Data Compression for Industrial IoT

A Survey of Algorithms and Techniques

The Industrial IoT Data Challenge

Volume

Massive scale. Estimates project 175 zettabytes of data by 2025, much of it from IIoT sensors.

Velocity

Data is generated in continuous, high-speed, real-time streams that must be processed immediately.

Constraints

IIoT devices are often low-power and operate on networks with limited bandwidth (e.g., LPWAN).

Why is Compression Critical in IIoT?

Reduce Network Load

Enables the transmission of massive sensor data volumes over constrained networks. This is vital for remote monitoring and real-time control, where bandwidth is a premium.

Lower Costs & Energy

Decreases storage costs on both edge and cloud platforms. Critically, it reduces data transmission time, which is the most energy-intensive operation for a battery-powered IIoT device.

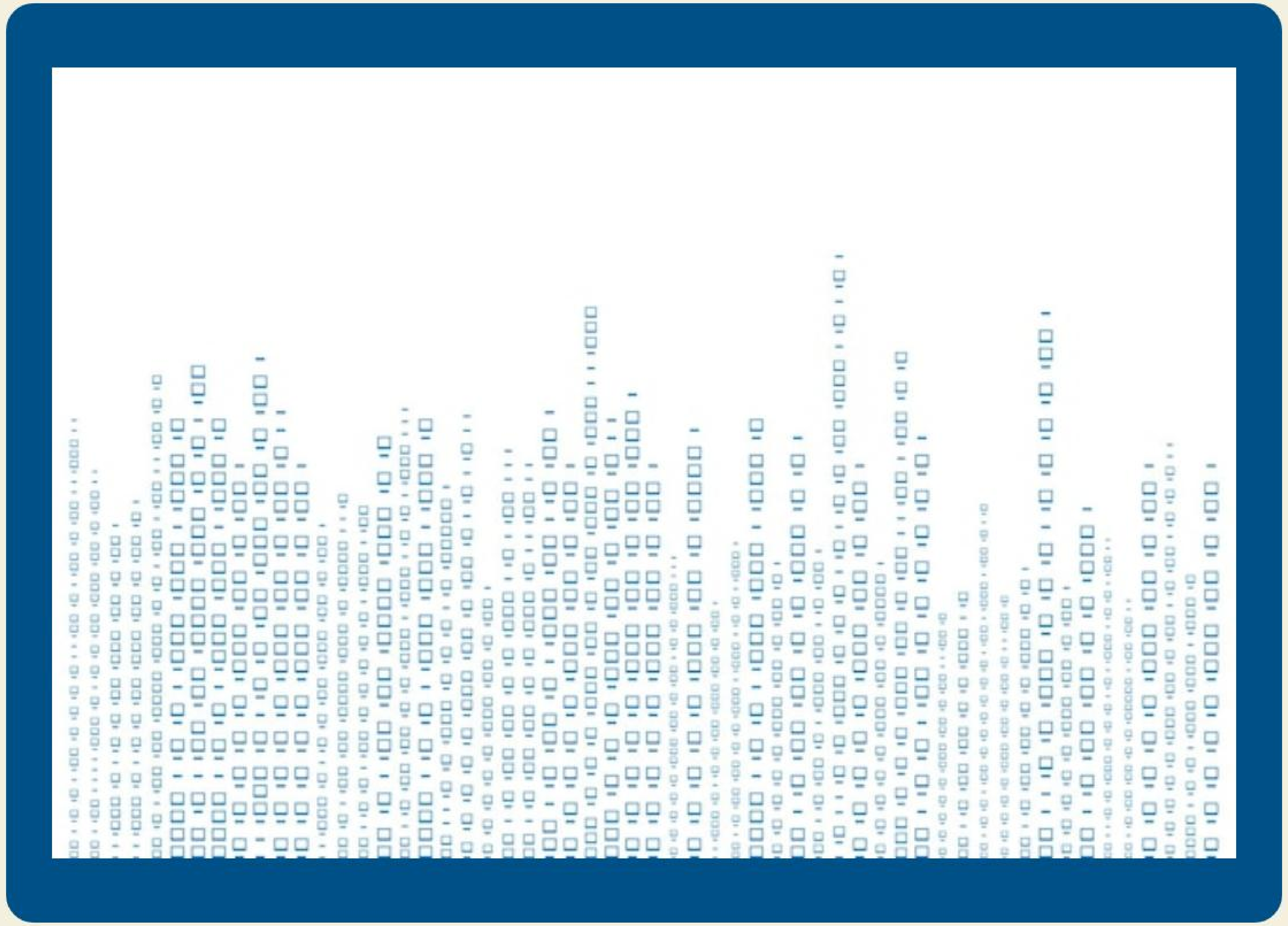
A Fundamental Choice: Lossless vs. Lossy

Lossless Compression

Perfect reconstruction. No data is lost.

This approach is essential when 100% data accuracy is non-negotiable.

- **Use Case:** Critical industrial process data, sensor readings for compliance, and executable files.
- **How it works:** Identifies and removes statistical redundancy in data.
- **Examples:** Huffman Coding, LZW, Run-Length Encoding (RLE).



Lossy Compression

Trades accuracy for higher compression.

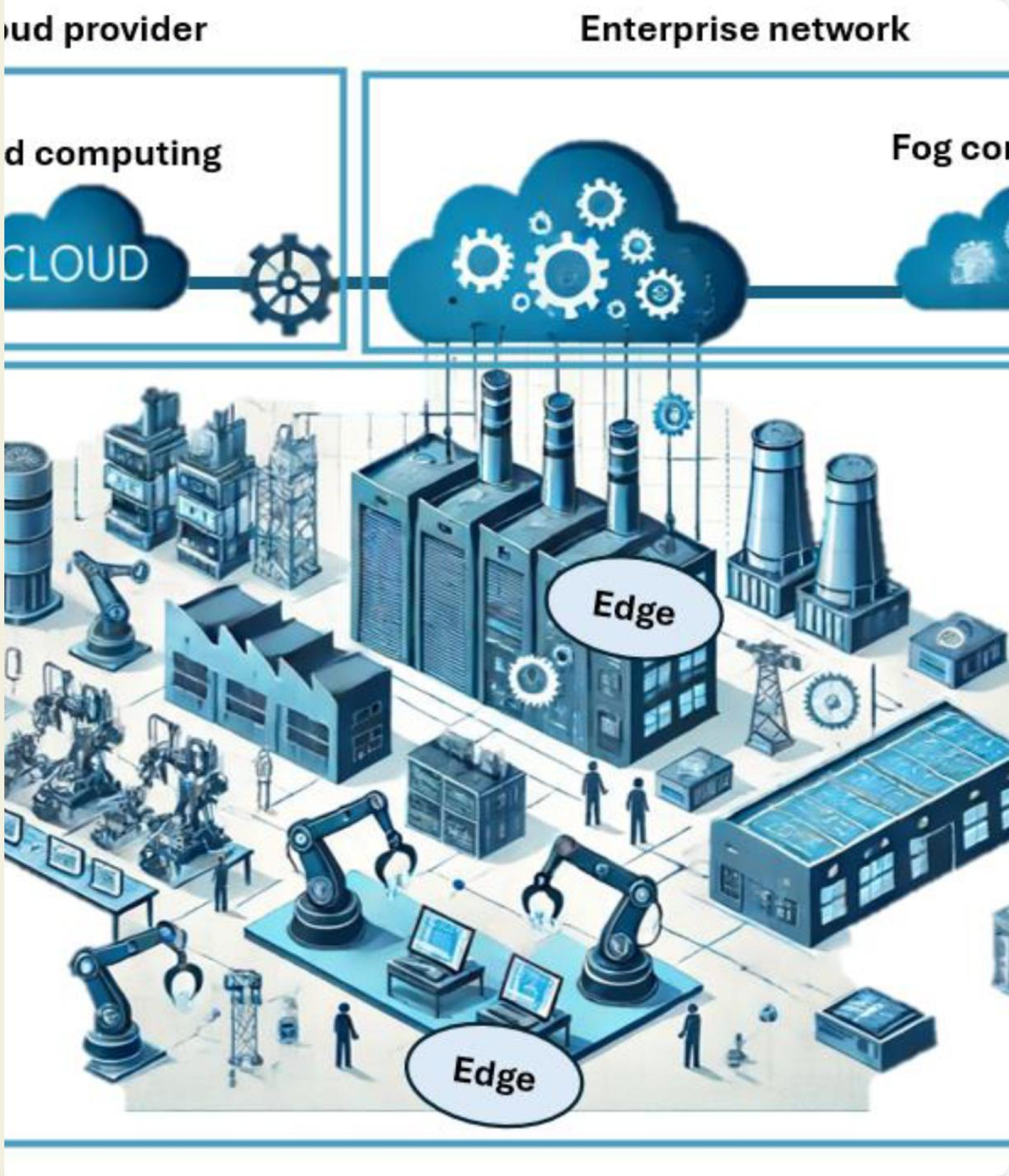
This method achieves much smaller file sizes by permanently removing "unnecessary" or less perceptible information.

Use Case:

Video feeds (security, monitoring), audio, and some sensor data (e.g., vibration) where minor inaccuracies are acceptable.

Examples:

JPEG (for images), MP3 (for audio), and specialized time-series algorithms like autoencoders.



Lossless Spotlight: Huffman Coding

- **Technique:** An entropy-based, variable-length encoding algorithm.
- **How it Works:** It assigns short binary codes to the most frequently occurring data symbols and longer codes to infrequent symbols.
- **IIoT Challenge:** Traditional Huffman requires building a frequency tree, which can be memory-intensive for low-power edge devices.
- **Adaptation:** "Adaptive Huffman" or "Huffman Deep Compression (HDC)" use sliding windows to manage memory and adapt to changing data streams.

Lossless Spotlight: LZW

> "LZW is a dictionary-based algorithm that builds a 'dictionary' of string patterns and replaces them with a single code."

— Used in GIF and TIFF. Effective for repetitive data, common in IIoT sensor logs. —

How Are Algorithms Compared?

For IIoT, it's a trade-off. A high Compression Ratio (blue) is great, but low CPU, Memory, and Energy use (green) are often more critical for edge devices.

Emerging Trends in IIoT Compression

TinyML / AI

Using deep learning (e.g., autoencoders) on edge devices to "learn" data patterns for highly efficient, context-aware compression.

Hybrid Models

Combining lossy pre-processing (like quantization) with a lightweight lossless algorithm (like LZW) to get the best of both worlds.

Fog Architectures

Using multi-layered compression: simple RLE at the sensor, more complex Zstd at the fog, and full archival at the cloud.

Case Study: Zstandard (Zstd)

2.5x

Faster than LZW

A Modern Standard

Zstandard is a modern algorithm offering high compression ratios (comparable to `gzip -9`) but with significantly faster compression and decompression speeds. It's becoming a standard for real-time, high-throughput data streams in modern systems.

Thank You

Questions?

Image Sources

https://img.freepik.com/premium-vector/binary-code-abstract-matrix-white-background-with-binary-computer-code_833685-3292.jpg

Source: www.freepik.com

https://pub.mdpi-res.com/sensors/sensors-24-04239/article_deploy/html/images/sensors-24-04239-g001.png?1720407339

Source: www.mdpi.com