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

Mingran Yang
Carnegie Mellon University and Massachusetts Institute of Technology
mingrany@andrew.cmu.edu

Junbo Zhang
Carnegie Mellon University
junboz2@andrew.cmu.edu

Akshay Gadre
Carnegie Mellon University
agadre@andrew.cmu.edu

Zaoxing Liu
Carnegie Mellon University and Boston University
zaoxing@cmu.edu

Swarun Kumar
Carnegie Mellon University
swarun@cmu.edu

Vyas Sekar
Carnegie Mellon University
vsekar@andrew.cmu.edu

Abstract

Wireless sensors have enabled a number of key applications. Due to their energy constraints, wireless sensors today communicate occasional short samples or pre-determined summary statistics of the data they collect. This means that computing every additional statistic at high fidelity incurs additional communication and energy overhead. This paper presents *Joltik*, a framework enabling general, future-proof, and energy-efficient analytics for low power wireless sensors. *Joltik* is *general* in that it summarizes sensed data from low-power devices without making assumptions on which specific statistical metric(s) are desired at the cloud and is *future-proof*, meaning it supports new, unforeseen metrics. *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 testbed evaluation of *Joltik* in LoRaWAN NUCLEO-L476RG boards and sensors. Across a range of realistic datasets, *Joltik* provides up to a 24.6× reduction in energy cost compared to transmitting raw data and outperforms many natural alternatives (e.g., sub-sampling, custom sketches, compressed sensing, and lossy compression) in terms of energy-accuracy trade-offs.

CCS Concepts

• **Networks** → **Network components**; • **Computer systems organization** → **Embedded and cyber-physical systems**; • **Hardware** → **Communication hardware, interfaces and storage**.

Keywords

Low-Power Wide-Area Network (LPWAN), Sensor Networks, Sketching Algorithm

MobiCom '20, September 21–25, 2020, London, United Kingdom

ACM ISBN 978-1-4503-7085-1/20/09.
https://doi.org/10.1145/3372224.3419204

ACM Reference Format:
Mingran Yang, Junbo Zhang, Akshay Gadre, Zaoxing Liu, Swarun Kumar, and Vyas Sekar. 2020. *Joltik*: Enabling Energy-Efficient "Future-Proof" Analytics on Low-Power Wide-Area Networks. In *The 26th Annual International Conference on Mobile Computing and Networking (MobiCom '20), September 21–25, 2020, London, United Kingdom.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3372224.3419204

1 Introduction

Low-Power Wide-Area Networks (LP-WANs) that span a large city or rural area, are increasingly deployed to sense various metrics of interest and send aggregate reports [39, 56]. These networks operate over an 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 these devices to communicate occasional point samples or short summary statistics (e.g., a mean value). More importantly, the exact list of statistics needed must be agreed upon *a priori*, as the raw data is too large to be stored or transmitted using the sensing devices.

Yet, this requirement of choosing the required statistics upfront is problematic, as we show in the following motivating scenario. As a solar farm, one may deploy multiple solar sensors which periodically send the 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 such summary statistics. If the sensors cannot be reprogrammed in the field, the operators need to commit to a small set of metrics with corresponding per-metric estimation algorithms at design time and cannot support future requirements. Even if the sensors are re-programmable in the field, at any given time we may only be able to enable a small subset of algorithms due to computation and energy constraints, resulting in blind spots for metrics that are currently disabled.

Ideally, we want a solution that enables downstream sensor data analytics that: (1) are *general* to support many applications with a single algorithm in contrast to using many per-metric algorithms, and *future-proof* to support possibly unforeseen metrics for the same data; (2) have *high fidelity* (e.g., at most 5% error [45]); and (3)

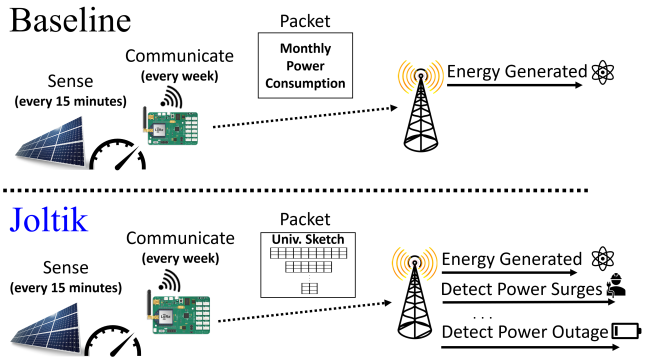


Figure 1: *Joltik* leverages universal sketches to generate future-proof general analytics for LP-WANs

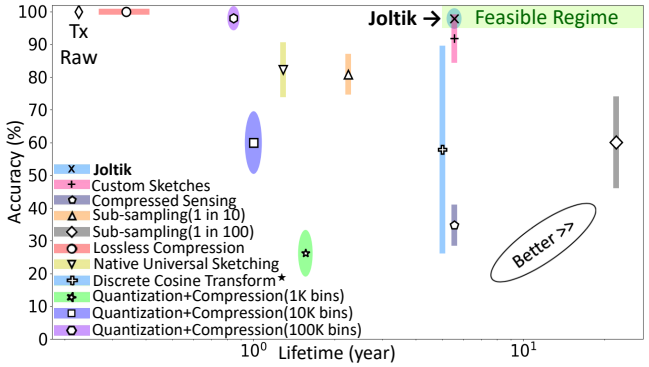


Figure 2: *Joltik* provides better energy-accuracy trade-off for "future-proof analytics" vs. prior approaches (*feasible regime* defined as estimation accuracy > 95% for all metrics of interest, and sensor battery lifetime > 5 years)

do not sacrifice client *energy-efficiency* (i.e., supporting multi-year deployments on a single battery charge). At first glance, these goals seem challenging to achieve simultaneously — barring sending raw data (which is energy inefficient due to the communication costs), it seems difficult to support a large spectrum of data analytics tasks on sensor data, let alone unforeseen requirements.

This paper presents *Joltik*, a framework that enables *general* and *energy-efficient* analytics for low power wireless sensors. *Joltik* operates under the power constraints of a low power sensor, computing aggregates over significantly more sensed samples (better accuracy), yet compressing them within a single packet transmission using universal sketching (generality). Further, *Joltik* can compute a wide range of unforeseen metrics without additional energy overhead at the sensor (energy-efficiency). *Joltik* can enable all these benefits on off-the-shelf clients with a simple software update.

Joltik builds on recent theoretical advances in *universal sketching* [10, 12–14]. At a high-level, sketching, or streaming algorithms estimate (approximately) specific properties of a data stream, with provable memory-accuracy trade-offs [4, 16, 19, 47, 51]. Unlike previous sketches that support a narrow metric of interest (e.g., heavy hitters [16, 19, 51], quantiles [1]), universal sketches can simultaneously support many already known and possibly unforeseen estimation tasks in a single algorithm [12, 45]. With universal sketching, we only require to collect a general sketch summary from the data at runtime and query any supported (and possibly unforeseen) statistics from the sketch anytime afterward (see Sec. 2.3). This makes universal sketching the perfect candidate for enabling general "future-proof" sensor data analytics.

Realizing these potential benefits of universal sketching on low-power sensor platforms, however, raises significant practical challenges. Universal sketches need to perform a large number of hash computations, store counter arrays for each layer, and communicate these to base stations for computing the metrics. Yet, to achieve reasonable fidelity requires space, computation, and communication well beyond the capabilities of a typical low-power client.

Joltik's key contribution is identifying and addressing these practical system bottlenecks of applying universal sketching in a low-power sensing context. Specifically, we identify the three key bottlenecks as storage, power constraints, and computation overheads. The rest of this paper addresses our three key design contributions to tackle these challenges for enabling general accurate analytics on power-starved sensors.

Efficient Storage (Sec. 4.1): These low-power devices have limited memory (hundreds of KBs) to store sketch data structures, and as such native implementations of universal sketching are infeasible. *Joltik* designs optimizations to ensure that the sketch data structure is compact (tens of 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.

Reducing Communication Cost (Sec. 4.2): Despite the above optimizations, the communication footprint is still too high especially in a lossy LP-WAN setting and consumes over 90% of battery life for LP-WAN sensors [20]. 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 costs.

Reducing Computation Cost (Sec. 4.3): Finally, as the client sensing frequency increases, the computation footprint becomes the major bottleneck as universal sketching entails computing numerous hashes and counter updates on low-power devices. We refactor the counter update computation in universal sketching to effectively reduce the computation cost. Our approach is based on an observation that the native universal sketch [45] unnecessarily adds the same item into the sketch by multiple times. Thus, in order to reduce computation while achieving similar accuracy, we can eliminate these redundant counter updates and conduct only one update per item.

Our end-to-end goal is to maximize sensor battery life while achieving the desired level of system accuracy under clients' computation/memory constraints. Building on the above design optimizations, we show how users can configure *Joltik*'s sketch parameters to suit application-specific accuracy requirements, while accounting for the energy costs and resource constraints.

We implement *Joltik* using LoRaWAN NUCLEO-L476RG boards and sensors, and conduct proof-of-concept experiments on a 10-node LoRaWAN testbed spanning a university building. We further emulate a variety of sensor deployment scenarios using publicly available datasets ([15, 30, 66]) of temperature, pressure, soil moisture, and solar energy measurement. We demonstrate *Joltik*'s performance against a variety of baselines including sending raw data,

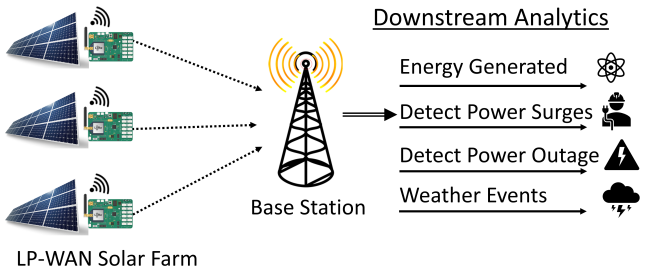


Figure 3: A motivating example: A solar farm may need general analytics for various applications

sub-sampling, compressed sensing, custom per-metric sketches, and various lossy and lossless data compression strategies. Our evaluation demonstrates that: (1) Given an energy budget, *Joltik* achieves significantly better accuracy when compared to all baselines (see Fig. 2); (2) *Joltik* reduces storage requirements by 5× compared to native universal sketches without loss in accuracy; and (3) *Joltik* can measure multiple sensor statistics at the same time without extra energy costs.

Contributions and Roadmap: In summary, this paper 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 (Sec. 3);
- A practical low CPU, low memory footprint and low power realization of universal sketches on commodity sensor platforms (Sec. 4);
- An end-to-end system realization of *Joltik* on a real-world LoRaWAN testbed (Sec. 5) and demonstration of its benefits compared to alternatives (Sec. 6).

2 Background and Motivation

2.1 Motivating scenario

Let us look at an example of a solar farm to motivate *Joltik* (Fig. 3). 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 [65] (i.e., total energy use) or detect day-night cycles, and therefore operate solely on battery power to function, independent of the grid [38].

In this section, we motivate *Joltik* with a real-world scenario and highlight the limitations of current approaches. We then briefly discuss the theoretical foundations underlying *Joltik*'s abilities.

Solar Sensor	Metric	Definition
Energy Generated	L1-norm (L_1)	$\sum f_i$
Power usage	α-Heavy Hitters	$f_i \geq \alpha \sum f_i$
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	$f_c \geq \alpha \sum f_c$
Voltage Range	Tail Detection	Quantiles
Power Outages	Zero-draw Time	$f_0 \geq \alpha \sum f_i$

Table 1: Metrics relevant to a solar farm

In addition, we may have other analytical 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 analytical tasks. Table 1 shows a subset of possible (but non-exhaustive) statistics for this application setting.

Approach	Energy	Accuracy	Generality
Sub-sampling [26, 28]	✓	×	✓
Lossless Compression [40, 60]	×	✓	✓
Lossy Compression [27, 57]	✓	×	✓
Sparse Recovery [52]	✓	✓	×
Data-centric Aggregation [53, 70]	✓	✓	×
Joltik	✓	✓	✓

Table 2: In contrast to prior work, *Joltik* guarantees energy-efficiency, accuracy, and generality (See Sec. 7)

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 re-programmability were feasible, the operators still have to make some unfortunate runtime trade-offs. Since the sensors only have 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 the voltage range of malfunctioning sensors, or investigate the effect of weather events. It is expensive to physically visit and reconfigure the clients manually. While over-the-air (OTA) updates [42] 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 that can provide accurate, general, and possibly future-proof analytics for a wide array of metrics without constant manual intervention.

2.2 Strawman Solutions

In this context, we explore a few strawman approaches (see Sec. 7 for a more detailed related work overview). We note that our experimental evaluation compares *Joltik* with each of the below alternative design choices to demonstrate significant improvements in the accuracy vs. energy consumption trade-off (shown in Sec. 6).

Sub-Sampling [26, 28]: The company may circumvent computation limitation by allowing the sensors to select some k applications out of the 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.

Lossless compression [40, 60]: Using lossless compression on raw data is hard to achieve a significantly longer battery life compared to transmitting raw data. The reason is that, compression ratio using lossless compression is typically low (e.g., 4.5× in previous work [60]). Given that low power clients usually have a limited storage capacity [73], when the client is operating at a high

sampling rate, the client needs to transmit much more frequently, resulting in high energy consumption. Even if the client operates at a low sampling rate, the benefit of lossless compression over transmitting raw data is limited due to the low compression ratios.

Lossy compression [27, 57]: If the client uses lossy compression, the battery life is extended as the client now transmits much fewer data. However, there will be a large amount of information loss sacrificing the data accuracy. For instance, quantization or rounding can reduce the amount of data to be transmitted by limiting the number of data bins or decimals. Yet, this approach will affect data precision and increase estimation error of statistics requiring a high data precision. Transform coding schemes such as Discrete cosine transform (DCT) [2] are also potentially great candidates in ensuring increased client battery life but fail to guarantee high fidelity as some high-frequency components can be discarded.

Sparse Recovery [52]: 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 the raw 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.

Sketching [4, 16, 19, 51]: Sketching algorithms (sketches) are a class of approximation algorithms to estimate various statistics of a given data stream with small space and high fidelity. Summarizing the sensed data with sketches can enable highly accurate estimates for a specific metric of interest (e.g., count-min sketches [19] for heavy hitters). Sketches typically rely on 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, typical sketching algorithms not only require separate algorithms to compute multiple statistics, which is memory- and compute-heavy, but also fail to be future-proof as they cannot directly support other (possibly new) metrics of interest.

In summary, the above discussion suggests that an ideal approach should be general and future-proof by enabling high-fidelity estimates of current and future relevant metrics without making assumptions on data distributions. In this regard, recent advances in universal sketching [11, 12] appear promising. Most importantly, universal sketching can estimate a wide spectrum of statistics under arbitrary data distributions without the need to specify the metrics of interest beforehand, leaving room for future analytical tasks.

2.3 Background on Universal Sketching

Conceptually, a universal sketch maintains a unified 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 i^{th} unique element. Fortunately, many natural statistical summarizations of interest fall within this family as seen in Table 1. For instance, we evaluate several real-world datasets in Sec. 6 that need to estimate G-sum. In the Indoor Solar dataset [66] containing continuous output voltages of a solar panel, we can compute the L2-norm of the voltages to quantify the panel quality or estimate the entropy to detect anomaly events. In the LoRa Farm dataset [15] from a soil moisture sensor, we can estimate the change of the moisture values to detect environmental changes. The theory results show that if g is monotonic increasing and upper bounded by $O(f_i^2)$, then a single universal sketch can compute these G-sum functions. (A detailed analysis is out of the scope of this paper, and we refer our readers to relevant references [11, 12, 14, 45].)

However, a conceptual understanding of the computation/memory structure of sketch data structures is relevant for us in order to use it in the *Joltik* context. At a high-level, universal sketching leverages basic L_2 heavy hitter algorithms (e.g., count sketch [16]) as building blocks[1]. As a basic L_2 sketch, a count sketch maintains several arrays of counters (e.g., a matrix of counters with r rows and d columns) with independent hash functions; For each data item, it will update a randomly located counter in every row based on the corresponding hash index and use a heap to store current top-k heavy hitters. Thus, a universal sketch composes multiple layers of count sketch instances simultaneously. Each count 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 the most frequent as well as the tail of the histogram distribution. These layers then track the heavy hitters to identify the key contributors to the G-sum functions. As depicted in Fig. 4a, 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 recursively aggregated result from bottom to top can then be used to compute the desired statistic. Prior work has shown that this aggregation can be performed as an unbiased estimator of G-sum [45, 64] with bounded additive errors. 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 1. The low-power sensor computes a universal sketch over sensed samples and reports it to the base station, significantly reducing the amount of data to be transmitted. The base station can then compute the metrics 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 is to identify and address them as we discuss next.

3 *Joltik* – 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* Workflow:** *Joltik* operates as follows:

(1) **Configuration:** A *Joltik* client is configured once during its operation to configure the behavior of the sketch during its lifetime.

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

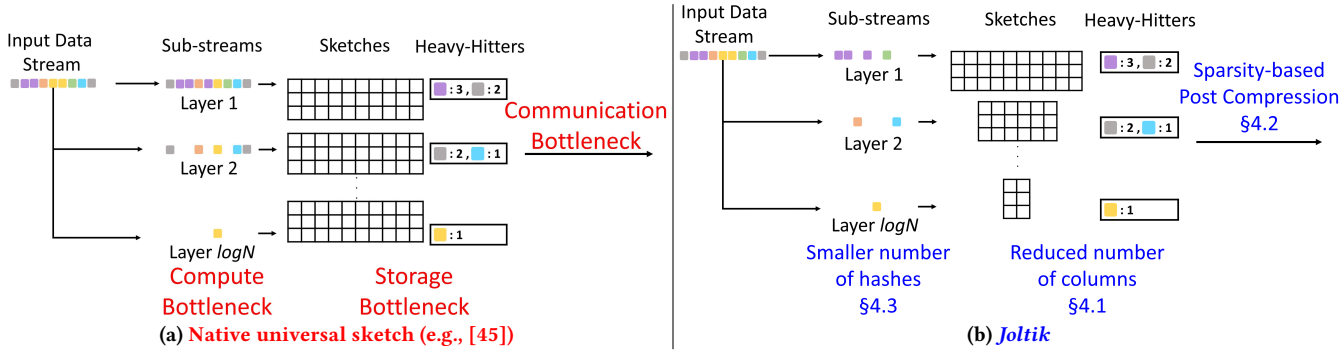


Figure 4: *Joltik* addresses the three key bottlenecks (compute, storage and communication) in prior universal sketching implementations to enable accurate general analytics for low-power clients

This decision is taken based on the battery life and accuracy requirements for a particular 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 ϵL_2-additive error guarantee (which means the difference between the measurement and the ground truth is bounded by ϵL_2) for all supported metric estimations based on the sketch size (see Sec. 4.1 for details).

(2) **Sensing and Computing:** Every collected data sample from the sensor will be fed into the universal sketch on board using the embedded micro-controller. Instead of storing the raw samples, *Joltik* keeps a universal sketch for each measurement interval.

(3) **Communication to the base station:** *Joltik* devices send the compressed form of the computed sketch over the wireless medium periodically, e.g., once per hour or day. The base station uses the sketch to estimate metrics for downstream analytics.

***Joltik* – System Challenges:** To achieve the benefits of universal sketching in practice for low-power clients, *Joltik* addresses the following challenges as depicted in Fig. 4:

Achieving high fidelity with small memory footprint: A canonical universal sketch [45] 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 Sec. 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 reduce the communication cost to some extent. Yet, every bit matters for low-power wireless transmission. In Sec. 4.2, we present our approach to dynamically reduce sketch counter sizes without affecting the accuracy of metrics for estimation.

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 independent hashes) for each collected sample, inducing computation overhead and additional energy consumption. In Sec. 4.3, we identify a simple-yet-effective opportunity to halve the computation requirement without affecting accuracy.

Tunable configuration to meet specific needs: *Joltik*'s data structure can be configured to meet application-specific goals in terms of energy and accuracy trade-off. In Sec. 4.4, we derive the relationship of *Joltik*'s parameters vs. energy as well as accuracy, and show how these configurations can be automatically derived demonstrating *Joltik*'s flexibility.

4 Detailed Design

In this section, we discuss our contributions in reducing the overall energy footprint and making universal sketches feasible on low-power sensor platforms, as shown in Fig. 4b. We also discuss how a sensor deployment can practically configure *Joltik* system parameters to get suitable accuracy-lifetime trade-offs.

4.1 Reducing Memory Footprint

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 UnivMon [45] presents a canonical realization of this approach by implementing several equal-sized count sketch [16] 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. For instance, if UnivMon allocates 600 KB memory for a 12-layer sketch to estimate thousands of distinct elements, each count sketch instance uses 50 KB memory. Reducing the total size to 60 KB for sensor deployment would mean that each count sketch gets only 5 KB, causing 10× more hash collisions and significantly larger errors. As shown in Fig. 11 in Sec. 6, we see greater than 30% errors.

Strawman solutions: Given that the naïve memory reduction on UnivMon causes significant errors, we can consider several other possible memory reduction methods: 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. 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, and leading to irrelevant failed results frequently (e.g., 1 failure out of 5 trials).

Our approach: 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 enable high fidelity estimates by 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 a smaller influence on the final statistic than the more frequent elements. (2) The lower layers need to handle a 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.

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 a geometrically smaller number of columns at each layer. Our empirical evaluation in Sec. 6 of the above demonstrates that a ratio of 1/2 provides a good energy-accuracy trade-off. Thus, every layer is allocated 1/2 the number of columns as the previous layer.

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 [16] stated in Theorem 4.1 below:

Theorem 4.1 ([16]). *For $0 < \delta, \epsilon > 1$, 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 4.1, the count sketch instances in the universal sketches are able to maintain ϵL_2-additive errors for any workload distribution. In our approach, we allocate less space for the lower layer count sketches, leading to increased ϵ values in these layers based on Theorem 4.1. For instance, if we decrease the sketch size by 2×, then the ϵ error will increase by $O(\sqrt{2})$. However, this additive error bound increase will not convert to a larger error as the actual errors depend on both ϵ and the actual L_2 norm. Intuitively, even though ϵ value increases as memory reduces, this increase is mitigated by a simultaneous reduction in the value of the L_2 norm that is multiplied with ϵ in the overall error bound in Theorem 4.1. Indeed, L_2 value always decreases in lower layers because the subsampling between sketch layers will reduce the data size and "filter out" large heavy hitters in the lower layers with high probability. Thus, allocating more space for estimating large heavy hitters in the upper layers is a wise choice. Across all of our tested sensor datasets, we confirm the observation that L_2 norm's decreasing rate is significantly higher than ϵ's increasing rate, resulting in the minimal overall impact on accuracy.

4.2 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 4.56 million 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 already achieve a 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× reduction in energy consumption. Furthermore, our memory optimization (Sec. 4.1) provides a further 5× reduction by using only a 60 KB data structure. That said, even with these techniques in place, we want to explore opportunities to further reduce the amount of data to be transmitted due to the large asymmetry in the energy cost of computation vs. communication.

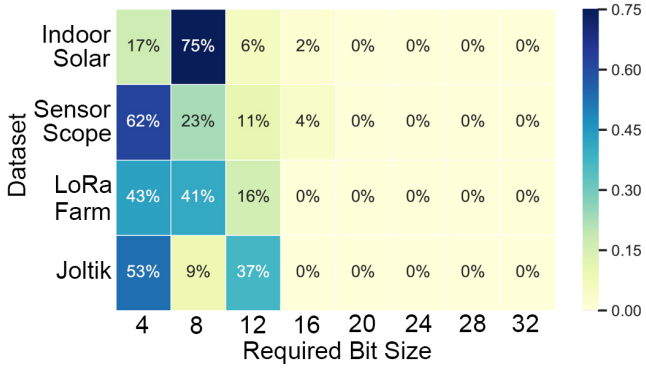
Recall that each layer of a sketch structure consists of two parts: a count sketch and a heavy hitter heap. Next, we describe how we can effectively reduce the communication footprint of transmitting these two data structures from the sensor to the base station.

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.

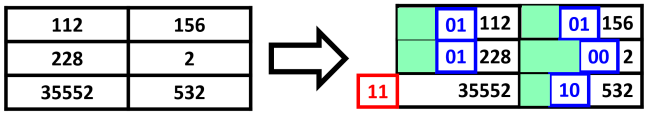
Joltik's compression of count sketches derives from a simple yet important observation that, in a certain count sketch, only a small portion of counters tend to have large values, as shown in Fig 5a. This indicates that one can compress a sketch by assigning different bit sizes to each counter.

Specifically, *Joltik* assigns two extra bits to indicate 4 levels of counter lengths, namely 4, 8, 12, and 16, and uses the corresponding number of bits to represent a certain counter. 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, this method incurs extra costs. However, it is very unlikely for a 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 very low accuracy. While *Joltik*'s compression scheme appears very simple, we show in Sec. 6 that it outperforms several common 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 makes this a worthwhile trade-off.

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, to track the top heavy hitters. It is sufficient to reconstruct a heap without heap values, as long



(a) Percentage of counters with different required bit sizes: 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.



(b) *Joltik* uses two bits to indicate counter length and represents each counter with corresponding number of bits.

Figure 5: *Joltik*'s compression scheme

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.

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.

Discussion on robustness and bit-settings: Our compression algorithm may theoretically end up transmitting more bits than the original sketch if a majority of sketch counters take more than 12 bits to store. However, empirically such scenarios are very rare in all datasets we use.

In addition, if some counter values take more than 16 bits, the compression scheme may fail. Suppose we are using a sensor with 1 Hz sampling frequency (a total sample number of 86,400 per day) and we transmit sketch summaries daily. In this case, a counter value exceeding 16 bits means that more than 65,536 out of 86,400 sensor readings are exactly the same, a situation that is extremely unlikely in real-world datasets. However, in case that this happens, in our compression scheme, we scan through all the counters before transmission. We can set the four levels of counter length based on the current counter property to maximize the compression performance and avoid failure (e.g., 4, 8, 12, and 16 bits, or 4, 8, 12, and 20 bits). Further, we can reduce the sampling frequency to 0.1 Hz (leading to a maximum counter value of 8640) or transmit every 2 hours (leading to a maximum counter value of 7200).

As our compression on the sketch summaries to be transmitted is lossless, different bit-settings in the compression scheme have no impact on data fidelity. The less extra bits we add, the longer the sensor battery life will be, which is why we set four levels of counter length differently to maximize compression performance.

4.3 Reducing Computation Overhead

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

Problem: The universal sketch algorithm processes each element using a series of operations, such as hash computations, 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 [43] demonstrates that the top two CPU performance bottlenecks in the universal sketch are (a) hash computations and (b) counter updates. Thus, the key for computation reduction is to reduce the number of hash computations and counter updates.

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 the final accuracy. At a high-level, our approach (as shown in Fig. 6) is to 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 [45] (Fig. 6a), one must update the top several layers if an element has been sub-sampled in consecutive Layers 1 to i since the hashes on these layers all return 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 Fig. 6b, 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 heavy hitters a, b have been reported on Layer i, we will use a, b's results to update Layers $i-1$ to 1 as if a, b were tracked in these upper layers. Likewise, we repeat this step for all the heavy hitter reported in other upper layers.

Intuitively, if all sketching layers are allocated with equal amounts of memory, tracking elements on their last layers provides similar (or better) accuracy as moving elements from dense upper layers to sparse lower layers yields the same (or smaller) actual errors for these elements. However, this technique seems contradictory to the idea of allocating less space for lower layers as described in Sec. 4.1. As we will show later in the section, combining this technique with the memory reduction technique does not affect the actual accuracy given the robustness of the subsampling approach and the natural skewness of the data.

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 [45], 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 [45], our approach only needs $11H + 5C$ computation as we only update one sketch at a time, reducing the cost by approximately 2×.

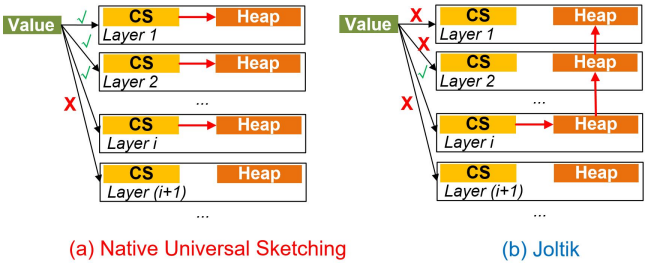


Figure 6: (a) Original update (update count sketch and heap in each layer) vs. (b) Optimized update (only update the count sketch in the last possible layer, then update the heap in the last layer and all previous layers). In this figure, "✓" means updating, "×" means not updating.

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 Sec. 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 subsampling 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 Sec. 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.

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}: it describes how frequently individual sensors should collect data; (2). Period of transmission T: it decides how frequently *Joltik* client should transmit.

Joltik then tunes its sketch size S_s (in Bytes) to identify suitable 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.[2]

Lifetime Estimation: We show how lifetime can be represented based on the above three variables (namely R_{cl}, T, and S_s). Specifically, we first compute the energy consumption per day E_{day}, and then calculate an expected lifetime.

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

For a low-power client, its battery is usually modeled based on its processing cost and its transmission cost, as in [20]. 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 [20]: (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 [20], 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:

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

Error Bound: Following the analysis in [12] and [16], 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 $1/\epsilon^2$.

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: (a). When it is necessary to transmit frequently (hence T is small) due to application requirements, one can potentially use less memory with an acceptably small loss of accuracy, but beyond a certain point the accuracy will be significantly reduced due to smaller memory; (b) 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 3 shows several sample configurations to illustrate this energy and accuracy trade-off. Here, E_b is taken to be a typical value of $3000 * 3600$ mA*s (i.e., 3000 mAh) based on an AA battery, and we use a public Indoor Solar dataset [66] for evaluation (See Sec. 6 for more details).

Let's say the user has a specific requirement of collecting and sending data frequently (e.g., 10 Hz and once per day), *Joltik* would suggest several candidate structure sizes, and provide corresponding estimate of their battery life and accuracy. Table 3 shows three such possible selections, namely 30 KB, 60 KB, and 90 KB. It is important to note that the user can get a significantly longer battery life while maintaining accuracy, if R_{cl} and T are set to 5 Hz and 2 days, respectively. However, this comes at the cost of latency of reporting. Thus, *Joltik* empowers the user to configure the parameters by proposing several candidates that best suit their needs.

5 Implementation

We implement *Joltik* in C using Mbed [50] compiler at both client and the base station. The code for implementing *Joltik* and the dataset collected on campus is available at [48]. We use commodity off-the-shelf sensors and RF boards to sense at the clients and communicate to the base station.

Requirements		Candidate Configurations		
R_{cl}	T	S_s	Lifetime	Accuracy
10 Hz	1 day	30 KB	2751 days	94.30% ± 3.27%
		60 KB	1613 days	96.61% ± 2.25%
		90 KB	1141 days	97.50% ± 1.92%
5 Hz	2 days	30 KB	5468 days	94.28% ± 3.47%
		60 KB	3215 days	96.50% ± 2.17%
		90 KB	2276 days	98.15% ± 2.08%

Table 3: *Joltik* deployment example

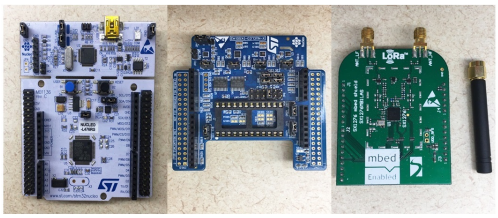


Figure 7: Hardware Components of *Joltik*: (a) Microcontroller (NUCLEO-L476RG); (b) Sensor Board (X-NUCLEO-IKS01A2); (c) Transceiver (SX1276 Long Range Transceiver).

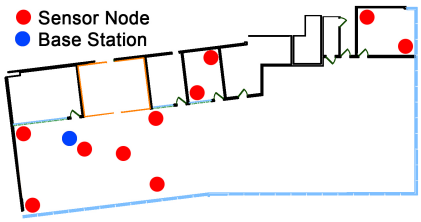


Figure 8: Sensor locations in the building

Sensor Node. The *Joltik* sensor node consists of three parts — sensor board, micro-controller (MCU), and transceiver:

- *Sensor Board:* We use the X-NUCLEO-IKS01A2 (Fig. 7b), 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.
- *MCU:* We use the NUCLEO-L476RG board (Fig. 7a) , which contains a STM32L476RGT6U MCU in the LQFP64 package. This MCU runs *Joltik* to generate sketch summaries and assembles LoRaWAN packets.
- *RF Frontend:* We use the SX1276 LoRa Transceiver (Fig. 7c) to communicate with the base station. In our experiments, we set this transceiver to operate at 915MHz, SF 10, and 125 KHz bandwidth.

Base Station. *Joltik* base station consists of two parts — the MCU and the 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 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 Sec. 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 section, we first evaluate *Joltik* on a proof-of-concept deployment in a campus building with ten pressure sensors and one base station (as shown in Fig. 8). We also study the performance of *Joltik* by emulating three varieties of sensor deployments with three other large real-world datasets. Across these sensor datasets, we compare *Joltik* with other alternatives, and the results are shown in Sec. 6.1. Then, we compare the performance of *Joltik* with the native universal sketching algorithm, and the results are shown in Sec. 6.2.

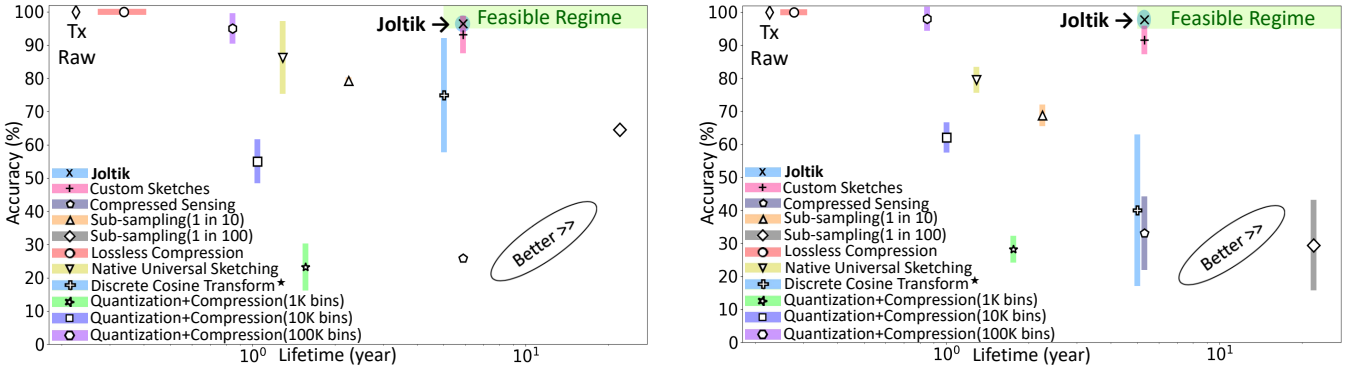
Note that when configuring *Joltik*, we follow the guidelines in Sec. 4.4 and balance the energy and accuracy trade-off. In addition, when comparing with other schemes such as compressed sensing or custom per-metric sketches, we use the same energy budget.

Our major findings are as follows:

- *Joltik* achieves significantly better accuracy (97.9%) compared to sub-sampling (60.1%), native universal sketching (82.3%), compressed sensing (34.9%), and discrete cosine transform (57.8%) using the same amount of 60 KB memory across datasets and across evaluation tasks.
- *Joltik* achieves a longer sensor lifetime (5.6 years), compared to canonical lossless compression (0.3 years) and lossy compression (0.85 years) given similar accuracy requirements.
- *Joltik* reduces 96% of the sensor power consumption compared to transmitting raw data while maintaining high accuracy (> 95%) on the tested tasks.
- *Joltik* supports a range of analytical tasks without extra power consumption on the sensor nodes.

Datasets: In addition to our own dataset, we also consider three real-world datasets from previous work: (1) **Indoor Solar [66]**: 2 years of joint high accuracy power and ambient condition traces at 6 diverse indoor locations for energy harvesting systems. (2) **SensorScope [30]**: Environmental data from past SensorScope deployments [63]. In our experiments, we use the global current value to test system performance. (3) **LoRa Farm [15]:** 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.

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



(a) Indoor Solar dataset **(b) LoRa Farm dataset**

Figure 9: *Joltik* Accuracy-Energy Trade-off: *Joltik* provides better battery lifes while providing better accuracy.

Battery life estimation: To estimate the power consumption of *Joltik* and other baselines, we leverage prior current models of our board [20]. The power model is based on the actual current measurements from the Semtech SX1276 board which we use for evaluation. This model estimates the required current for the computation, communication, reception, and sleeping modes of the RF board. Using these current and time numbers, we report the device's total power consumption (including sampling, processing, and transmitting) for one day. We then leverage these to estimate the battery life of the client by using a standard AA battery as the energy source (typical for an LP-WAN client).

Baselines: We compare *Joltik* with multiple baselines. Note that an exhaustive comparison to all lossy compression techniques is out of the scope of this paper. Here, we consider two commonly used lossy compression techniques – canonical lossy compression and transform coding lossy compression. Below are detailed descriptions of all baselines:

(a). *Tx Raw* – transmitting all raw data.

(b). *Sub-sampling* – downsampling raw data with a defined ratio and transmitting sub-sampled data.

(c). *Lossless Compression* – we use Huffman coding [33].

(d). *Compressed Sensing* – we use CoSaMP [52], a widely used compressed sensing algorithm.

(e). *Canonical Lossy Compression* (marked as "Quantization + Compression" in Fig. 9) – we tested several compression techniques: i). quantization (data binning) [27] followed by differential encoding and entropy coding (Huffman coding) [33]; ii). rounding [59] followed by differential encoding and entropy coding; iii). quantization followed by entropy coding. Technique i) performs the best and is selected to represent lossy compression in Fig. 9.

(f). *Transform Coding Lossy Compression*[3] – performing Discrete Cosine Transform (DCT) [2] on raw data samples and using the top K (in our experiments, $K = 8,000$) coefficients to represent the raw data.

(g). *Custom Per-metric Sketches* (marked as "Custom Sketch" in Fig. 9) – specifically designed sketches for the above metrics; in our experiments, we use Count-Min Sketch [19] for HH, Hyperloglog [22] for Cardinality, entropy estimation algorithm [17] for Entropy, and Count Sketch [16] for L2-norm.

(h). *Native Universal Sketching* – the original version of universal sketching [45].

6.1 End-to-end System Performance

We evaluate *Joltik* on the following performance metrics: accuracy, device power consumption, and multi-task handling (i.e., computing multiple metrics simultaneously) using the same energy budget.

Energy-Accuracy Trade-off: We run experiments on all four datasets 30 times (note that sampling across hash layers is probabilistic) and report the *median* and the *standard deviation* of errors. Note that we only report the experiment results using *Indoor Solar* and *LoRa Farm* datasets in Fig. 9 due to limited space, but experiments using the other two datasets have similar results, and Fig. 2 shows the results from all four datasets.

Result: As depicted in Fig. 9, *Joltik* achieves significantly better accuracy in all tested tasks and datasets over sub-sampling, compressed sensing, custom per-metric sketches, and lossy compression. We also notice that, for sub-sampling, as we increase the rate from 10 to 100, 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 for sparser datasets, the compressed sensing approach fails miserably on dense data, achieving only about 30% accuracy. While one would assume custom per-metric sketches to outperform *Joltik* in terms of accuracy, remember that when we constrain the system to have equal energy consumption, each custom sketch only gets $\frac{1}{4}$ of the total energy. Thus, *Joltik* outperforms the custom per-metric sketch method. DCT also fails to guarantee high fidelity, as we are only using the top 8,000 coefficients to represent the raw data, and the information kept in high-frequency components is lost.

We also tested *Joltik*'s performance to canonical lossy compression baselines. *Joltik* outperforms all of them in relation to the energy-accuracy trade-off. The problem with quantization or rounding is that, when reducing the number of data bins or decimals, low data precision will lead to a high error rate in statistics like HH (in the worst case, no matched heavy hitters can be found, leading to a 100% error rate). Also, the battery life improvement provided by

[3]Given the complexity of DCT algorithm and limited computation capacity of our sensor hardware, DCT result shown in Fig. 9 is extrapolated: We report accuracy result by running DCT on real sensor datasets, and report estimated lifetime by calculating sensing and communication energy cost. ⋆ in Fig. 9 denotes extrapolated result.

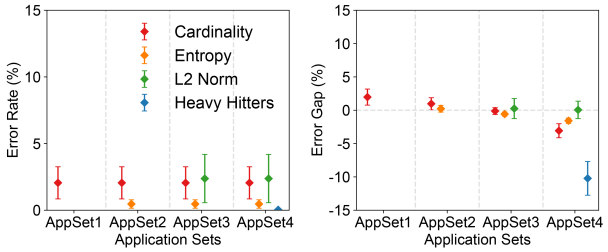


Figure 10: Generality and Multi-task Handling. Left: Error rate of *Joltik* estimating four application sets using the same energy budget. Right: Error gap between *Joltik* and custom per-metric sketches estimating four application sets using the same energy budget. positive values imply *Joltik* is worse and vice versa). Both experiments use *Joltik* dataset with 1 Hz sensor sampling rate.

these canonical lossy compression techniques compared to transmitting raw data is limited (3.75× when keeping 100 k data bins and 7.1× when keeping 1 k data bins). The reason is that, when we perform quantization on the raw data samples, we are reducing the number of bits needed for each sample, rather than reducing the total number of samples. Suppose the sensor is sampling at 1 Hz, transmitting 86,400 32-bit data samples per day (a total of 345.6 KB). After quantization, we would transmit 86,400 16-bit data samples per day (a total of 172.8 KB), leading to a 2× reduction in the size of transmission data. Differential coding and entropy coding further compress the data by another 3.5×, resulting in a 7× total improvement. On the other hand, *Joltik*, with its S_s set as 60 KB, transmits only around 16 KB per day (our technique in Sec. 4.2 provides a 3.84× compression ratio), hence it provides a much longer battery life than these lossy compression techniques.

Joltik is also more energy-efficient over baseline approaches of transmitting raw data (24.6×) and lossless compression (16.4×). 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 5.6 years.

At first glance, sub-sampling, custom per-metric sketches, DCT, and canonical lossy compression (with 1 k data bins) 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. Fig. 9 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.

Generality: To evaluate generality, we deploy both *Joltik* and custom per-metric 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 custom per-metric sketches operate under the same energy budget (5 years of battery lifetime using a typical AA battery). For custom per-metric sketches, when evaluating multi-task handling, we divide the power budget uniformly across different custom sketches.
Result: Fig. 10 shows that running multiple tasks on *Joltik* using the same energy budget does 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, which require a variety of features from sensory 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 custom per-metric sketches (*Joltik* error rate subtracted by custom per-metric sketch error rate; i.e., a positive value implies *Joltik* is worse and vice versa). As expected, when more tasks are needed, *Joltik* performs significantly better than custom per-metric sketches in terms of accuracy. This is due to the consequently decreasing power budget for individual tasks.

6.2 Evaluating *Joltik*'s Optimizations

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

Memory Footprint: We evaluate our approach described in Sec. 4.1 by deploying both *Joltik* and the original universal sketching algorithm [45] to perform 2 analytical tasks (HH and L2-norm). We vary the amount of memory for both approaches to study the impact on accuracy.
Result: As depicted in Fig. 11, *Joltik* achieves low-error estimation for both 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.

Communication Overhead: We evaluate *Joltik*'s approach for reducing the communication overhead by comparing the power consumption of *Joltik*'s scheme with lossless compression schemes such as Huffman encoding, LZW, Delta Encoding, and native universal sketch.
Result: Our results in Table 4 demonstrate a 3.84× reduction in the size of transmission data and a 3.76× reduction in 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.

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. Fig. 12 shows increasing benefits as sensors collect raw data more frequently, demonstrating a 2× reduction in the sketch processing power. (At lower sampling frequencies, transmission costs dominate.)

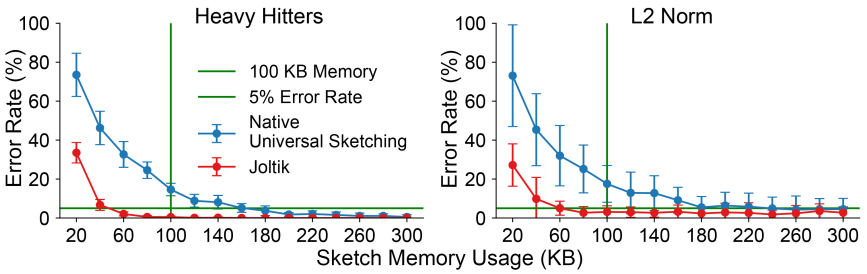


Figure 11: Evaluation of memory optimization: Error rates of *Joltik* and the original universal sketching on *Joltik* Dataset with 1 Hz sensor sampling rate (feasible region is within 5% error rate and 100KB sensor memory limit).

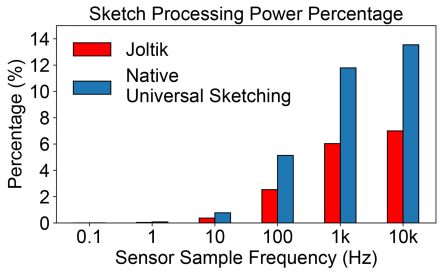


Figure 12: Evaluation of computation optimizations: The percentage of sketch processing power out of total sensor power.

Compression Method	Compression Ratio	Energy (mAs)
Reducing 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 4: Comparing Sketch Compression Methods

7 Related Work

Low-Power IoT: There has been much work done in Low-Power Wide-Area Networks (LP-WANs) for synchronization [3, 55], association [23, 41], optimizing power [5, 20], improving scalability [21, 24, 58], and client power adaptation [74]. Recent trends [31, 37] demonstrate moving complex functions off the low-power client to the more powerful base stations. LP-WAN clients typically can communicate only a few KBs of data every communication cycle with the above approaches increasing its throughput by 3 – 5×. Within the same communication constraints, *Joltik*'s approach can improve the accuracy of measured metrics from the same LP-WAN client and can enable better quality estimates of a wide range of statistics, potentially with a simple software update.

Aggregation in Sensor Networks: Retrieving information from a large number of IoT clients or sensor nodes has been widely studied. While some approaches, such as compressed sensing [8, 9, 29, 34, 52, 71], leverage the sparsity of information to retrieve the data, other approaches use machine learning [61, 69] or statistical sampling [28, 68] to retrieve information from a large number of sensors in a network. There is also a rich literature on lossless compression [67], such as dictionary based [49, 60] and predictive coding based [32, 40], with a caveat of providing relatively modest compression without affecting the accuracy of statistics. On the other hand, lossy compression, such as quantization/rounding based [27, 57] or transform coding (e.g., discrete cosine transform [2]) based, trades some accuracy for further compression on data size. The last approach for aggregation [6, 7, 35, 53, 54, 62, 70] in wireless sensor networks exploits the spatio-temporal correlations to minimize the information required to calculate specialized metrics. In contrast, LP-WAN technologies present a new challenge, as the communication is centralized instead of distributed, which means each client communicates directly with the base station. This means we cannot leverage hierarchy or spatio-temporal correlations to alleviate the energy deficit by reducing the amount of data to be communicated. Instead, as shown in Table 2, *Joltik* complements these solutions by providing a generalized analytics framework that does not rely on assumptions about the sensed data and can operate in a centralized manner.

Sketching for Data Analytics: Sketching algorithms for aggregate statistics have been explored in various contexts, including stream data processing [4, 10, 18, 44, 51], database [16, 19, 25] and network telemetry [36, 43, 45, 46, 72, 75]. Perhaps the closest related work is UnivMon [45], which enables real-time general network telemetry. While UnivMon has focused on adopting and optimizing universal sketches for high-power network switches, *Joltik* instead optimizes for the battery life of the low-power client communicating with the base station. This presents new challenges in each component (hashing, storage, and transmission) of universal sketching, which *Joltik* alleviates to enable general, accurate, and energy-efficient analytics for low-power clients.

8 Conclusion and Future Work

This paper presents *Joltik*, a framework for general and energy-efficient analytics on low-power IoT clients. *Joltik* enables this by optimizing universal sketching for storage, computation, and communication, making it compatible with low-power clients. A detailed evaluation 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 computation resources at the edge and cloud to leverage rich sensed information for various applications, while maintaining the energy-efficiency of individual sensors.

Acknowledgement

We would like to thank the shepherd and reviewers for their valuable comments and insightful feedback. This work was supported in part by seed grants from the Kavčić-Moura program and the CMU Scott Institute, the Cylab IoT initiative, the NSF (grants 1942902, 1837607), and CONIX Research Center, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

References

[1] P. K. Agarwal, G. Cormode, Z. Huang, J. M. Phillips, Z. Wei, and K. Yi. Mergeable summaries. *ACM Transactions on Database Systems (TODS)*, 38(4):1–28, 2013.

[2] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE transactions on Computers*, 100(1):90–93, 1974.

[3] A. Ali and W. Hamouda. On the cell search and initial synchronization for NB-IoT LTE systems. *IEEE Communications Letters*, 21(8):1843–1846, 2017.

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

[5] 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*, 31(6):8–15, 2017.

[6] 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)*, 2003.

[7] 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.*, 22(6):1130–1140, Sept. 2006.

[8] 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*, pages 134–142. ACM, 2006.

[9] R. G. Baraniuk, V. Cevher, M. F. Duarte, and C. Hegde. Model-Based Compressive Sensing. *IEEE Transactions on Information Theory*, 56(4):1982–2001, Apr 2010.

[10] 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*, pages 261–276, 2016.

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

[12] V. Braverman and R. Ostrovsky. Zero-one frequency laws. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, STOC '10, page 281–290, New York, NY, USA, 2010. Association for Computing Machinery.

[13] V. Braverman and R. Ostrovsky. 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*, pages 58–70. Springer, 2013.

[14] V. Braverman, R. Ostrovsky, and A. Roytman. Zero-one laws for sliding windows and universal sketches. In *Proc. of APPROX/RANDOM*, 2015.

[15] 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*, pages 26–28. ACM, 2019.

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

[17] P. Clifford and I. Cosma. A simple sketching algorithm for entropy estimation over streaming data. In *Artificial Intelligence and Statistics*, pages 196–206, 2013.

[18] G. Cormode and M. Garofalakis. Sketching probabilistic data streams. In *Proc. of ACM SIGMOD*, 2007.

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

[20] 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)*, pages 60–71. IEEE, 2018.

[21] 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*, pages 309–321. ACM, 2017.

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

[23] 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.

[24] 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.

[25] E. Gan, J. Ding, K. S. Tai, V. Sharan, and P. Bailis. Moment-based quantile sketches for efficient high cardinality aggregation queries. *Proc. VLDB Endow.*, 11(11):1647–1660, July 2018.

[26] S. Gandhi, S. Suri, and E. Welzl. Catching elephants with mice: sparse sampling for monitoring sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 6(1):1–27, 2010.

[27] A. Gersho and R. M. Gray. *Vector quantization and signal compression*, volume 159. Springer Science & Business Media, 2012.

[28] A. Gilbert and P. Indyk. Sparse recovery using sparse matrices. *Proceedings of the IEEE*, 98(6):937–947, 2010.

[29] 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*, STOC '07, pages 237–246, New York, NY, USA, 2007. ACM.

[30] Guillermo Barrenetxea. Sensorscope Data. https://doi.org/10.5281/zenodo.2654726, 2019. Accessed: 2019-12-12.

[31] 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.

[32] 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.

[33] D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[34] 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*, FOCS '08, pages 199–207, Washington, DC, USA, 2008. IEEE Computer Society.

[35] 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*, MobiCom '00, pages 56–67, New York, NY, USA, 2000. ACM.

[36] N. Ivkin, R. B. Basat, Z. Liu, G. Einziger, R. Friedman, and V. Braverman. I know what you did last summer: Network monitoring using interval queries. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3(3):1–28, 2019.

[37] 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*, pages 391–403. ACM, 2019.

[38] 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*, COMPASS'19, July 2019.

[39] A. Lavric and V. Popa. Internet of things and lora™ low-power wide-area networks: a survey. In *2017 International Symposium on Signals, Circuits and Systems (ISSCS)*, pages 1–5. IEEE, 2017.

[40] Y. Liang and W. Peng. Minimizing Energy Consumptions in Wireless Sensor Networks via Two-modal Transmission. *SIGCOMM Comput. Commun. Rev.*, 40(1):12–18, Jan. 2010.

[41] X. Lin, A. Adhikary, and Y.-P. E. Wang. Random access preamble design and detection for 3GPP narrowband IoT systems. *IEEE Wireless Communications Letters*, 5(6):640–643, 2016.

[42] LinkLabs. Firmware-over-the-air (fota) with lora. https://www.link-labs.com/blog/firmware-over-the-air-fota-with-lora, 2017.

[43] 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*, pages 334–350. ACM, 2019.

[44] Z. Liu, N. Ivkin, L. Yang, M. Neyrinck, G. Lemson, A. Szalay, V. Braverman, T. Budavari, R. Burns, and X. Wang. Streaming algorithms for halo finders. In *2015 IEEE 11th International Conference on e-Science*, pages 342–351. IEEE, 2015.

[45] 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.

[46] Z. Liu, S. Zhou, O. Rottenstreich, V. Braverman, and J. Rexford. Memory-efficient performance monitoring on programmable switches with lean algorithms. 2019.

[47] 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.

[48] M. Yang, J. Zhang, A. Gadre, Z. Liu, S. Kumar, and V. Sekar. Joltik source code. https://github.com/Joltik-project/Joltik, 2020.

[49] F. Marcelloni and M. Vecchio. Enabling energy-efficient and lossy-aware data compression in wireless sensor networks by multi-objective evolutionary optimization. *Information Sciences*, 180(10):1924–1941, 2010.

[50] Mbed. Mbed development tools. https://os.mbed.com/.

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

[52] D. Needell and J. A. Tropp. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Applied and computational harmonic analysis*, 26(3):301–321, 2009.

[53] S. Pattem, B. Krishnamachari, and R. Govindan. The Impact of Spatial Correlation on Routing with Compression in Wireless Sensor Networks. *ACM Trans. Sen. Netw.*, 4(4):24:1–24:33, Sept. 2008.

[54] 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.*, 2003.

[55] 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*, pages 289–300. ACM, 2019.

[56] S. Randhawa and S. Jain. Data aggregation in wireless sensor networks: Previous research, current status and future directions. *Wireless Personal Communications*, 97(3):3355–3425, 2017.

[57] K. R. Rao and P. C. Yip. *The transform and data compression handbook*. CRC press, 2018.

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

[59] S. M. Rump, T. Ogita, and S. Oishi. Accurate floating-point summation part ii: Sign, k-fold faithful and rounding to nearest. *SIAM Journal on Scientific Computing*, 31(2):1269–1302, 2009.

[60] 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*, SenSys '06, pages 265–278, New York, NY, USA, 2006. ACM.

[61] 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*, 61:217–238, 2014.

[62] A. Scaglione and S. Servetto. On the Interdependence of Routing and Data Compression in Multi-hop Sensor Networks. *Wirel. Netw.*, 11(1-2):149–160, Jan. 2005.

[63] 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.

[64] 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.

[65] Semtech. Smart Electricity Metering using LoRa. https://www.semtech.com/lora/lora-applications/smart-electricity-metering, 2020.

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

[67] T. Srisooksai, K. Keamarungsi, P. Lamsrichan, and K. Araki. Practical Data Compression in Wireless Sensor Networks: A Survey. *J. Netw. Comput. Appl.*, 35(1):37–59, Jan. 2012.

[68] A. Stein and C. Ettema. An overview of spatial sampling procedures and experimental design of spatial studies for ecosystem comparisons. *Agriculture, Ecosystems & Environment*, 94(1):31–47, 2003.

[69] 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*, 142:283–297, 2017.

[70] 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)*, 2004.

[71] J. A. Tropp and A. C. Gilbert. Signal Recovery From Random Measurements Via Orthogonal Matching Pursuit. *IEEE Trans. Inf. Theor.*, 53(12):4655–4666, Dec. 2007.

[72] 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.

[73] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Computer networks*, 52(12):2292–2330, 2008.

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

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