Approximate Query Service on Autonomous IoT Cameras

Mengwei Xu*
Peking University
mwx@pku.edu.cn

Xiwen Zhang
Purdue ECE
zhan2977@purdue.edu

Yunxin Liu
Microsoft Research
yunxin.liu@microsoft.com

Gang Huang
Peking University
hg@pku.edu.cn

Xuanzhe Liu†
Peking University
xzl@pku.edu.cn

Felix Xiaozhu Lin
Purdue ECE
xzl@purdue.edu

ABSTRACT

Elf is a runtime for an energy-constrained camera to continuously summarize video scenes as approximate object counts. Elf's novelty centers on planning the camera's *count actions* under energy constraint. (1) Elf explores the rich action space spanned by the number of sample image frames and the choice of per-frame object counters; it unifies errors from both sources into one single bounded error. (2) To decide count actions at run time, Elf employs a learning-based planner, jointly optimizing for past and future videos without delaying result materialization. Tested with more than 1,000 hours of videos and under realistic energy constraints, Elf continuously generates object counts within only 11% of the true counts on average. Alongside the counts, Elf presents narrow errors shown to be bounded and up to 3.4× smaller than competitive baselines. At a higher level, Elf makes a case for advancing the geographic frontier of video analytics.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; • **Information systems** → **Data analytics**.

KEYWORDS

Video Analytics; IoT Cameras; Approximate Query

ACM Reference Format:
Mengwei Xu, Xiwen Zhang, Yunxin Liu, Gang Huang, Xuanzhe Liu, and Felix Xiaozhu Lin. 2020. Approximate Query Service on Autonomous IoT Cameras. In *The 18th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '20), June 15–19, 2020, Toronto, ON, Canada.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3386901.3388948

1 INTRODUCTION

A case for autonomous cameras Today's IoT cameras and their analytics mostly target urban and residential areas with ample resources, notably electricity supply and network bandwidth. Yet, video analytics has rich opportunities in more diverse environments where cameras are "off grid" and connected with highly constrained networks. These environments include construction sites, interstate highways, underdeveloped regions, and farms. There, cameras must be *autonomous*. First, they must be energy independent. Lacking wired power supply, they typically operate on harvested energy, e.g., solar or wind [7, 14, 72]. Second, they must be compute independent. On low-power wide-area network where bandwidth is low (e.g., tens of Kbps [34]) or even intermittent, the cameras must execute video analytics on device and emit only concise summaries to the cloud. Section 2 will offer more evidence.

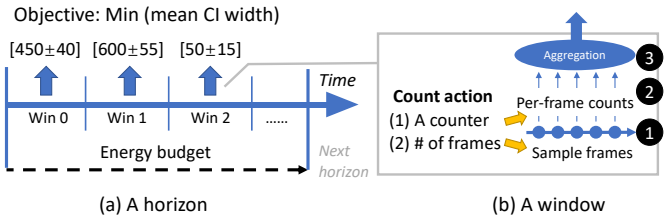


Figure 1: The target analytics and its execution. For simplicity, only counts for one object class are shown.

Target query: object counting As an initial effort to support analytics on autonomous cameras, this work focuses on a key query type: *object counting* with bounded errors. Inexpensive IoT cameras produce large videos [19, 20]. To extract insights from video scenes, a common approach is to summarize a scene with object counts. This is shown in Figure 1: a summary consists of a stream of object counts, one count for each video timespan called an *aggregation window*. Object counting is already known vital in urban scenarios; the use cases include counting customers in retailing stores for better merchandise arrangement [65]; counting audiences in sports events for avoiding crowd-related disasters [2, 62, 106]. Beyond urban scenarios, object counting further enables rich analytics: along interstate highways, cameras estimate traffic from vehicle counts, cheaper than deploying inductive loop detector [36, 69, 76, 89]; on a large cattle farm, scattered cameras count cattle and therefore monitor their distribution, more cost-effective than livestock wearables [5, 13, 47]; in the wilderness, cameras count animals to track their behaviors [55, 78, 81, 95].

Elf and its operation This paper presents Elf, a runtime for an autonomous camera to produce error-bounded object counts with frugal resources, especially limited energy. The counts are annotated with confidence intervals (CIs) as shown in Figure 1. CI is a

*Work performed while visiting Purdue
†Corresponding author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MobiSys '20, June 15–19, 2020, Toronto, ON, Canada
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7954-0/20/06…$15.00
https://doi.org/10.1145/3386901.3388948

common notion in approximate query processing (AQP) [32, 33, 41, 45, 49, 104]. A narrower CI suggests higher confidence in the count, hence more useful to users.

Elf builds atop periodic energy budgets, an abstraction commonly provided by existing energy-harvesting OSes [52, 96]. As illustrated in Figure 1, at fixed time intervals of a *horizon* (e.g., one day), the OS replenishes energy budget to be used by Elf in the next horizon. With given energy budget, Elf periodically wakes up the camera to capture video frames (❶); it runs neural network (NN) object counters on each captured frame (❷); by aggregating per-frame object counts, Elf materializes a per-window aggregated object count for each window (❸); Elf emits the sequence of aggregated counts by uploading them to the cloud either in real time or upon user request.

The central problem: count planning The theme of Elf is to produce aggregated object counts with high confidence under limited energy. Elf's operational objective for a horizon $\mathscr{H}$ is to maximize the overall confidence in all the emitted aggregated counts belonging to $\mathscr{H}$ while respecting the energy budget for $\mathscr{H}$. This is shown in Figure 1(a).

The above theme sets Elf apart from various visual analytics systems [56, 58, 60, 97, 101, 102, 107]. While prior systems focused on per-object results (e.g., accurate object labels), Elf takes one step further: it directly addresses the need for statistical summaries of videos, optimizing for aggregated counts with narrow errors. As a result, while prior systems focused on tradeoffs inherent in vision operators, Elf exploits higher-order tradeoffs between frame quantities and errors in per-frame counts. Its design thus has two unique aspects.

(1) Per window: characterizing count actions and outcome Of an aggregation window w, Elf navigates in a large space of *count actions*: it not only chooses the number of sampled frames but also chooses an NN, i.e., a counter, to count objects on individual frames. This is shown in Figure 1(b). For the window w, different count actions lead to disparate confidences in the aggregated count as well as disparate energy expenditures. There is no silver-bullet action. Notably, a more energy-expensive counter does not necessarily lead to higher confidence: while per-frame counts have lower errors, Elf can afford to process fewer frames, leading to lower confidence in the aggregated count.

Among all possible count actions for w, which ones should Elf consider? Our answer is an *energy/CI front* – all the count actions that lead to the narrowest CIs at different energy expenditure levels. To quantify the energy/CI front, Elf integrates as one unified CI two errors: i) Elf sampling frames rather than processing all possible frames; ii) per-frame counts being inexact. Prior AQP systems addressed the former [32, 33, 45] but never the two integrated to our knowledge. Elf unifies the two errors with novel modeling and approximation, as will be discussed in Section 5.

(2) Across windows: making joint count decisions on the go Of a window w, the energy/CI front reflects the confidence *return* from Elf's energy *investment*. As we will empirically show, windows often have different energy/CI fronts. As a result, investing the same amount of energy on different windows yields disparate CIs in their respective object counts. As Elf's objective is to maximize the overall confidence for a horizon, it must decide heterogeneous count actions for windows jointly, e.g., to invest more energy on windows where CI width sees more substantial reduction.

To do so, Elf addresses a dilemma between the needs for *global knowledge* and for *timely decisions*: i) to decide the optimal count action for a window w, Elf must compare the energy/CI fronts of all windows in the horizon; ii) to timely materialize the aggregated count of w, Elf must decide the count action for w without seeing *future* windows. To this end, Elf *predicts* the action count for each upcoming window based on the observation of past windows. Based on reinforcement learning, Elf mimics what an oracle planner, which (impractically) knows a horizon's all past and future windows, *would* decide.

Elf runs on off-the-shelf embedded hardware. Our tests on over 1,000-hour real-world videos demonstrate Elf's efficacy: with the energy level of a small solar panel, Elf continuously produces aggregated object counts that are only within 11% of the truth counts. At 95% confidence level, the CI widths are as narrow as 17% on average. We make the following contributions.

- *The design space:* We explore object counting on autonomous, energy-constrained cameras. We identify the central design problem: characterizing count actions and choosing them at run time.
- *The problem formulation:* From the large space of count actions, we formulate an energy/CI front as the viable actions a camera should consider. To quantify the front, we propose novel techniques for unifying errors from multiple sources into a single CI.
- *The runtime mechanisms:* To plan count actions at run time, we design a novel, learning-based planner, which mimics the decisions made by an oracle under the same observation of past videos.
- *The implementation:* We report a prototype Elf. Running on commodity hardware and with energy from a small solar panel, Elf continuously produces video summaries with high confidence.

To our knowledge, Elf is the first software system executing video object counting under energy constraint. Our experiences make a case for advancing the geographic frontier of video analytics.

2 BACKGROUND

2.1 Autonomous Cameras

Compute: commodity SoCs Similar to commodity IoT cameras [19, 20], autonomous cameras incorporate embedded application processors, e.g., those with Armv7a or MIPS32 cores at a few GHz and a few GBs of memory. We do not assume special-purpose hardware for vision [7, 63], as their wide adoption remains to be seen. The cameras run commodity OSes like Linux [15], which supports POSIX apps and frameworks, e.g., TensorFlow.

Network In off-grid scenarios we target, low-power wide-area networks (LPWAN) emerge as the norm [4]. One popular standard is LoRaWAN [34]: engineered for long-range, low-power communications, it mandates deeply duty-cycled (<1%) transmissions with data rate as low as a few Kbps [29]. Such networks motivate cameras to upload concise video summaries only, e.g., object counts over time windows. By contrast, uploading image frames (720P), even one in every one minute (inadequate for deriving useful object counts as our evaluation will show) would consume at least 40 Kbps per camera [29], unsustainable on LPWAN.

Energy source A key parameter of our system is the typical range of energy budgets. We use as a reference energy source a small (20cm × 20cm) solar panel backed by a rechargeable energy buffer. Such an energy harvester costs as low as $30, commonly seen in embedded prototypes and production [1, 53].

In two ways, we estimate the daily energy budget available from such an energy source. First, we measure the energy harvested by our small solar panel in the Midwestern US as 12.6 – 20.5 Wh/day. Second, we follow the prior studies on solar-powered embedded systems [53] and their typical parameters: a solar panel rated at 5 W, an energy buffer as a supercapacitor of 60Wh, and solar irradiance of 2490 – 9629 Wh/m^2/day (major US cities, according to the National Solar Radiation Database [10]). The energy available for use is 12.45 – 48.15 Wh/day. We conclude to design and evaluate Elf with energy budgets in the range of 10 – 30 Wh/day.

Energy budgeting At run time, the camera OS allocates an energy budget prior to each horizon. OS energy budgeting has been well studied [51, 52, 92, 96] and is orthogonal to this work. In a nutshell, it sets the horizon length to reflect periodic/temporal energy availability, e.g., one day; it allocates energy budgets in order to stay energy neutral given future energy availability (e.g., sunlight in the next few days). As such, the OS may set different energy budgets for different horizons. With the OS-level energy budgeting, Elf neither has to be the only application running on an autonomous camera nor has to run one query only.

2.2 Video Summary via Object Counting

At camera deployment time or run time, a user specifies her analytics as a continuous query $\langle \tau, \alpha, K \rangle$:

- *Aggregation window length* (τ) defines the temporal granularity, e.g., 30 mins, at which object counts are aggregated.
- *Confidence level* (α) specifies the desired probability of the query answer covering the ground truth count, e.g., 95%. Based on the desired probability Elf generates confidence interval (CI), a bounded error widely adopted in analytics systems [32, 33, 41, 45, 49, 104].
- *Object classes* (K, optional). By default, the camera counts all the object classes recognizable by modern vision object detectors (e.g., 80 classes by YOLOv3 trained on COCO [64]). The user may narrow down the counted classes to a subset C, e.g., cars and humans.

Elf answers a query with a stream of aggregated counts, $\{[\mu_{i,j} \pm \delta_{i,j}]\}$. In the sequence, each tuple corresponds to one window; $\mu_{i,j}$ is the output aggregated count of object class j in window i; $\delta_{i,j}$ is the CI width, which covers the true count with α probability. For example, under $\alpha = 95\%$ when Elf emits a count 1000 ± 100, it indicates that the probability for the true count to fall in (900, 1100) is 95%. A smaller δ indicates higher confidence in the count, making it more useful. Note that aggregation windows and CI are well-known concepts used by popular analytics systems [32, 33]. Through further analysis, users may compose object counts accompanied by CIs into higher-level statistical summaries that describe trends in long videos, e.g., "there is a 90% probability that cars crossing this intersection have doubled in the past week".

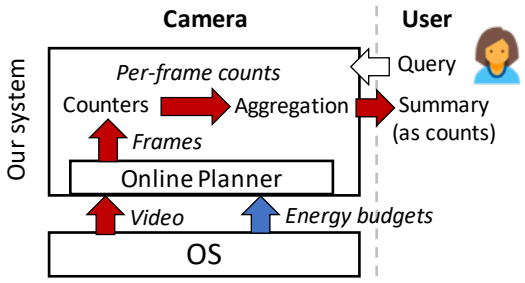


Figure 2: The Elf architecture

2.3 Object Counters on Individual Frames

As described in §1, to derive the object count for a window, Elf first counts objects on individual frames sampled from that window. To count per-frame objects, we follow a common approach of executing object detectors [18, 38, 48, 100]. The state-of-the-art object detectors are neural networks (NN). We choose a set of generic, popular detectors as listed in Table 3. Note that: i) We are aware of object detectors specialized for particular videos [60] which are compatible with our design. We motivate and validate our design with generic, well-known detectors for ease of experiments and result reproducibility. ii) We are aware of recent work using NN to emit counts without first detecting objects [59]; we dismiss such an approach due to its tedious per-video, per-class training and much lower accuracy than object detectors observed in our experiments.

The wide selection of NN counters offers diverse energy/error tradeoffs at the frame level. This is crucial, as no single counter can yield the narrowest CIs for all the windows under a given energy budget. For instance, YOLOv3, an expensive counter, incurs low error in per-frame counts; however, with its high energy cost (e.g., ~50J per frame on an Arm device, see §6) the system can only afford processing one frame in every 2 minutes (with a 10Wh/day budget), which eventually leads to inferior CIs. We will show more evidence in §3.

3 THE ELF DESIGN

Figure 2 shows the architecture of Elf. A user installs a query to the camera either at the deployment time or over the air later. Elf plans its execution in the scope of a horizon $\mathscr{H}$.

3.1 System Operation

Energy expenditure Elf executes the query by respecting the received energy budget for the horizon $\mathscr{H}$. It spends the energy on the following activities: (1) E_{cap} for capturing frames; (2) E_{count} for executing counters on frames; (3) E_{agg} for deriving aggregated counts; (4) E_{upload} for uploading the aggregated counts. The first two activities dominate the Elf's energy consumption (>99.9% as we measured): in each window (e.g., 30 min), Elf acquires tens of MB pixels from image sensors and runs several trillions of FLOPs. Activities (1) (2) hence will be our focus. By comparison, the latter two consume negligible energy: (3) only incurs hundreds of arithmetic operations per window and (4) only uploads concise numerical counts no more than a few hundred bytes per window.

Count action For each window belonging to horizon $\mathscr{H}$, Elf picks the number of frames to sample and an object counter. For the window, the action determines the confidence in the window's aggregated count and energy expenditure (including both E_{cap} and E_{count}). Note that Elf must run the same counter on all frame samples from a window in order to integrate per-frame count error in a tractable way (see §5 for details); from a window, it must draw frame samples uniformly in time to avoid sampling bias [46].

Objective: overall confidence as mean CI width Operating under the energy budget for $\mathscr{H}$, Elf seeks to maximize the overall confidence for $\mathscr{H}$. Our implementation defines the overall confidence as reciprocal to the mean of CI widths of the counts from all the windows in $\mathscr{H}$. Mean CI width is commonly used to characterize the overall confidence in a set of aggregates [33]. Elf is also compatible with alternative overall confidence metrics, such as median and minmax of CI widths of all windows in $\mathscr{H}$.

3.2 Count Action & Outcome

We next dive into the relation between the count action and the outcome. To simplify discussion, we focus on counting for a single object class (i.e., one aggregated count per window); yet the discussion applies to multiple classes if we consider the mean CI width over all the counted classes.

For a window w:

$$\underbrace{\langle N_w, C_w \rangle}_{\text{Action}} \rightarrow \underbrace{< E_w, \delta_w >}_{\text{Outcome}}$$

where N_w is the counter choice and C_w is the number of frames to sample; E_w is the energy spent on the window (including both E_{cap} and E_{count}) and δ_w is the CI width for the aggregated count of the window. For instance, Elf may execute an *expensive* counter (e.g., a deeper NN) to produce *more* exact per-frame counts with *higher* per-frame energy; or a *cheaper* counter (e.g., a shallower NN) produces *less* exact counts with *lower* energy. Note that while varying sample quantity is widely exploited by prior AQP systems, varying per-sample errors (in our case, through the counter choice) is a less explored opportunity as enabled by NNs.

The action/outcome plot All possible $N \times C$ combinations present rich actions available to Elf. They are visualized in action/outcome plots in Figure 3. On one plot, picking one counter (a curve in the plot) and the number of sample frames (a point on the curve with the number annotated), Elf generates an aggregated count with specific CI width (Y-axis) at energy consumption (X-axis). Along a given curve, as sample number increases, CI width narrows, according to sampling theory [46], and energy consumption increases. All such plots for all windows in $\mathscr{H}$ form the basis for Elf to make count decisions.

How to derive an action/outcome plot? For a specific window w, the plot is determined by (1) the true object count in w; (2) the count variation during the course of w; (3) the per-frame count errors of candidate counters. §5 will present quantification details.

The design implications are as follows:

• **No counter is silver bullet**. For a given window, no single counter (e.g. the most expensive or the cheapest ones) *always* results in the narrowest CI. As exemplified in Figure 3 (left): when the window's energy expenditure is low (<0.5 kJ), cheap counters result in narrower CIs; as the energy expenditure increases, expensive counters start to excel. This is because with less energy the CI is primarily bottlenecked by the inadequacy of sample frames, and cheap counters allow more sample frames. With abundant energy, even running more expensive counters Elf can afford adequate frames; hence, the CI starts bottleneck at the errors in per-frame counts.

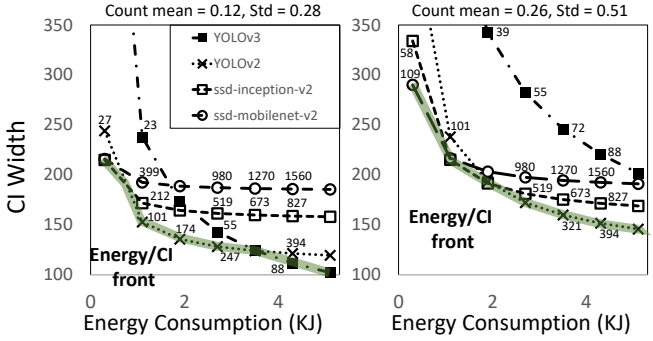


Figure 3: A comparison of action/outcome plots for two example windows, showing their disparate outcomes and energy/CI fronts. On each plot: X-axis shows the window's system energy expenditure (E_{cap} and E_{count}); Y-axis shows the CI width of the aggregated count for that window; each curve corresponds to a counter choice, on which each point corresponds to an amount of frames to sample (annotated along curves).

• **Operate on the energy/CI front only.** On an action/outcome plot, the bottom-left segments from multiple curves constitute an energy/CI front as highlighted in Figure 3. The front contains all the "optimal" count actions that lead to the minimum CI width (Y-axis) with different amounts of energy spent on the window (X-axis). The implications are that: i) Elf should always operate on the energy/CI front, picking a point of count action from the front according to how much energy it is willing to spend on that window. ii) As Elf considers spending additional energy on a window, the gradient of the front indicates *prospect* of confidence gain, i.e., the rate of CI width reduction in response to additional energy investment.

• **Make joint decisions across windows.** The energy/CI fronts vary across windows, as exemplified by a comparison of Figure 3 (left) and (right). This means the same amount of energy expenditure on different windows will result in different CI widths. For example, increasing the energy from 2.0 kJ to 3.0 kJ would reduce the CI width by 26 in Figure 3 (right) but only 9 in (left). Intuitively, windows with higher object counts and higher variations (e.g. video clips during rush hours) would see higher CI reduction than otherwise (e.g., midnight). Since the objective of Elf is to minimize the mean CI width across a horizon (§3.1), it shall decide count actions across windows *jointly*, resulting in heterogeneous actions and outcomes. As will be shown in the evaluation, the resultant energy expenditures across windows could differ by 9x.

Putting it together The above observations suggest an energy planning strategy as follows. Based on energy/CI fronts for individual windows, the system iteratively invests energy slices to the window that sees the highest prospect of confidence gain. For that invested window, the system picks the optimal counter and frame count according to the window's energy/CI front. Intuitively, the system is inclined to invest more energy on windows with higher object counts and variations; the system is therefore more likely to sample more frames and pick more expensive counters – as energy budget permits – on such windows. This strategy is the basis of the oracle planner in §4.1.

4 COUNT PLANNING

Challenge To plan count actions towards the objective described in §3.1, Elf shall address a twist of two needs:

- The need for **global** knowledge. To optimize for the whole horizon $\mathscr{H}$, Elf shall make joint count decisions across windows (§3.2), i.e. comparing the energy/CI fronts of all windows, which reflect their confidence gains from potential energy investment. Yet, not until the end of $\mathscr{H}$ can Elf estimate the fronts of all windows belonging to $\mathscr{H}$, as Elf needs to see their respective object counts and count variation (see §5 for details).

- The need for **on-the-go** decisions. Deferring count actions to the end of $\mathscr{H}$ (as indicated above) raises two problems. i) Stale results: Elf will not be able to emit the counts for all windows until the end of $\mathscr{H}$. As $\mathscr{H}$ may span hours or days, doing so prevents users from observing fresh object counts. ii) Excessive capture: Prior to each window, Elf must decide the number of frames to *capture*, which limits the number of frames Elf can *process* later. Deciding count actions at the end of $\mathscr{H}$ forces Elf to play safe, capturing excessive frames for each window. Besides the two, holding captured frames until the end of $\mathscr{H}$ increases camera storage pressure and risk of privacy breach.

Approach overview Elf addresses both needs above with an online planner. The key idea is to make online decisions by mimicking what an oracle planner would do. Specifically, the oracle planner works *offline*: with full knowledge of a video, it decides what count actions *should have been* for windows in the video by considering the energy/CI front of all the windows jointly. Trained with the oracle decisions and true object counts from the videos, the online planner makes decisions that the oracle *would* make with a similar observation of recent windows.

4.1 The Oracle Planner (Offline)

The oracle planner works based on impractical assumptions: i) it knows the energy/CI fronts of all windows and ii) the amount of captured frames exactly matches the amount needed in processing. The oracle plans count actions by solving an energy allocation problem: it iteratively allocates energy slices to the window that exhibits the highest CI width reduction. More specifically, for a horizon:

- *Initialization:* The oracle planner dispatches energy to each aggregation window so that each window has a minimum number M of frame samples. This is because in statistics, an estimation from sampling is only regarded meaningful when the sample size is sufficiently large. We pick $M = 30$ by following common practice in statistics [61]. The oracle picks the cheapest NN counter for each aggregation window in order to start from the lowest possible energy consumption.

- *Iteration:* The oracle repeatedly allocates a small, fixed amount of energy to individual windows. To pick the next window W for receiving an energy slice, the oracle examines the energy/CI fronts of all windows, selecting W to be the one having the highest gradient $|\frac{\partial CI}{\partial E}|$ at the current operating point $\langle$ energy, CI width $\rangle$. By allocating the energy slice, the oracle updates the count action for W and slides W's operating point along its energy/CI front.

- *Terminate:* The oracle planner stops when it uses up the energy budget. At this moment, the NN counters and numbers of frames for all windows are the final count decisions.

In evaluation, we consider the oracle as the upper bound of the overall confidence achievable by Elf.

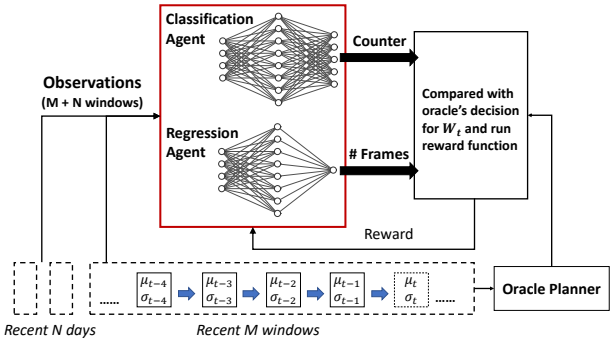


Figure 4: Training RL agents for online count decision. μ and σ represent the object count and count variation (observations) for each window.

4.2 The Learning-based Planner (Online)

Overview Prior to Elf deployment, we run the oracle planner offline on sample video footage from the target camera; in our implementation we use 3-day video footage. Using the video and the oracle decisions as the training dataset, we train the online planner. Deployed on the camera, the online planner continuously outputs count decisions based on the object counts and count variation that Elf emits recently, including those from recent windows and the windows at similar times in recent days.

Rationale: Why could an online planner work? An online planner acts only based on its observation of the past; yet, as we will demonstrate in §7, it can output decisions closely matching the oracle that knows both the past and the future. We attribute the efficacy to the temporal correlation among object counts and count variations in a video feed, a pervasive video characteristic. For instance, the car count in 9 AM – 9:30 AM correlates to the counts from half-hour windows prior to 9 AM of the same day and to the count from 9 AM – 9:30 AM of prior days. While the correlation is still not deterministic enough for Elf to directly predict counts as analytics results, the correlation provides sufficient *hints* for Elf to plan count actions and manage energy.

Intuitively, the count action picked by oracle for a window W primarily depends on the relative significance of W's object count and variation with regard to those of other windows. Due to the temporal correlation discussed above, both information is encoded in the sequence of past object counts and count variations prior to W. As such, the online planner can use the past sequence to predict the oracle's decision for W.

Why reinforcement learning (RL)? Essentially, the online planner takes sequential actions to optimize a long-term objective (i.e. mean CI width). This pattern well suits RL, a general framework for sequential decision making. In RL, an "agent" (e.g., our planner) interacts with its environment (e.g., the energy budget and all energy/CI fronts) in discrete time steps to maximize its cumulative reward. If the environment is fully observable to the agent, the agent takes an action based on the current environment state at each time step. The action takes the environment to a new state, and the agent receives a reward accordingly. If the environment is only partially observable (e.g., our planner only knows past, but not future, energy/CI fronts), the agent takes an action based its observation o_t of the environment.

To apply RL, we face the following challenges.

• (C1) Long delay in rewarding. The count actions of all windows jointly decide the mean CI width for the whole horizon $\mathscr{H}$. Hence, not until the end of $\mathscr{H}$ can Elf evaluate its past count actions and assign reward/penalty accordingly. This makes RL training difficult.

• (C2) Hard constraint on the accumulative outcome. Of our online planner, the total energy expenditure across multiple steps (i.e. windows) should respect a constraint – the energy budget for $\mathscr{H}$. Yet, the standard RL lacks mechanism to enforce such a constraint to our knowledge.

We address (C1) to reward the planner in training timely and frequently – at every time step: rather than reasoning about the long-term impact of the planner's decision, we consider how much the decision deviates from the oracle's decision. We address (C2) by making the planner *implicitly* learn to respect an energy budget, as we train the planner to follow the decisions made by the oracle that operates under the same energy budget. To handle the unlikely events that the planner burns out the budget before a horizon's end, we devise a backstop mechanism to be described later.

RL formulation As illustrated in Figure 4, we formulate:

• *A time step* is an aggregation window.

• *The observation* vector consists of the object counts and count variation of the most recent M windows and the same-time windows in the recent N days. Our experiment empirically chooses $M = 4$ and $N = 1$.

• *The agents* are two that receive the same observation and pick the number of sample frames and the counter, respectively. We instantiate the two as a regression agent and a classification agent. Both agents are NNs with the same multi-layer perceptron (MLP) architecture. The NNs are small: the input layer has only 10 input units; both NNs have two hidden layers each with 64 hidden neurons only. Each NN has less than 5K parameters and 55KB in size.

• *The reward/penalty* is an agent's decision deviation from the oracle's decision. For the regression agent, its reward function is: $r_{t,reg} = -|N_{t,agent} - N_{t,oracle}|$ where $N_{t,agent}$ and $N_{t,oracle}$ are the frame amounts from the agent and from the oracle, respectively. For the classification agent, the reward function is: $r_{t,cls} = 1\ if (C_{t,agent} = C_{t,oracle})\ otherwise\ 0$, where $C_{t,agent}$ and $C_{t,oracle}$ are the NN counters chosen by the agent and the oracle planner respectively.

• Respecting *energy budgets*. To train the RL agents for operating on a variety of energy budgets that the agents may receive from the OS at run time, we discretize the target energy budget range into multiple levels (e.g., 10–30 Wh/day at 5 Wh/day steps) and train a pair of RL agents for each energy level.

Offline training & cost We follow a standard approach: we use the Actor-Critic framework [93] and A2C [3], a synchronous, deterministic training algorithm as a variant of A3C [71].

Before deploying RL agents in real-world systems, we first run the oracle planner on a video segment (3 days in our experiment) to collect training data and train the RL agents offline. Our experience shows modest training effort in general. For most video scenes under test (listed in Table 2), we find a 3-day video sufficient for training. The training overhead primarily comes from: i) running the oracle planner, including deriving the energy/CI fronts by testing all candidate counters on the videos; ii) training the RL agents. The former takes a few hours on a commodity GPU workstation with one Nvidia Quadro 6000 and can be further accelerated by additional GPUs or TPU; the latter takes as low as tens of minutes on the same workstation.

After deployment, we expect the trained RL agents to operate for a long period of time autonomously. Our experiments show their stable accuracy over our longest video lasting 2 weeks. In real-world deployment, we expect to only retrain the RL agents in case of substantial changes in video scene, e.g. due to cameras being relocated or small scene changes accumulated over time. Users may initiate retraining based on their knowledge on camera deployment, or as preventive maintenance. Similar to the initial training, users would need to retrieve a recent video segment from the camera, run the oracle over the video on their development machines, and update the RL agents on the camera.

Online prediction & cost Once trained, the online planner is deployed as part of Elf on camera. As RL is incapable of guaranteeing perfect decisions, what if the planner mispredicts (despite unlikely)? In particular, burning out energy before a horizon ends would leave no energy for the remaining windows and therefore no counts produced for them. To this end, Elf incorporates a backstop mechanism alongside the planner.

Over the course of a horizon, Elf monitors the energy balance, e.g., the remainder from the budget. When the balance drops down to the "bare minimum", i.e., the amount needed by the remainder windows to run the cheapest counter with the smallest frame count needed to be statistically significant (e.g., 30), Elf bypasses the planner for the remaining windows and follows the minimum count actions. If the online planner acts conservatively and has not used up the energy when a horizon ends, Elf returns the unused energy to the OS.

Notation	Description
$x = x_1..x_n$	Inexact counts on sampled frames observed by a given NN counter
$\bar{x}$, S	The mean and standard deviation of x
μ_x	A random variable representing the mean of inexact per-frame counts (from the given NN counter) on all frames
μ	A random variable representing the mean of exact per-frame counts on all frames
E', E'', θ	E' (or E'') is the distribution of the deviation between μ and μ_x when $\bar{x}$ is above (or below) a threshold θ, respectively
V', V''	The distributions of μ, when $\bar{x}$ is above (or below) θ, respectively
v'_α, v''_α	The CI widths for μ with confidence level α, when $\bar{x}$ is above (or below) θ, respectively

Table 1: Notations used in Section 5

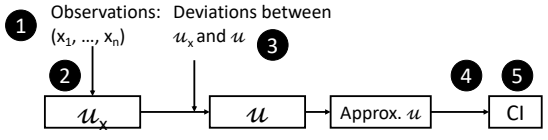


Figure 5: The workflow of deriving the CI

We encourage Elf's conservative energy expenditures through tuning reward functions for our RL agents. The rationale is that we prefer (slight) budget underutilization to early burn out: while the former only sees minor loss in the overall confidence, the latter results in significant CI widths for a series of windows.

The planner itself incurs negligible energy overhead. For each window, it performs around 8K multiply-adds operations. By contrast, even the cheapest NN counter in our selection performs 0.8 billion multiply-adds operations *per frame*. We estimate the planner's energy consumption is at least four orders of magnitude smaller than the per-frame counter.

5 INTEGRATING MULTI-SOURCE ERRORS

The problem Given a set of per-frame counts, Elf needs to derive an aggregated count accompanied by a *single* CI – for both planning (i.e., constructing energy/CI fronts for all windows) and for materializing aggregated counts.

Table 1 summarizes the notations used below. Our *input* is a series of inexact per-frame counts ($x = x_1...x_n$) as observed by one given NN counter (❶). Our *output* is a CI as an estimation for μ, i.e., the mean of *exact* per-frame counts on *all* the frames (❺).

Approach overview To model the random variable μ, we first model a random variable μ_x, the mean of *inexact* per-frame counts on *all* the frames; we derive μ_x from x, the observed *inexact* counts on *sampled* frames (❷). We then model the deviation between μ_x and μ as caused by errors in per-frame counts (❸). Eliminating the deviation from μ_x, we derive the distribution of μ, from which we estimate its CI via approximation (❹).

We demonstrate the validity of the resultant CIs with experiments on real-world videos (§7.2).

Modeling μ_x The theory of statistics gives us:

$$\frac{\bar{x} - \mu_x}{S/\sqrt{n}} \sim T_{n-1} \tag{1}$$

where $\bar{x}$ and S are the mean and standard deviation of the observed inexact counts (x), and T_{n-1} is the well-known t-distribution with n - 1 degrees of freedom [46]. This holds regardless of x's actual distribution, e.g., normal or Poisson, as long as the sample population is sufficiently large (e.g., 30 which is also the minimal sampling number for each aggregation window used by Elf), according to the central limit theorem [103].

Modeling the deviation between μ_x and μ The deviation is because μ_x incorporates errors in per-frame counts that μ does not incorporate. We model the deviation with a heuristics: the deviation has positive correlation with μ. The rationale is that an NN counter is likely to incur more false positives/negatives on video frames with more objects. Based on our experiments, when the absolute value of μ exceeds a threshold θ (as we will empirically determine), the distribution of deviation E' is best modeled as the ratio between μ and μ_x (i.e., $\mu/\mu_x \sim E'$); when μ is small, the distribution E'' is best modeled as a linear offset between the mean of true counts and that of observed counts (i.e., $\mu - \mu_x \sim E''$). Notably, E' and E'' cannot be unified as one distribution. For instance, a small μ results in a large ratio; including those outliers in the overall error distribution results in high distribution variance as observed from real videos. As Elf cannot directly observe μ at run time, it uses $\bar{x}$ to estimate if μ exceeds the threshold θ.

Determining the parameters of deviations The means and standard deviations of E' and E'' are crucial to modeling μ, which we obtained through offline profiling: for each camera, profiling once at the camera's deployment time, and profiling again if the video scene changes significantly, e.g., the camera being relocated. This is based on twofold observation below.

- The distributions of deviation are stable over time, with a given NN counter and a given video feed. As an example, we measured the distributions with two counters, `YOLOv2` and `ssd mobilenet-v2` and video segments from Jackson; we tested two week-long video segments that are one month apart. The Bhattacharyya coefficient [37] between the two videos are 0.93 and 0.86 for the two NN counters, respectively.

- The distributions are independent of the observed object counts x, as we confirmed with chi-square test [46] on our video datasets. Given such independence, we can integrate μ_x and E in order to estimate the distribution of μ, as shown below.

Note that across different NN counters the above distributions of deviation (E) are often disparate, even on the same video feed. This explains why we only use one NN counter in one aggregation window.

Modeling μ We use V' and V'' to denote the distributions of μ:

$$\mu = \begin{cases} (\bar{x} + S/\sqrt{n} \cdot t) \times e' \sim V' & \text{if } \bar{x} > \theta \\ (\bar{x} + S/\sqrt{n} \cdot t) + e'' \sim V'' & \text{if } \bar{x} \le \theta \end{cases} \tag{2}$$
$$where: t \sim T_{n-1}, e' \sim E', e'' \sim E''$$

Hence, the CI of μ, denoted as v'_α and v''_α, is as follows:

$$CI = \begin{cases} [\bar{x} \times \mu(e') \pm v'_\alpha] & \text{if } \bar{x} > \theta \\ [\bar{x} + \mu(e'') \pm v''_\alpha] & \text{if } \bar{x} \le \theta \end{cases} \tag{3}$$

$$where \quad P(|V' - \bar{x} \times \mu(e')| \le v'_\alpha) = \alpha\%$$
$$P(|V'' - \bar{x} - \mu(e'')| \le v''_\alpha) = \alpha\%$$

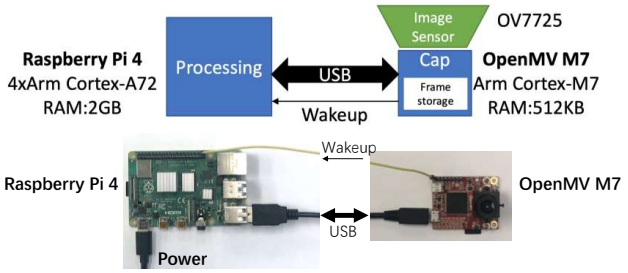


Figure 6: Our hardware prototype for testing Elf. The platform consists of two interconnected SoCs for frame capturing and processing, respectively.

Typically, the CI widths are derived through Monte Carlo simulation [87]: randomly picking the same number of samples from t-distribution and NN counter's error distribution (E', E'') respectively, and combining them as Equation 2. The results are expected to follow the distribution of μ, from which one gets the CI.

Approximating the distribution of μ The downside of Monte Carlo simulation is high compute overhead; this is exacerbated by that Elf must run the simulation repeatedly online for planning, as $\bar{x}$ and S can only be observed online. Fortunately, we observe that each of V' and V'' is close to a normal distribution with a similar cumulative distribution function (CDF). Hence, we approximate the CI width by treating V' and V'' as normal distributions with standard deviation $\sigma(\mu)$. Based on Equation 2, we derive $\sigma(\mu)$, the standard deviation of μ, as below.

$$\sigma^2(\mu) = \begin{cases} (\sigma^2(u_x) + \bar{x}^2)(\mu^2(e') + \sigma^2(e')) - \bar{x}^2\mu^2(e') & \text{if } \bar{x} > \theta \\ \sigma^2(u_x) + \sigma^2(e'') & \text{if } \bar{x} \le \theta \end{cases} \tag{4}$$

$$where: \quad \sigma^2(u_x) = \frac{S^2}{n}\sigma(t_{n-1}) = \frac{S^2(n-1)^2}{n(n-3)^2}$$

As V' and V'' are approximated as normal distributions, we have their CI widths as:

$$v'_\alpha = z_{\alpha/2}\sigma(\mu) \quad v''_\alpha = z_{\alpha/2}\sigma(\mu) \tag{5}$$

where $z_{\alpha/2}$ is the Z-score for the given confidence level [46], e.g., 1.96 when α is 95% and 2.576 when α is 99%.

The approximation above also sheds light on how different factors affect CI widths. For instance, higher variations (S) in observed object counts and larger per-frame count errors ($\sigma(e')$ or $\sigma(e'')$) contribute to $\sigma(\mu)$, resulting in wider CI widths ($v'(\alpha)$ or $v''(\alpha)$). A larger amount of samples (n) reduces the CI widths.

From mean counts to aggregated counts With the above steps Elf estimates the mean count *per frame*, e.g., "the average number of cars on the road at 1FPS is $[0.5 \pm 0.1]$". To get aggregated counts, Elf multiplies the mean by the amount of frames, e.g., "the total number of cars in 30 minutes is $[900 \pm 180]$". See Section 6 for details.

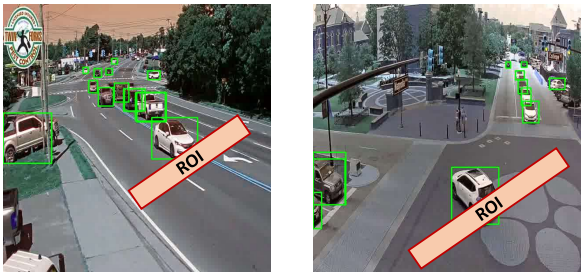


Figure 7: ROI-based object counting

6 IMPLEMENTATION

Heterogeneous hardware decoupling frame capture/processing Commodity IoT cameras are often energy-inefficient at sampling *sparse* image frames: to capture one frame, the whole camera wakes up from deep sleep and falls back to sleep afterward, spending several seconds. We measured that the energy for capturing a frame is almost the same as the energy for processing the frame (YOLOv2 on Raspberry Pi 4). While the camera may defer processing images (e.g., until window end) for amortizing the wake-up energy cost, it cannot defer periodic frame capture.

To make periodic image capture efficient, we build a hardware prototype with a pair of heterogeneous processors, as shown in Figure 6. The prototype includes one capture unit, a microcontroller running RTOS and capturing frames periodically with rapid wakeup/suspend; and one processing unit, an application processor running Linux and waking up only to execute NN counters. Our evaluation §7.4 validates the necessity of heterogeneity.

NN counters Elf builds on NNPACK-accelerated darknet [11] for YOLO NNs and TensorFlow [26] for other NNs. It uses OpenCV [12] for image processing.

ROI-based instance counting To avoid double-counting objects in adjacent frames, a known computer vision challenge, our implementation adopts a common heuristics that exploit region of interest (ROI) [18, 38]. Shown in Figure 7, an ROI for a video specifies an image region as well as t, the maximum time that an interesting object takes to travel through the region. Accordingly, the object count within a time period is the total number of objects intersecting with ROI on all the frames sampled over the time period at the intervals of t. We are aware of enhancements for mitigating double counting, e.g., by tracking objects across frames [90, 100]. Such computer vision enhancements are compatible with Elf: they add per-frame compute cost that is minor compared to object detection which changes little of our core challenge: the relation between count actions and outcomes; they are also orthogonal to our core contributions for producing statistical results with limited energy.

7 EVALUATION

Our evaluation answers the following questions:
§7.2 Can Elf provide useful counts with valid, narrow CIs under realistic energy constraints?
§7.3 Whether the key designs of Elf are significant?
§7.4 How will new hardware impact Elf's performance?

Video	Length	GT count	description
Jackson [24]	2 weeks	1,386/3,060	An intersection in Jackson Hole, WY
Auburn [21]	1 week	495/1,908	Toomers Corner in Auburn, AL
Cross [22]	2 weeks	329/4,412	A three-way cross, location unknown
Taipei [25]	1 week	1,658/4,284	An intersection in Taipei
Hampton [23]	1 week	1,267/4,824	An interstate road in Hamptons, NY

Table 2: Videos for evaluation. GT count: mean/max ground-truth count of all half-hour windows. Target object: vehicle.

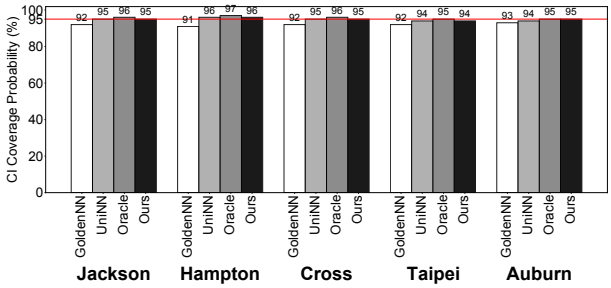


Figure 8: Elf produces CIs that cover the true counts at the target confidence level (95%, the horizontal line).

NN Counters	Input	mAP	Energy
YOLOv3 (Golden, GT) [85]	608x608	33.0	1.00
YOLOv2 [84]	416x416	21.6	0.22
faster rcnn inception-v2 [86]	300x300	28.0	0.40
ssd inception-v2 [68]	300x300	24.0	0.08
ssd mobilenet-v2 [88]	300x300	22.0	0.05
ssdlite mobilenet-v2 [88]	300x300	22.0	0.04

Table 3: The NN counters used in this work. mAP: mAP accuracy on COCO dataset [64]. Energy: normalized energy as measured on RPI 4.

7.1 Methodology

Videos We evaluate Elf on 5 long videos from different cameras (Table 2). Each video lasts 1-2 weeks; altogether, they constitute 1176 hours and ∼800 GB of data. We intentionally select the videos to cover diverse scenes, e.g., intersections and highways. Our selection of videos is challenging to error-bounded object counting: most frames contain few objects in ROIs; aggregated counts typically have small mean values and high variation. Finding optimal count strategies for such sparse data is difficult as shown in prior work [104]. We preprocess the videos by decoding them into 1 FPS images to accommodate our ROI-based counting (§6). Of each video, we use the first three days of video to train our RL-based planner, and the remaining days for testing.

We report the results of counting *vehicles*. While Elf by design supports counting multiple object classes (§2.2), to our knowledge, there are no publicly available videos that last for days while containing diverse object classes, each class with sufficient instances for meaningful counting. Notably, video benchmarks popular in computer vision research only last seconds or minutes each [17, 100], inadequate for exercising Elf.

Despite our best effort in finding long benchmark videos, we acknowledge the limitation in scene diversity of our video datasets. In the traffic videos we use, object counts are more likely to exhibit high temporal correlation, matching our rationale of using RL as discussed in Section 4.2. Beyond traffic videos, we expect such temporal correlation in a variety of video scenes, e.g., cattle monitoring. Nevertheless, on videos where such temporal correlation is weaker, e.g. counting *rare* wild animals, we expect the RL-based planner to make more misprediction. For such videos, while Elf's characterization of count actions still holds, it would need additional heuristics for planning.

Metrics To quantify Elf's query answers, we report:

- *CI coverage probability*: the measured chance that CIs produced by Elf covering the ground truth. The probability is expected to exceed the desired confidence level specified by users (§2).
- *Mean CI width*, normalized to the mean counts. For example, for a list of CIs as $\{[\mu_i \pm \delta_i]\}$, the mean CI width is $(\sum \delta_i)/(\sum \mu_i)$. A low value indicates high overall confidence in Elf's answers.
- *Mean error*, defined as $(\sum |\mu_i - g_i|)/(\sum g_i)$ where g_i is the true count. This metric shows by how much Elf's approximate counts deviate from the ground truth.

Energy we report the whole-camera energy measured from the hardware prototype.

NN counters and ground truth counts Table 3 lists the NN counters used in experiments and their respective energy consumptions. Following prior work [56, 60, 101], we treat as the ground truth the counts returned by the most expensive NN, named the "golden" NN counter (YOLOv3).

Alternative designs We compare Elf to the following designs:

- GoldenNN runs the golden NN counter with the same amount of sample frames in all windows.
- UniNN runs one *single* NN counter with the *same* amount of frames in all windows. To make this design competitive, we set its NN counter to be the one with the best average performance over all test horizons of a given video. Unlike GoldenNN, the counter of UniNN may be different on separate videos. The design uses our technique (§5) to provide CIs.
- Oracle uses the oracle planner descried in §4.1, representing the best attainable performance. Note that Oracle is built atop impractical assumptions and delays materializing aggregated counts.

System parameters We set our parameters as typically used in prior systems: we use 30 minutes as the aggregation window length [94], 24 hours as a horizon [96], and 95% as the default confidence level [32, 70]. As discussed in §2.1, the typical harvested energy from a small solar panel is 10Wh – 30Wh, which we use in experiments.

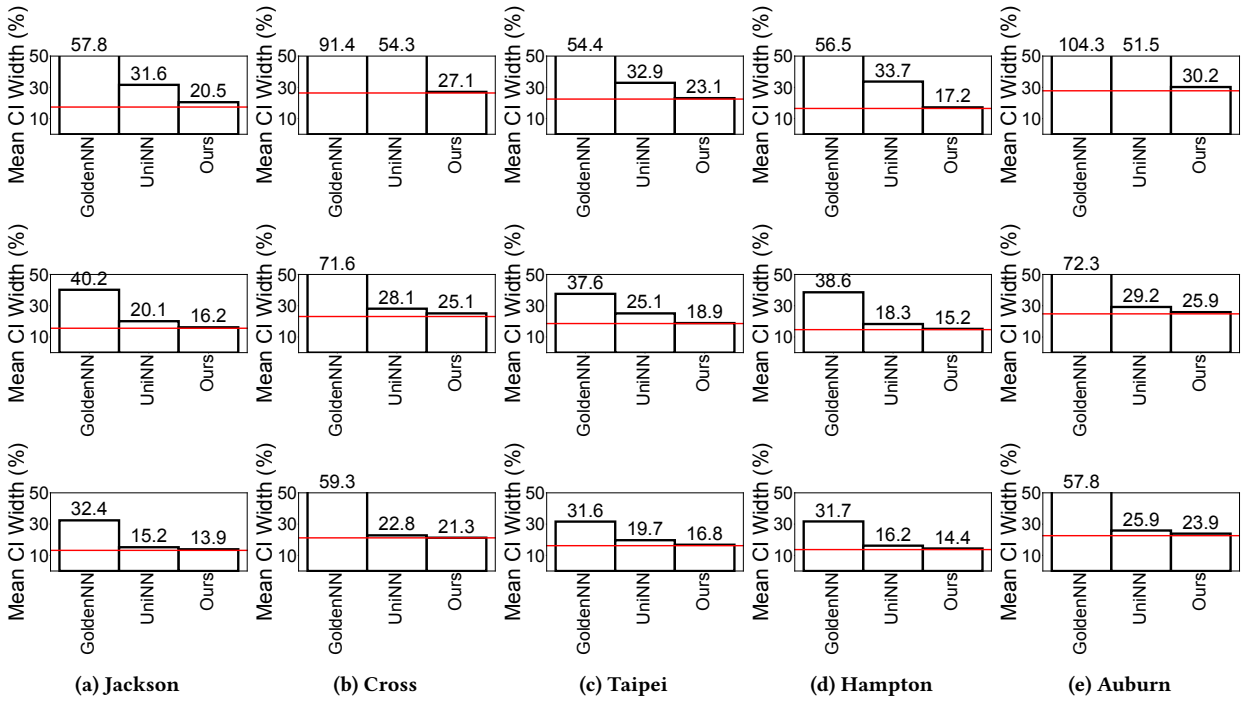


Figure 9: With given energy budgets (1st/2nd/3rd rows: 10Wh/20Wh/30Wh per day) and videos (columns), Elf provides narrower CIs (Y-axis) than `GoldenNN` and `UniNN`, and very close to `Oracle` (the horizontal line).

7.2 End-to-End Performance

Elf provides valid CIs Our experiments show that the CIs cover the ground truth count at the target coverage probability, which validates our technique for integrating errors (§5). Figure 8 shows the CI coverage probability, suggesting that Elf has met the specified confidence level (95%). The results are averaged over multiple experiment setups, i.e., energy budgets and CI width targets, as described in §7.3.

Figure 8 also shows the CI coverage by the alternative designs. Employing our error integration technique, `Oracle` and `UniNN` also meet the target confidence level. `GoldenNN` results in noticeably lower coverage probability below the target. The reason is that, when running `GoldenNN` under energy constraint, the system can only afford processing a small number of frames (< 30) using the golden, expensive counter. Such a small sample is insufficient for deriving statistically meaningful aggregates.

Elf emits useful counts with limited energy. Elf produces small mean errors and mean CI widths. On one hand, the aggregated counts emitted by Elf are within 14.8%, 12.4%, and 11.1% of the ground truth with an energy budget of 10 Wh/day, 20 Wh/day, and 30 Wh/day, respectively. On the other hand, as shown in Figure 9, Elf presents mean CI widths of 22.1%, 19.3%, and 17.3%, respectively. Such results are on a par with state-of-the-art video counting approaches and analytics systems [31, 66, 79, 108]

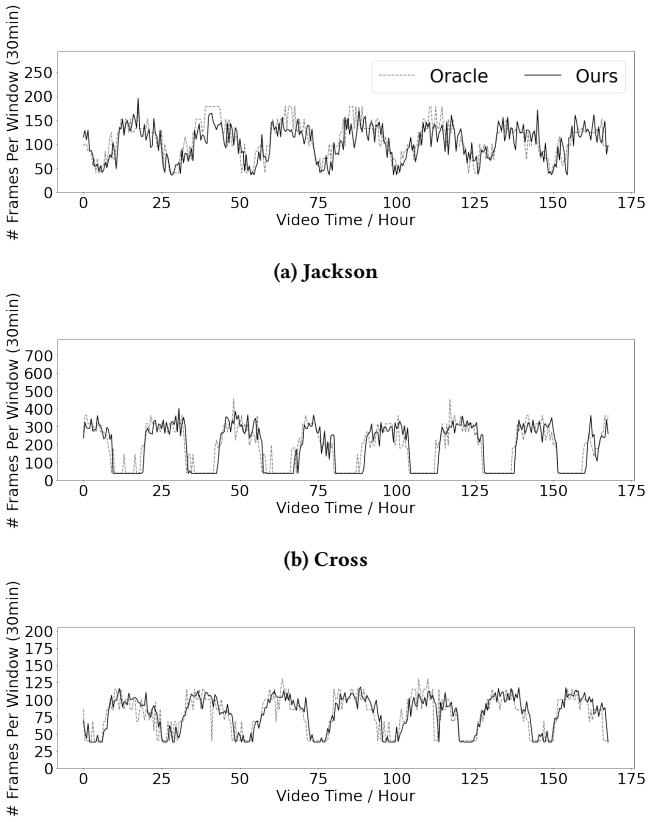
7.3 Validation of Key Designs

7.3.1 Exploiting diverse NN counters. Elf significantly outperforms using one counter only. As shown in Figure 9, compared to `GoldenNN`, Elf's mean CI widths are smaller by 66.6%, 59.8%, and 56.2% (note: not percentage points) on average with an energy budget of 10Wh/-day, 20Wh/day, and 30Wh/day, respectively. On some videos, e.g., Auburn, Elf's CI is up to 3.4× smaller.

Diving deeper, we find that while the golden NN counter produces more exact counts per frame, the system can only sample less than 30 frames per aggregation window. By contrast, Elf is able to pick moderately less accurate counters while sampling 5×–9× more frames. The large sample quantities outweigh the modest increase in per sample errors and result in overall higher confidence.

7.3.2 Heterogeneous count actions across windows. Elf consistently outperforms `UniNN`, a static count action optimized for all windows in a video. As shown in Figure 9, compared to `UniNN`, Elf's mean CI widths are smaller by 41.1%, 16.6%, and 9.7% on average with an energy budget of 10Wh/day, 20Wh/day, and 30Wh/day, respectively.

The advantages of Elf are twofold. First, `UniNN` only uses one single NN counter that performs best throughout an entire video. By contrast, Elf utilizes the energy/CI front to select the proper NN counter for each individual aggregation window. We observe that: within one horizon, Elf switches among 2 – 5 different NN counters across windows; across different horizon, videos, and energy budgets, Elf's counter choices are even more diverse. Accordingly, Elf picks a wide range of sample quantities across windows, e.g., up to 5× difference (30–160 frames) with 10Wh/day energy budget. Second, with a static count action `UniNN` allocates the same energy on each window. However, as video characteristics are disparate across time, the return of the same amount of energy varies substantially across different aggregation windows as shown in §3.2 and



(a) Jackson

(b) Cross

(c) Taipei

Figure 10: The amount of per-window frames by Elf's learning-based planner as compared to the oracle planner.

Figure 3. By comparison, Elf identifies such disparity and adjusts energy accordingly (§4). As we measured, the difference in energy allocations across windows can be up to 7×.

7.3.3 Imitating the oracle planner. The confidence in Elf's results is close to that of `Oracle`, showing the efficacy of our learning approach. As shown in Figure 9, compared to `Oracle`, Elf's mean CI widths are only wider by 7.2%, 4.4%, and 4.1% on average with an energy budget of 10/20/30 Wh/day, respectively. Note that `Oracle` is impractical and cannot deliver timely results as Elf does.

Zooming in, we find Elf's planner predicts the oracle's decisions with good accuracy, as exemplified by the videos in Figure 10. First, the amount of per-window frame by Elf is within 15.5% of the amount by the oracle. Second, Elf picks the same counters (out of six counter options) as the oracle does in 56%–92% of all the windows (mean/median=76%/78%, not shown in Figure 10). We further examine when our planner deviates noticeably from the oracle, finding out that these are the windows i) showing *irregular* variations in object counts or ii) where the oracle picks a rarely used NN counter. In the former situation, temporal correlation in object counts is weaker, rendering RL less effective; in the latter situation, as our planner does not see such rarely used NNs enough during training, it is less likely to pick them at run time. Note that

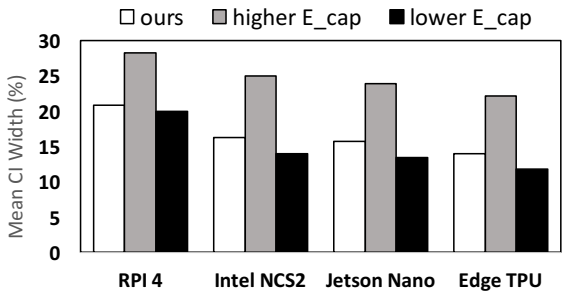


Figure 11: Elf on different hardware (video: Jackson, energy budget: 10Wh/day). Higher E_{cap} indicates using less efficient hardware to capture, while lower E_{cap} indicates using an ultra low-power capture hardware.

i) even when our planner deviates from the oracle, the planner's decision is often the second optimal, incurring minor efficiency loss; ii) the deviation in resource planning does not affect Elf's statistical guarantee for analytics results.

Elf's planner strictly respects energy budgets allocated to it. Recall that we tune our RL for conservative energy expenditures in order to minimize occurrences of early energy burn out before the end of a horizon, as described in Section 4.2. Thanks to such a design, Elf has energy leftover at the ends of 64.6% of the horizons; on average the fraction of unused energy is as small as 9.5%, which will be returned to the OS. Elf's per-window energy expenditures are within that of the oracle planner as low as 8.2% on average.

7.4 The Impact of Hardware

In the current Elf prototype, we choose commodity hardware (RPI4) that is popular for low monetary cost and programming ease. Next we evaluate how Elf behaves with different hardware.

Compute hardware Elf is relevant with accelerators that run NNs with higher energy efficiency; the extra efficiency may further expand the applicability of Elf.

We test three accelerators: Intel NCS2 [8] ($80), Jetson Nano [9] ($100), and Edge TPU [6] ($150). With the results shown in Figure 11, our observations are two. First, compared to RPI4, more efficient accelerators reduce the mean CI width noticeably (by 22.1%–33.1%), primarily because Elf affords processing more frames per window. Second, even a cutting-edge accelerator (e.g., TPU) cannot reduce the CI width to near zero. The error mainly comes from frame sampling, showing the efficiency of the accelerator has not reached a level where Elf can afford processing *every* frame. This suggests our core design – count action planning – to be relevant in the near future. Third, to yield similar CI widths, the accelerators need a much smaller energy budget, hence more modest energy sources. For instance, with an Edge TPU, Elf can operate on a solar panel 16.1× smaller while producing the aggregated counts with same confidence. Such miniaturization simplifies deployment of cameras and may suit them to low solar irradiance, or indoor environments.

Image capture hardware Efficient image capture as in our prototype (Figure 11) matters. When Elf performs image capture with the less efficient Cortex-A72, we measured the per-frame energy for capture is almost 10× compared to our prototype which uses

Cortex-M7 for capture. With less energy available for frame processing, Elf resorts to cheaper counters or fewer frames to sample, leading to much wider CIs (an increase of 35.6% – 59.0%) as shown in Figure 11 ("higher E_{cap}").

Efficient capture will be increasingly important as more efficient compute hardware emerges. We consider ultra low-power capture hardware [40] that reduces capture energy by 10× and estimate its impact as shown in Figure 11 ("lower E_{cap}"). Compared to Cortex-M7 used in our prototype, the ultra low-power capture reduces the mean CI width across all compute hardware options, with more significant reduction when accelerators in use: in the latter cases, image capture will contribute a higher fraction of system energy, or even become the bottleneck of the system energy.

8 RELATED WORK

Object counting is a key video query. It has been extensively studied in the computer vision literature and shown to enable other scientific investigations [2, 16, 36, 55, 62, 69, 76, 78, 81, 95, 106].

Energy harvesting systems A variety of systems, ranging from tiny embedded devices to datacenters, operate on harvested energy [43, 44, 50–52, 92, 96]. Elf shares their motivation and may further build atop some of their mechanisms, e.g., energy budget. Nevertheless, these prior systems never directly address approximate visual analytics as Elf does.

Battery-free cameras take a radical approach toward miniaturization [7, 73–75, 77]. With frugal energy available on device, these cameras are often restricted to occasionally sending out captured images or running lightweight compute such as background subtraction. By contrast, Elf targets battery-powered cameras (§2); with orders of magnitude more energy, these cameras can run richer analytics built on more capable NNs. Elf therefore explores a new design space disparate from that of battery-free cameras.

Specialized hardware Systems like XNOR.ai AI Camera [7] and RedEye [63] embrace hardware specialized for NN or vision. The gained efficiency may shift some design parameters of Elf, e.g., operating on smaller solar panels or smaller capacitors as discussed in Section 7.4. These systems, however, do not eliminate the need for approximate analytics, as vision algorithms are still racing to higher accuracy at higher compute expense.

Optimizing video analytics Many systems have been proposed for video analytics, being real-time [39, 57, 58, 67, 83, 97–99, 105, 107] or retrospective [56, 60, 82, 101, 102]. In most of these systems, compute depends on edge/cloud infrastructures, as opposed to running solely on device which is needed by autonomous cameras. Most, if not all, prior systems focus on per-objects results instead of statistical summaries of videos, as discussed in Section 1.

Background subtraction is a common technique for skipping similar frames without full-fledged processing [56, 60]. It monitors if adjacent frames captured at higher frame rate (e.g., 1FPS) contain mostly the same pixels. Elf can barely use background subtraction for skipping any frames: Elf samples frames sparsely in time (e.g., one per minute); therefore, adjacent frames often differ on substantial pixels.

Answering query with approximation and sampling Approximate query processing (AQP) [49] speeds up queries over large data. Typical AQP approaches include online aggregation (OLA) [32, 45, 54, 80] and offline synopses generation [28, 30, 42]. Besides, many sampling strategies [27, 35, 42, 91, 104] have been proposed to improve performance. AQP systems often answer queries with approximation and bounded error as Elf does. However, they are mostly designed for relational data but not videos; they do not run inaccurate operators (e.g., NNs) on data. They do not face many challenges as Elf does, such as integrating multiple errors and operating under energy budget.

9 CONCLUSIONS

Elf is an analytics system to answer object counting queries on autonomous cameras. Elf combines inaccurate NNs and sampling technique for video queries. It contributes a novel mechanism to make on-the-fly count decisions within and across multiple time windows. It takes a novel approach to integrating errors from different sources. Tested on large videos, Elf provides good estimation of object counts with bounded, narrow errors.

ACKNOWLEDGMENT

We thank the anonymous reviewers and our shepherd, Dr. Aakanksha Chowdhery, for their valuable feedback. The authors affiliated with Peking University were supported by the National Key R&D Program of China under the grant number 2018YFB1004800, the National Natural Science Foundation of China under grant number 61725201, the R&D projects in key areas of Guangdong Province under grant number 2020B010164002, the Beijing Outstanding Young Scientist Program under grant number BJJWZYJH01201910001004, and partially supported by the Key Laboratory of Intelligent Passenger Service of Civil Aviation. The authors thank NVIDIA for GPU donation.

REFERENCES

[1] 2017. 50% of Traffic Lights to Run on Solar Energy this Year. https://thecostaricanews.com/50-traffic-lights-run-solar-energy-year/.
[2] 2017. First Workshop on Video Analytics in Public Safety. https://www.nist.gov/sites/default/files/documents/2017/01/19/ir8164.pdf.
[3] 2017. OpenAI Baselines: ACKTR & A2C. https://openai.com/blog/baselines-acktr-a2c/.
[4] 2018. Low Power Wide Area Network Market Size. https://www.gminsights.com/industry-analysis/low-power-wide-area-network-lpwan-market.
[5] 2019. digitalanimal's cattle tracking system. https://digitanimal.com/cattle/?lang=en.
[6] 2019. Edge TPU. https://cloud.google.com/edge-tpu/.
[7] 2019. The first ever battery-free AI technology. https://www.xnor.ai/solar-powered-ai.
[8] 2019. Intel Neural Compute Stick 2. https://software.intel.com/en-us/neural-compute-stick.
[9] 2019. Jetson Nano Developer Kit. https://developer.nvidia.com/embedded/jetson-nano-developer-kit.
[10] 2019. National Solar Radiation Database. https://nsrdb.nrel.gov/.
[11] 2019. NNPACK-accelerated Darknet. https://github.com/digitalbrain79/darknet-nnpack.
[12] 2019. OpenCV 3.3. https://opencv.org/opencv-3-3/.
[13] 2019. Postscapes' cattle tracking system. https://www.postscapes.com/cattle-tracking-systems/.
[14] 2019. Solar Powered Security Camera Buyer's Guide. https://reolink.com/solar-powered-security-cameras-buying-guide/.
[15] 2019. A Tale of Reversing an Embedded System. https://www.defcon.org/images/defcon-21/dc-21-presentations/Manning-Lanier/DEFCON-21-Manning-Lanier-GoPro-or-GTFO-Updated.pdf.
[16] 2019. Traffic Video Analytics - a case report. https://www.microsoft.com/en-us/research/publication/traffic-video-analytics-case-study-report/.
[17] 2019. US Highway 101 Dataset. https://www.fhwa.dot.gov/publications/research/operations/07030/index.cfm.

[18] 2019. Vehicle prediction using tensorflow object counting API. https://github.com/ahmetozlu/vehicle_counting_tensorflow.

[19] 2019. Wyze Camera v2 1080p. https://www.wyze.com/product/wyze-cam-v2/.

[20] 2019. YI Home Camera. https://www.amazon.com/YI-Security-Surveillance-Monitor-Android/dp/B01CW4AR9K.

[21] 2019. Youtube live streaming: Auburn. https://www.youtube.com/watch?v=hMYIc5ZPJL4.

[22] 2019. Youtube live streaming: Cross. https://www.youtube.com/watch?v=049ltZb9JP8.

[23] 2019. Youtube live streaming: Hampton. https://www.youtube.com/watch?v=y3NOhpkoR-w.

[24] 2019. Youtube live streaming: Jackson Town. https://www.youtube.com/watch?v=1EiC9bvVGnk.

[25] 2019. Youtube live streaming: Taipei. https://www.youtube.com/watch?v=1y5dcfnv-Ss.

[26] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A System for Large-Scale Machine Learning. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). USENIX Association, Savannah, GA, 265–283. https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi

[27] Swarup Acharya, Phillip B Gibbons, and Viswanath Poosala. 2000. Congressional samples for approximate answering of group-by queries. In Acm Sigmod Record, Vol. 29. ACM, 487–498.

[28] Swarup Acharya, Phillip B Gibbons, Viswanath Poosala, and Sridhar Ramaswamy. 1999. The Aqua approximate query answering system. In ACM Sigmod Record, Vol. 28. ACM, 574–576.

[29] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne. 2017. Understanding the Limits of LoRaWAN. IEEE Communications Magazine 55, 9 (Sep. 2017), 34–40. https://doi.org/10.1109/MCOM.2017.1600613

[30] Pankaj K Agarwal, Graham Cormode, Zengfeng Huang, Jeff M Phillips, Zhewei Wei, and Ke Yi. 2013. Mergeable summaries. ACM Transactions on Database Systems (TODS) 38, 4 (2013), 26.

[31] Sameer Agarwal, Henry Milner, Ariel Kleiner, Ameet Talwalkar, Michael Jordan, Samuel Madden, Barzan Mozafari, and Ion Stoica. 2014. Knowing when you're wrong: building fast and reliable approximate query processing systems. In Proceedings of the 2014 ACM SIGMOD international conference on Management of data. ACM, 481–492.

[32] Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, and Ion Stoica. 2013. BlinkDB: Queries with Bounded Errors and Bounded Response Times on Very Large Data. In Proceedings of the 8th ACM European Conference on Computer Systems (Prague, Czech Republic) (EuroSys '13). ACM, New York, NY, USA, 29–42. https://doi.org/10.1145/2465351.2465355

[33] Nitin Agrawal and Ashish Vulimiri. 2017. Low-Latency Analytics on Colossal Data Streams with SummaryStore. In Proceedings of the 26th Symposium on Operating Systems Principles (Shanghai, China) (SOSP '17). ACM, New York, NY, USA, 647–664. https://doi.org/10.1145/3132747.3132758

[34] Aloÿs Augustin, Jiazi Yi, Thomas Clausen, and William Townsley. 2016. A study of LoRa: Long range & low power networks for the internet of things. Sensors 16, 9 (2016), 1466.

[35] Brian Babcock, Surajit Chaudhuri, and Gautam Das. 2003. Dynamic sample selection for approximate query processing. In Proceedings of the 2003 ACM SIGMOD international conference on Management of data. ACM, 539–550.

[36] David Beymer, Philip McLauchlan, Benjamin Coifman, and Jitendra Malik. 1997. A real-time computer vision system for measuring traffic parameters. In Proceedings of IEEE conference on computer vision and pattern recognition. IEEE, 495–501.

[37] Anil Bhattacharyya. 1943. On a measure of divergence between two statistical populations defined by their probability distributions. Bull. Calcutta Math. Soc. 35 (1943), 99–109.

[38] Debojit Biswas, Hongbo Su, Chengyi Wang, Jason Blankenship, and Aleksandar Stevanovic. 2017. An automatic car counting system using OverFeat framework. Sensors 17, 7 (2017), 1535.

[39] Christopher Canel, Thomas Kim, Giulio Zhou, Conglong Li, Hyeontaek Lim, David G. Andersen, Michael Kaminsky, and Subramanya R. Dulloor. 2019. Scaling Video Analytics on Constrained Edge Nodes. In Proceedings of the 2nd SysML Conference (Palo Alto, California, USA). 12.

[40] Ismail Cevik, Xiwei Huang, Hao Yu, Mei Yan, and Suat Ay. 2015. An ultra-low power CMOS image sensor with on-chip energy harvesting and power management capability. Sensors 15, 3 (2015), 5531–5554.

[41] Kaushik Chakrabarti, Minos Garofalakis, Rajeev Rastogi, and Kyuseok Shim. 2001. Approximate query processing using wavelets. The VLDB Journal—The International Journal on Very Large Data Bases 10, 2-3 (2001), 199–223.

[42] Surajit Chaudhuri, Gautam Das, and Vivek Narasayya. 2007. Optimized stratified sampling for approximate query processing. ACM Transactions on Database Systems (TODS) 32, 2 (2007), 9.

[43] Alexei Colin, Graham Harvey, Brandon Lucia, and Alanson P Sample. 2016. An energy-interference-free hardware-software debugger for intermittent energy-harvesting systems. ACM SIGPLAN Notices 51, 4 (2016), 577–589.

[44] Alexei Colin, Emily Ruppel, and Brandon Lucia. 2018. A reconfigurable energy storage architecture for energy-harvesting devices. In Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems. 767–781.

[45] Tyson Condie, Neil Conway, Peter Alvaro, Joseph M. Hellerstein, Khaled Elmeleegy, and Russell Sears. 2010. MapReduce Online. In Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation (San Jose, California) (NSDI'10). USENIX Association, Berkeley, CA, USA, 21–21. http://dl.acm.org/citation.cfm?id=1855711.1855732

[46] Wilfrid J Dixon and Frank J Massey Jr. 1951. Introduction to statistical analysis. (1951).

[47] Maroto-Molina Francisco, Navarro-García Jorge, Príncipe-Aguirre Karen, Gómez-Maqueda Ignacio, Guerrero-Ginel Jose, Garrido-Varo Ana, and Pérez-Marín Dolores. 2019. A Low-Cost IoT-Based System to Monitor the Location of a Whole Herd. Sensors (2019). https://doi.org/10.3390/s19102298

[48] Takashi Furuya and Camillo J Taylor. 2014. Road intersection monitoring from video with large perspective deformation. Ph.D. Dissertation. University of Pennsylvania.

[49] Minos N Garofalakis and Phillip B Gibbons. 2001. Approximate Query Processing: Taming the TeraBytes.. In VLDB. 343–352.

[50] Graham Gobieski, Brandon Lucia, and Nathan Beckmann. 2019. Intelligence beyond the edge: Inference on intermittent embedded systems. In Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems. ACM, 199–213.

[51] Íñigo Goiri, William Katsak, Kien Le, Thu D Nguyen, and Ricardo Bianchini. 2014. Designing and managing data centers powered by renewable energy. IEEE Micro 34, 3 (2014), 8–16.

[52] Íñigo Goiri, Kien Le, Thu D Nguyen, Jordi Guitart, Jordi Torres, and Ricardo Bianchini. 2012. GreenHadoop: leveraging green energy in data-processing frameworks. In Proceedings of the 7th ACM european conference on Computer Systems. ACM, 57–70.

[53] Moeen Hassanalieragh, Tolga Soyata, Andrew Nadeau, and Gaurav Sharma. 2016. UR-SolarCap: An Open Source Intelligent Auto-Wakeup Solar Energy Harvesting System for Supercapacitor-Based Energy Buffering. IEEE Access 4 (2016), 542–557.

[54] Joseph M. Hellerstein, Peter J. Haas, and Helen J. Wang. 1997. Online Aggregation. In Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data (Tucson, Arizona, USA) (SIGMOD '97). ACM, New York, NY, USA, 171–182. https://doi.org/10.1145/253260.253291

[55] Jarrod C Hodgson, Shane M Baylis, Rowan Mott, Ashley Herrod, and Rohan H Clarke. 2016. Precision wildlife monitoring using unmanned aerial vehicles. Scientific reports 6 (2016), 22574.

[56] Kevin Hsieh, Ganesh Ananthanarayanan, Peter Bodik, Shivaram Venkataraman, Paramvir Bahl, Matthai Philipose, Phillip B. Gibbons, and Onur Mutlu. 2018. Focus: Querying Large Video Datasets with Low Latency and Low Cost. In 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18). USENIX Association, Carlsbad, CA. https://www.usenix.org/conference/osdi18/presentation/hsieh

[57] Samvit Jain, Junchen Jiang, Yuanchao Shu, Ganesh Ananthanarayanan, and Joseph Gonzalez. 2018. ReXCam: Resource-Efficient, Cross-Camera Video Analytics at Enterprise Scale. CoRR abs/1811.01268 (2018). arXiv:1811.01268 http://arxiv.org/abs/1811.01268

[58] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. 2018. Chameleon: Scalable Adaptation of Video Analytics. In Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication (Budapest, Hungary) (SIGCOMM '18). ACM, New York, NY, USA, 253–266. https://doi.org/10.1145/3230543.3230574

[59] Daniel Kang, Peter Bailis, and Matei Zaharia. 2018. BlazeIt: Fast Exploratory Video Queries using Neural Networks. arXiv preprint arXiv:1805.01046 (2018).

[60] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. 2017. NoScope: Optimizing Neural Network Queries over Video at Scale. Proc. VLDB Endow. 10, 11 (Aug. 2017), 1586–1597. https://doi.org/10.14778/3137628.3137664

[61] Sitanshu Sekhar Kar and Archana Ramalingam. 2013. Is 30 the magic number? Issues in sample size estimation. National Journal of Community Medicine 4, 1 (2013), 175–179.

[62] P Karpagavalli and AV Ramprasad. 2013. Estimating the density of the people and counting the number of people in a crowd environment for human safety. In 2013 International Conference on Communication and Signal Processing. IEEE, 663–667.

[63] Robert LiKamWa, Yunhui Hou, Julian Gao, Mia Polansky, and Lin Zhong. 2016. RedEye: analog ConvNet image sensor architecture for continuous mobile vision. ACM SIGARCH Computer Architecture News 44, 3 (2016), 255–266.

[64] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In European conference on computer vision. Springer, 740–755.

[65] Alan J Lipton, Peter L Venetianer, Niels Haering, Paul C Brewer, Weihong Yin, Zhong Zhang, Li Yu, Yongtong Hu, Gary W Myers, Andrew J Chosak, et al. 2015. Video analytics for retail business process monitoring. US Patent 9,158,975.

[66] Fei Liu, Zhiyuan Zeng, and Rong Jiang. 2017. A video-based real-time adaptive vehicle-counting system for urban roads. PloS one 12, 11 (2017), e0186098.

[67] Peng Liu, Bozhao Qi, and Suman Banerjee. 2018. EdgeEye: An Edge Service Framework for Real-time Intelligent Video Analytics. In Proceedings of the 1st International Workshop on Edge Systems, Analytics and Networking (Munich, Germany) (EdgeSys'18). ACM, New York, NY, USA, 1–6. https://doi.org/10.1145/3213344.3213345

[68] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. Ssd: Single shot multibox detector. In European conference on computer vision. Springer, 21–37.

[69] Xu Liu, Zilei Wang, Jiashi Feng, and Hongsheng Xi. 2016. Highway vehicle counting in compressed domain. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 3016–3024.

[70] Aleksander Maricq, Dmitry Duplyakin, Ivo Jimenez, Carlos Maltzahn, Ryan Stutsman, and Robert Ricci. 2018. Taming performance variability. In 13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18). 409–425.

[71] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. In Proceedings of The 33rd International Conference on Machine Learning (Proceedings of Machine Learning Research), Maria Florina Balcan and Kilian Q. Weinberger (Eds.), Vol. 48. PMLR, New York, New York, USA, 1928–1937. http://proceedings.mlr.press/v48/mniha16.html

[72] Habibzadeh Mohamadhadi, Moeen Hassanalieragh, Akihiro Ishikawa, Tolga Soyata, and Gaurav Sharma. 2017. Hybrid Solar-Wind Energy Harvesting for Embedded Applications: Supercapacitor-Based System Architectures and Design Tradeoffs. IEEE Circuits & Systems Magazine 17, 4 (2017), 29–63.

[73] Saman Naderiparizi, Mehrdad Hessar, Vamsi Talla, Shyamnath Gollakota, and Joshua R Smith. 2018. Towards battery-free {HD} video streaming. In 15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18). 233–247.

[74] Saman Naderiparizi, Aaron N Parks, Zerina Kapetanovic, Benjamin Ransford, and Joshua R Smith. 2015. WISPCam: A battery-free RFID camera. In 2015 IEEE International Conference on RFID (RFID). IEEE, 166–173.

[75] Saman Naderiparizi, Yi Zhao, James Youngquist, Alanson P Sample, and Joshua R Smith. 2015. Self-localizing battery-free cameras. In Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing. ACM, 445–449.

[76] Milind Naphade, David C Anastasiu, Anuj Sharma, Vamsi Jagrlamudi, Hyeran Jeon, Kaikai Liu, Ming-Ching Chang, Siwei Lyu, and Zeyu Gao. 2017. The nvidia ai city challenge. In 2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI). IEEE, 1–6.

[77] Shree K Nayar, Daniel C Sims, and Mikhail Fridberg. 2015. Towards self-powered cameras. In 2015 IEEE International Conference on Computational Photography (ICCP). IEEE, 1–10.

[78] Mohammad Sadegh Norouzzadeh, Anh Nguyen, Margaret Kosmala, Alexandra Swanson, Meredith S Palmer, Craig Packer, and Jeff Clune. 2018. Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning. Proceedings of the National Academy of Sciences 115, 25 (2018), E5716–E5725.

[79] Min-hwan Oh, Peder A Olsen, and Karthikeyan Natesan Ramamurthy. 2019. Crowd counting with decomposed uncertainty. arXiv preprint arXiv:1903.07427 (2019).

[80] Niketan Pansare, Vinayak R Borkar, Chris Jermaine, and Tyson Condie. 2011. Online aggregation for large mapreduce jobs. Proc. VLDB Endow 4, 11 (2011), 1135–1145.

[81] Jason Remington Parham, Jonathan Crall, Charles Stewart, Tanya Berger-Wolf, and Daniel Rubenstein. 2017. Animal population censusing at scale with citizen science and photographic identification. In 2017 AAAI Spring Symposium Series.

[82] Alex Poms, Will Crichton, Pat Hanrahan, and Kayvon Fatahalian. 2018. Scanner: Efficient Video Analysis at Scale. ACM Trans. Graph. 37, 4, Article 138 (July 2018), 13 pages. https://doi.org/10.1145/3197517.3201394

[83] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen. 2018. DeepDecision: A Mobile Deep Learning Framework for Edge Video Analytics. In IEEE INFOCOM 2018 - IEEE Conference on Computer Communications. 1421–1429. https://doi.org/10.1109/INFOCOM.2018.8485905

[84] Joseph Redmon and Ali Farhadi. 2017. YOLO9000: better, faster, stronger. In Proceedings of the IEEE conference on computer vision and pattern recognition. 7263–7271.

[85] Joseph Redmon and Ali Farhadi. 2018. Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767 (2018).

[86] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems. 91–99.

[87] Reuven Y Rubinstein and Dirk P Kroese. 2016. Simulation and the Monte Carlo method. Vol. 10. John Wiley & Sons.

[88] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 4510–4520.

[89] S. Sheik Mohammed Ali, B. George, L. Vanajakshi, and J. Venkatraman. 2012. A Multiple Inductive Loop Vehicle Detection System for Heterogeneous and Lane-Less Traffic. IEEE Transactions on Instrumentation and Measurement 61, 5 (May 2012), 1353–1360. https://doi.org/10.1109/TIM.2011.2175037

[90] Honghui Shi. 2018. Geometry-aware traffic flow analysis by detection and tracking. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 116–120.

[91] Lefteris Sidirourgos, PA Boncz, ML Kersten, et al. 2011. Sciborq: Scientific data management with bounds on runtime and quality. (2011).

[92] Rahul Singh, David Irwin, Prashant Shenoy, and Kadangode K Ramakrishnan. 2013. Yank: Enabling green data centers to pull the plug. In Presented as part of the 10th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 13). 143–155.

[93] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In Proceedings of the 12th International Conference on Neural Information Processing Systems (Denver, CO) (NIPS'99). MIT Press, Cambridge, MA, USA, 1057–1063. http://dl.acm.org/citation.cfm?id=3009657.3009806

[94] Madjid Tavana and Srikanta Patnaik. 2018. Recent Developments in Data Science and Business Analytics. Springer.

[95] Jan C van Gemert, Camiel R Verschoor, Pascal Mettes, Kitso Epema, Lian Pin Koh, and Serge Wich. 2014. Nature conservation drones for automatic localization and counting of animals. In European Conference on Computer Vision. Springer, 255–270.

[96] Deepak Vasisht, Zerina Kapetanovic, Jongho Won, Xinxin Jin, Ranveer Chandra, Sudipta Sinha, Ashish Kapoor, Madhusudhan Sudarshan, and Sean Stratman. 2017. Farmbeats: An iot platform for data-driven agriculture. In 14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17). 515–529.

[97] Chengcheng Wan, Muhammad Santriaji, Eri Rogers, Henry Hoffmann, Michael Maire, and Shan Lu. 2019. ALERT: Accurate Anytime Learning for Energy and Timeliness. arXiv preprint arXiv:1911.00119 (2019).

[98] Junjue Wang, Brandon Amos, Anupam Das, Padmanabhan Pillai, Norman Sadeh, and Mahadev Satyanarayanan. 2017. A Scalable and Privacy-Aware IoT Service for Live Video Analytics. In Proceedings of the 8th ACM on Multimedia Systems Conference (Taipei, Taiwan) (MMSys'17). ACM, New York, NY, USA, 38–49. https://doi.org/10.1145/3083187.3083192

[99] Junjue Wang, Ziqiang Feng, Zhuo Chen, Shilpa George, Mihir Bala, Padmanabhan Pillai, Shao-Wen Yang, and Mahadev Satyanarayanan. 2018. Bandwidth-Efficient Live Video Analytics for Drones Via Edge Computing. In 2018 IEEE/ACM Symposium on Edge Computing, SEC 2018, Seattle, WA, USA, October 25-27, 2018. 159–173. https://doi.org/10.1109/SEC.2018.00019

[100] Peter Wei, Haocong Shi, Jiaying Yang, Jingyi Qian, Yinan Ji, and Xiaofan Jiang. 2019. City-scale vehicle tracking and traffic flow estimation using low frame-rate traffic cameras. In Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers. ACM, 602–610.

[101] Mengwei Xu, Tiantu Xu, Yunxin Liu, Xuanzhe Liu, Gang Huang, and Felix Xiaozhu Lin. 2019. Supporting Video Queries on Zero-Streaming Cameras. arXiv preprint arXiv:1904.12342 (2019).

[102] Tiantu Xu, Luis Materon Botelho, and Felix Xiaozhu Lin. 2019. VStore: A Data Store for Analytics on Large Videos. In Proceedings of the Fourteenth EuroSys Conference 2019 (Dresden, Germany) (EuroSys '19). ACM, New York, NY, USA, Article 16, 17 pages. https://doi.org/10.1145/3302424.3303971

[103] Taro Yamane. 1973. Statistics: An introductory analysis. (1973).

[104] Ying Yan, Liang Jeff Chen, and Zheng Zhang. 2014. Error-bounded sampling for analytics on big sparse data. Proceedings of the VLDB Endowment 7, 13 (2014), 1508–1519.

[105] S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, and Q. Li. 2017. LAVEA: Latency-Aware Video Analytics on Edge Computing Platform. In 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS). 2573–2574. https://doi.org/10.1109/ICDCS.2017.182

[106] B Yogameena and C Nagananthini. 2017. Computer vision based crowd disaster avoidance system: A survey. International journal of disaster risk reduction 22 (2017), 95–129.

[107] Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodik, Matthai Philipose, Paramvir Bahl, and Michael J. Freedman. 2017. Live Video Analytics at Scale with Approximation and Delay-Tolerance. In 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17). USENIX Association, Boston, MA, 377–392. https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/zhang

[108] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. 2016. Single-image crowd counting via multi-column convolutional neural network. In Proceedings of the IEEE conference on computer vision and pattern recognition. 589–597.