Joltik:

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

Mingran Yang, Junbo Zhang, Akshay Gadre,
Zaoxing Liu, Swarun Kumar, and Vyas Sekar

Carnegie Mellon University

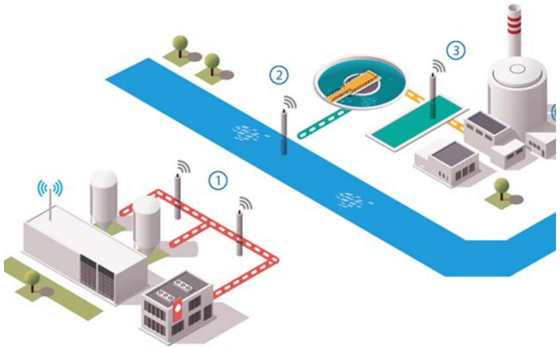
LPWAN provides opportunity for many applications

√ Long range
√ Low power

LoRa™ NB-IoT™

LTE-M sigfox

Low-power wide-area network (LPWAN) Technologies

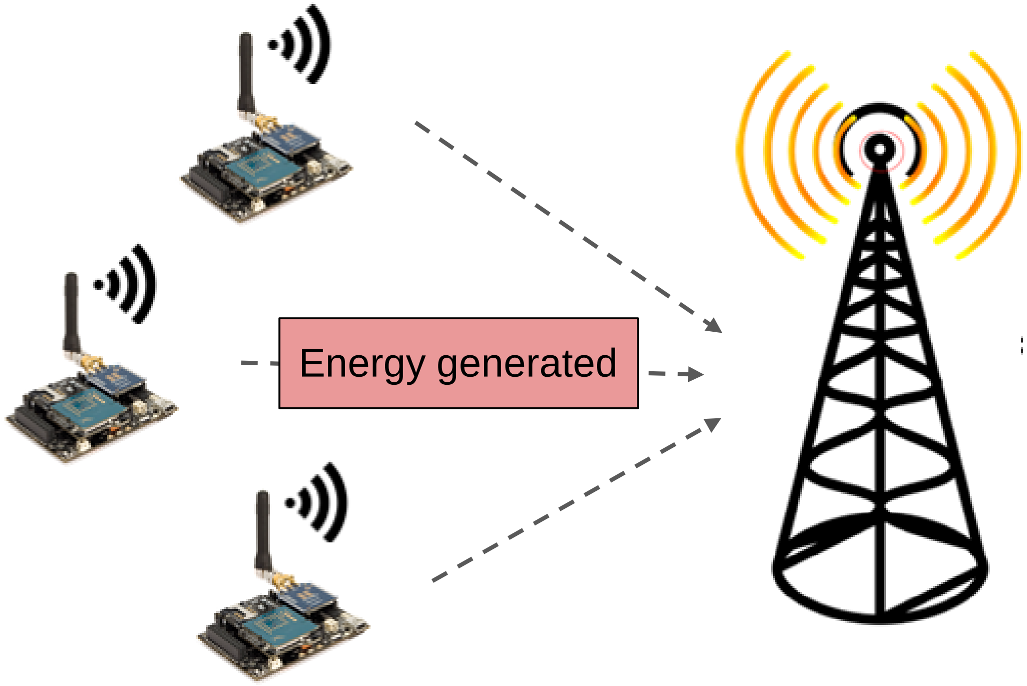


Industrial monitoring

Smart city

Smart farming

Generality vs. Power trade-off for LPWAN applications



Downstream Analytics

Energy generated

Power outages

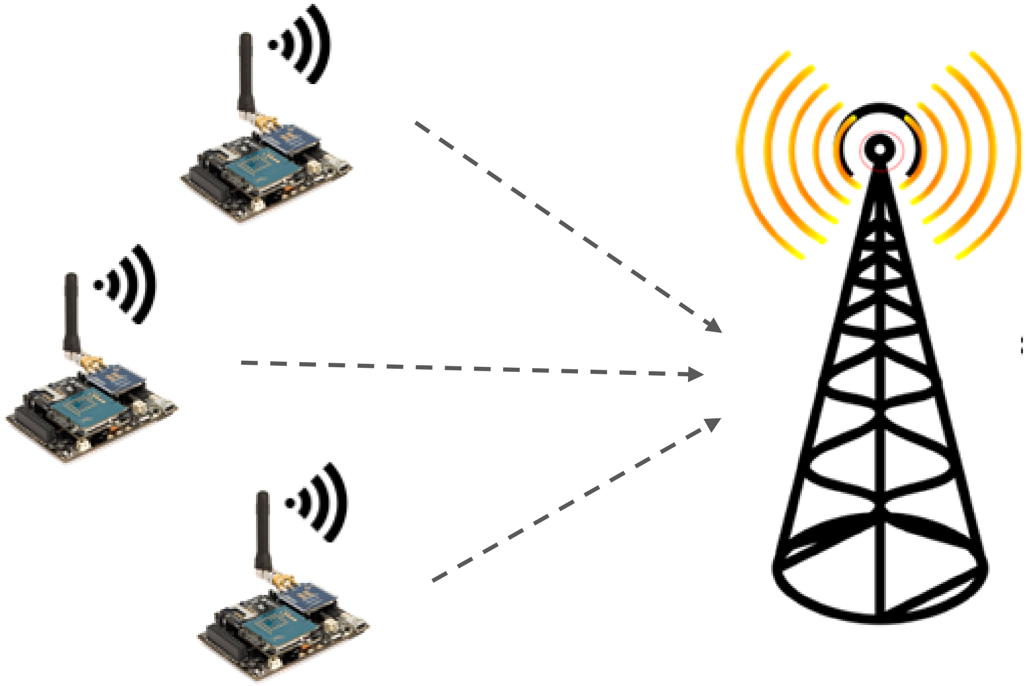
Weather events

……

- Occasional short samples
- Pre-determined summary statistics

New tasks → additional energy overhead

System design goal 1: generality



Downstream Analytics

Energy generated →

Power outages →

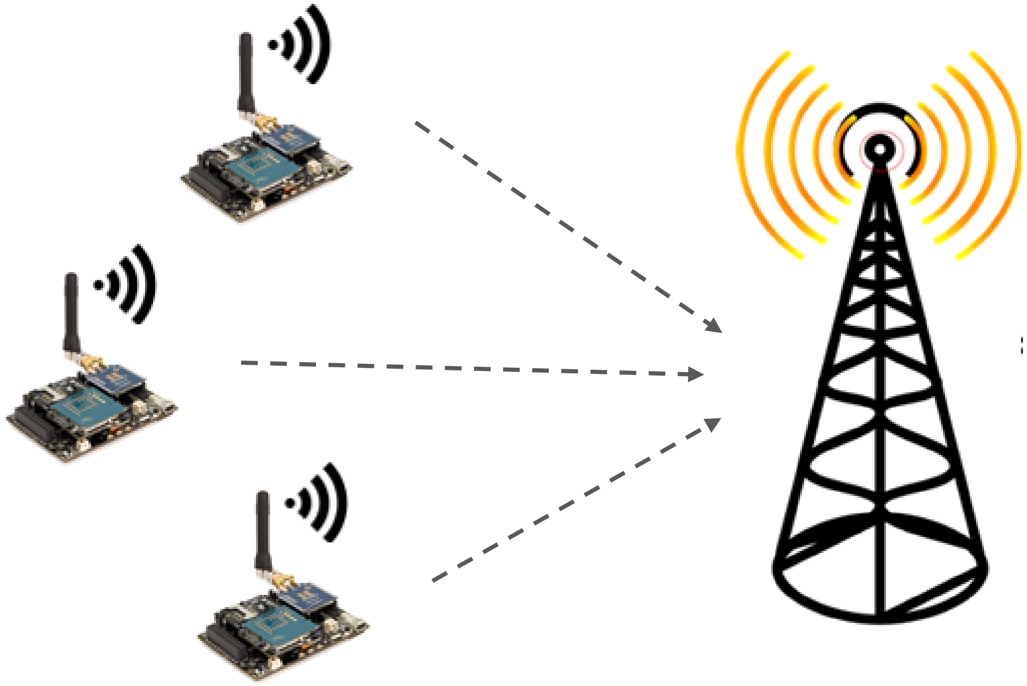
Weather events →

…… →

↓

Support estimation on multiple statistics simultaneously

System design goal 2: high-fidelity



<u>Downstream Analytics</u>

Energy generated

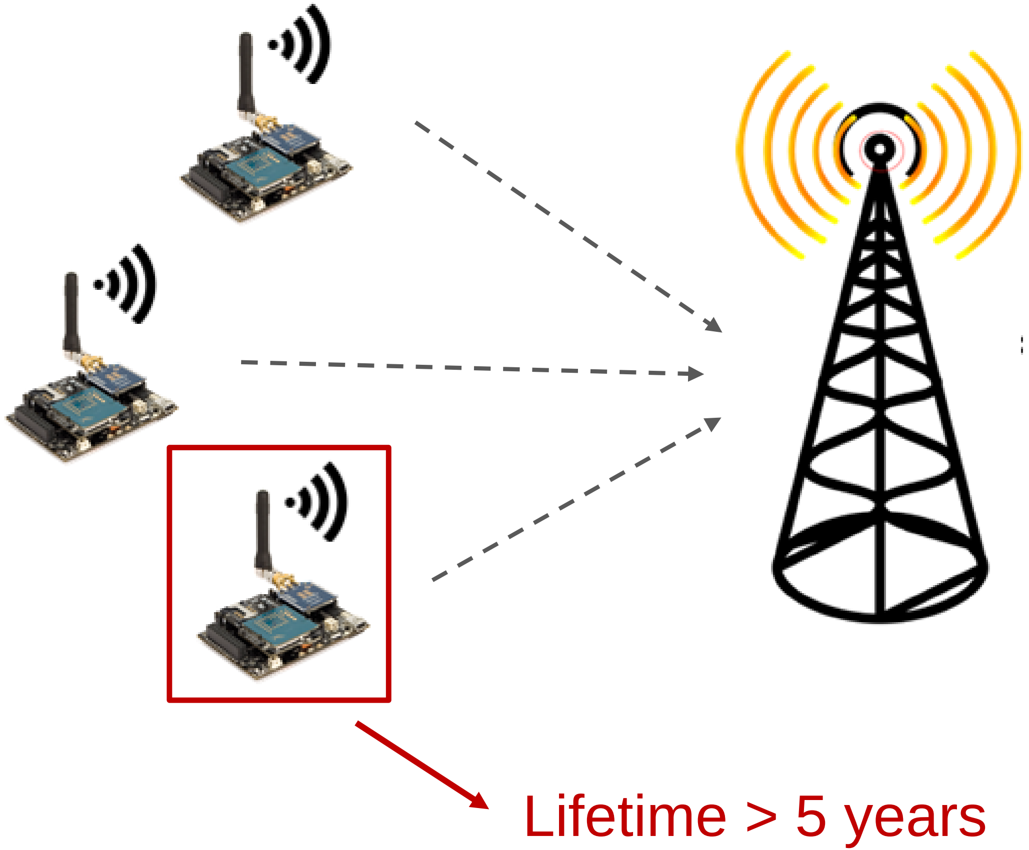
Power outages

Weather events

……

Error rate < 5%

System design goal 3: energy-efficiency



Downstream Analytics

Energy generated

Power outages

Weather events

……

Existing solutions and limitations

Approach	Energy-efficiency	High-fidelity	Generality
Sub-sampling	√	X	√
Lossless compression	X	√	√
Lossy compression	√	X	√
Sparse recovery	√	√	X
Data-centric aggregation	√	√	X

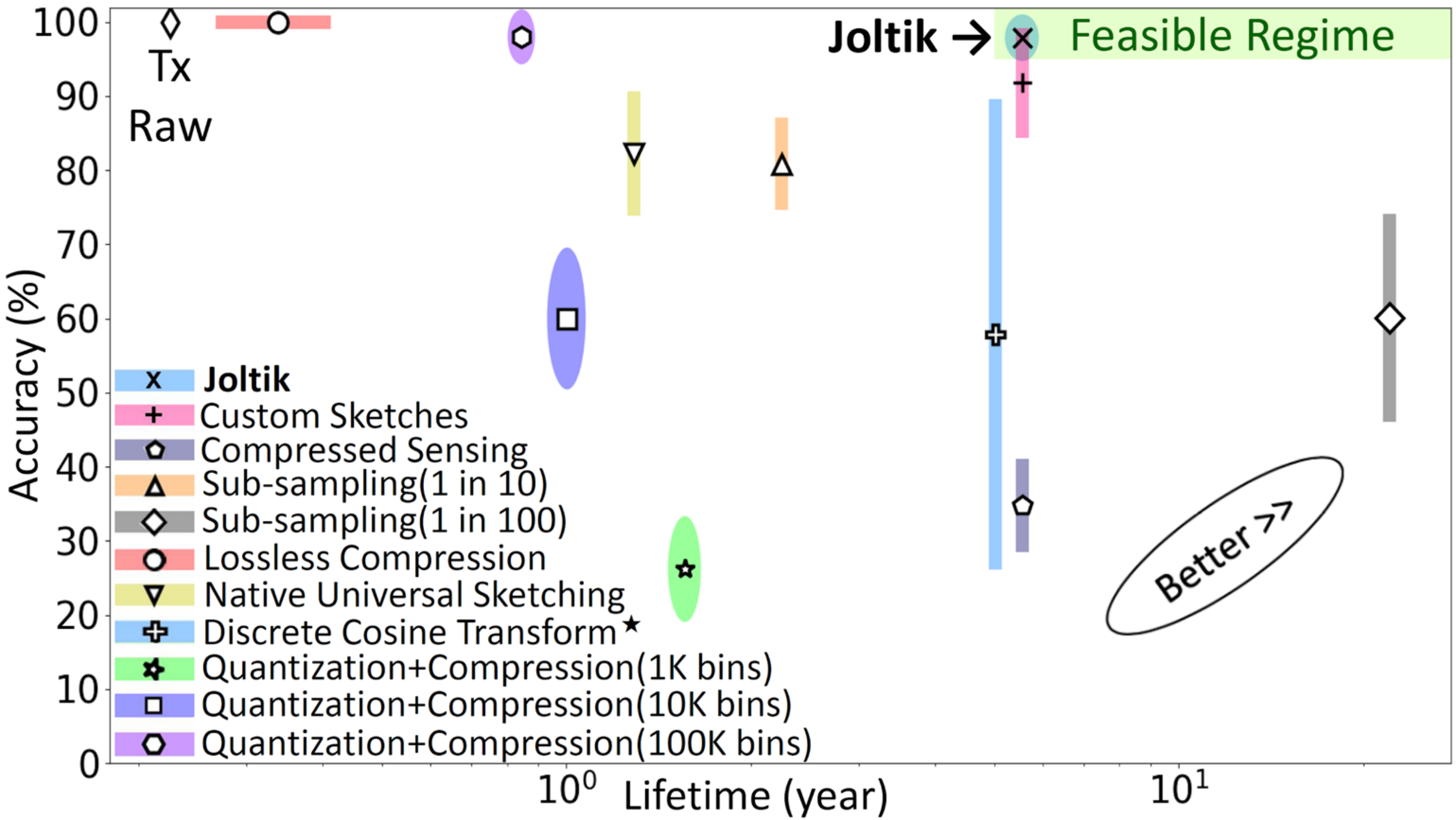
Joltik vs. existing solutions

Approach	Energy-efficiency	High-fidelity	Generality
Sub-sampling	√	X	√
Lossless compression	X	√	√
Lossy compression	√	X	√
Sparse recovery	√	√	X
Data-centric aggregation	√	√	X
Joltik	√	√	√

Universal sketching

- Support the estimation on broad class of statistics
- No requirements on raw data / network topology

Joltik vs. existing solutions

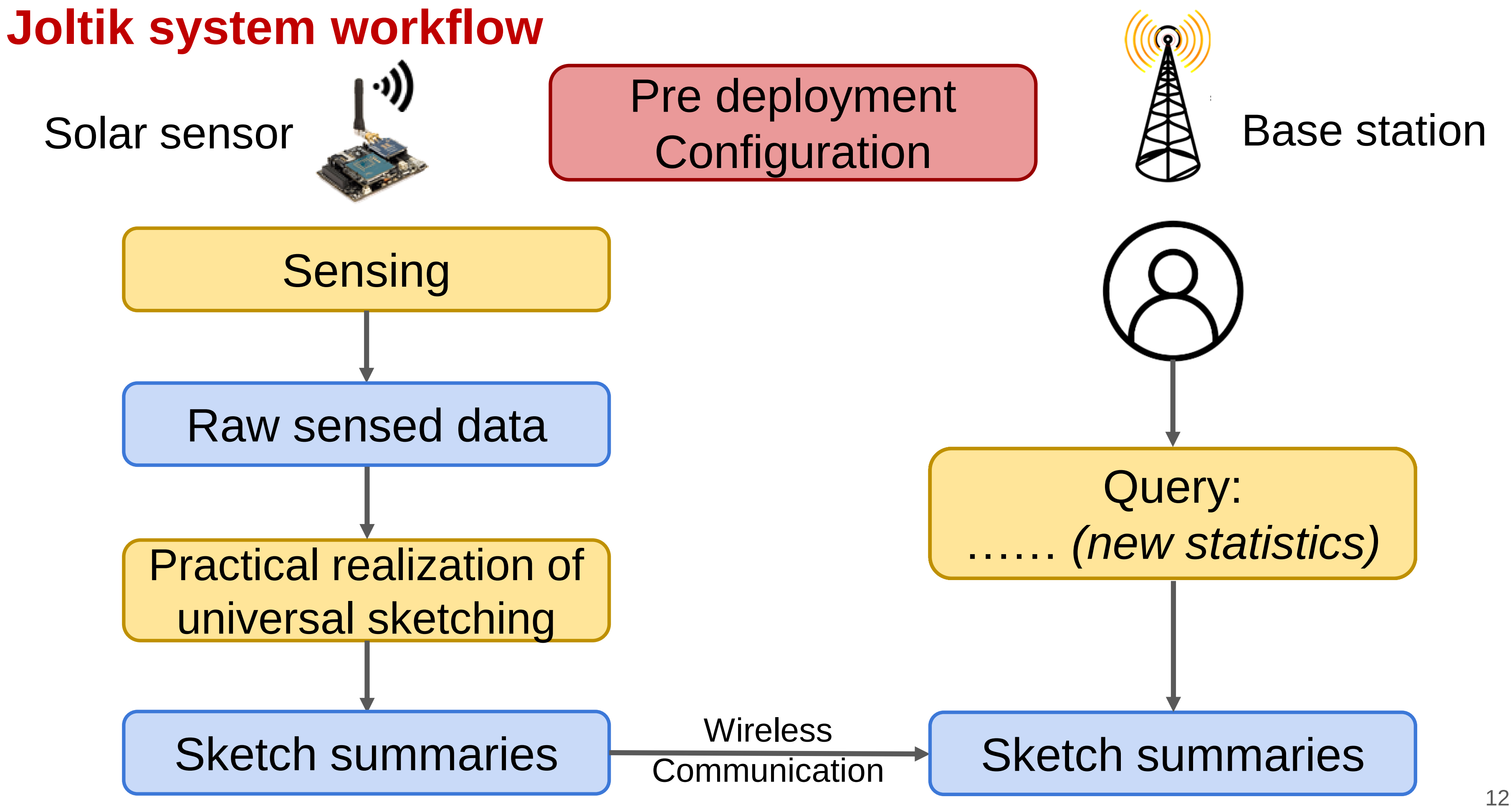


Outline for this talk

- Motivation
- System workflow and challenges
- System design
    - Low memory footprint - reducing memory footprint
    - Low power - reducing communication footprint
    - Low CPU - reducing computation overhead
- Implementation and evaluation
- Conclusions

Outline for this talk

- Motivation
- System workflow and challenges
- System design
  - Low memory footprint - reducing memory footprint
  - Low power - reducing communication footprint
  - Low CPU - reducing computation overhead
- Implementation and evaluation
- Conclusions



Joltik system workflow
Solar sensor
Pre deployment Configuration
Base station
Sensing
Raw sensed data
Practical realization of universal sketching
Sketch summaries
Wireless Communication
Query:
…… (new statistics)
Sketch summaries

Background on universal sketching

- Sketching algorithm:

e.g., count-min sketch serves as a **frequency table** of events in a data stream

- Universal Sketching:

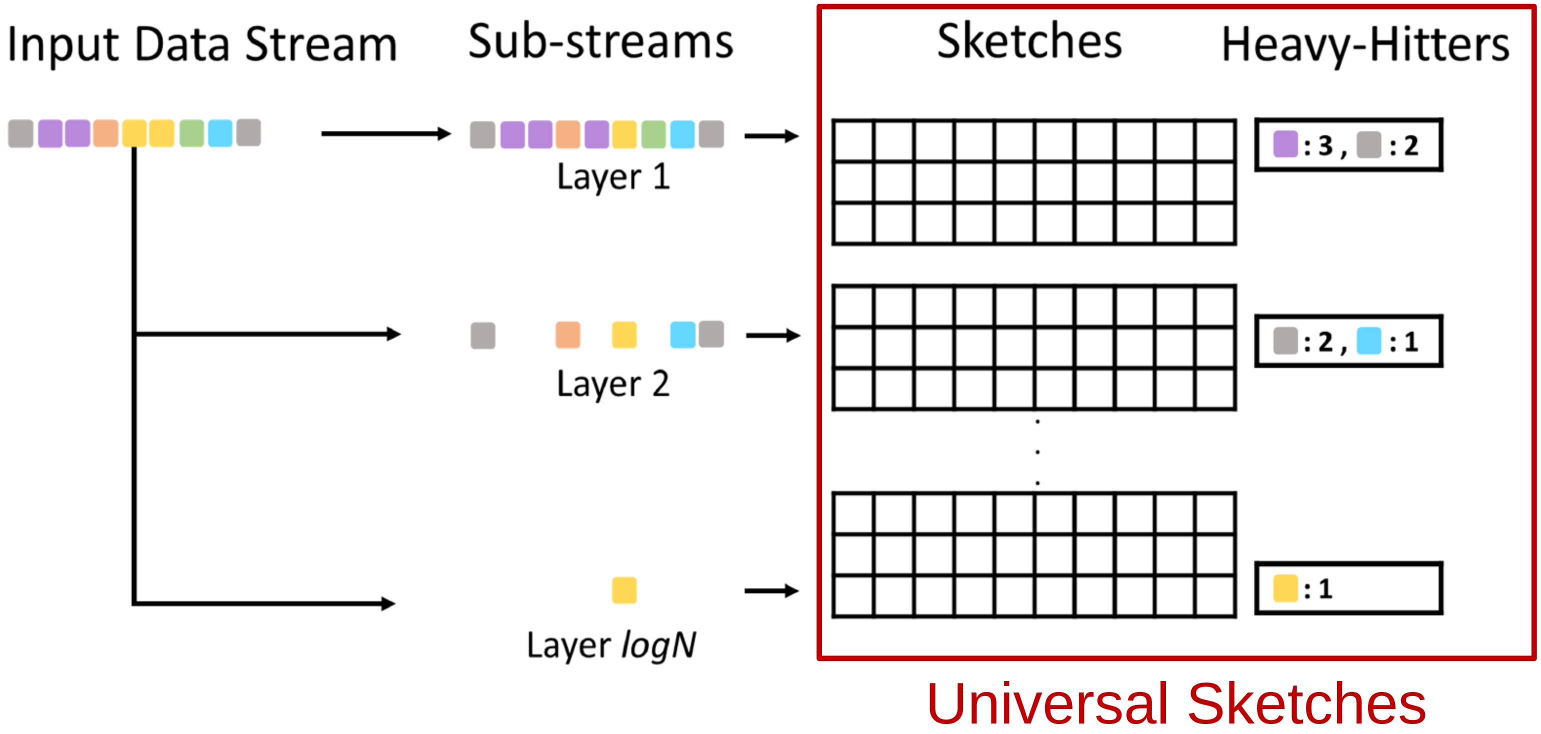
Why universal sketching is a promising solution?

Universal sketching can estimate **many already known (generality)**, or even **possibly unforeseen (future-proof)** statistics at the same time.

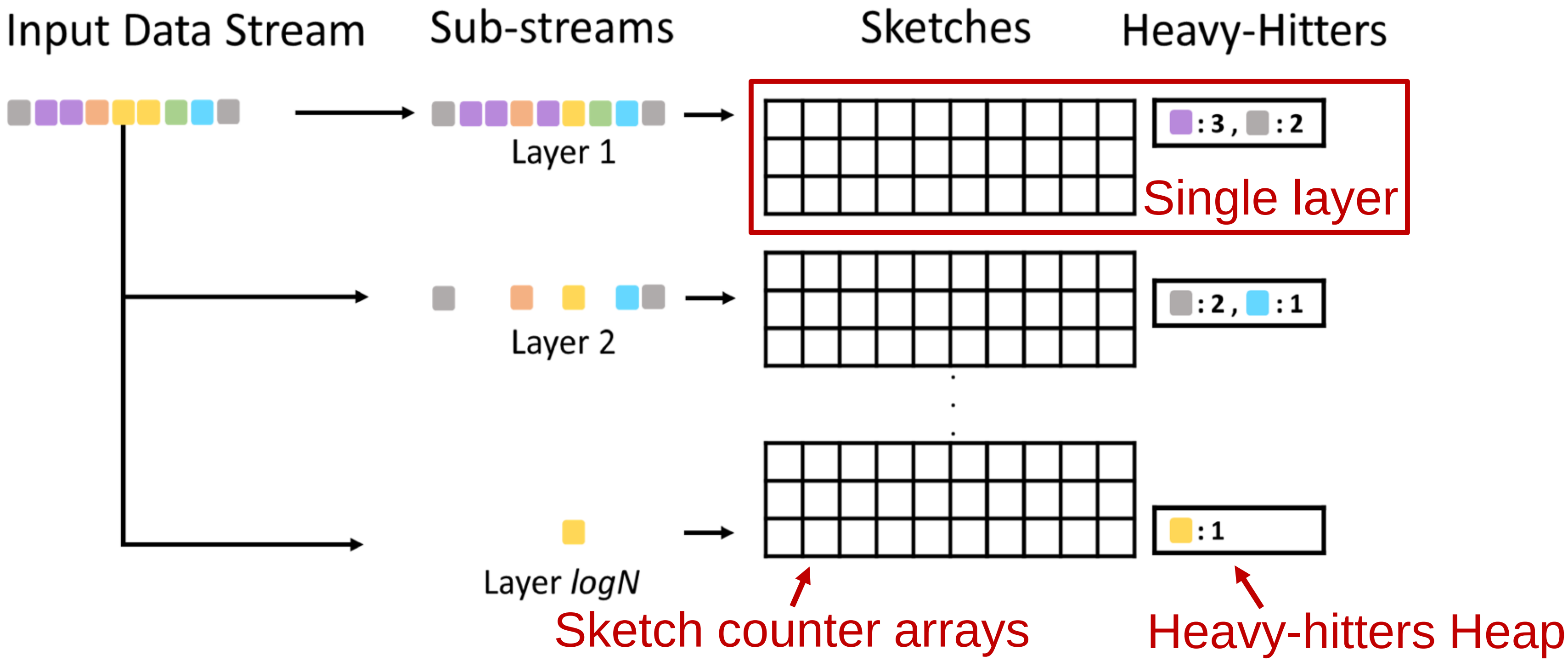
Estimation tasks in solar sensor	Statistics supported by universal sketching
Energy generated	L1-norm
Power outages	Zero-draw time
Weather events	Change
Anomaly detection	Entropy
Voltage volatility	L2-norm
…...	…...

Table: Statistics relevant to solar farm

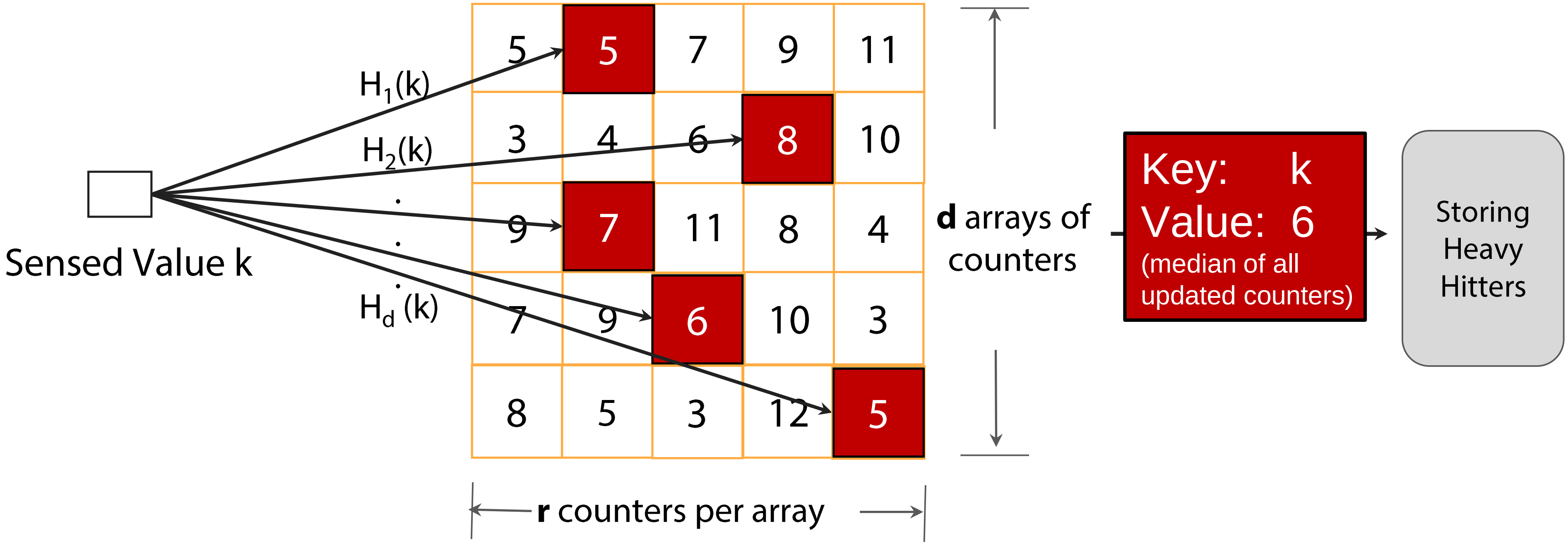
Data structure in universal sketching algorithm



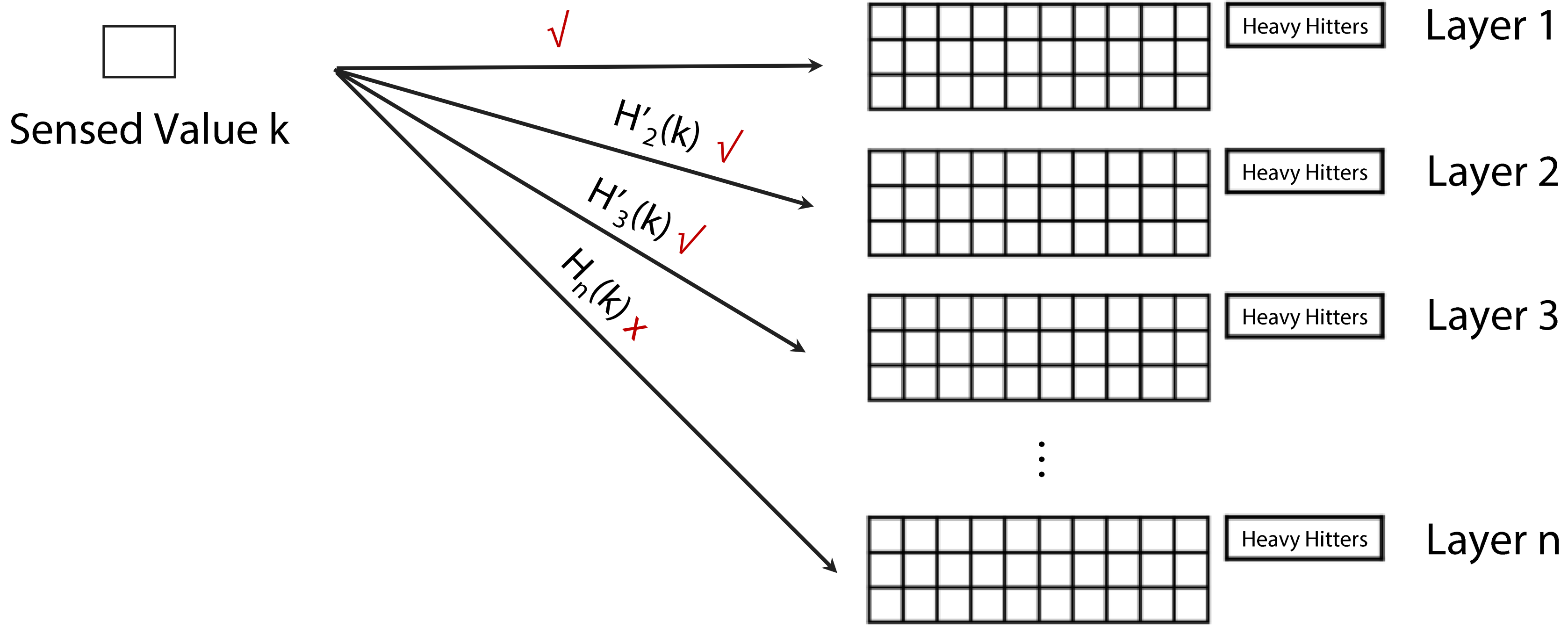
Data structure in universal sketching algorithm



How sensed value updates a single layer



How sensed value updates the entire structure

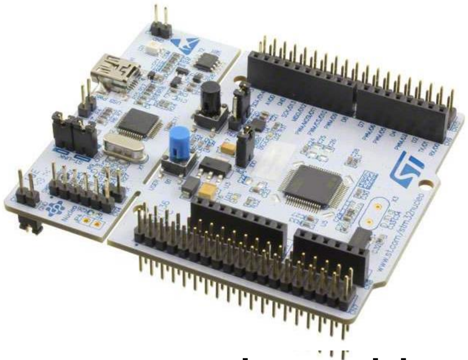


Joltik system challenges

- Challenge 1: Reduce memory footprint of universal sketches

Native Universal Sketching ← Hundreds of KBs memory

e.g., 300 KB to achieve 95% accuracy on a real-world dataset

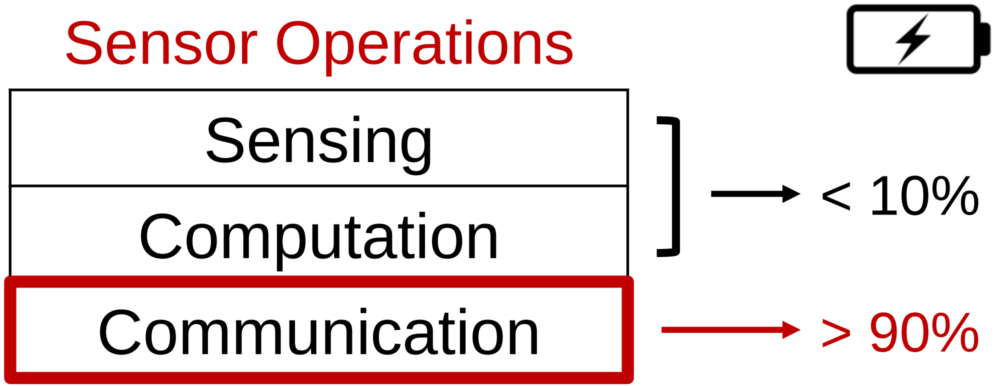


← Limited on-chip memory

e.g., the sensor board in our testbed only has 128 KB memory

Joltik system challenges

- Challenge 2: Reduce communication footprint



Joltik system challenges

- Challenge 3: Reduce computation overhead of sketch update

Universal sketch operations

Hash computations
Arithmetic calculations
Counter updates
Heap updates
……

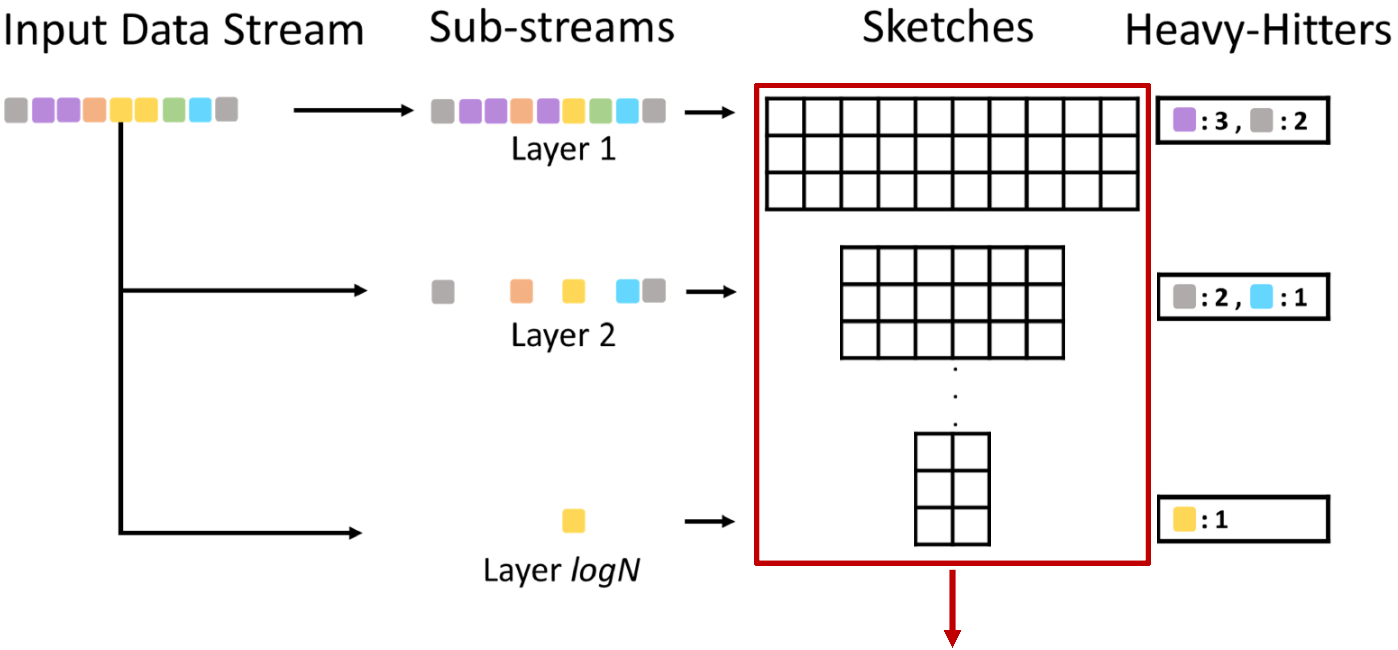
Outline for this talk

- Motivation
- System workflow and challenges
- System design
  - Low memory footprint - reducing memory footprint
  - Low power - reducing communication footprint
  - Low CPU - reducing computation overhead
- Implementation and evaluation
- Conclusions

Outline for this talk

- Motivation
- System workflow and challenges
- System design
  - Low memory footprint - reducing memory footprint
  - Low power - reducing communication footprint
  - Low CPU - reducing computation overhead
- Implementation and evaluation
- Conclusions

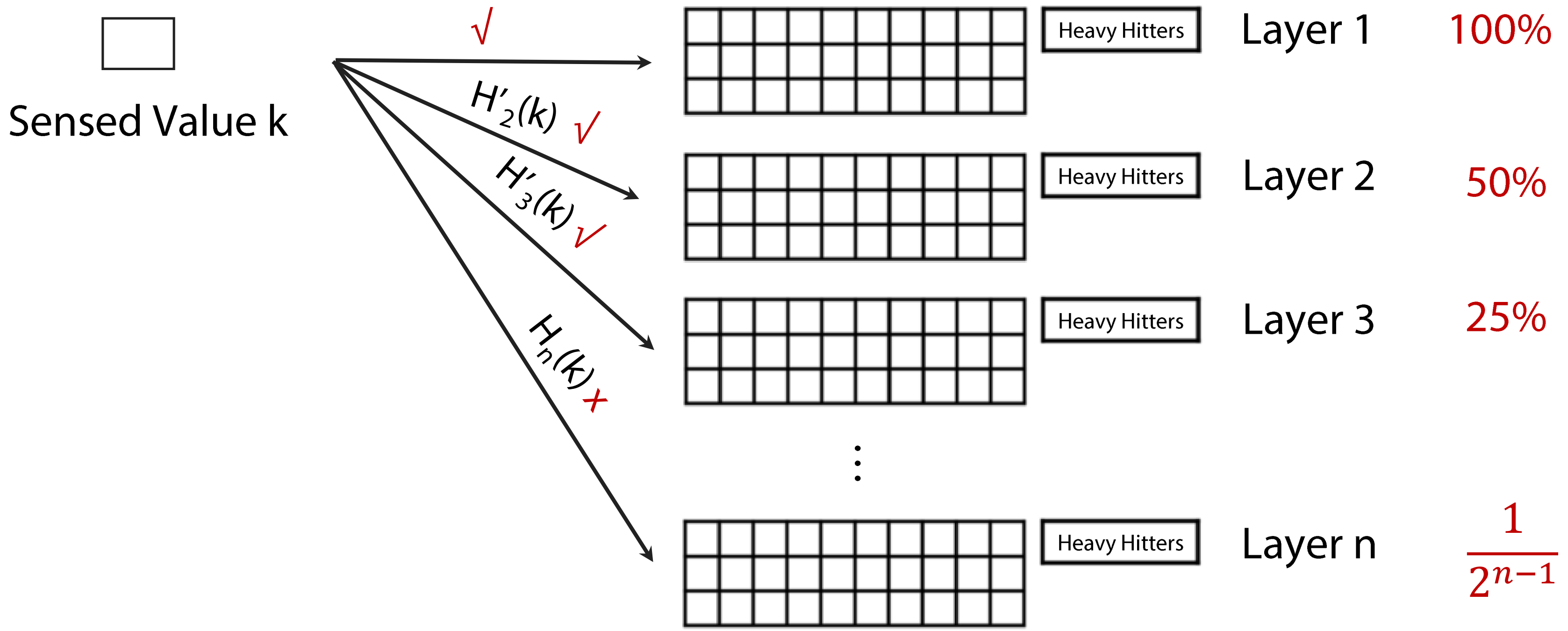
Reducing memory footprint: “Inverted Pyramid”



Gradually reduce
number of columns

Insight of "Inverted Pyramid" structure

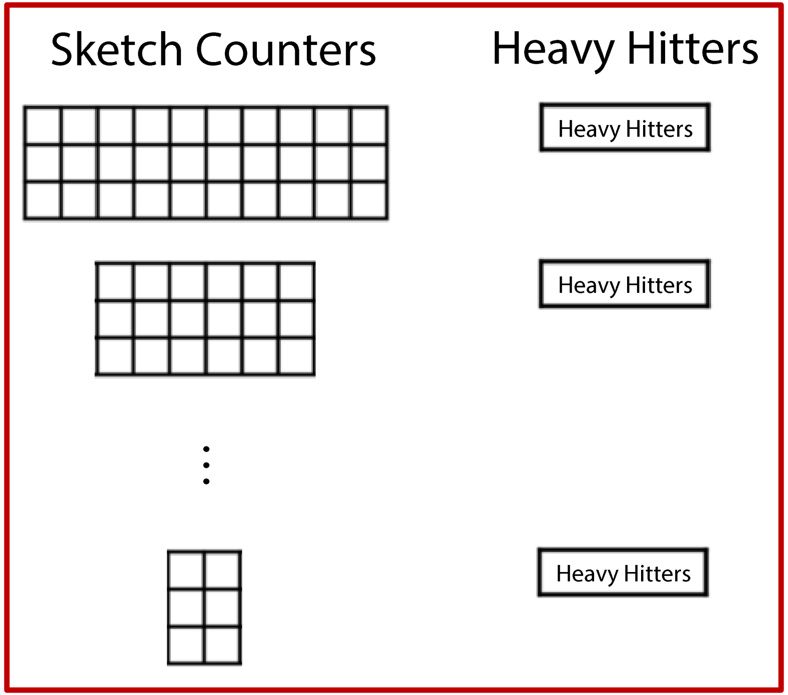
- Lower layers need to handle much smaller number of samples



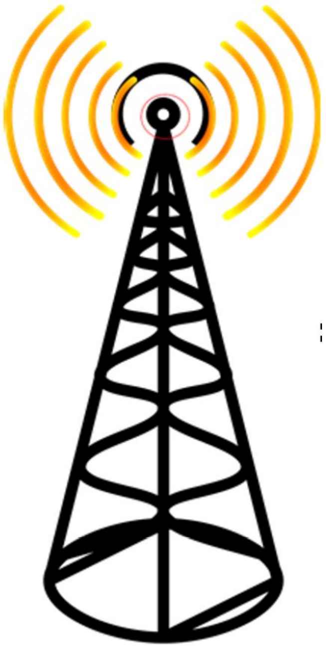
Outline for this talk

- Motivation
- System workflow and challenges
- System design
  - Low memory footprint - reducing memory footprint
  - Low power - reducing communication footprint
  - Low CPU - reducing computation overhead
- Implementation and evaluation
- Conclusions

Reducing communication footprint



Data packet

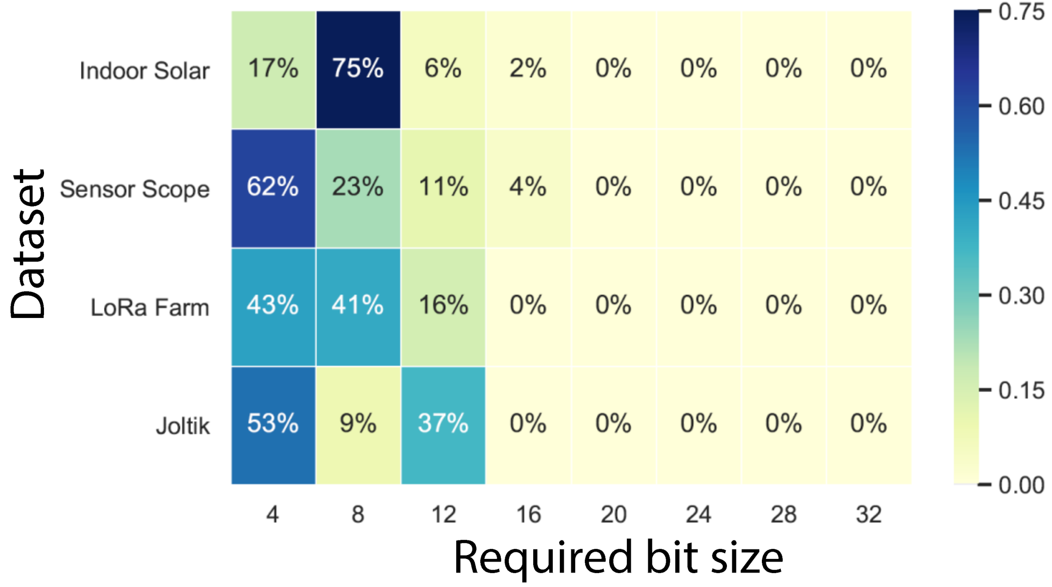


Observation on sketch counters

- Only small part of counters have large values

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

A sketch counter example

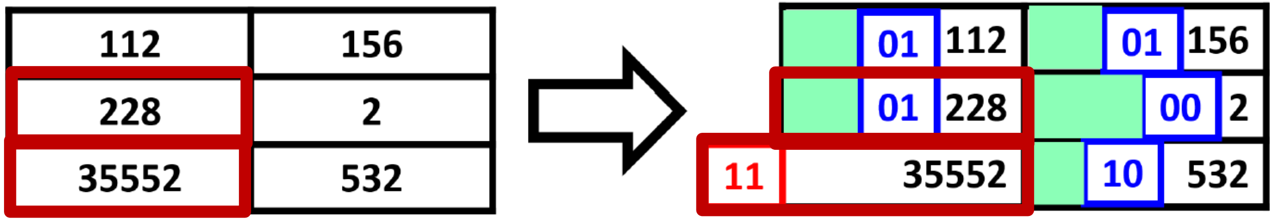


Lossless encoding of sketch counters

Required Bitsize	Prefix	Bitsize before/after compress
4	00	16 / 6
8	01	16 / 10
12	10	16 / 14
16	11	16 / 18

Example:

Rare case!



All counters are originally 16 bits before encoding

Efficiently transmitting heavy-hitters heap

Transmit ←

Key	Value
910	35552
804	34409
983	29374
905	19395
827	18890
…	…

→ Reconstruct at base station

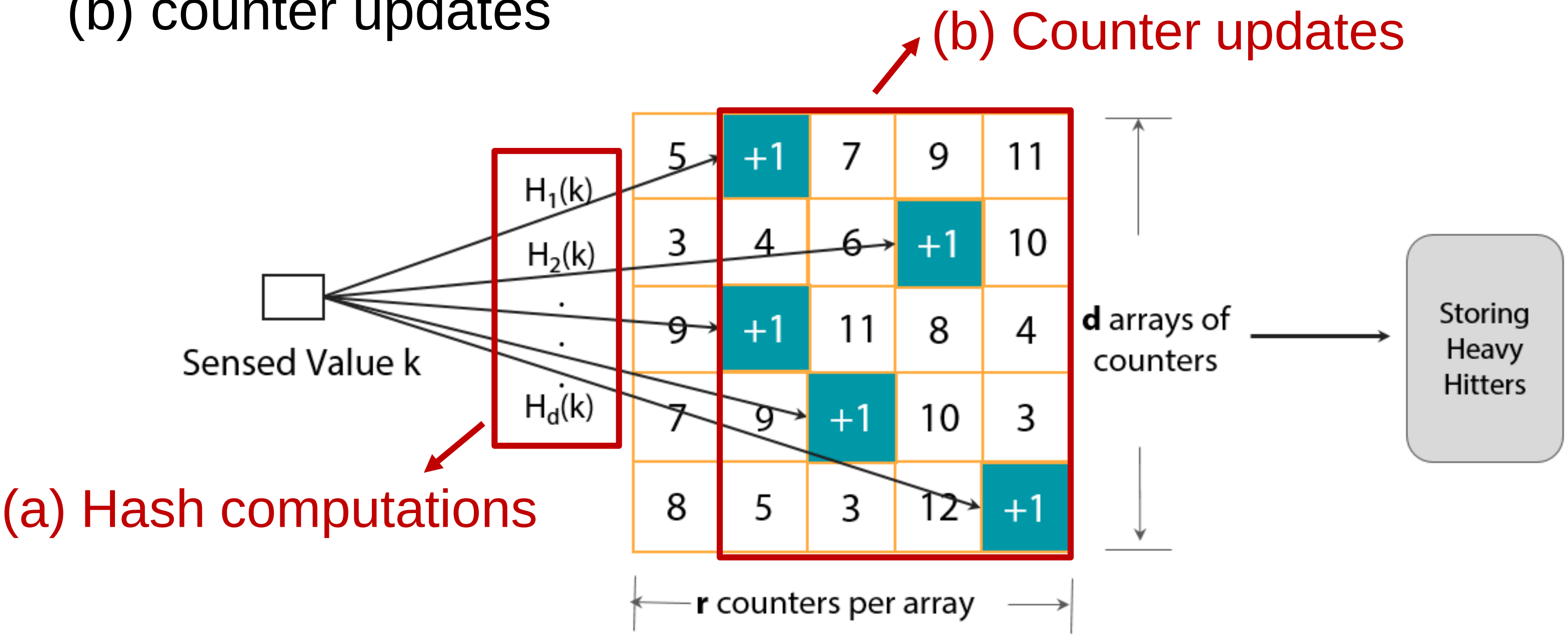
A heavy-hitter heap example

Outline for this talk

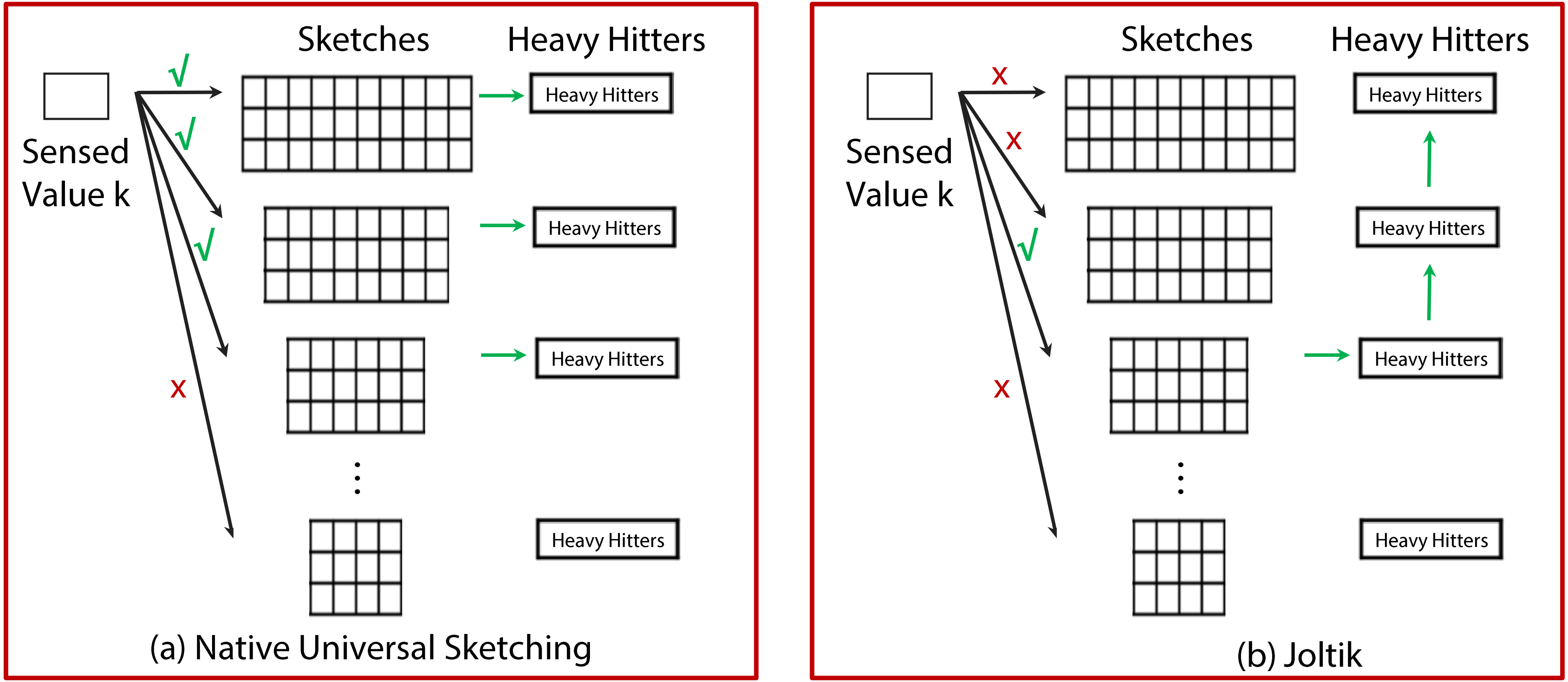
- Motivation
- System workflow and challenges
- System design
  - Low memory footprint - reducing memory footprint
  - Low power - reducing communication footprint
  - Low CPU - reducing computation overhead
- Implementation and evaluation
- Conclusions

Bottleneck analysis

- Top two CPU performance bottlenecks:
  (a) hash computations
  (b) counter updates



Reduced sketch update



Intuition: less samples in lower layers → less collisions → more accurate result

End-to-end deployment

- User input:
  (a) Data collection rate - *Rcl*
  (b) Period of transmission - *T*
- Parameter tuned by Joltik:
  (a) Sketch Size - *S*

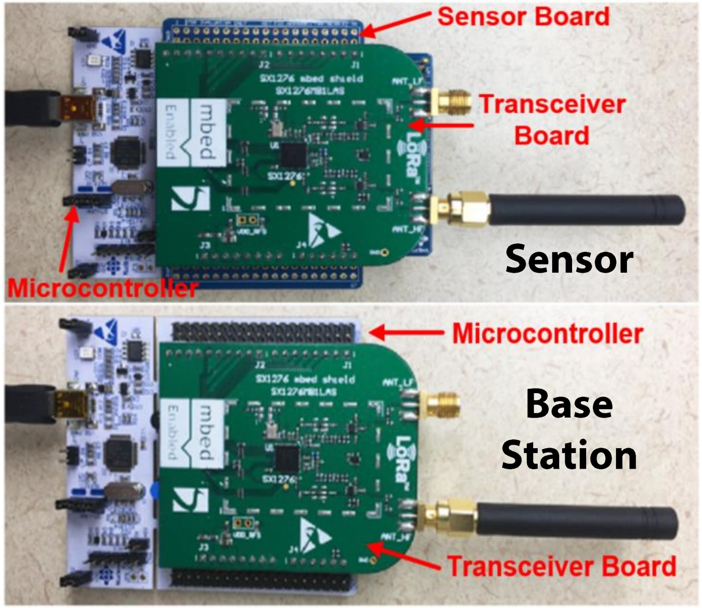
Rcl	***T***	***S***	**Lifetime**	**Accuracy**
10 Hz	1 day	30 KB	2751 days	94.30%±3.27%
10 Hz	1 day	60 KB	1613 days	96.61%±2.25%
10 Hz	1 day	90 KB	1141 days	97.50%±1.92%

Table: Joltik deployment example

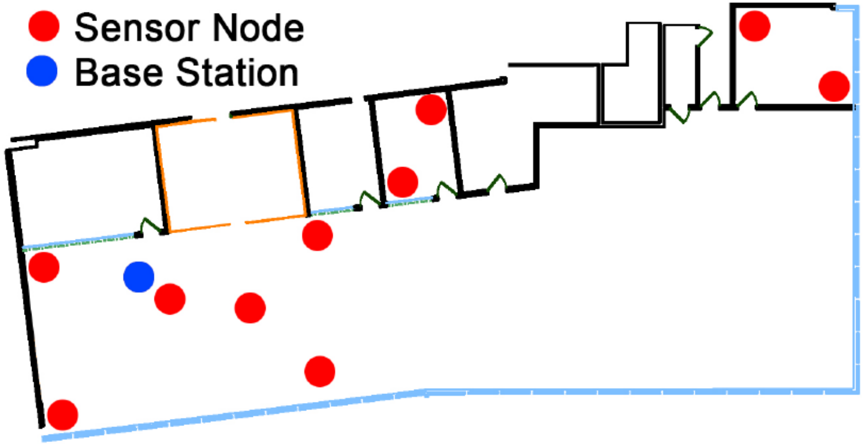
Outline for this talk

- Motivation
- System workflow and challenges
- System design
  - Low memory footprint - reducing memory footprint
  - Low power - reducing communication footprint
  - Low CPU - reducing computation overhead
- Implementation and evaluation
- Conclusions

Joltik implementation and evaluation setup

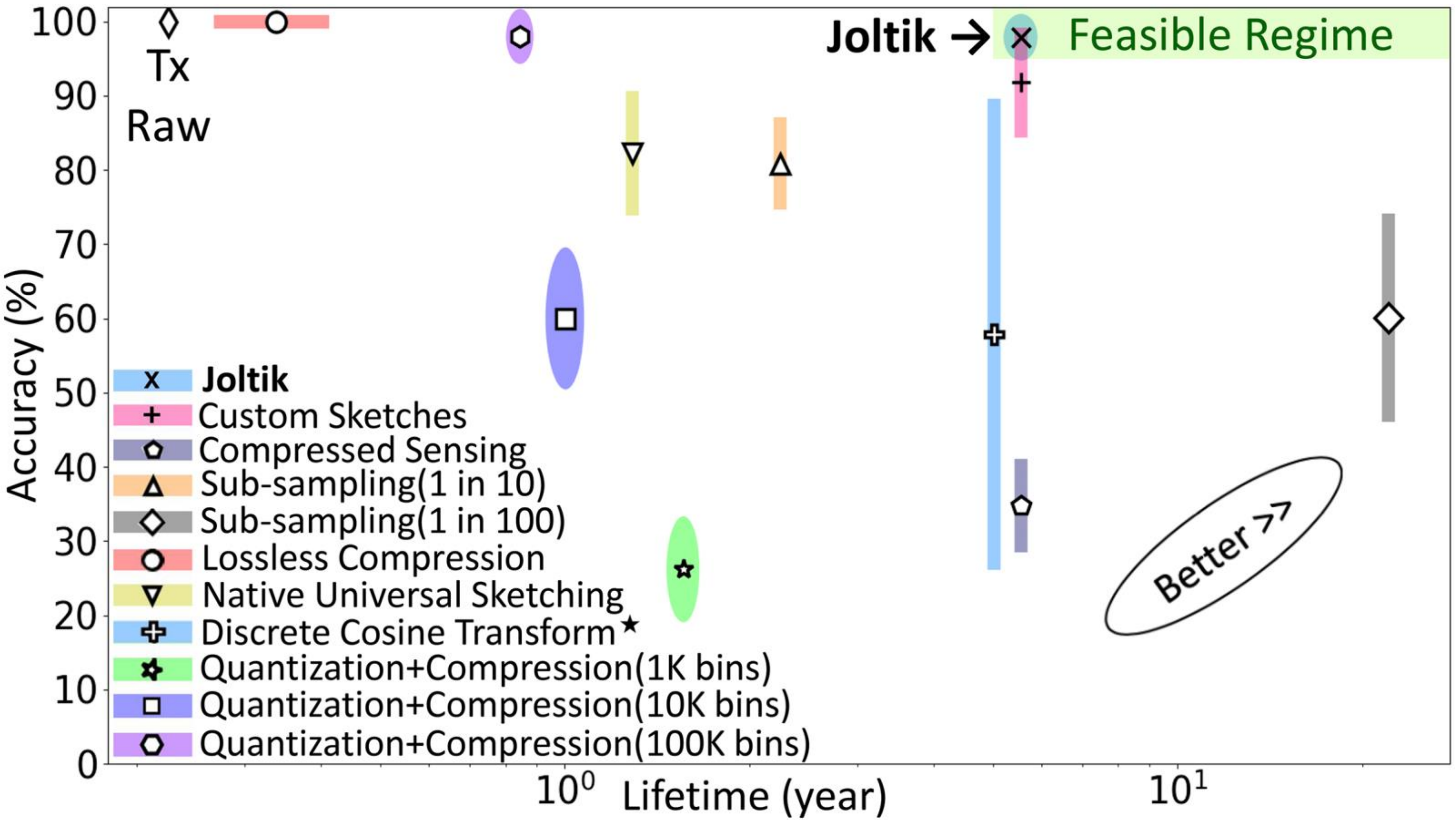


- Microcontroller: NUCLEO-L476RG
- Sensor board: X-NUCLEO-IKS01A2
- RF Frontend: SX1276 LoRa Transceiver

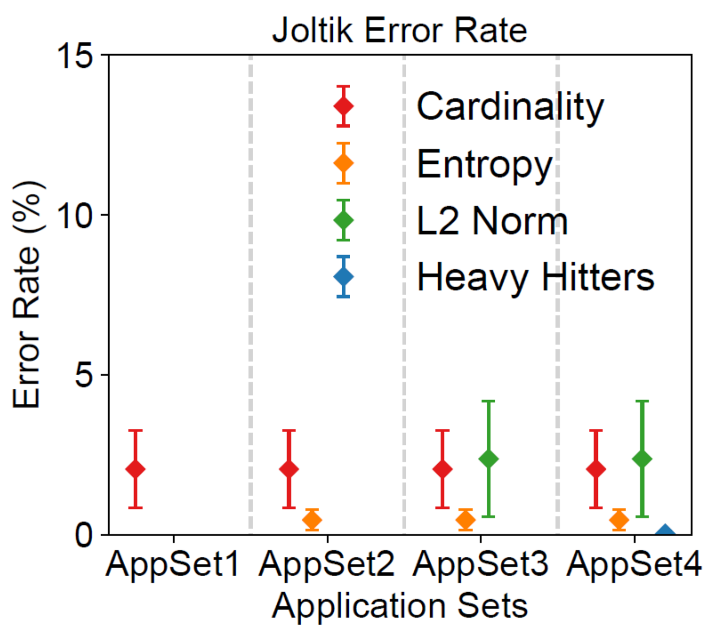


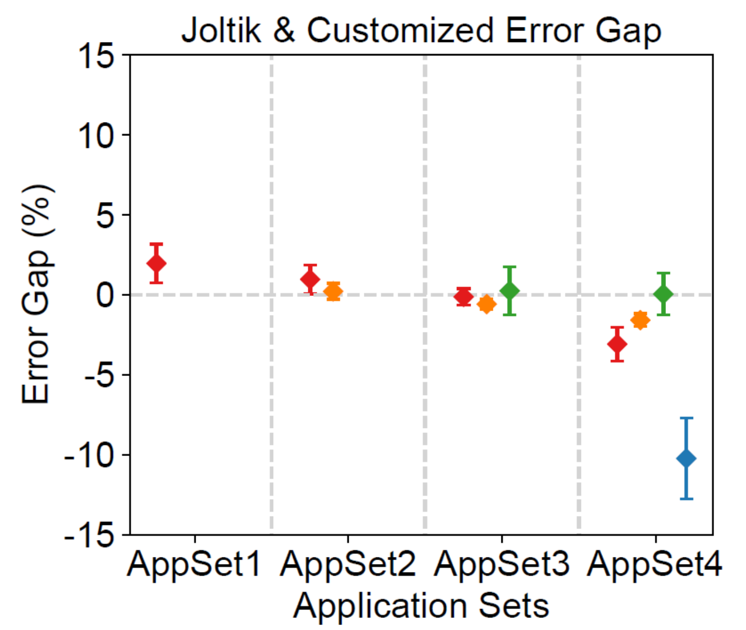
Real world testbed in a campus building at CMU

Joltik provides better energy-accuracy trade-off for "future-proof" analytics



Joltik supports multi-task handling and provides generality





(Positive values imply Joltik is worse and vice versa)

AppSet1 = {Cardinality}
AppSet2 = {Cardinality, Entropy}

AppSet3 = {Cardinality, Entropy, L2}
AppSet4 = {Cardinality, Entropy, L2, HH}

Summary

- Goal
  - Build a general, accurate, and energy-efficient sensor analytics framework
- Our approach
  - Propose a novel architecture by leveraging universal sketching
  - Low memory footprint, low power and low CPU realization of universal sketching
- Joltik: Enabling energy-efficient “future-proof” analytics on LPWAN
  - Support general and “future-proof” analytics
  - Guarantee >5-year sensor battery life and <5% error rate on a range of statistics

https://github.com/Joltik-project/Joltik

Mingran Yang mingrany@andrew.cmu.edu