Joltik: Enabling Energy-Efficient "Future-Proof" Analytics on Low-Power Wide-Area Networks

Submitted in partial fulfillment of the requirements for

the degree of

Master of Science

in

Information Networking

Mingran Yang

B.S., Electronic and Information Science and Technology, Fudan University

Carnegie Mellon University
Pittsburgh, PA

May, 2020

Acknowledgements

First and foremost, I would like to thank my thesis advisor Prof. Vyas Sekar for his dedicated help and guidance over the past two years. Vyas is the person who taught me the core idea of research, and also the person who motivated me to do networking research (finally leads me towards the path of pursuing a Ph.D.). More importantly, Vyas always encouraged me to be confident and speak up, which will truly benefit me for long. I am very grateful for the opportunity to do research at Vyas's group.

Next, I would like to thank Prof. Swarun Kumar, Dr. Zaoxing (Alan) Liu, Junbo Zhang and Akshay Gadre for their collaborations on the *Joltik* project. I am deeply impressed by their insightful ideas and I learned a lot from working together with all of them. Thank Swarun and Alan for mentoring me, and thank Junbo and Akshay for being so helpful along the project.

Then, I would like to acknowledge the funding support for this work. This work was supported in part by the CONIX Research Center, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA, the Kavčić-Moura Endowment Fund, and a seed grant from the Scott Institute

I woud also like to thank Prof. Justine Sherry and Prof. Peter Steenkiste. "15-641 Computer Networks" is one of my favourite courses at CMU and this course sparked my interest in the area of networking. Also, being TA for Justine and Peter is truly a valuable experience.

I would like to thank Vyas's research group for their help, advice and feedback: thank you Zaoxing Liu, Yucheng Yin, Zinan Lin, Sekar Kulandaivel, Tianlong Yu, Guyue Liu, Soo-Jin Moon, Antonis Manousis, Daehyeok Kim, Rahul Sharma and

Hun Namkung.

I want to thank my friends in Cylab: Tian Li, Yucheng Yin, Zaoxing Liu and Weizhao Tang. I am very grateful for your help and support, and I will always miss the time we hangout together. I also want to thank Miao Yu for helping me understand concepts in computer security.

I want to thank all of the faculties and staff at the Information Networking Institute. I really treasured my experience at INI, both inside and outside of the classroom.

I also want to thank all my friends for being together with me along the journey. A special thank you to Weichao Duan for his support and companion along the way.

Finally, I would like to thank my family for their unconditional love. Thank you my father Bin Yang and my mother Zhiwen Yan, for always taking care of me, and for supporting each decision that I made.

Abstract

Wireless sensors have enabled a number of key applications. Despite many advances in sensing, computation and wireless technologies, simultaneously achieving *energy-efficiency*, *fidelity*, and *generality* across many (possibly unforeseen) metrics, has remained elusive. To this end, this paper presents *Joltik*, a framework enabling general and energy efficient analytics for low power wireless sensors. *Joltik* is built upon recent theoretical advances in *universal sketching*, which can enable a *Joltik* sensor node to report a compact summary of observed data to enable a large class of statistical summaries. We address key system design and implementation challenges with respect to communication, memory and computation bottlenecks that arise in practically realizing the potential benefits of universal sketching in the low-power regime.

We present a proof-of-concept evaluation of *Joltik* in LoRaWAN NUCLEO-L476RG boards and sensors and conduct experiments in a local testbed. Across a range of realistic datasets, *Joltik* provides significant reduction in energy cost compared to transmitting raw data and outperforms many natural alternatives (e.g., sub-sampling, custom sketches, compressed sensing) in terms of energy-accuracy tradeoffs.

Table of Contents

List of Tables

List of Figures

1

Introduction

Over the past several years, we have seen a significant interest towards developing the technology of wireless sensors. With this technology development, low-power Wide-Area Networks (LP-WANs) that span a large city or rural area are increasingly deployed for many real-world applications. For instance, these sensor networks can be used to do environmental monitoring or animal tracking [64]. Among all these applications, sensors are sensing various metrics of interest and send aggregate reports. Constrained by the environment or the sensor itself, these sensor networks typically operate over extended range (miles) on stringent battery constraints (e.g., ten-year battery life).

Even though modern sensing chips can energy-efficiently sense the environment at high sampling rates, storage and communication constraints restrict the devices to send the occasional point sample or short summary statistics (e.g., a mean value). More importantly, the exact list of statistics needed must be agreed upon *a priori*, given the raw data is too large to store in the sensing devices.

Yet, this requirement of choosing the required statistics up front is problematic, as we show in the following motivating scenario. As a solar farm, you have deployed

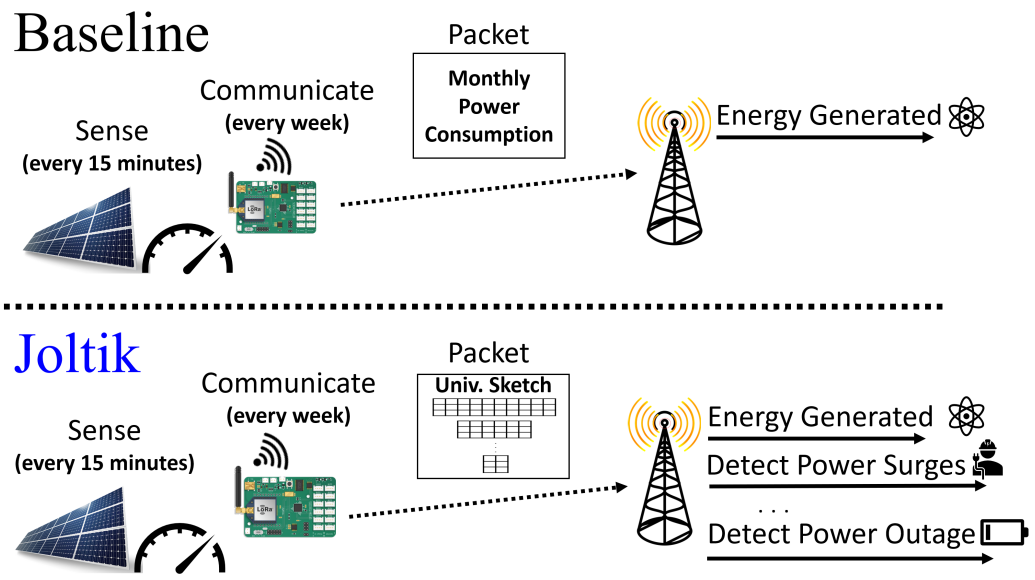


Figure 1.1: *Joltik* leverages universal sketches to generate future-proof general analytics for LP-WANs.

multiple solar sensors which periodically send amount of energy generated every few minutes. After deployment, however, new regulations or workload demands may require the operator to detect new types of anomalies or outages (e.g., weather events that might lead to blackouts) that require new kinds of statistical estimates over the raw data.

Unfortunately, today we do not have good methods to efficiently and accurately compute a broad spectrum of summary statistics. If the sensors cannot be reprogrammed in the field, the operators need to commit to a small set of metrics at design time and cannot support future requirements. Even if the sensor platforms are reprogrammable in the field, at any given time we may only be able to enable a small subset due to computation/energy constraints. Thus, there will be blind spots for metrics that are currently disabled.

Ideally, we want a solution that enables downstream sensor data analytics that: (1) is *general* to support many applications and *future-proof* to support possibly unforeseen statistics for future requirements; (2) computed with *high fidelity* (e.g., at

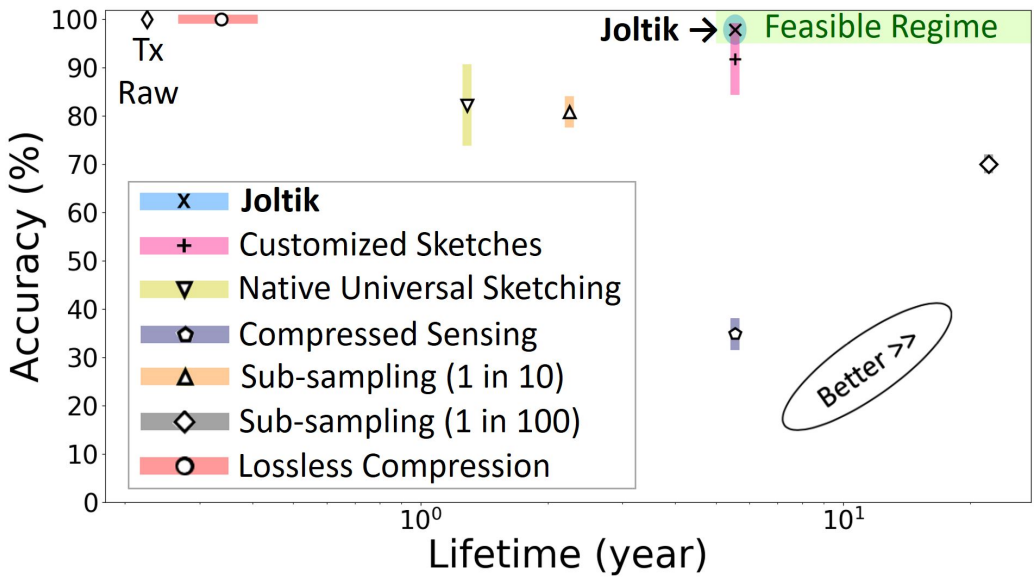


Figure 1.2: *Joltik* provides better energy-accuracy trade-off for "future-proof analytics" vs. prior approaches.

most 5% error); and (3) does not sacrifice client *energy efficiency* (i.e., support multi-year deployments on a single battery charge). At first glance, this seems impossible — barring sending raw data (which is energy inefficient due to the communication costs), it seems challenging to support a large spectrum of data analytics tasks on sensor data, let alone unforeseen requirements.

To this end, this thesis presents *Joltik*, a framework that can enable *general and energy-efficient* analytics for low power wireless sensors. *Joltik* builds on recent theoretical advances in *universal sketching* [10, 62, 61, 11]. At a high-level, sketching or streaming algorithms estimate (approximately) specific properties of a data stream, with provable memory-accuracy tradeoffs [3, 18, 15, 42, 39]. Unlike previous sketches that support a narrow metric of interest (e.g. heavy hitters [18, 15, 42], quantiles [1]), universal sketches can simultaneously support many (and possibly unforeseen) estimation tasks [10, 38]. Thus, they can serve as the fundamental basis for general "future-proof" sensor data summarization.

Realizing these potential benefits, however, raises significant practical challenges

in low-power sensor platforms. To see why, it is useful to understand the universal sketching algorithm at high level (see Chapter 2.4 for details). Given a stream of sensed inputs, each one of n unique values, the sketch iteratively subsamples the incoming stream $\log n$ times into smaller sub-streams using $\log n$ hash functions. For each of the $\log n$ sub-streams, we maintain a hashed counter array storing per-key counts (with some potential loss of information) in a data structure called a Count Sketch [15], which can be conceptually viewed as a counting Bloom filter with a custom counter update strategy.

Our contribution in designing and implementing *Joltik* is in identifying and addressing practical system bottlenecks of realizing the benefits of universal sketching in a low-power sensing context. Specifically, we identify three key bottlenecks. First, these devices have limited memory (few hundred kilobytes) to store the sketch data structures, and as such the native implementations of universal sketch are infeasible. Second, even though sketches are much smaller than transmitting raw data, this footprint is still too high especially in a lossy LP-WAN setting, and consumes over 90% of battery-life for LP-WAN sensors [19]. Third, the compute footprint is also high as it entails computing numerous hashes and counter updates on low-power devices.

To tackle the above challenges, we have three key system design ideas:

(1) **Efficient Storage (Chapter 4.1):** We design optimizations to ensure that the sketch data structure is compact (i.e., a few KBs) for low-power sensors. Instead of retaining the same number of counters per-layer in the universal sketch, our approach carefully provisions a smaller number of counters at lower layers resulting in significant memory savings.

(2) **Reducing Communication Cost (Chapter 4.2):** We design a custom compression scheme that relies on the natural structure of the sketch data structure

– i.e. only a small fraction of counters in the sketch are large, while the rest are small. Thus, we dynamically resize sketch counters prior to transmission to reduce communication cost.

(3) **Reducing Computation Cost (Chapter 4.3):** We refactor the counter update computation based on the insight that the only update which really matters for an element is the one to its final layer where an element would have been added. Thus, by only making updates to the final layer, we can halve the effective compute footprint of the sketching data structure.

Our end-to-end goal is to maximize sensor battery-life for a desired level of system accuracy and given the computation/memory constraints (Chapter 4.4). Building on the above design optimizations, we show how sensor operators can tune *Joltik*'s sketch parameters to suit application-specific accuracy requirements, while accounting for the energy costs and resource constraints.

We implement *Joltik* in LoRaWAN NUCLEO-L476RG boards and sensors and conduct proof-of-concept experiments in a LoRaWAN testbed spanning a university building. We further emulate a variety of sensor deployment scenarios using publicly available datasets for solar energy measurement, temperature, pressure and soil moisture. We demonstrate *Joltik*'s performance against a variety of baselines including compressed sensing, subsampling, specialized sketches, varied data compression strategies and sending raw data. Our evaluation demonstrates that: (1) Given an energy budget, *Joltik* achieves significant better accuracy when compared to all baselines (see Figure 1.2); (2) *Joltik* reduces storage requirements by 5× compared to traditional universal sketches without loss in accuracy; and (3) *Joltik* can measure multiple sensor statistics at the same time without extra energy cost.

Contributions: In summary, this thesis makes the following contributions:

- A novel architecture for enabling general, energy-efficient, and high fidelity

sensor analytics by observing an opportunity to leverage universal sketching (Chapter 3);

- A practical low-CPU, low memory footprint and low power realization of universal sketches on commodity sensor platforms (Chapter 4);
- An end-to-end system realization of *Joltik* on a real-world LoRaWAN testbed (Chapter 5) and demonstrating the benefits compared to alternatives (Chapter 6);

Thesis outline: Chapter 1 briefly introduces the existing problem in LP-WANs and discusses our contributions in solving this problem. Chapter 2 gives a motivation scenario and highlights the limitations of current approaches, and finally discusses the theoretical foundations underlying *Joltik*. Chapter 3 gives an overview on the *Joltik* system and analyzes the system challenges. Chapter 4 describes three design ideas in detail and introduces end-to-end system deployment. Chapter 5 introduces system implementation. In Chapter 6, we provide the evaluation result of *Joltik* system. The conclusions and future work are discussed in Chapter 7.

2

Background

We start a motivation scenario to show the design goal of *Joltik* system. Then we highlight the limitations of current approaches. We also introduce the related work and briefly discuss the theoretical foundations underlying *Joltik*.

2.1 Motivating Scenario

Let us look at an example of a solar farm to understand the *Joltik*'s motivation (Figure 2.1). Many companies are already moving towards low-power wireless platforms for monitoring solar power production. These meters are typically used to either send monthly energy generated [53] (i.e. total energy use) or detect day-night cycles, and therefore operate solely on battery power to function independent of the grid [33].

In addition, we may have other analytics tasks of interest; say detection of an anomaly or volatility of the voltage generated. This might be facilitated for example, by calculating the entropy of the drawn current. More generally, there may be multiple kinds of statistical summaries of interest computed over the raw sensor stream for various analytics tasks. Table 2.1 shows a subset of possible statistics for this application setting, with a caveat that this list is by no means exhaustive.

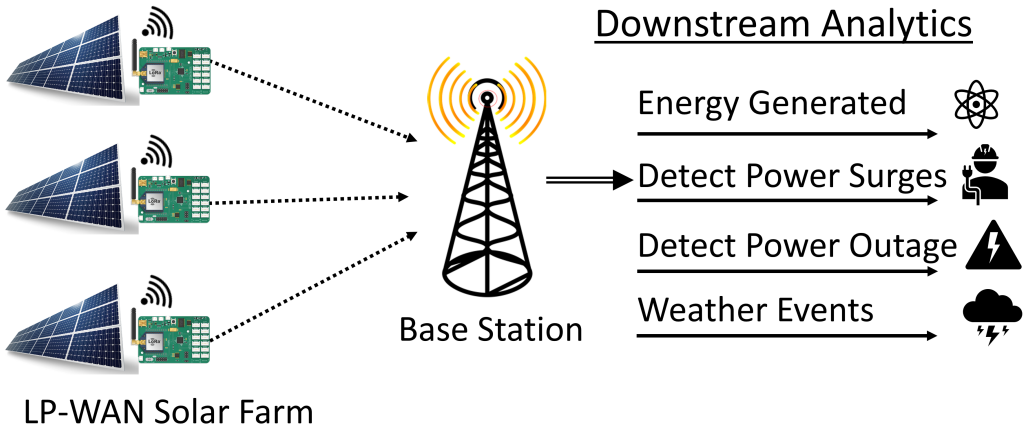


Figure 2.1: A motivating example: A solar farm may need general analytics for various applications.

Solar Sensor	**Metric**	**Definition**
Energy Generated	L1-norm (L_1)	$\sum f_i$
Power usage	α-Heavy Hitters	$f_i \geqslant \alpha \sum f_i$
Voltage Volatility	L2-Norm (L_2)	$\sum f_i^2$
Anomaly Detection	Entropy	$-\sum \frac{f_i}{L_1} \log \frac{f_i}{L_1}$
Weather Event	Change	$f_c \geqslant \alpha \sum f_c$
Voltage Range	Tail Detection	Quantiles
Power Outages	Zero-draw Time	$f_0 \geqslant \alpha \sum f_i$

Table 2.1: Metrics relevant to a solar farm

Today, it is challenging if not impossible to support such "general" analytics. The operators have to decide at deployment time which set of analytics tasks need to be supported. Since the low-power clients cannot compute all possible set of statistics, we may not be able to support a wide spectrum of downstream tasks.

Even if field reprogrammability were feasible, the operators still have to make some unfortunate runtime trade-offs. Since the sensors have only finite computation, storage and power resources, we may not be able to simultaneously run all possible services and some of these tasks will suffer from fundamental "blind spots".

Now, consider the scenario where a new type of analytical capability emerges, say to study voltage range of malfunctioning sensors or investigate the effect of weather

Approach	Energy	Accuracy	Generality
Sub-sampling [24, 26]	✓	×	✓
Compression [48, 35]	×	✓	✓
Sparse Recovery [43]	✓	✓	×
Data-centric Aggregation [59, 44]	✓	✓	×
Joltik	✓	✓	✓

Table 2.2: In contrast to prior work, *Joltik* guarantees energy-efficiency, accuracy and generality.

events. It is expensive to physically visit and reconfigure the clients manually. While over-the-air (OTA) updates [34] may address this partially for high-end platforms, many clients cannot support this capability due to the large power draw. Thus, this new capability will require essentially a significant overhaul of the already deployed infrastructure.

The above case study demonstrates the need for an approach which can provide *accurate, general, and possibly future-proof* analytics for wide array of metrics without constant manual intervention.

2.2 Strawman Solutions and Limitations

In this context we explore several strawman solutions and highlight their limitations when doing sensor analytics tasks.

Sub-Sampling [24, 26]: The company may circumvent computation limitation by allowing the sensors to select some k applications out of pool of all applications and then leverage spatial and temporal correlation over the sparse samples to estimate all metrics. Yet, this will still lead to lower fidelity and blind spots in measuring these metrics. Further, this approach will need the pool *a priori* which is not conducive to unknown applications.

Compression [48, 35]: Using lossless compression on raw data cannot be supported by these low power clients for two reasons. First, the clients transmits much less frequently than it senses and hence the small storage resources cannot

support high sampling rate. Second, many compression algorithms are computationally intensive affecting the power budget of the low power clients. If it uses lossy compression, it will sacrifice accuracy of the raw data which will also increase the error in estimating the metrics.

Sparse Recovery [43]: This class of approaches can operate efficiently at clients (e.g., compressive sensing is linear) without sacrificing accuracy like the above approaches. However, it makes assumptions about the sparsity of data which may not be true. Further, this assumption can miss out on the tail of the distribution which may affect its efficacy for many relevant metrics as we will see.

Sketching [3, 18, 15, 42] Summarizing the sensed data streams using sketching algorithms for specific functions would enable high fidelity data-independent estimates of a metric (such as count-min sketches[18] for heavy hitters). They typically rely on using hash functions to identify the frequency of various events. These hash functions are customized to store only the necessary details to compute a function. However, this approach still remains agnostic to possibly new functions that can typically be determined very easily on raw data but cannot be computed due to the hash-based compression. Furthermore, this approach is also not future-proof as the sketches cannot support other metrics of interest.

2.3 Related Work

The related work of this thesis mainly falls into the following three categories: low-power IoT, aggregation in sensor networks and sketching for data analytics.

2.3.1 Low-Power IoT

There has been much work done in Low-Power Wide-Area Networking (LP-WANs) for synchronization [2, 46], association [36, 22], optimizing power [4, 19], improving scalability [47, 20, 23], and client power adaptation [65]. Recent trends [32, 28]

demonstrate moving complex functions off the low-power client to the more powerful base stations. *Joltik* complements prior work by developing a novel approach to retrieve general data analytics from resource-starved low-power IoT clients.

Further, advances in energy-harvesting technologies have enabled intermittently powered devices [40, 54] which sense the environment periodically based on available energy. However, as intermittent sensors are essentially sub-sampling sensed data from the environment, they sacrifice accuracy due to missing out on important information when sleeping.

2.3.2 Aggregation in Sensor Networks

Retrieving information from large number of IoT clients or sensor nodes has been widely studied. While some approaches, such as compressed sensing [7, 43, 8, 60, 30, 25], leverage the sparsity of information to retrieve the data, other approaches use machine learning [58, 49] or statistical sampling [26, 57] to retrieve information from large number of sensors in a network. There is also a rich literature on lossless compression [56], such as dictionary based [48, 41] and predictive coding based [35, 29]. However, these solutions have the caveat of providing relatively modest compression without affecting the accuracy of statistics. The last approach for aggregation [31, 45, 5, 6, 50, 44, 59] in wireless sensor networks exploits the spatio-temporal correlations to minimize the information required to calculate specialized metrics. As shown in Table 2.2, *Joltik* complements these solutions by providing a generalized analytics framework that does not rely on assumptions about the sensed data.

2.3.3 Sketching for Data Analytics

Sketching algorithms for aggregate statistics have been explored in various contexts, including stream data processing [3, 42, 17, 9], database [15, 18, 15] and network

telemetry [66, 38, 37, 63]. In contrast, *Joltik* presents the first system to leverage the universal sketching paradigm for generalized aggregation in wireless low-power clients and making them storage, computation and energy efficient.

2.4 Background on Universal Sketching

The above discussion suggests that an ideal approach should enable high-fidelity estimates of relevant metrics without making any assumptions on the distribution of data and is energy-efficient. In this regard, recent advances in the theory of universal sketching [10, 12] appear promising.

At a high level, a universal sketch maintains a single sketch structure that can enable estimation for every function drawn from a broad class of functions instead of keeping one individual sketch per estimation task. More specifically, this class of estimation tasks can be represented in the following form: G-sum=$\sum g(f_i)$, where f_i is the frequency of the ith unique element. Fortunately, many natural statistical summarizations of interest fall within this family as seen in Table 2.1. The theory results show that if g is monotonic and upper bounded by $O(f_i^2)$, a single universal sketch can compute these G-sum functions. (A detailed analysis is outside our scope and we refer readers to the relevant references [10, 62, 12, 38].)

However, a conceptual understanding of the computation/memory structure of the sketch is relevant for us in order to use it in the *Joltik* context. Structurally, a universal sketch maintains multiple layers of "heavy hitter" sketches; i.e., items that have high frequencies.[1] Each sketch applies independent hash function h_j (returns 0 or 1) to the input data stream to subsample at every layer (from the previous layer). This enables them to give equal weightage to both most frequent as well as the tail of the histogram distribution. These layers then track the heavy hitters

[1] Preciously, universal sketch requires to track L_2-heavy hitters, defined as items whose frequencies are larger than some α fraction of L_2 for $0< \alpha <1$, where $L_2 = \sqrt{\sum f_i^2}$.

to identify the key contributors to the G-sum functions. As depicted in Figure 3.1, the intuition here is that the layered structure of universal sketch is designed for sampling representative elements with diverse frequencies and these elements can be used to estimate G-sum with bounded errors. If only one layer of heavy hitter sketch is used, we can only find frequent elements, lacking representatives from less frequent elements.

During the offline phase, we use the heavy-hitters at each layer and process the sketch iteratively from the bottom layer to the top. The top-layer can then be used to compute the desired statistic. Prior work has shown that this aggregation can be performed to be an unbiased estimator of G-sum [52, 38]. This enables universal sketches to provide general analytics for all G-sum functions at the base station without knowing them *a priori*.

Revisiting our solar farm scenario, this can be a good fit for the applications as shown in Table 2.1. The low-power sensor computes a universal sketch over sensed samples and report it to the base station significantly reducing the amount of data to be transmitted. The base station can then compute the metric of interest (e.g., energy generated) using the reconstruction algorithm.

While this is a promising starting point, there are practical challenges that arise due to the storage and energy (i.e., from computation and communication) constraints in a LP-WAN context. Our contribution in *Joltik* is to identify and address these problems as we discuss next.

3

Joltik System Overview

Joltik is a sensor analytics framework that simultaneously achieves *generality* to support a large range of metrics of interest, *fidelity* in estimating these diverse statistics, and *energy-efficiency* for optimized client battery life. *Joltik* achieves these three design goals at the same time by leveraging universal sketching algorithm (as introduced in Chapter 2.4). Yet, deploying universal sketching algorithm directly on resource-constrained sensor devices introduces additional technical challenges. In this chapter, we first introduce the workflow of *Joltik* system (Chapter 3.1), then discuss the system challenges (Chapter 3.2).

3.1 *Joltik* Workflow

Joltik system contains two main components, namely *Joltik* client (sensor node) and *Joltik* base station. The entire system operates as follows:

(1) **Configuration:** A *Joltik* client is configured only once during its operation to configure the behavior of the sketch during its life time. This decision is taken based on the battery-life and accuracy requirements for a particular

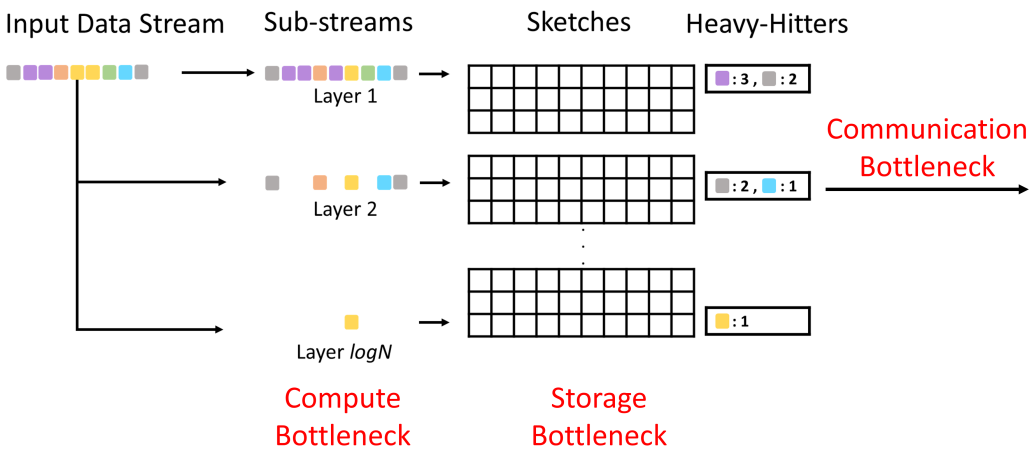


Figure 3.1: Native universal sketch workflow (e.g.,[38]).

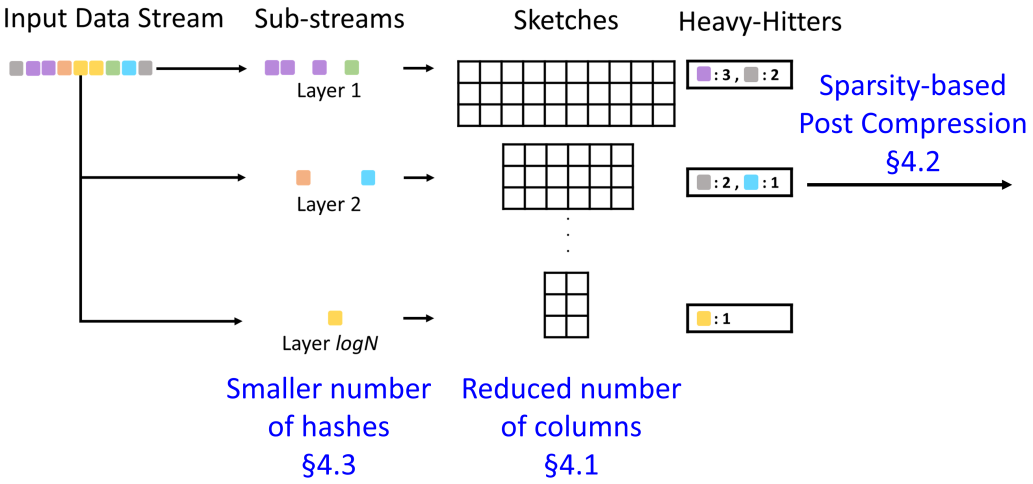


Figure 3.2: *Joltik* system workflow.

client. Based on the energy profile of the client, *Joltik* is configured to perform under a budget of total energy E and available memory M. This allows *Joltik* to provide an ϵ-additive error guarantee for all supported metric estimations based on the sketch size.

(2) **Sensing and Computing:** Every collected data sample from the sensor will be fed into the universal sketch on board using the embedded MCU. Instead

of storing the raw samples, *Joltik* only keeps a universal sketch for each measurement interval.

(3) **Communication to base station:** *Joltik* devices send the computed (and possibly compressed) sketch over the wireless medium infrequently; e.g., once per hour or per day. The base station uses this sketch to estimate various metrics needed for downstream analytics.

3.2 System Challenges

Figure 3.1 shows how native universal sketching works, and Figure 3.2 shows the workflow of *Joltik*. These two figures together address the three key bottlenecks (computation, storage and communication) in prior universal sketching implementations to enable accurate general analytics for low-power clients. In order to achieve the benefits of universal sketching in practice for low-power clients, *Joltik* addresses the following challenges as depicted in the figures:

- **Acheiving high fidelity with smaller memory footprint:** A canonical universal sketch [38] requires several hundreds of KBs or even a few MBs to obtain highly accurate results. However, low-power clients have an embedded MCU with limited on-chip memory (e.g., NUCLEO-L476RG LoRaWAN board has 128KB SRAM). Given this tight memory budget, we need a compact universal sketch but still provide high accuracy. We describe our approach to reduce the memory footprint in Chapter 4.1.

- **Optimizing communication with the base station:** Wireless communication is the first-order energy consumer in wireless sensor platforms. By transmitting the sketch data structure instead of raw data, we already reduce the communication cost to some extent. But every bit matters for low-power

wireless transmission. In Chapter 4.2, we present our approach to dynamically reduce the sketch counter sizes without affecting the accuracy of the estimated metrics.

- **Reducing energy footprint of sketch update:** Low-power MCUs have limited computation resources (e.g., Cortex-M3 CPU with 32MHz). Universal sketching implemented as-is requires multiple hash computations (e.g., 5-10 hashes) for each collected sample, inducing compute overhead and additional energy consumption. In Chapter 4.3, we identify a simple-yet-effective opportunity to halve the compute requirement without affecting accuracy.

In this chapter we introduce the workflow of *Joltik* and also the system design challenges. We will then dive into the detailed system design in the following chapter.

4

Detailed Design

In this chapter, we discuss our contributions in tackling the three challenges discussed in Chapter 3.2 (reducing the overall energy footprint and making universal sketches feasible on low-power sensor platforms, as shown in Figure 3.2). We also discuss how a sensor deployment can practically configure various *Joltik* system parameters to get the best accuracy-lifetime tradeoffs.

4.1 Reducing Memory Footprint

4.1.1 Problem

Recall that the universal sketching algorithm maintains multiple "L2 heavy hitter" instances for subsampled streams, and each instance needs a separate heap data structure to record the top heavy hitters for that particular substream. Prior work on universal monitoring (UnivMon) [38] presents a canonical realization of this approach by implementing several equal-sized count sketch [15] components.

If we leverage this native universal sketching implementation directly and try to reduce the memory size, we would need to reduce the memory footprint of each count sketch uniformly. By default, UnivMon allocates 600 KB memory for a 12-layer

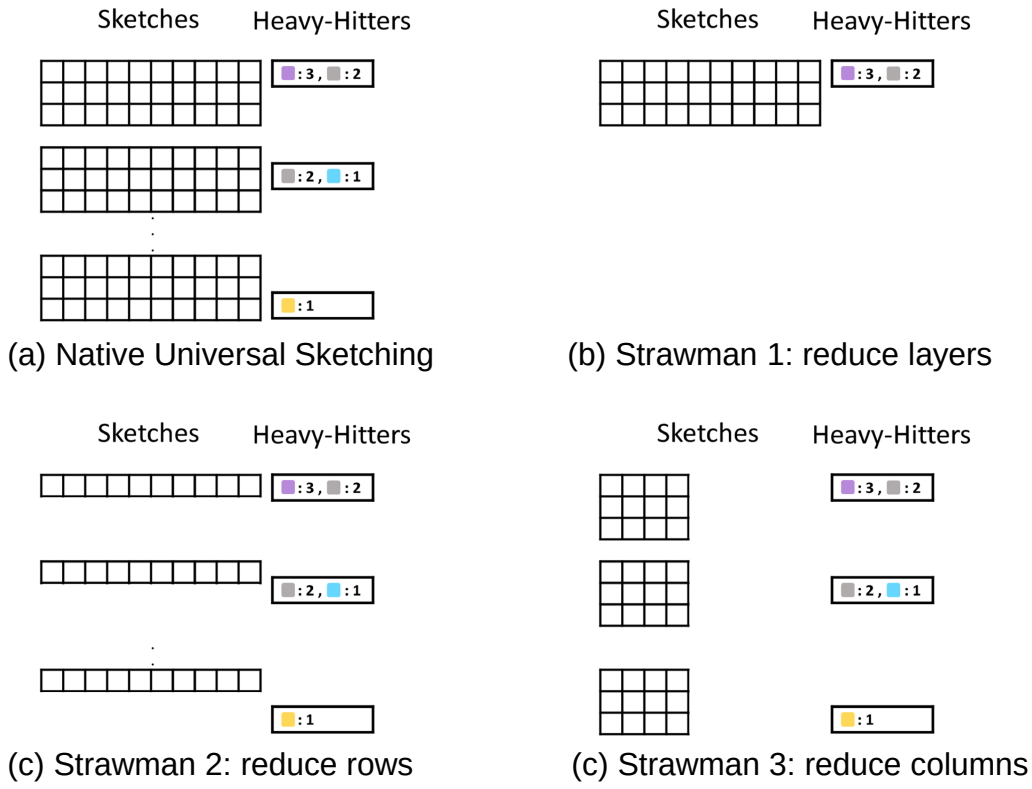


Figure 4.1: Starman solutions for reducing memory footprint.

sketch, with each count sketch taking 50KB memory. Reducing this size to 60KB for sensor deployment would mean that each count sketch only gets 5KB, causing massive hash collisions and significantly larger errors.

4.1.2 Strawman Solutions

Given that the memory reduction in native universal sketching algorithm causes significant errors, we can consider several other possible methods (Figure 4.1):

- **Strawman solution 1: reduce layers.** First, We can try to reduce the number of layers instead of smaller sketches per layer. However, the structure of universal sketch with $O(\log n)$ layers (for n unique elements) is key to maintain its generality and fidelity for supported statistics. Reducing the number of

layers will affect the algorithm's capabilities to detect less frequent elements and, in turn, the accuracy.

- **Strawman solution 2: reduce rows.** A second approach could be to reduce the number of rows of hash functions in each sketch. However, reducing the number of rows will affect the confidence interval of obtaining accurate results reducing the quality of results, leading to irrelevant failed results frequently (e.g., 1 failure out of 5 trials).

- **Strawman solution 3: reduce columns.** The third approach to reduce memory could be to reduce the number of columns in each sketch. However, reducing the number of columns will lead to more hash collisions in each layer, resulting in inaccurate heavy hitter estimation in each layer, and finally cause an overall accuracy degradation.

4.1.3 Our Approach: "Inverted Pyramid" Structure

Our approach to reduce memory footprint is by designing an "Inverted Pyramid" data structure, as shown in Figure 4.2. The main insight here is that the upper layers contribute significantly more to any G-sum function than the lower layers. In a sensor deployment, we argue that we can still enable high fidelity estimates by significantly reducing the size of lower layer sketches for two reasons: (1) The lower layer sketches are responsible for identifying the sizes of small "heavy hitters" (i.e., elements appear less frequently in the whole data but are relatively frequent in that particular layer) from lower-layer substreams. The errors from those infrequent elements have smaller influence on the final statistic than the more frequent elements. (2) The lower layers need to handle much smaller number of samples. Since the actual errors in estimating the heavy hitters are proportional to data samples, reducing the sizes of lower-layer sketches will not yield significantly higher errors than upper layers.

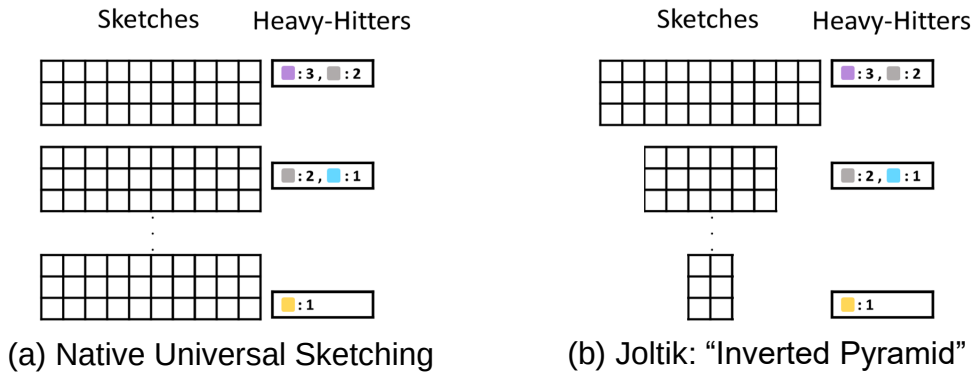


Figure 4.2: Our approach to reduce memory footprint: "Inverted Pyramid" structure.

With this insight, we maintain larger (i.e., higher fidelity) sketches for upper layers and smaller sketches for lower layers, as an "inverted pyramid" structure. We gradually reduce the number of columns in each sketch as we move to lower layers. Optimally tuning the relative sizes of each layer will be workload dependent. In practice, we choose this reduction to geometrically smaller number of columns at each layer. Our empirical evaluation in Chapter 6 of the above demonstrates that ratio of 1/2 provides a good energy-accuracy tradeoff. Thus, every layer is allocated 1/2 the number of columns as the previous layer.

4.1.4 Impact on Accuracy

To understand why our inverted pyramid memory allocation can preserve high accuracy in practice while significantly reducing the memory size, we analyze this strategy based on the accuracy bound of the Count Sketch [15] stated in Theorem 1 below:

Theorem 1 ([15]). *For $\delta > 0$, let f_i be the actual frequency and $\tilde{f}_i$ be the estimated frequency of element i in the dataset. The Count Sketch algorithm estimates $\tilde{f}_i = f_i \pm \epsilon L_2$ using $O(\log \frac{1}{\delta} \cdot \frac{1}{\epsilon^2})$ space with 1-δ probability, where L_2 is the L_2 norm of the vector with all element frequencies.*

Given Theorem 1, our mechanism does allocate smaller space for lower layers, leading to larger ϵ additive errors in the worst case. For instance, if we decrease the sketch size by $2\times$, then the ϵ error will increase by $O(\sqrt{2})$. However, this worst case error bound increase will not convert to a larger error in practice as real-world sensor datasets are naturally skewed. Intuitively, this is because even though ϵ error increases as memory reduces, this is mitigated by a simultaneous reduction in the value of L_2 that is multiplied with ϵ in the overall error bound in Theorem 1. Indeed, L_2 always decreases in lower layers because the subsampling will reduce the data size and "filter out" large heavy hitters in the lower layers with high probability. Across all of our tested sensor datasets, the decreasing rate of L_2 is much higher than ϵ's increasing rate, leading to minimal overall impact on accuracy.

4.2 Reducing Communication Footprint

4.2.1 Why Reducing Communication Footprint

Energy is a key constraint in wireless sensor deployment, and typically the communication component dominates the energy costs. For instance, on our experimentation platform with a NUCLEO-L476RG as MCU and SX1276 LoRa Transceiver as radio transceiver, our measurements show that $4,560,000$ computation cycles can be performed for the same energy usage as transmitting a single byte. Thus, to extend sensor lifetime given limited battery resources, we need to minimize the amount of data we need to transmit as much as possible.

We do note that by using sketching and the memory optimizations above, we have already achieved significant reduction in communication costs vs. sending raw data. Suppose a sensor operates at a sampling rate of 10 Hz, transmitting all raw data would result in 3.5 MB data to be communicated every day. Using even an unoptimized universal sketch requires only 300 KB per day; a $12\times$ reduction in energy consumption. Furthermore, our memory optimization (Chapter 4.1) provides

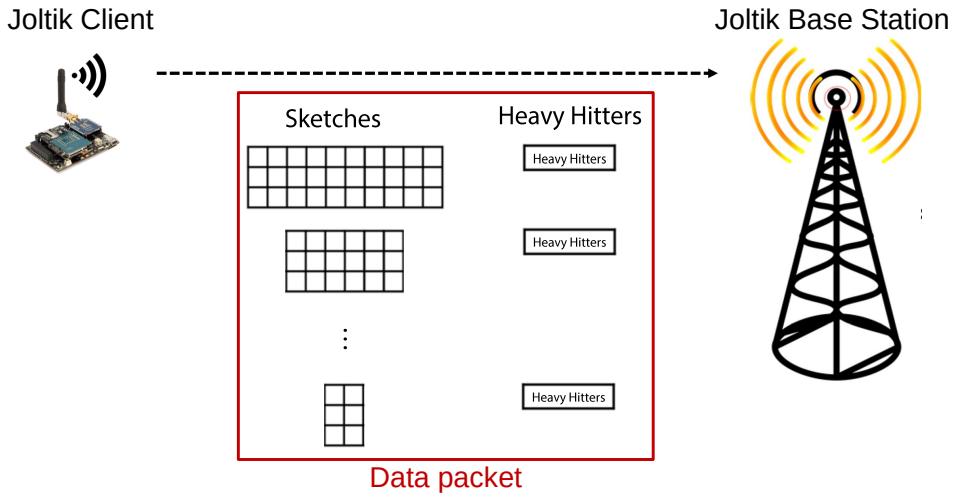


Figure 4.3: The transmitted data packet in *Joltik* system.

a further 5× reduction by using only a 60KB data structure. That said, even with these techniques in place, we want to explore further opportunities for reducing the amount of data to be transmitted due to the large asymmetry in the energy cost of computation vs. communication.

Recall that each layer of a sketch structure consists of two parts: a count sketch and a heavy hitter heap. (As shown in Figure 4.3) Next, we describe how we can effectively reduce the communication footprint of transmitting these two data structures from the sensor to the base station.

4.2.2 Lossless Encoding of Sketch Structure

When transmitting a sketch structure, it is important to use only lossless compression techniques, as the base station would need the exact sketch information from the sensor to calculate the application metrics accurately.

Joltik's compression of count sketches derives from a simple yet important observation that, in a certain count sketch, only a small portion of counters tend to have

5	3	7930	9	11
3	4	30	7	10
9	184	11	8	4
7	9	5	2859	3

Figure 4.4: An example of sketch counter (4 rows × 5 columns).

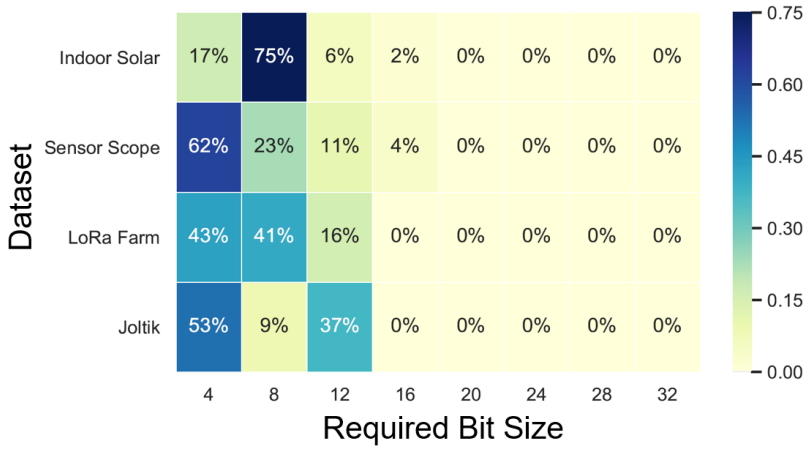


Figure 4.5: Percentage of counters with different required bit size: we test 30 runs on each of the four datasets with a sampling rate of 10 Hz, a data collection time of one day and a sketch structure of 60 KB in memory.

large values (Figure 4.4 shows a sketch counter example, and Figure 4.5 shows the percentage of required bit size). This indicates that one can compress a sketch by assigning different bit sizes to each counter.

Specifically, *Joltik* assigns two extra bits to indicate 4 levels of counter lengths, namely 4, 8, 12 and 16, and use corresponding number of bits to represent a certain counter (as shown in Figure 4.6). For example, 112 needs more than 4 bits, thus will be represented as 0101110000 (10 bits), where the first two bits indicate that counter length is 8-bit. Inevitably, for counter values which need more than 12 bits,

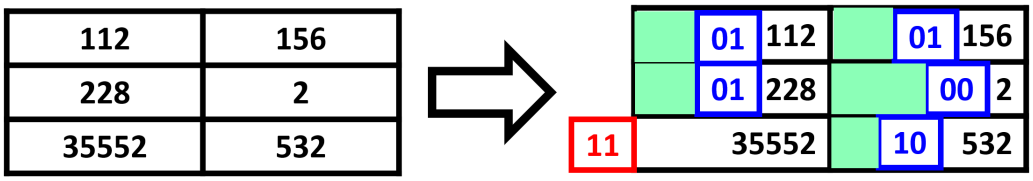


Figure 4.6: *Joltik* uses two bits to indicate counter length and represents each counter with corresponding number of bits.

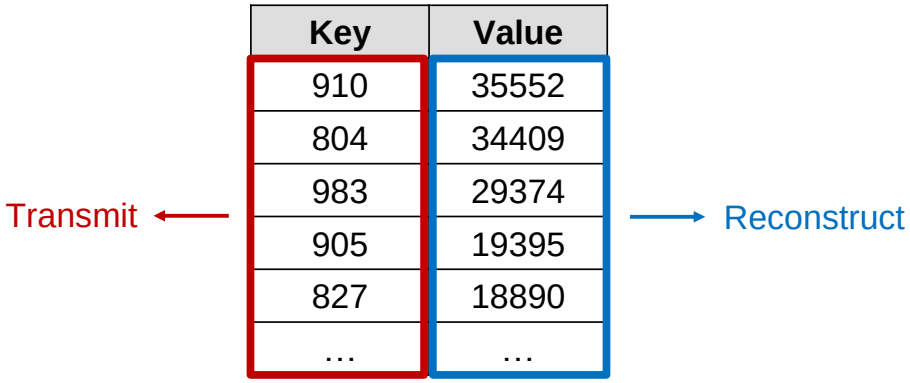


Figure 4.7: An example of *Joltik*'s heavy-hitters heap.

this method incurs extra costs. However, it is very unlikely for a well-conditioned count sketch to have a large number of such values, as this indicates that the count sketch is already near-saturated and would result in a very low accuracy.

While *Joltik*'s compression scheme appears very simple, we show in Chapter 6 that it outperforms several of the most prevalent lossless compression schemes. We also note that under *Joltik*'s compression scheme, compressing a single counter requires only about $1/10^5$ energy of transmitting it, which is worthwhile to perform.

4.2.3 Efficiently Transmitting Heavy-hitter Heap Data

As in original universal sketching algorithms, *Joltik* also maintains a heap in each layer, structured with key-value pairs, so that it is aware of the top heavy hitters (as shown in Figure 4.7). Yet, we note that it is sufficient to reconstruct a heap

without heap values, as long as the corresponding count sketch and heap keys are successfully recovered, since the heap values are just estimates yielded by the count sketch. Thus, *Joltik* only transmits heap keys when sending data, and leaves the task of reconstructing a complete heap to base stations. This discarding process does no harm to our sketching algorithm and will not influence its accuracy.

4.2.4 Impact on Accuracy

By construction, both of these optimizations are lossless. Thus, by design they have no impact on the accuracy of the sketch-based estimates.

4.3 Reducing Computation Overhead

In addition to reducing the memory and communication footprints, *Joltik* designs a new updating strategy for universal sketch to reduce computation overhead in micro-controllers.

4.3.1 Problem

The universal sketch algorithm processes each element using a series of operations, such as hash computation, arithmetic calculations, counter updates, and heap updates. These computations incur high CPU overhead on the sensor data processing, and bring extra energy cost and large processing latency. A recent study [37] demonstrates that the top two CPU performance bottlenecks in the universal sketch are (a) hash computations and (b) counter updates.

4.3.2 Our Approach

To further optimize the energy and reduce processing delay, we introduce a new counter updating strategy that can accelerate the computation by 2× without decreasing accuracy. At a high-level, our approach (as shown in Figure 4.8) is to

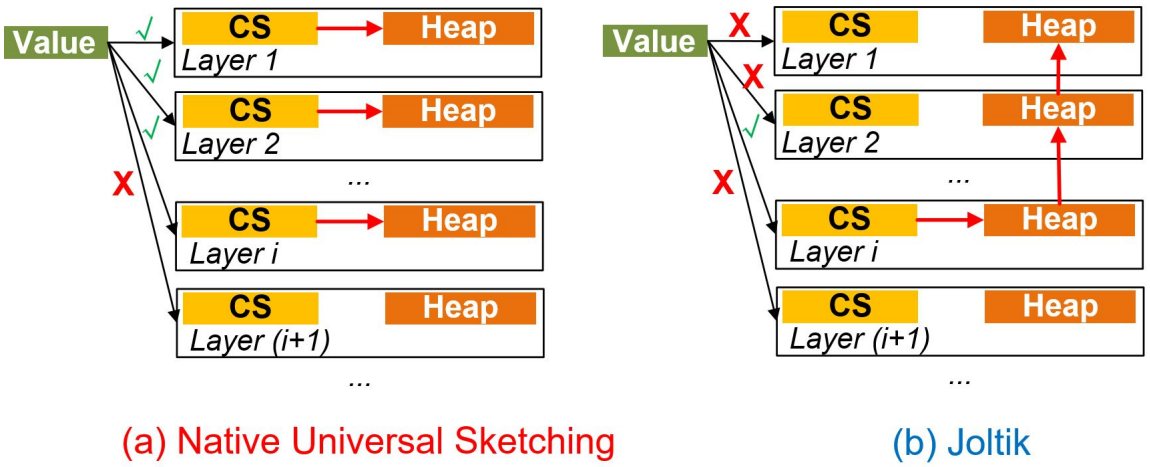


Figure 4.8: (a) Original update (update count sketch and heap in each layer) vs. (b) Optimized update (only update count sketch in last possible layer, then update heap in last layer and all previous layers). In this figure, "✓" means updating, "×" means not updating.

perform only a subset of updates to the layered Count Sketch instances and we analyze that the reduced updates will not affect the final accuracy.

Specifically, we change the updating strategy in the following two aspects:

(1) In prior universal sketch implementations [38] (Figure 4.8(a)), one must update the top several layers if an element has been sub-sampled in consecutive Layers 1 to i by hashes that return 0 or 1. In this case, in total i layers of Count Sketches have been updated. Instead, *Joltik* chooses to only update the lowest sampled layer for every element. For example in Figure 4.8(b), we will only update Layer i.

(2) When reporting the heavy hitters from all layers, we will use the heavy hitters captured in the lower layers to update all its upper layers. For instance, if Layer i's heap has heavy hitters a, b, we will need to insert a and b to Layers $i-1$ to 1. For the rest of the layers, we will update their previous upper layers as Layer i.

Intuitively, tracking elements on their last layers and using them to update all their upper lows seems contradicting on the idea of saving memory from lower uppers in Chapter 4.1. For the heavy hitters that were previously tracked in upper layers, we now need to report them from smaller-sized lower layers, resulting in worse accuracy guarantees. However as we will show later in the section, combining this technique with the memory reduction in Chapter 4.1 does not affect the actual accuracy given the natural skewness of the data.

4.3.3 Computational Benefits

Recall that the probability that a certain element is preserved from layer i to layer $i+1$ is $\frac{1}{2}$ (as a hash returns 0 or 1), we can prove that the universal sketch updates two layers of sketches on average for each element. Denote one hash computation as H and one counter update as C, and assume each count sketch instance has 5 rows of counters. Thus, in [38], the per-element computation is $21H + 10C$ as 1 hash is needed for deciding which layers to update, 20 hashes and 10 counter updates are required to operate two Count Sketch instances. Compared with [38], our approach only needs $11H + 5C$ computation as we only update one sketch at a time, reducing the cost by approximately $2\times$.

4.3.4 Impact on Accuracy

Viewed in isolation, the above approach does not result in any accuracy loss as only updating the last layer will provide the same accuracy guarantee for every item. That said, we do acknowledge that combining it with the memory reduction (as shown in Chapter 4.1) can result in worst-case errors since the counter fidelity at lower layers is lower. In practice, however, the actual added errors are negligible due to two reasons: (1) Based on the sub-sampling probability, very few heavy hitters will be preserved in bottom layers. (2) Even if a heavy hitter is indeed selected to update a

very bottom layer, the error will still be reasonable as our approach further reduces the L_2 norm of the data processed through that layer (see Chapter 4.1).

4.4 End-to-end Deployment

In this section, we describe how *Joltik*'s optimizations on computation, communication and storage are integrated to achieve end-to-end improvements in the energy vs. accuracy trade-off. Specifically, we show how both total energy and system accuracy can be determined from *Joltik*'s sketch structure size, so that it can be tuned to meet application-specific requirements.

4.4.1 Configure Sketch-related Parameter

Consider the perspective of sensor network operators who want to benefit from deploying *Joltik*. The user provides as input parameters that describe their sensor network deployment, following which *Joltik* suggests a menu of candidate configurations that trade-off battery-life and accuracy. The users can then choose a configuration that they think best suitable for their application. Specifically, *Joltik* needs the following parameters from the user: (1) Data collection rate R_{cl}: describes how frequently individual sensors should collect data and (2) Period of transmission T: decides how frequently *Joltik* client should transmit.

Joltik then tunes the following sketch size S_s in *Joltik* (in Bytes) to identify suitable alternative configuration options that trade off between client battery life and estimation accuracy. We measure battery life across these instances by assuming a total energy budget of E_b, which is determined by the sensor's battery capacity.[1]

[1] Note we express energy consumption, including E_b, with the unit of mA*s in our analysis, since we use a fixed voltage of 3V across all operations.

4.4.2 Lifetime Estimation

We show how lifetime can be represented based on the above three variables. Specifically, we first compute the energy consumption per day E_{day}, and then calculate an expected lifetime.

For a low-power client, its battery is usually modeled based on its processing cost and its transmission cost, as in [19]. Based on this battery model, we note that E_{day} consists of four main energy costs (all on a per-day scale), all of which depend on input and sketch parameters [19]: (1) The processing cost per sample (proportional to R_{cl}, the number of samples per day); (2) The compression cost (proportional to S_s/T); (3) The sleep cost (decreases linearly with R_{c1}); (4) The transmission cost (proportional to S_s)

Following a similar analysis in [19], we can define E_{day} as a function $f(.)$ of R_{cl}, S_s and T:

$$E_{day} = f(R_{cl}, S_s, T)$$

Thus, an expected lifetime can be estimated as:

$$\text{Lifetime} = E_b/E_{day} = E_b/f(R_{cl}, S_s, T)$$

4.4.3 Error Bound

Following the analysis in [13] and [15], we know that universal sketch requires space that is $O(\log n \cdot \log(1/\delta) \cdot 1/\epsilon^2)$ to process n unique elements, where ϵ is the additive error in L_2 norm of the frequency vector, and it further influences accuracy. Specifically, that means S_s should be proportional to $1/\epsilon^2$.

4.4.4 Energy and Accuracy Trade-off

Using the above methodology, S_s links lifetime and error bound together with its different impacts: for lifetime, coarsely we have Lifetime$\propto 1/(kS_s + b)$, where k and b are constants; while for accuracy, coarsely we have Error$\propto 1/\sqrt{S_s}$. This gives two

important takeaways for a potential user: (1) When it is necessary to transmit frequently (hence T is small) due to application requirements, one can potentially use less memory with acceptably small loss in accuracy, but beyond a certain point the accuracy will be significantly reduced due to smaller memory; (2) When a larger transmission period is acceptable, one can transmit less frequently and apply larger sketch structure to push towards a smaller ϵ, hence improving accuracy while maintaining a reasonable lifetime.

In practice, we envision suggesting several candidate configurations so that users can flexibly explore a suitable trade-off between lifetime and accuracy. Table 4.1 shows three sample configuration examples in our experiments to illustrate this energy and accuracy trade-off. Here, E_b set to a typical value of $3000 * 3600$ mA*s based on an AA battery.

R_{cl}	T	S_s	**Lifetime**	**Accuracy**
8 Hz	2 days	30 KB	4665 days	95.5%
8 Hz	2 days	50 KB	3264 days	96.3%
8 Hz	2 days	70 KB	2594 days	96.7%

Table 4.1: *Joltik* deployment example

5

Implementation

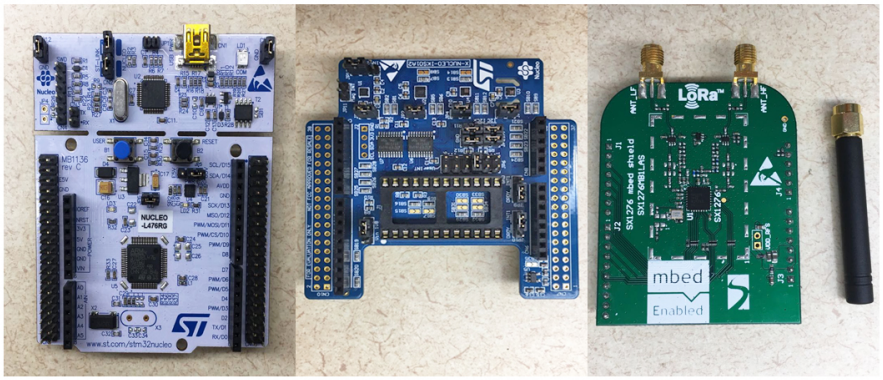
We implement *Joltik* in C using mbed compiler at both client and the base station. We use commodity off-the-shelf sensors and RF boards to sense at the clients and communicate to the base station. In this chapter, we introduce the hardware components of *Joltik* system and our real-world testbed.

5.1 *Joltik* Sensor Node

The *Joltik* sensor node consists of three parts — sensor board, microcontroller (MCU) and transceiver:

- *Sensor Board:* We use the X-NUCLEO-IKS01A2 (Figure 5.1(b)), a motion MEMS and environmental sensor expansion board for the STM32 Nucleo. This board integrates the motion MEMS accelerometer, gyroscope, magnetometer and environmental sensors for humidity, temperature and pressure on one board. We use the LPS22HB MEMS pressure sensor on board to collect pressure dataset on campus.

- *MCU:* We use the NUCLEO-L476RG board (Figure 5.1(a)), which contains



(a) Microcontroller (b) Sensor Board (c) Transceiver Board and Antenna

Figure 5.1: Hardware Components of *Joltik*: (a) Microcontroller (NUCLEO-L476RG); (b) Sensor Board (X-NUCLEO-IKS01A2); (c) Transceiver (SX1276 Long Range Transceiver).

a STM32L476RGT6U MCU in LQFP64 package. This MCU runs *Joltik* to generate sketch summaries, and assembles LoRaWAN packets.

- *RF Frontend:* We use the SX1276 LoRa Transceiver (Figure 5.1(c)) to communicate with the base station. In our experiments, we set this transceiver to operate at 915MHz, operating at SF 10 and 125 KHz bandwidth.

5.2 *Joltik* Base Station

Joltik base station consists of two parts — the MCU and a LoRa RF transceiver:

- *MCU:* We use the same MCU board as the client. This MCU receives LoRa packets from sensor nodes. Application metrics could either be calculated at this MCU (case 1), or at the cloud (case 2). In case 1, the MCU rebuilds sketch summaries, and run universal sketching offline algorithm to generate results on application metrics. In case 2, the MCU is connected to a computer or network, and uses serial communication to transmit all received LoRa packets

to the cloud. In our experiments as described in Chapter 6, we use the MCU to directly calculate the application metrics (case 1).

- *RF Frontend:* We use the same RF board as the client for LoRa communication. Traditionally a base station is equipped with much larger bandwidth to receive multiple packets simultaneously. However, we overcome this problem by requiring our clients to transmit across fixed time-slots enabling isolation.

6

Evaluation

In this chapter, we first evaluate *Joltik* on a proof-of-concept deployment using four real-world datasets (as introduced in Chapter 6.1). Across these sensor datasets, we compare *Joltik* with other alternatives in Chapter 6.2. Then, we compare the performance of *Joltik* with the native universal sketching algorithm, and the results are shown in Chapter 6.3.1.

Our major findings are as follows:

- *Joltik* achieves significantly better accuracy (97.9%) compared to sub-sampling (70.1%), native universal sketching (82.3%) and compressed sensing (34.9%) using the same amount of 60 KB memory across datasets and across evaluation tasks.

- *Joltik* reduces the power consumption of the sensor nodes by 96% compared to transmitting raw data while maintaining high accuracy ($> 95\%$) on the tested tasks.

- *Joltik* supports a range of analytics tasks without extra power consumption on the sensor nodes.

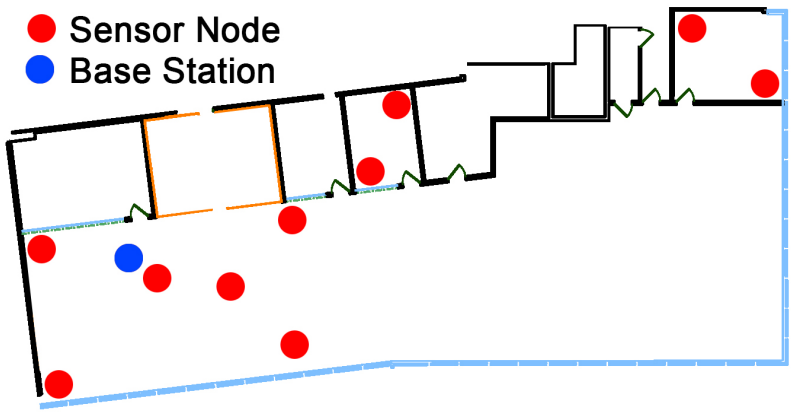


Figure 6.1: Sensor locations in building.

6.1 Evaluation Setup

6.1.1 Real-world Testbed

We build a real-world proof-of-concept testbed in a campus building with ten pressure sensors and one base station (as shown in Figure 6.1). Using this real-world testbed, we collect pressure data for weeks using 1Hz sensor sampling rate (we call this "*Joltik* dataset" in the following context). We also study the performance of *Joltik* by emulating three varieties of sensor deployments with three other large real-world datasets.

Note that when configuring *Joltik*, we follow the guidelines in Chapter 4.4 and balance the energy and accuracy trade-off. In addition, when comparing with other schemes such as compressed sensing or customized sketch, we use the same energy budget across them.

6.1.2 Datasets

In addition to our own dataset, we also consider three real-world datasets from previous work:

(1) **Indoor Solar [55]**: 2 years of joint high accuracy power and ambient condition

traces at 6 diverse indoor locations for energy harvesting systems.

(2) **SensorScope [27]**: Environmental data from past SensorScope deployments [51]. In our experiments, we use the global current value to test system performance.

(3) **LoRa Farm [14]:** Soil moisture and temperature measurements from an underground sensor network on a farm site in Western Australia. In our experiments, we use the water content value to test system performance.

6.1.3 Baselines

Across experiments, we compare *Joltik* system with 3 baselines:

(1) **sub-sampling:** downsamples raw samples with a defined ratio and transmits sub-sampled data.

(2) **customized sketches:** designed specifically for a single application function.

(3) **CoSaMP:** one of the widely used compressed sensing algorithms [43].

6.2 End-to-end System Performance

We evaluate *Joltik* on the following performance metrics: accuracy, device power consumption and multi-task handling using same energy budget.

6.2.1 Energy-Accuracy Trade-off

To evaluate energy-accuracy tradeoff of *Joltik*, we choose 4 analytic tasks as motivated by our solar farm example: Heavy Hitter (HH), Cardinality, Entropy Estimation (Entropy), and L2 norm (L2). For heavy hitters, we detect the top 100 most frequent values in a day and estimate their mean relative errors. We report *relative error* $= \frac{s - s_{real}}{s_{real}}$, where s_{real} is the ground truth of a task and s is the measured

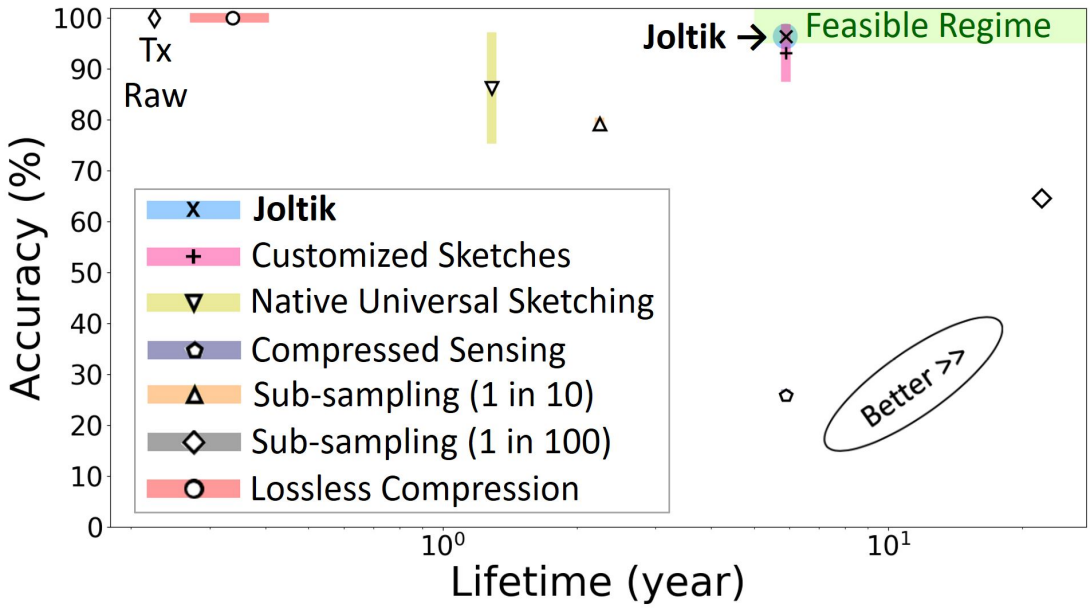


Figure 6.2: *Joltik* energy-accuracy trade-off tested using *Indoor Solar* dataset: *Joltik* provides better battery lifes while providing better accuracy.

value. We run the experiments on the *LoRa Farm* and *Indoor Solar* datasets 30 times (note that sampling across hash layers is probabilistic) and report the *median* and the *standard deviation* of the measure error.

To estimate the power consumption of *Joltik*, we leverage prior current models of our board [19], and report the device total power consumption (including sampling, processing and transmitting) for one day. We then leverage this to estimate the battery life of the client by using a standard AA battery as the energy source. Across all experiments, *Joltik* and customized sketch transmit result once per day. Sub-sampling, lossless compression and transmit raw transmit result every 15 min.

Result: As depicted in Figure 6.2 and Figure 6.3, *Joltik* achieves significantly better accuracy in all tested tasks, sampling frequencies and datasets over the baselines . We also notice as we increase the subsampling factor from 10 to 1000, sensors lose more and more information, leading to higher error rates. This is particularly large for tasks that focus on tail distributions such as cardinality. While CoSaMP works

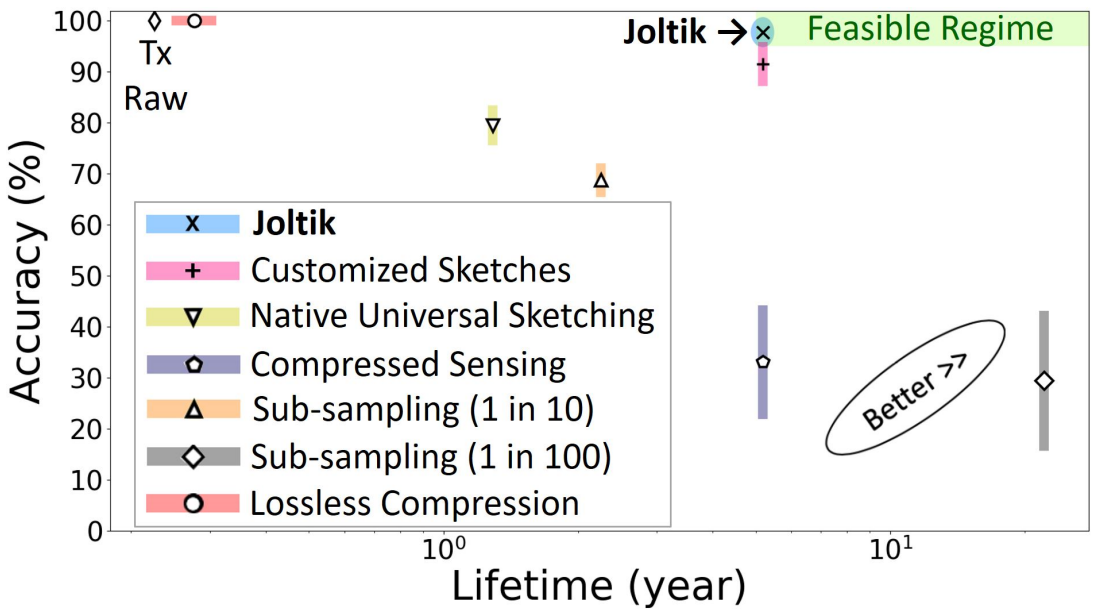


Figure 6.3: *Joltik* energy-accuracy trade-off tested using *LoRa Farm* dataset: *Joltik* provides better battery lifes while providing better accuracy.

for sparser datasets, the compressed sensing approach fails miserably on dense data, achieving only about 30% accuracy. While one would assume customized sketch to outperform *Joltik* in terms of accuracy, remember that when we constrain the system to have equal energy consumption, each of the customized sketches get $\frac{1}{4}$ of the total energy used by *Joltik*. Thus, *Joltik* outperforms customized sketch method.

Joltik is also significantly more energy-efficient over baseline approaches of transmitting raw data (24.6x) and lossless compression (16.4x). This is due to the fact that lossless compression can only compress the raw data by 1.5×. This means that if the low power sensor operates on a typical AA battery, these approaches would provide an insufficient battery life of 80 days and 135 days, respectively. In comparison, *Joltik* can allow the client to provide high fidelity analytics for 5.5 years.

At first glance, sub-sampling and customized sketch seem pretty energy-efficient. However, we can see that they achieve so by sacrificing accuracy. Thus, it is necessary to understand how these algorithms trade-off energy for accuracy. Our results

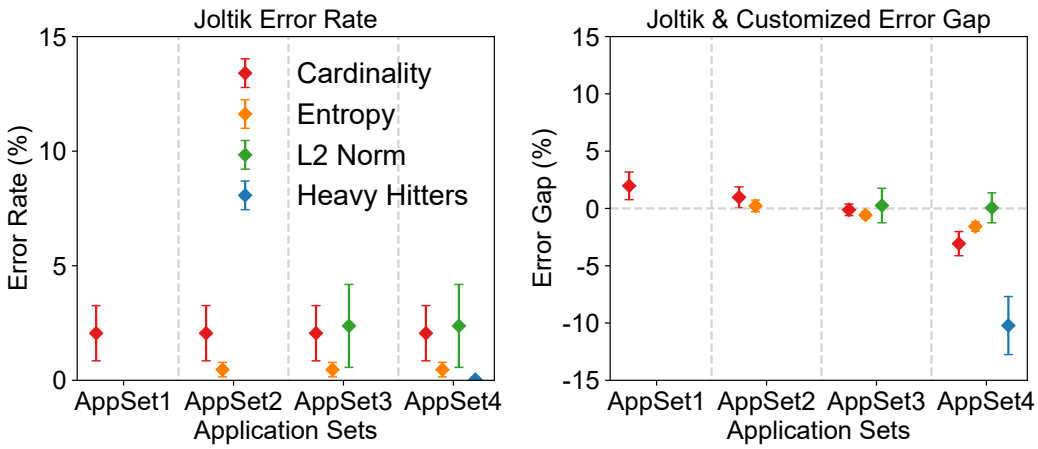


Figure 6.4: Generality and Multi-task Handling. Left: Error rate of *Joltik* estimating four application sets using same energy budget. Right: Error gap between *Joltik* and customized sketch estimating four application sets using same energy budget. positive values imply *Joltik* is worse and vice versa). Both experiments use *Joltik* dataset with 1Hz sensor sampling rate.

in Figure 6.2 and Figure 6.3 clearly demonstrates that *Joltik* has a better energy consumption and accuracy trade-off over these approaches across applications. This result highlights the ability of *Joltik* to achieve energy-efficiency and high-fidelity at the same time.

6.2.2 Generality

To evaluate generality, we deploy both *Joltik* and customized sketches to do four different sets of estimation tasks: AppSet1 = {Cardinality}, AppSet2 = {Cardinality, Entropy}, AppSet3 = {Cardinality, Entropy, L2} and AppSet4 = {Cardinality, Entropy, L2, HH}. For all the estimation task sets, *Joltik* and customized sketches are allocated the same energy budget (5 years of battery lifetime using a typical 9,720 As capacity AA battery). For the customized sketch, we use Hyperloglog [21] for cardinality, entropy estimation algorithm [16] for entropy, Count Sketch [15] for L2, and Count-Min Sketch [18] for HH. When doing multiple estimation tasks using cus-

tomized sketches, we divide the power budget uniformly across different customized sketches.

Result: Figure 6.4 shows that running multiple tasks on *Joltik* using the same energy budget will not incur accuracy deduction in individual tasks, since the universal sketch maintained in the sensor preserves information for all of these tasks. In various machine learning and data analytics scenarios that need a variety of features from sensed data, *Joltik* can be an energy-efficient and accurate alternative to the approach of simply sending all raw data. We also show the "error gap" between *Joltik* and customized sketches (error result of *Joltik*- error result of customized sketches. i.e., positive values imply *Joltik* is worse and vice versa). As expected, when more tasks are needed, *Joltik* will perform significantly better than customized sketches in terms of accuracy. This is due to the consequently decreasing power budget for individual tasks.

6.3 Evaluating *Joltik*'s Optimizations

In this section, we compare the performance of *Joltik* vs. native universal sketching in terms of memory footprint, computation-efficiency and communication overhead with the original universal sketching algorithm. We will show how *Joltik* solves the three system challenges and why *Joltik* is more feasible for low-power wireless sensors compared to the original universal sketching algorithm.

6.3.1 Memory Footprint

We evaluate our approach described in Chapter 4.1 by deploying both *Joltik* and the canonical universal sketching algorithm [38] to perform 2 analytics tasks (HH and L2-norm). We vary the amount of memory for both tested approaches to study the impact on accuracy.

Result: As depicted in Figure 6.5, *Joltik* achieves low-error estimation for both

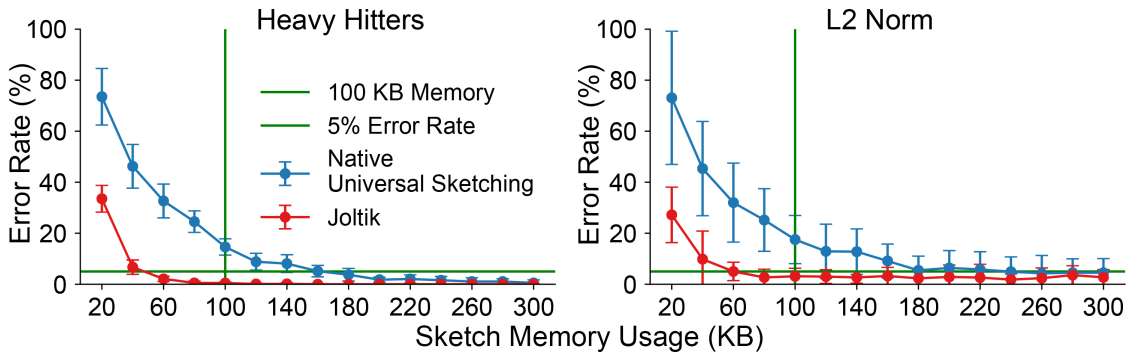


Figure 6.5: Evaluation on memory optimization. Error rates of *Joltik* and the original universal sketching on *Joltik* Dataset with 1Hz sensor sampling rate. For all experiments, the feasible region is within 5% error rate and 100KB sensor memory limit.

heavy hitters and L2-norm at significantly smaller sketch sizes. A key thing to notice is that for memory sizes smaller than 100 KB (typical on a low-power client), only *Joltik* can provide high accuracy for most supported tasks.

6.3.2 Communication Overhead

Compression Methods	**Compression Rate**	**Energy (mAs)**
Reduce bitsize per counter (***Joltik***)	3.835	5479.7
Huffman	3.575	5868.4
LZW	3.57	5960.2
Delta Encoding	3.415	6136.8
No Compression (*Native Universal Sketching*)	1	20616.1

Table 6.1: *Joltik* compression methods comparison

We evaluate *Joltik*'s approach for reducing the communication overhead by comparing the power consumption of *Joltik*'s scheme with other state-of-the-art compression schemes such as Huffman encoding, LZW, Delta Encoding and the native universal sketch.

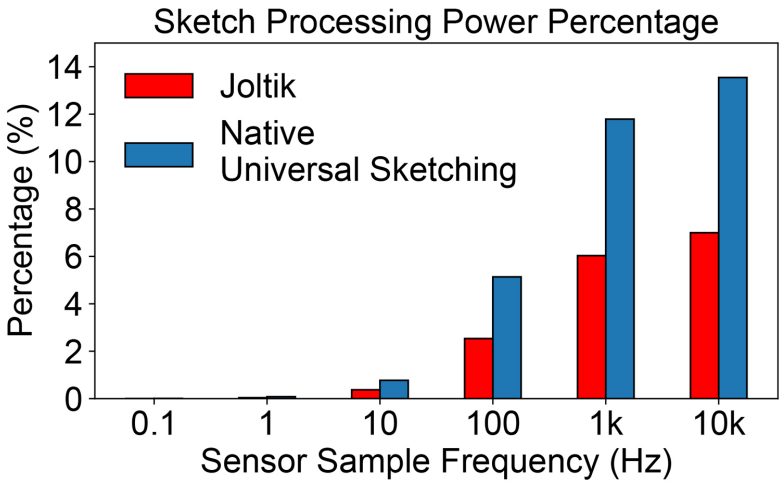


Figure 6.6: Evaluation on computation optimization, showing the percentage of sketch processing power out of total sensor power.

Result: Our results in Table 6.1 demonstrate a 3.76× reduction on power consumption over the native universal sketching algorithm without losing any information at the base station. It also performs comparably to many of the prior encoding schemes in terms of compression ratio.

6.3.3 Computation Overhead

Finally, we evaluate the impact of *Joltik*'s computation optimization of reducing the number of hashes per element. Since our approach does not affect the accuracy of the output metric, we focus on the reduction in power consumption as a function of sensor sampling frequency. Figure 6.6 shows increasing benefits for sensors that sample significantly more demonstrating a 2× reduction in the sketch processing power. (At lower sampling frequency, transmission costs dominate.)

7

Conclusions and Future Work

This thesis presents *Joltik*, a framework for doing analytics tasks on low-power IoT clients. Compared to existing approaches, *Joltik* could simultaneously achieves *generality* to support a large range of metrics of interest, *fidelity* in estimating these diverse statistics, and *energy-efficiency* for optimized client battery life. *Joltik* enables this by observing an opportunity to leverage universal sketching algorithm. To make the algorithm compatible with low-power clients, we optimize the native universal sketching for storage, computation and communication. To evaluate the performance of the proposed *Joltik* system, we present an end-to-end system realization of *Joltik* on a real-world LoRaWAN testbed. And a detailed testbed evaluation of *Joltik* demonstrates 96% reduced power consumption compared to transmitting raw data, 97.9% accuracy in computing valuable statistics on client data within a storage space of 60 kB.

We believe *Joltik* presents an interesting opportunity in developing sketches for low-power IoT clients that are particularly amenable to highly data-intensive machine learning algorithms including deep learning. This will allow the vast compute resources at the edge and cloud to leverage rich sensed information for various appli-

cations, while maintaining the energy-efficiency of individual sensors. For the future work, making *Joltik* system more resistant to packet loss during wireless transmissions could be a very interesting direction.

Bibliography

[1] P. K. Agarwal, G. Cormode, Z. Huang, J. M. Phillips, Z. Wei, and K. Yi, "Mergeable summaries," *ACM Transactions on Database Systems (TODS)*, vol. 38, no. 4, pp. 1–28, 2013.

[2] A. Ali and W. Hamouda, "On the cell search and initial synchronization for NB-IoT LTE systems," *IEEE Communications Letters*, vol. 21, no. 8, pp. 1843–1846, 2017.

[3] N. Alon, Y. Matias, and M. Szegedy, "The space complexity of approximating the frequency moments," in *Proc. of ACM STOC*, 1996.

[4] P. Andres-Maldonado, P. Ameigeiras, J. Prados-Garzon, J. Navarro-Ortiz, and J. M. Lopez-Soler, "Narrowband IoT data transmission procedures for massive machine-type communications," *IEEE Network*, vol. 31, no. 6, pp. 8–15, 2017.

[5] T. Arici, B. Gedik, Y. Altunbasak, and L. Liu, "PINCO: a pipelined in-network compression scheme for data collection in wireless sensor networks," *Proceedings. 12th International Conference on Computer Communications and Networks (IEEE Cat. No.03EX712).*

[6] S. J. Baek, G. d. Veciana, and X. Su, "Minimizing Energy Consumption in Large-scale Sensor Networks Through Distributed Data Compression and Hierarchical Aggregation," *IEEE J.Sel. A. Commun.*, vol. 22, no. 6, pp. 1130–1140, Sep. 2006. [Online]. Available: https://doi.org/10.1109/JSAC.2004.830934. [Accessed 2020-03-25].

[7] W. Bajwa, J. Haupt, A. Sayeed, and R. Nowak, "Compressive wireless sensing," in *Proceedings of the 5th international conference on Information processing in sensor networks.* ACM, 2006, pp. 134–142.

[8] R. G. Baraniuk, V. Cevher, M. F. Duarte, and C. Hegde, "Model-Based Compressive Sensing," *IEEE Transactions on Information Theory*, vol. 56, no. 4, p. 1982–2001, Apr 2010. [Online]. Available: http://dx.doi.org/10.1109/TIT.2010.2040894. [Accessed 2020-03-25].

[9] V. Braverman, S. R. Chestnut, D. P. Woodruff, and L. F. Yang, "Streaming space complexity of nearly all functions of one variable on frequency vectors," in *Proc. of PODS*, 2016.

[10] V. Braverman and R. Ostrovsky, "Zero-one frequency laws," in *Proceedings of the forty-second ACM symposium on Theory of computing*, 2010, pp. 281–290.

[11] V. Braverman, S. R. Chestnut, D. P. Woodruff, and L. F. Yang, "Streaming space complexity of nearly all functions of one variable on frequency vectors," in *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, 2016, pp. 261–276.

[12] V. Braverman, R. Krauthgamer, and L. F. Yang, "Universal streaming of subset norms," *CoRR*, vol. abs/1812.00241, 2018.

[13] V. Braverman and R. Ostrovsky, "Zero-one frequency laws," in *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, ser. STOC '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 281–290. [Online]. Available: https://doi.org/10.1145/1806689.1806729. [Accessed 2020-03-25].

[14] R. Cardell-Oliver, C. Hübner, M. Leopold, and J. Beringer, "Dataset: LoRa Underground Farm Sensor Network," in *Proceedings of the 2nd Workshop on Data Acquisition To Analysis*. ACM, 2019, pp. 26–28.

[15] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," in *Proc. of ICALP*, 2002.

[16] P. Clifford and I. Cosma, "A simple sketching algorithm for entropy estimation over streaming data," in *Artificial Intelligence and Statistics*, 2013, pp. 196–206.

[17] G. Cormode and M. Garofalakis, "Sketching probabilistic data streams," in *Proc. of ACM SIGMOD*, 2007.

[18] G. Cormode and S. Muthukrishnan, "An Improved Data Stream Summary: The Count-Min Sketch and Its Applications," *J. Algorithms*, 2005.

[19] A. Dongare, R. Narayanan, A. Gadre, A. Luong, A. Balanuta, S. Kumar, B. Iannucci, and A. Rowe, "Charm: exploiting geographical diversity through coherent combining in low-power wide-area networks," in *2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE, 2018, pp. 60–71.

[20] R. Eletreby, D. Zhang, S. Kumar, and O. Yağan, "Empowering low-power wide area networks in urban settings," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 2017, pp. 309–321.

[21] P. Flajolet, É. Fusy, O. Gandouet, and F. Meunier, "Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm," 2007.

[22] A. Gadre, R. Narayanan, A. Luong, A. Rowe, B. Iannucci, and S. Kumar, "Frequency Configuration for Low-Power Wide-Area Networks in a Heartbeat," in *USENIX NSDI*, 2020.

[23] A. Gadre, F. Yi, A. Rowe, B. Iannucci, and S. Kumar, "Quick (and Dirty) Aggregate Queries on Low-Power WANs," in *ACM/IEEE IPSN*, 2020.

[24] S. Gandhi, S. Suri, and E. Welzl, "Catching elephants with mice: sparse sampling for monitoring sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 6, no. 1, pp. 1–27, 2010.

[25] A. C. Gilbert, M. J. Strauss, J. A. Tropp, and R. Vershynin, "One Sketch for All: Fast Algorithms for Compressed Sensing," in *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, ser. STOC '07. New York, NY, USA: ACM, 2007, pp. 237–246. [Online]. Available: http://doi.acm.org/10.1145/1250790.1250824. [Accessed 2020-03-25].

[26] A. Gilbert and P. Indyk, "Sparse recovery using sparse matrices," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 937–947, 2010.

[27] Guillermo Barrenetxea, "Sensorscope Data," https://doi.org/10.5281/zenodo.2654726, 2019, [Accessed 2020-03-25].

[28] M. Hessar, A. Najafi, V. Iyer, and S. Gollakota, "TinySDR: Low-Power SDR Platform for Over-the-Air Programmable IoT Testbeds," in *USENIX NSDI*, 2020.

[29] F. Huang and Y. Liang, "Towards Energy Optimization in Environmental Wireless Sensor Networks for Lossless and Reliable Data Gathering," *2007 IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems*, 2007.

[30] P. Indyk and M. Ruzic, "Near-Optimal Sparse Recovery in the L1 Norm," in *Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science*, ser. FOCS '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 199–207. [Online]. Available: https://doi.org/10.1109/FOCS.2008.82. [Accessed 2020-03-25].

[31] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '00. New York, NY, USA: ACM, 2000, pp. 56–67. [Online]. Available: http://doi.acm.org/10.1145/345910.345920. [Accessed 2020-03-25].

[32] M. Khazraee, Y. Guddeti, S. Crow, A. C. Snoeren, K. Levchenko, D. Bharadia, and A. Schulman, "SparSDR: Sparsity-proportional Backhaul and Compute for SDRs," in *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 2019, pp. 391–403.

[33] N. Klugman, J. Adkins, S. Berkouwer, K. Abrokwah, I. Bobashev, P. Pannuto, M. Podolsky, A. Susenot, R. Thatte, C. Wolfram, J. Taneja, and P. Dutta, "Hardware, apps, and surveys at scale: Insights from measuring grid reliability in accra, ghana," in *ACM SIGCAS Conference on Computing and Sustainable Societies*, ser. COMPASS'19, July 2019.

[34] L. Labs, "Firmware-over-the-air (fota) with lora," https://www.link-labs.com/blog/firmware-over-the-air-fota-with-lora, 2017, [Accessed 2020-03-25].

[35] Y. Liang and W. Peng, "Minimizing Energy Consumptions in Wireless Sensor Networks via Two-modal Transmission," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 1, pp. 12–18, Jan. 2010. [Online]. Available: http://doi.acm.org/10.1145/1672308.1672311. [Accessed 2020-03-25].

[36] X. Lin, A. Adhikary, and Y.-P. E. Wang, "Random access preamble design and detection for 3GPP narrowband IoT systems," *IEEE Wireless Communications Letters*, vol. 5, no. 6, pp. 640–643, 2016.

[37] Z. Liu, R. Ben-Basat, G. Einziger, Y. Kassner, V. Braverman, R. Friedman, and V. Sekar, "Nitrosketch: Robust and general sketch-based monitoring in software switches," in *Proceedings of the ACM Special Interest Group on Data Communication*. ACM, 2019, pp. 334–350.

[38] Z. Liu, A. Manousis, G. Vorsanger, V. Sekar, and V. Braverman, "One sketch to rule them all: Rethinking network flow monitoring with univmon," in *Proc. of ACM SIGCOMM*, 2016.

[39] Y. Lu, A. Montanari, B. Prabhakar, S. Dharmapurikar, and A. Kabbani, "Counter Braids: A Novel Counter Architecture for Per-FlowMeasurement," in *Proc. of ACM SIGMETRICS*, 2008.

[40] B. Lucia, V. Balaji, A. Colin, K. Maeng, and E. Ruppel, "Intermittent computing: Challenges and opportunities," in *2nd Summit on Advances in Programming Languages (SNAPL 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[41] F. Marcelloni and M. Vecchio, "Enabling energy-efficient and lossy-aware data compression in wireless sensor networks by multi-objective evolutionary optimization," *Information Sciences*, vol. 180, no. 10, p. 1924–1941, 2010.

[42] A. Metwally, D. Agrawal, and A. E. Abbadi, "Efficient computation of frequent and top-k elements in data streams," in *Proc. of ICDT*, 2005.

[43] D. Needell and J. A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Applied and computational harmonic analysis*, vol. 26, no. 3, pp. 301–321, 2009.

[44] S. Pattem, B. Krishnamachari, and R. Govindan, "The Impact of Spatial Correlation on Routing with Compression in Wireless Sensor Networks," *ACM Trans. Sen. Netw.*, vol. 4, no. 4, pp. 24:1–24:33, Sep. 2008. [Online]. Available: http://doi.acm.org/10.1145/1387663.1387670. [Accessed 2020-03-25].

[45] D. Petrovic, R. Shah, K. Ramchandran, and J. Rabaey, "Data funneling: routing with aggregation and compression for wireless sensor networks," *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications, 2003.*

[46] C. G. Ramirez, A. Sergeyev, A. Dyussenova, and B. Iannucci, "LongShoT: long-range synchronization of time," in *Proceedings of the 18th International Conference on Information Processing in Sensor Networks*. ACM, 2019, pp. 289–300.

[47] R. Ratasuk, B. Vejlgaard, N. Mangalvedhe, and A. Ghosh, "NB-IoT system for M2M communication," in *2016 IEEE wireless communications and networking conference*. IEEE, 2016, pp. 1–5.

[48] C. M. Sadler and M. Martonosi, "Data Compression Algorithms for Energy-constrained Devices in Delay Tolerant Networks," in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, ser. SenSys '06. New York, NY, USA: ACM, 2006, pp. 265–278. [Online]. Available: http://doi.acm.org/10.1145/1182807.1182834. [Accessed 2020-03-25].

[49] L. Sanchez, L. Muñoz, J. A. Galache, P. Sotres, J. R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis *et al.*, "SmartSantander:

IoT experimentation over a smart city testbed," *Computer Networks*, vol. 61, pp. 217–238, 2014.

[50] A. Scaglione and S. Servetto, "On the Interdependence of Routing and Data Compression in Multi-hop Sensor Networks," *Wirel. Netw.*, vol. 11, no. 1-2, pp. 149–160, Jan. 2005. [Online]. Available: http://dx.doi.org/10.1007/s11276-004-4752-y. [Accessed 2020-03-25].

[51] T. Schmid, H. Dubois-Ferrière, and M. Vetterli, "SensorScope: Experiences with a Wireless Building Monitoring Sensor Network," *Workshop on Real-World Wireless Sensor Networks (REALWSN'05)*, 2005. [Online]. Available: http://infoscience.epfl.ch/record/51001. [Accessed 2020-03-25].

[52] R. Schweller, A. Gupta, E. Parsons, and Y. Chen, "Reversible sketches for efficient and accurate change detection over network data streams," in *Proc. of ACM IMC*, 2004.

[53] Semtech, "Smart Electricity Metering using LoRa," https://www.semtech.com/lora/lora-applications/smart-electricity-metering, 2020, [Accessed 2020-05-07].

[54] F. K. Shaikh and S. Zeadally, "Energy harvesting in wireless sensor networks: A comprehensive review," *Renewable and Sustainable Energy Reviews*, vol. 55, pp. 1041–1054, 2016.

[55] L. Sigrist, A. Gomez, and L. Thiele, "Dataset: Tracing Indoor Solar Harvesting," in *Proceedings of the 2nd Workshop on Data Acquisition To Analysis*. ACM, 2019, pp. 47–50.

[56] T. Srisooksai, K. Keamarungsi, P. Lamsrichan, and K. Araki, "Practical Data Compression in Wireless Sensor Networks: A Survey," *J. Netw. Comput. Appl.*, vol. 35, no. 1, pp. 37–59, Jan. 2012. [Online]. Available: http://dx.doi.org/10.1016/j.jnca.2011.03.001. [Accessed 2020-03-25].

[57] A. Stein and C. Ettema, "An overview of spatial sampling procedures and experimental design of spatial studies for ecosystem comparisons," *Agriculture, Ecosystems & Environment*, vol. 94, no. 1, pp. 31–47, 2003.

[58] J. M. Talavera, L. E. Tobón, J. A. Gómez, M. A. Culman, J. M. Aranda, D. T. Parra, L. A. Quiroz, A. Hoyos, and L. E. Garreta, "Review of IoT applications in agro-industrial and environmental fields," *Computers and Electronics in Agriculture*, vol. 142, pp. 283–297, 2017.

[59] C. Tang, C. Raghavendra, and V. Prasanna, "Power Aware Coding for Spatio-Temporally Correlated Wireless Sensor Data," *2004 IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE Cat. No.04EX975)*.

[60] J. A. Tropp and A. C. Gilbert, "Signal Recovery From Random Measurements Via Orthogonal Matching Pursuit," *IEEE Trans. Inf. Theor.*, vol. 53, no. 12, pp. 4655–4666, Dec. 2007. [Online]. Available: https://doi.org/10.1109/TIT.2007.909108. [Accessed 2020-03-25].

[61] R. O. V. Braverman, "Generalizing the layering method of indyk and woodruff: Recursive sketches for frequency-based vectors on streams," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 2013, pp. 58–70.

[62] R. O. V. Braverman and A. Roytman, "Zero-one laws for sliding windows and universal sketches," in *Proc. of APPROX/RANDOM*, 2015.

[63] T. Yang, J. Jiang, P. Liu, Q. Huang, J. Gong, Y. Zhou, R. Miao, X. Li, and S. Uhlig, "Elastic sketch: Adaptive and fast network-wide measurements," in *Proc. of ACM SIGCOMM*, 2018.

[64] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[65] C. Yu, L. Yu, Y. Wu, Y. He, and Q. Lu, "Uplink scheduling and link adaptation for narrowband Internet of Things systems," *IEEE Access*, vol. 5, pp. 1724–1734, 2017.

[66] M. Yu, L. Jose, and R. Miao, "Software defined traffic measurement with opensketch," in *Proc. of USENIX NSDI*, 2013.