We consider an object detection scenario, and measure accuracy using mean average precision (mAP) relative to a maximal data rate and resolution (*i.e.*, uncompressed) base case. mAP assigns equal weights to each class detected, regardless of number of objects per class detected. Equal weight works fine for balanced datasets (*e.g.*, an image dataset with a similar number of objects per class), but poorly for real-world video datasets, because some objects appear much more frequently than others (*e.g.*, cars are common in videos of in-

Table 2: Energy Model Parameters (p implies pixel)

Param	Value	Units	Description
a	8.27E-09	W/p	Resolution-dependent image sensor active energy coefficient
b	1.30E-01	W	Resolution-independent image sensor active energy coefficient
c	1.42E-01	W	Image sensor idle power
f	1.20E+07	Hz	Image sensor clock frequency
T_{exp}	1.66E-02	s	Camera exposure time
$P_{sys,idle}$	2.429	W	System processor baseline power
$P_{\Delta GPU}$	0.335	W	Increase in power with active GPU
$P_{\Delta CPU}$	0.211	W	Increase in power with active CPU
e_1	1.20E-08	J / p	Data-dependent H.264 energy
e_2	1.62E-02	J	Data-independent H.264 energy
h	0.25	unitless	Inter-chip transmission overhead
p	14	bits/p	Bits per RAW pixel
BR	4.00E+09	bits/s	MIPI bit rate
$P_{\Delta comm}$	4.07E-02	W	MIPI power consumption

Table 3: Energy Model Variables

Data	Units	Description
R	pixels	Full resolution image size
R_d	pixels	Downsampled image size
R_{fovea}	pixels	Foveated image size
$R_{compress}$	pixels	Compressed image size
T_{ISP}	s	Time for ISP to process pixel data
$T_{compress}$	s	Image compression time
T_{cache1}	s	Difference image calculation and bounding box generation time
T_{cache2}	s	Downsampled cache update time
T_{TX}	s	Transmission time
$P_{\Delta TX}$	W	Radio power consumption
T_{wait}	s	System idle time between frames
T_{frame}	s	Time for one video frame ($1/framerate$)

tersections). Many of these infrequent object are the result of classification noise in our automated labeling process for ground truth; thus, equal weighting creates artificially low scores that are heavily affected by noise. Therefore, we perform a weighted average across detected classes with weights based on the ground truth number of objects in each class. We call this metric Weighted mAP (WmAP).

H.264 compression is used as an evaluation baseline. It suffers from high accuracy degradation when its energy consumption is similar to that of the Scene Cache. It is important to note that the Scene Cache is implemented as a software-based solution, while the H.264 baseline uses a hardware video encoder on our test platform. Custom Scene Cache hardware may further improve our results.

Energy Consumption: As described in Section 4, we model energy consumption as a function of per-frame data transfer, computation time, and hardware-specific parameters. The energy model is used to evaluate the energy implications of using Scene Cache in a practical setup. We compare it with the H.264 compression baseline.

Datasets: We select clips from the CDW-2012 dataset [7] with no camera motion for tuning the cache. The evaluated frames range in size from 320×240 to 720×576 pixels and videos range in length from 1,099 to 2,050 frames. We use YOLOv3 [12], a state-of-the-art object detection algorithm, to determine the impact Scene Caching has on WmAP (see Section 5) in comparison to the H.264 compression baseline. The WmAP values are averaged across a given video.

We developed our own dataset for evaluating Scene Cache parameter generalization. This dataset contains surveillance footage released onto YouTube by Digital Vision Security (DVS) [4]. The footage in the dataset all come from a stationary camera pointed at a traffic intersection. It contains cars, buildings, and pedestrians. We selected three clips, each capturing the same intersection at daytime, nighttime, and during rain. There are over 100,000 frames of 480p video in total. We obtained permission from Digital Vision Security to redistribute the clips for research purposes. The dataset has been published on the web [13].

H.264 Accuracy Modeling: We estimate the video quality of the Raspberry Pi's on-board Broadcom VideoCore IV H.264 encoder by performing H.264 encoding on our datasets using FFmpeg. This software-to-software encoding is necessary to preserve the locations of the objects for object detection, but it precludes testing using the camera-to-software pipeline of the VideoCore IV encoder. To correlate the settings of the encoders, we play a source video from one of our datasets on a screen and record it into an H.264 encoded video using a Raspberry Pi Camera. We encode the same source video using FFmpeg. In both the VideoCore IV and FFmpeg, we set the Constant Rate Factor to a typical value of 23. We select the other FFmpeg parameters that produce the video with the most similar bitrate to the video produced by the VideoCore IV.

6. Experimental Results

We evaluate the Scene Cache on the CDW and DVS datasets. The CDW dataset is used for tuning the cache parameters; the DVS dataset is used to test their generality.

6.1. Cache Tuning

To tune the cache, we evaluate the detection accuracy and data transfer of different versions of the cache on the CDW dataset. Data transfer is positively correlated with Scene Cache energy consumption and has smaller fixed overhead, making it easier to compare directly to detection accuracy. We sweep four downsample rates and eighteen difference threshold values for a total of 72 different Scene Caches.

To determine which caches provide the best accuracy-energy tradeoffs we use a criterial equation to reduce the multiobjective data transfer vs. detection performance criteria to one dimension:

$$S = \alpha * (1.0 - D_x) + (1.0 - \alpha) * A_{cc}. \quad (8)$$

S is the cache score, with higher values corresponding to better caches. D_x is the ratio of data volume transferred by the Scene Cache to that transferred by a conventional image pipeline. A_{cc} is the WmAP score of the cache. $\alpha \in [0, 1]$ is a hyperparameter that trades off the relative importance of data transfer and classification accuracy. A higher value of α favors caches that reduce the amount of data transferred while a lower value of α favors caches that are more accurate.

We sweep α, optimizing S values. Table 4 shows the best cache for α from 0.1 and 0.9, their values for WmAP, percent data transfer, and energy consumption. The caches are labeled (d,t), where d is downsample rate and t is difference threshold. Results for two H.264 encoders and the conventional pipeline with full-resolution video are also shown.

By coincidence, the hardware- and software-based H.264 encoders had nearly identical energy consumption for the dataset. Both energy equations are essentially linear functions of data volume transferred. The hardware encoder has a higher energy overhead and a lower incremental energy cost per pixel of data transfer compared to the software encoder. The results for different videos in this dataset happen to lie on both sides of their intersection point, making the final average for each encoder similar to the other.

This table demonstrates that for nearly all included caches, the Scene Cache achieves better WmAP than H.264. However, as we approach the compression ratio of the H.264 encoder, we find that the Scene Cache does not have a high WmAP. Regardless, the Scene Cache has much better WmAP than H.264 for the same energy consumption in

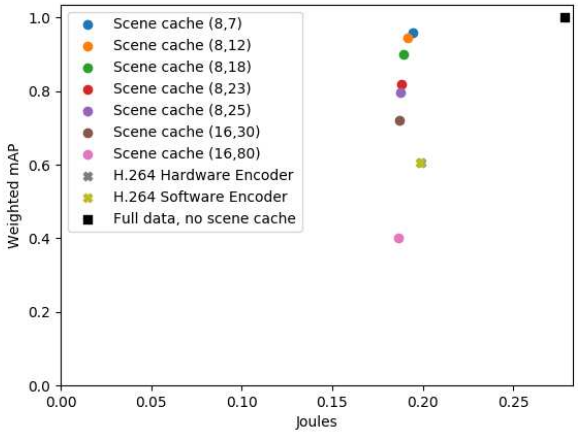

Figure 4: Cache WmAP in relation to energy consumption for CDW dataset.

almost all cases: 0.61 for the H.264 encoders; 0.95 (55.7% improvement) for Scene Cache (8, 12) (see Figure 4). Further, note that the Scene Cache dramatically reduces energy consumption in comparison to the uncompressed baseline. The trend was similar for the DVS data set (see Table 5).

Table 4: Generalized Cache Selection on CDW Dataset

α	Cache	WmAP	% Data Transfer	Energy (J)
0.1	(8,7)	0.96	39.5%	0.195
0.3	(8,12)	0.95	27.9%	0.192
0.5	(8,18)	0.90	19.5%	0.189
0.7	(8,25)	0.80	11.4%	0.188
0.9	(16,80)	0.40	1.34%	0.186
H.264 (Hardware)		0.61	1.15%	0.199
H.264 (Software)		0.61	1.15%	0.199
No Compression		1.00	100%	0.279

6.2. Cache Generalization Testing

We performed testing on the DVS dataset with caches tuned for CDW to see how well they would work on a dataset they had not been trained on. We used Scene Caches (8,12) and (8,18) since both are optimal for multiple values of α. The energy vs. accuracy results are shown in Table 5 with H.264 encoders and the uncompressed baseline results for comparison.

In this graph the two caches again achieve better WmAP than H.264, but they only achieve better energy consumption in some cases. Scene Cache (8,18) has a mAP of 0.78 (20% improvement over H.264) and uses 0.229 J/frame of energy, which is 2.0% lower than the H.264 software encoder but 8.8% higher than the H.264 hardware encoder.

Properties of the datasets influence the results. The DVS dataset has higher resolution images than CDW, on average. Therefore, software encoding requires more energy than hardware encoding. The software encoder has a lower fixed overhead but higher incremental energy cost per pixel while the converse is true for the hardware encoder (see Table 2); the hardware encoder is therefore more efficient for large images.

The large image difference does not fully explain why the hardware encoder has lower energy consumption than the Scene Caches. The CDW dataset has one test video (PETS2006) with a frame size slightly larger than DVS. When energy is analyzed for just that video, the energy consumption for the software and hardware encoders and the uncompressed baseline are very similar to DVS, but all of the Scene Caches listed in Table 4 use less energy than either of the H.264 encoders. This implies that the frame size does not explain the discrepancies between the DVS and CDW results.

Table 5: Scene Cache Performance on DVS Dataset

Cache	WmAP	Energy (J)
(8,12)	0.86	0.267
(8,18)	0.78	0.229
H.264 (Hardware)	0.65	0.211
H.264 (Software)	0.65	0.234
No Compression	1.00	0.525

A more plausible explanation is that Scene Cache performance depends on image content. The caches that performed well on CDW may not be as optimal for DVS. Retuning the caches on DVS may improve energy consumption results.

While the Scene Cache may not generalize well between the CDW and DVS datasets, preliminary Scene Cache results trained on 80% of the CDW data and tested on 20% of the CDW data showed similar energy and accuracy results to Table 4. Thus Scene Cache can be trained to work well with new data in the same application. This is a reasonable constraint for our problem domain of stationary cameras since the application is generally fixed.

6.3. Energy Breakdown

Figure 5 shows the amount of energy used by each part of the pipeline for experiments on CDW and DVS for Scene Cache (8,18). Hardware H.264 encoding and a pipeline using no compression are shown for comparison. Transmission, compression, ISP, and system idle energy consumption dominate.

The qualitative difference between the Scene Cache and H.264 encoder is a tradeoff between transmission and com-

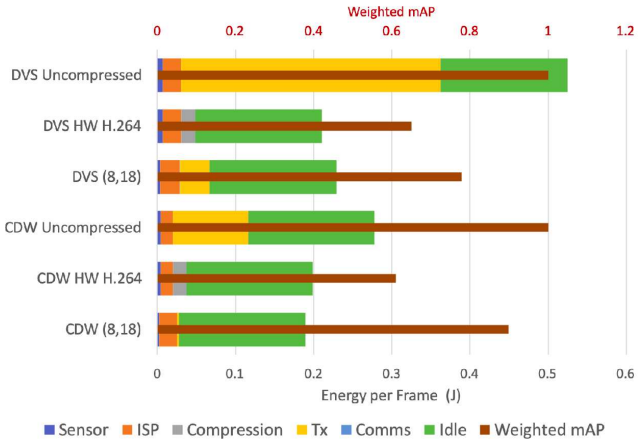

Figure 5: Energy consumption breakdown & WmAP scores.

pression energy. Since transmission energy is correlated with data volume, this implies a tradeoff between the benefit of data compression and its energy cost. Both hardware and software H.264 encoders achieve high enough compression ratios that their transmission energies are negligible. The Scene Cache imposes very little overhead and is more accurate, but does not reduce transmissions as much as H.264. It is important to note that the Scene Cache produces higher WmAP scores than H.264 at similar energy cost.

The system idle energy is constant for both the Scene Cache and the other compression algorithms, often dominating the energy cost per frame. This reduces the flexibility of system designers in making compression tradeoffs to save energy.

The Scene Cache reduces the amount of data processed by the sensor and ISP. This results in reduced sensor energy, which has little impact at the system level since the sensor already consumes little energy in the base case. The energy increase of the ISP is due to the multi-round nature of the Scene Cache incurring the data-independent energy overhead twice. This could be improved through more extensive redesign of the ISP for multi-round image sensing. Communication energy is not visible in the plot as it consumes less than 0.1% of the system energy.

7. Conclusion

We demonstrated Scene Cache, a novel, application specific compression framework for embedded machine vision systems. Without specialized hardware, it achieves significantly better object detection results than hardware-aided H.264 at similar or improved levels of energy consumption. A new energy model and video dataset aid in evaluation. When tuned on a dataset, Scene Caching improves object detection WmAP by 0.34 (55.7% improvement) while enabling slightly (3.6%) lower energy consumption than H.264. When generalized to datasets it was not tuned on, Scene Cache still improves object detection by 0.13 (20% improvement) while using similar energy to H.264 (8.8% higher than hardware and 2.0% lower than software).

Overall, the Scene Cache enables the tradeoff of system energy consumption for application performance in machine vision systems. The low computational cost of the Scene Caching leaves room for further improvement, such as making sampling decisions based on statistical importance of image regions to the machine vision task. Such improvements would truly enable goal-based sensing for highly efficient machine vision.

References

[1] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. D. Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza, J. Kusnitz, M. Debole, S. Esser, T. Delbruck, M. Flickner, and D. Modha. A low power, fully event-based gesture recognition system. In *Proc. Conf. on Computer Vision and Pattern Recognition*, pages 7388–7397, July 2017.

[2] Mireille Bélanger, Igor Allaman, and Pierre J. Magistretti. Brain energy metabolism: Focus on astrocyte-neuron metabolic cooperation. *Cell Metabolism*, 14(6):724–738, 2011.

[3] Jon Cropley. Top video surveillance trends for 2015. Technical report, IHS Technology, 2015.

[4] Digital Vision Security, Inc. Digital vision security - video camera surveillance systems & solutions. https://www.youtube.com/channel/UCSeMkpNPR7UGFOlBfZ0bRNg, 2014–2020.

[5] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes challenge 2012 (VOC2012) results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[6] L. M. Feeney and M. Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *Proc. Joint Conf. of the IEEE Computer and Communications Society*, volume 3, pages 1548–1557 vol.3, April 2001.

[7] N. Goyette, P. Jodoin, F. Porikli, J. Konrad, and P. Ishwar. Changedetection.net: A new change detection benchmark dataset. In *Proc. Conf. on Computer Vision and Pattern Recognition Workshops*, pages 1–8, June 2012.

[8] Robert LiKamWa, Bodhi Priyantha, Matthai Philipose, Lin Zhong, and Paramvir Bahl. Energy characterization and optimization of image sensing toward continuous mobile vision. In *Proc. Int. Conf. Mobile Systems, Applications, and Services*, pages 69–82, 2013.

[9] Jeremy E. Niven and Simon B. Laughlin. Energy limitation as a selective pressure on the evolution of sensory systems. *J. of Experimental Biology*, 211(11):1792–1804, 2008.

[10] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video

object segmentation. In *Proc. Conf. on Computer Vision and Pattern Recognition*, 2016.

[11] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 DAVIS challenge on video object segmentation. *arXiv:1704.00675*, 2017.

[12] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.

[13] Benjamin Simpson and Yuchen Liu. DVS dataset. http://ziyang.eecs.umich.edu/tools.html#dvsdata, 2020.

[14] E. S. Lubana and R. P. Dick. Digital Foveation: an energy-aware machine vision framework. *IEEE Trans. on Computer-Aided Design*, pages 2371–2380, Nov. 2018.

[15] Yuhao Zhu, Anand Samajdar, Matthew Mattina, and Paul N. Whatmough. Euphrates: Algorithm-SoC co-design for low-power mobile continuous vision. In *Proc. Int. Symp. Computer Architecture*, pages 547–560, 2018.