

Article

Structure-Aware Lossless Compression for Cell-Level BESS Monitoring Data

Tianle Liu ^{1,*} , Tengjiao Lu ¹, Changlin Yang ^{2,*}, Peng Pan ¹, Chen Li ³ and Tengjiao He ⁴

¹ School of Communication Engineering, Hangzhou Dianzi University, Hangzhou 310018, China; 242080153@hdu.edu.cn (T.L.); panpeng@hdu.edu.cn (P.P.)

² Energy AI Department, Zhejiang Tianneng Energy Storage Technology Development Co., Ltd., Huzhou 313100, China

³ School of Engineering, Cardiff University, Cardiff CF24 3AA, UK; chenlivien@gmail.com

⁴ College of Information Science and Technology, Jinan University, Guangzhou 510632, China; htj2018@jnu.edu.cn

* Correspondence: tianle@hdu.edu.cn (T.L.); yangcl02@tn-ess.com (C.Y.)

Abstract

Large-scale battery energy storage systems (BESSs) continuously generate cell-level monitoring data. These data increase storage demand and communication load in cloud-edge systems. Exact measurement values are required for archival and diagnostic tasks, which motivates lossless compression. This paper releases an industrial cell-level dataset collected from a 215 kWh BESS and analyzes its compression-relevant properties. The data exhibit finite decimal precision, strong temporal dependence, spatial similarity, and different value ranges across measurement variables. Based on these properties, we propose a structure-aware lossless compression method. The method groups the data by measurement variable and reversibly scales the values to integers. It applies temporal and spatial differencing, followed by signed-value remapping, bit-width packing, and final lossless coding. Experiments on AMD and Raspberry Pi platforms evaluate global block-based compression using all 59,452 records and real-time compression. The proposed method achieves compression ratio of 0.779% in the global block-based setting, while its real-time ratio is 1.801%. The real-time configuration requires 0.198 ms per time record on AMD and 1.252 ms on Raspberry Pi.

Keywords: battery energy storage system; lossless compression; industrial BESS dataset

1. Introduction

Recent advances in battery technologies have accelerated the deployment of electrochemical energy storage in modern power systems [1]. Battery energy storage systems (BESSs), in particular, are increasingly deployed to facilitate renewable energy integration, enhance grid flexibility, and maintain a stable power supply [2–4]. They provide services such as peak shaving, load shifting, and frequency regulation. These services improve power-system stability and operational efficiency [5,6]. Reliable BESS operation depends on accurate monitoring. Typical measurements include cell voltage, current, temperature, state of charge (SoC), state of health (SoH), and other battery-state-related variables [7,8]. Recent BESS digital-twin studies also show that operational battery data can support SoC forecasting and secure operation in frequency-regulation applications [9]. Safe operation of large-scale BESSs therefore requires dense sensing infrastructures to support supervision, battery management, and operational decision-making [10].



Academic Editor: Yong-Joon Park

Received: 17 August 2026

Revised: 17 September 2026

Accepted: 21 September 2026

Published: 4 October 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article

distributed under the terms and

conditions of the [Creative Commons](#)

[Attribution \(CC BY\) license](#).

BESS monitoring data support many data-driven applications, including state estimation, health assessment, anomaly detection, and performance prediction [11,12]. In cloud-edge BESS monitoring architectures, edge-side battery management systems (BMSs) collect high-frequency monitoring data from many cells and sensors, while cloud platforms provide storage, analytics, and model training [9,13,14]. This deployment produces high-dimensional multivariate time-series data with strong temporal and spatial correlations. The resulting data volume increases storage demand and transmission delay, and places pressure on bandwidth-limited links [14–16]. At the same time, safety-critical BESS applications require exact reconstruction of monitoring data. Cell-level voltage is closely related to SoC, SoH, and early fault diagnosis, including incipient internal short-circuit detection [17]. Even small reconstruction errors may affect health monitoring and fault diagnosis.

Since safety-critical BESS monitoring requires exact reconstruction, lossless compression is the most relevant class of techniques. Existing lossless methods have been widely studied for time-series monitoring data, e.g., smart-grid, IoT, and multi-sensor data [18–20]. Typical approaches include differencing, run-length coding, dictionary coding, entropy coding, and their hybrid designs. Many differencing-based lossless time-series compression methods exploit temporal redundancy by encoding changes between adjacent samples [21,22]. Related work further uses sparse, multiplier-free difference sequences to design lightweight transforms for time-series compression on resource-constrained IoT devices [23]. For slowly varying signals, repeated values and short repeated sequences can also be exploited by run-length coding and dictionary coding, e.g., LZ77 [24]. Entropy coding such as Huffman coding and related hybrid schemes further reduce the average code length [25,26]. These methods provide general-purpose tools for reducing temporal, repeated-pattern, and symbol-level redundancy. Recent studies further show that compression for sensing and edge systems should consider data organization and deployment constraints. Domain-specific energy-data compression can improve compression efficiency by exploiting repeated patterns in energy time-series data, although some methods are nearly lossless rather than strictly lossless [27]. Edge-side compression must also balance data reduction, computation cost, and network bandwidth [28]. For multi-sensor systems, sensor measurements should be jointly considered instead of treating each sensor stream in isolation [29]. Generalized deduplication can further exploit similarity among small sensor records for lossless compression [30]. However, they do not explicitly exploit the structured organization of BESS monitoring data, where measurements are arranged by the time record, cells, and measurement variables. Battery-specific data reduction methods have also been investigated for massive battery-pack data. For example, frequency-division reconstruction and recording-frequency optimization have been used to reduce the storage cost of battery voltage data [31,32]. These methods can exploit battery signal correlations or reduce the recording frequency to lower storage cost. However, they reconstruct or record approximations of the original measurements. Therefore, existing studies either preserve exact reconstruction without fully using BESS-specific data structures, or exploit battery-specific characteristics at the cost of reconstruction fidelity. Table 1 compares the related methods with the proposed method in details.

Therefore, the key challenge is not merely applying a general compressor to BESS data. Instead, it is to develop a data-characteristic-driven lossless compression framework that characterizes industrial BESS monitoring data and translates its structural and numerical properties into reversible designs suitable for cloud-edge deployment. In this study, we propose a structure-aware lossless compression method for cell-level BESS monitoring data. The proposed method targets large-scale cloud-edge BESSs. It jointly exploits spatiotemporal correlations across cells and measurement variables. It also uses the finite decimal precision and variable-dependent value ranges of finite-precision battery measurements.

The framework includes precision-preserving integerization, multi-dimensional differencing, signed-value remapping, compact bit-width packing, and final lossless coding (FLC). It reduces storage and transmission overhead. It also reconstructs the original monitoring table exactly. The evaluation is conducted on a publicly available operational BESS dataset [33] released in this paper. The main contributions are summarized as follows.

- **Public industrial BESS dataset and characterization:** We release a public cell-level monitoring dataset collected from a 215 kWh industrial BESS [33]. Existing public battery datasets are mainly designed for cell aging, health estimation, electric vehicle (EV) charging analysis, or general energy metering. They do not fully represent BESS monitoring, where many cells and multiple measurement variables are recorded together in a BMS format. The released dataset contains operational measurements from 240 cells. We characterize the dataset from a compression perspective. The measurements have fixed decimal places and can be reversibly scaled into integers. The data also show temporal dependency and spatial similarity, which support lossless compression.
- **Structure-aware lossless compression:** We then design of a reversible framework based on these characteristics. It combines precision-preserving integerization with temporal and spatial differencing. Signed-value remapping, bit-width packing, and FLC are then applied. The required side information is retained to reconstruct the measurement values and missing-value indicators exactly.
- **Evaluation across computing platforms:** We use an AMD platform to represent cloud-side processing and a Raspberry Pi platform to represent edge-side processing. The evaluation covers global block-based compression, 60 min and 5 min block-based compression, and real-time compression. We report compression ratios for the transmission and storage models, compression time, decompression time, and peak memory consumption.

The remainder of this paper is organized as follows. Section 2 analyzes the compression-relevant characteristics of the BESS monitoring dataset. Section 3 presents the proposed lossless compression and reconstruction framework. Section 4 reports the experimental results. Section 5 concludes this paper.

Table 1. Comparison of the proposed method with related data compression methods.

Method	Designed for Battery Data	Lossless	Temporal Processing	Spatial Processing	Bit-Width Packing	Real-Time Compression	Cell-Level BESS Data
SPDFlow [20]	No	Yes	Real time-series model	N/A	N/A	Yes	No
Powerstrip [27]	No	No	Delta	No	Yes	N/A	No
Giorgi [29]	No	Yes	Differencing	N/A	N/A	N/A	No
Zhou et al. [31]	Yes	No	Sampling	Yes	N/A	N/A	No
Zheng et al. [32]	Yes	No	Sampling	N/A	N/A	N/A	No
Proposed method	Yes	Yes	Temporal differencing	Spatial differencing	Yes	Yes	Yes

Note: N/A means the item is not available in the study.

2. Compression-Relevant Characteristics of the BESS Dataset

This section characterizes the publicly released industrial cell-level BESS monitoring dataset in this paper, which was collected from a 215 kWh system on 4 March 2025. The dataset contains 59,452 records from 240 battery cells. Its average sampling intervals are every 1.489 s. Because the data come from one industrial deployment, the following analysis is limited to the compression-relevant properties observed in this deployment.

Figure 1 shows selected measurements, Table 2 summarizes the dataset characteristics, and the full dataset is publicly available in [33]. In the following discussion, cell IDs refer to the ID recorded for each battery cell in the dataset. A measurement column name specifies the measurement variable, e.g., voltage. Column order refers to the sequence of columns in the dataset. For each measurement variable, the columns follow the dataset order from 1 to 240. This order does not imply physical adjacency.

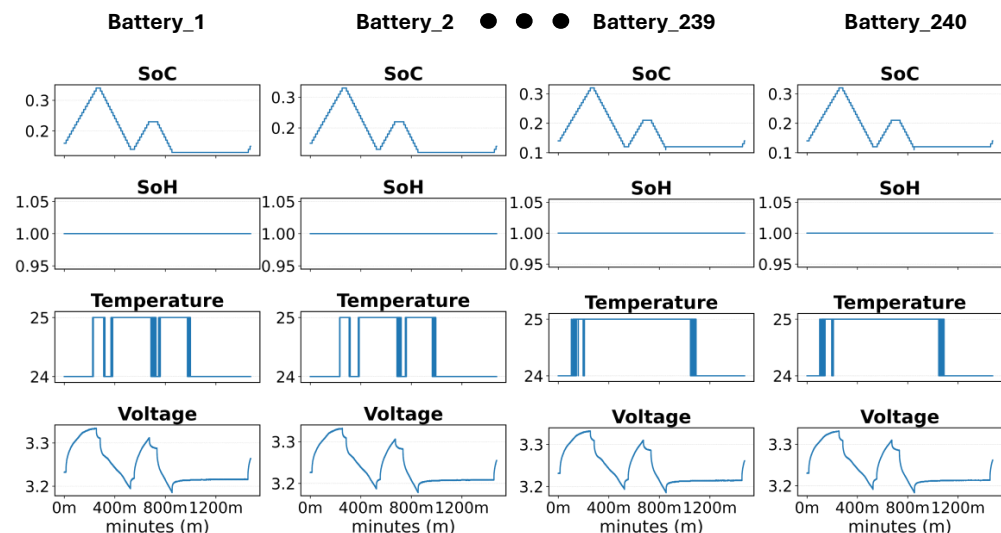


Figure 1. Recorded cell-level BESS monitoring data for four selected cells. The columns correspond to cells 1, 2, 239, and 240, and the rows correspond to SoC, SoH, temperature, and voltage. The horizontal axis shows elapsed time in minutes. The vertical axes show SoC and SoH as dimensionless ratios, temperature in °C, and voltage in V.

Table 2. Characteristics of the investigated 215 kWh BESS dataset.

Dataset Characteristic	Value
Number of Records	59,452
Number of Cells	240
Average Sampling Interval (s)	1.489
SoC Range	0.11–0.34
SoH Range	1.00–1.00
Temperature Range (°C)	24.0–26.0
Cell-Voltage Range (V)	3.177–3.335
Missing-Data Ratio (%)	0.308

2.1. Finite-Precision Measurements

The studied BESS sensor network records finite-precision measurements, including voltage, temperature, SoC, and SoH. Storing these values in floating-point format generally requires more bits than integer formats such as Int16. Since the recorded variables have finite decimal precision, they can be reversibly converted into integers using variable-wise scaling factors without losing the original recorded values. This integerization differs from lossy quantization: the scaling factors are matched to the recorded precision, so the original values can be exactly recovered. In addition, voltage, temperature, SoC, and SoH exhibit different value ranges and temporal variation patterns. The monitoring table should therefore be processed by measurement variable rather than as an unstructured byte stream. Such a variable-wise organization enables reversible integerization and spatial differencing among measurements of the same variable type. We next analyze the data characteristics of each monitored variable.

2.2. Voltage

In the spatial dimension, the cell-voltage values range from 3.177 V to 3.335 V in the investigated dataset. In the temporal dimension, the voltage of a single cell changes slowly. For instance, the voltage of cell 1 changes from 3.333 V to 3.334 V during the four-minute interval from 12:55:53 to 12:59:58.

2.3. Temperature

Battery temperature is a critical indicator for safety monitoring in energy storage systems. In practical deployments, temperature sensors are typically installed on only a subset of cells. The recorded data in [33] show that temperature varies slowly over time.

2.4. SoC

SoC reflects the remaining energy stored in a battery cell. Under the same current condition, the cell voltage generally increases or decreases with the increase or decrease in SoC over time. Moreover, the SoC values of different cells tend to evolve in similar increasing or decreasing trends. These properties indicate that SoC data contain both temporal and spatial redundancy.

2.5. SoH

SoH is typically used to describe the ratio of the current battery capacity or performance to its initial factory condition. As a result, SoH usually remains nearly constant over short sampling intervals. The dataset shows that, for a given cell, SoH often remains unchanged across the observed time records, and consecutive columns in database order frequently contain identical SoH values. This suggests that SoH data contain small temporal variation and repeated values in both the temporal and spatial dimensions.

2.6. Analysis

Overall, the released dataset exhibits several properties that are relevant to lossless compression. First, the measurements have finite decimal precision and can therefore be converted into integers using variable-wise scaling without information loss. Second, different measurement variables exhibit distinct value ranges and temporal variation characteristics, which motivates organizing the monitoring table by variable type rather than treating it as an unstructured byte stream. Third, most variables vary slowly over time, and consecutive columns in database order often have similar values. As a result, temporal, spatial, and temporal-spatial differencing usually produce residuals much smaller than the original scaled integers. However, similar values in consecutive columns in database order do not necessarily imply similar temporal increments, so spatial differencing of temporal residuals may not further reduce their range. For example, voltage values around 3.333 V are scaled to integers around 3333, whereas adjacent temporal or spatial differences are often only a few integer units after scaling. Similarly, SoH values can remain unchanged over short sampling intervals, producing many zero residuals. These narrow residual ranges make fixed-width Int16 or Int32 storage inefficient, since many residuals require fewer bits than the original integer width. This observation motivates compact bit-width representation, while signed-value remapping reduces the overhead of representing positive and negative residuals. These data characteristics support the use of precision-preserving integerization, spatiotemporal differencing, and compact bit-width representation in Section 3.

3. A Lossless Compression Framework for Large-Scale BESS

This section presents the proposed reversible compression framework for cell-level BESS monitoring data. The framework is designed according to the data characteristics analyzed in Section 2. The raw monitoring table is first transformed into an integer working matrix, with the side information required for exact reconstruction retained. Finite decimal precision motivates precision-preserving integerization, temporal dependency motivates temporal differencing, and spatial correlation motivates spatial differencing. The resulting signed residuals are then mapped to non-negative integers and encoded by reversible bit-width packing. Finally, a lossless coding stage is applied to further reduce the bitstream size. All reconstruction side information is stored together with the compressed data. The main notations are summarized in Table 3, and the flowchart of the framework is given in Figure 2.

Table 3. Main notation used in the proposed lossless compression framework.

Symbol	Definition
a, t, n	Measurement-variable, time, and cell or sensor indices
T, N_a	Number of time records and number of values per row for measurement variable a
$x_{t,n}^{(a)}, y_{t,n}^{(a)}$	Raw finite-precision value and corresponding integerized value
s_a	Scaling factor for measurement variable a
$\Delta^T, \Delta^C, \Delta^{RC}$	Temporal differencing, spatial differencing, and temporal-spatial differencing
$r_{t,n}^{(a)}, z_{t,n}^{(a)}$	Signed residual and non-negative remapped residual
m_t, M	Row bit width and number of header bits
s_t, S_t	Sign-bit side information and its length for row t
C_t	Encoded bitstream for row t
$\mathcal{M}_{\text{miss}}$	Missing-data mask

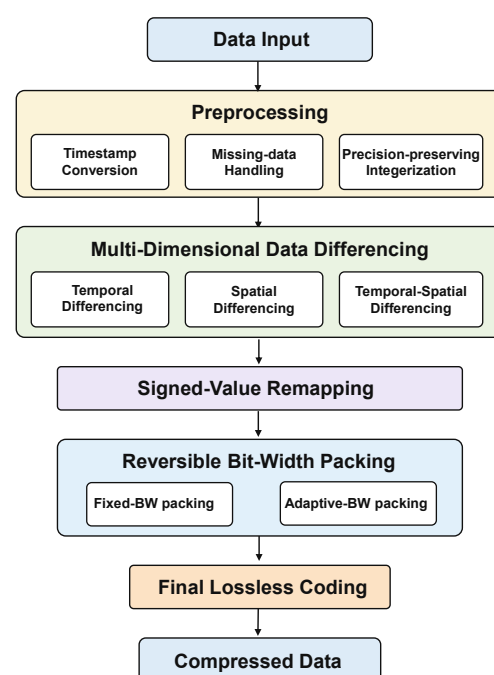


Figure 2. Flowchart of the proposed lossless compression framework.

3.1. Preprocessing

Before differencing, the proposed method converts the raw monitoring table into an integer working matrix. This step converts timestamp, i.e., the index of time strings to datetime values and finite-precision measurements to integers. It also records the side information required for exact reconstruction.

For missing data, columns whose entries are all missing are removed only from the working matrix. Their original indices and measurement-variable names are stored as side information. Occasional missing entries recorded by their time and column in $\mathcal{M}_{\text{miss}}$. Each recorded pair (t, d) indicates that the entry at time index t and column index d is missing. Only missing positions are recorded. After decompression, the mask restores its missing status. In the investigated dataset, 152,005 of the 49,345,160 active measurement values are missing, corresponding to 0.308%.

For each measurement variable a , the proposed method selects a scaling factor s_a according to its finite decimal precision. Here, a denotes the measurement variable, $t = 1, \dots, T$ denotes the time index, and $n = 1, \dots, N_a$ denotes the cell or temperature sensor index. A raw value $x_{t,n}^{(a)}$ is converted to an integer by

$$y_{t,n}^{(a)} = s_a x_{t,n}^{(a)}. \quad (1)$$

For example, a voltage value of 3.333 V is converted to 3333 when $s_a = 1000$. The scaling factor and integer storage type are stored as side information. During reconstruction, the original value is recovered by

$$x_{t,n}^{(a)} = \frac{y_{t,n}^{(a)}}{s_a}. \quad (2)$$

This operation is different from quantization. The scaling factor matches the finite decimal precision of the measurement variable, so the conversion is reversible. The scaling factor is selected so that $s_a x_{t,n}^{(a)}$ is an integer for every recorded finite-precision value of measurement variable a . If the scaled values fit in $[-2^{15}, 2^{15} - 1]$, they are stored as Int16; otherwise, Int32 is used. The side information is encoded with the compressed payload and included in the reported compressed size.

3.2. Multi-Dimensional Data Differencing

After integerization, the proposed method applies multi-dimensional differencing to the integer working matrix. Temporal differencing reduces redundancy between consecutive time records. Spatial differencing reduces spatial redundancy within each measurement variable. Temporal-spatial differencing first applies temporal differencing and then applies spatial differencing to the temporal-residual row. The reference values required for inverse differencing are stored as side information.

Temporal Differencing: Let $\mathbf{y}_t \in \mathbb{Z}^D$ denote the integer vector at time index t , where D is the number of measurement dimensions after preprocessing. Temporal differencing is defined as

$$\Delta^T \mathbf{y}_t = \mathbf{y}_t - \mathbf{y}_{t-1}, \quad t = 2, \dots, T. \quad (3)$$

The first vector $\mathbf{y}_1$ is stored as side information. The inverse transform is

$$\mathbf{y}_t = \mathbf{y}_1 + \sum_{k=2}^t \Delta^T \mathbf{y}_k, \quad t = 2, \dots, T. \quad (4)$$

The first integer vector $\mathbf{y}_1$ is retained as side information. Inverse temporal differencing reconstructs the integer vectors. The original measurements are then recovered using the variable-specific scaling factors.

Spatial and Temporal–Spatial Differencing: For spatial differencing, the proposed method processes each measurement variable separately. Let $\mathbf{Y}^{(a)} \in \mathbb{Z}^{T \times N_a}$ denote the integer matrix of measurement variable a . The element $y_{t,n}^{(a)}$ denotes the value with time index t and cell or sensor index n . Spatial differencing is defined as

$$\Delta^C y_{t,n}^{(a)} = y_{t,n}^{(a)} - y_{t,n-1}^{(a)}, \quad t = 1, \dots, T, \quad n = 2, \dots, N_a. \quad (5)$$

The first column $y_{t,1}^{(a)}$ is stored as side information. The inverse transform is

$$y_{t,n}^{(a)} = y_{t,1}^{(a)} + \sum_{j=2}^n \Delta^C y_{t,j}^{(a)}, \quad n = 2, \dots, N_a. \quad (6)$$

For temporal–spatial differencing, temporal differencing is first applied as

$$r_{t,n}^{(a)} = y_{t,n}^{(a)} - y_{t-1,n}^{(a)}, \quad t = 2, \dots, T. \quad (7)$$

Spatial differencing is then applied to the temporal residuals:

$$\Delta^{RC} y_{t,n}^{(a)} = r_{t,n}^{(a)} - r_{t,n-1}^{(a)}, \quad t = 2, \dots, T, \quad n = 2, \dots, N_a. \quad (8)$$

The first row $y_{1,:}^{(a)}$ and the first residual column $r_{t,1}^{(a)}$ are stored for exact reconstruction. The inverse spatial step reconstructs each temporal-residual row as

$$r_{t,n}^{(a)} = r_{t,1}^{(a)} + \sum_{j=2}^n \Delta^{RC} y_{t,j}^{(a)}, \quad n = 2, \dots, N_a. \quad (9)$$

The inverse temporal step then reconstructs the integer values as

$$y_{t,n}^{(a)} = y_{1,n}^{(a)} + \sum_{k=2}^t r_{k,n}^{(a)}, \quad t = 2, \dots, T. \quad (10)$$

3.3. Signed-Value Remapping

Differencing produces positive and negative residuals. Signed-value remapping converts these residuals into non-negative integers before bit-width packing. This step removes the need for a separate sign bit and maps small-magnitude residuals to small unsigned values.

Let $\hat{r}_i$ denote a signed residual after differencing. The remapped value z_i is defined as

$$z_i = \begin{cases} 2\hat{r}_i, & \hat{r}_i \geq 0, \\ -2\hat{r}_i - 1, & \hat{r}_i < 0. \end{cases} \quad (11)$$

where $z_i \in \mathbb{Z}_{\geq 0}$. The inverse mapping is

$$\hat{r}_i = \begin{cases} z_i/2, & z_i \text{ is even,} \\ -(z_i + 1)/2, & z_i \text{ is odd.} \end{cases} \quad (12)$$

In this implementation, the remapping is implemented by ZigZag encoding. After this remapping, all residuals become non-negative integers. Therefore, the following bit-width packing stage only needs to encode unsigned values and does not require an additional sign bit.

3.4. Reversible Bit-Width Packing

After differencing and signed-value remapping, the residuals are represented as integers with typically small magnitudes. Direct storage using fixed bit-width (fixed-BW) integer formats is inefficient because many high-order bits remain unused. Therefore, the proposed implementation applies bit-width packing to the values entering the packing stage. If signed-value remapping is enabled, the packed values are the remapped non-negative residuals. Otherwise, the residual magnitudes are packed, while their sign bits are stored together with the compressed payload. This sign-bit cost is included in the reported compressed size. For exact reconstruction, the bit-width information required by each packing block is also recorded in the payload. The following bit-width formulas are defined for one measurement-variable block. Each block selects its bit width independently.

Fixed-BW Packing: Fixed-BW packing uses one bit width for all remapped residuals in a compression block. Let $\{z_i\}_{i=1}^L$ denote the remapped residual sequence, where $z_i \in \mathbb{Z}_{\geq 0}$. More generally, z_i denotes the non-negative value encoded by the packing stage: it is the remapped residual when signed-value remapping is used, and the residual magnitude when signed-value remapping is not used. The required bit width is

$$m_{\text{fix}} = \left\lceil \log_2 \left(\max_{1 \leq i \leq L} z_i + 1 \right) \right\rceil. \quad (13)$$

Each value is represented using m_{fix} bits. The packed bitstream is

$$\mathcal{P}_{\text{fix}} = [m_{\text{fix}}]_{M_{\text{fix}}} \parallel [z_1]_{m_{\text{fix}}} \parallel \cdots \parallel [z_L]_{m_{\text{fix}}}, \quad (14)$$

where $[z_i]_m$ denotes the m -bit binary representation of z_i , $\parallel$ denotes bit-string concatenation, and M_{fix} is the number of header bits used to store m_{fix} . The total number of packed bits is

$$B_{\text{fix}} = M_{\text{fix}} + Lm_{\text{fix}}. \quad (15)$$

If the packed values are residual magnitudes rather than remapped residuals, the corresponding sign bits are stored within the measurement payload and are counted in the final compressed size.

Adaptive-BW Packing: Fixed-BW packing may waste bits when only a few residuals require a large bit width. Adaptive-BW packing reduces this overhead by assigning one bit width to each residual row. Let

$$\mathbf{z}_t = [z_{t,1}, z_{t,2}, \dots, z_{t,N_a}] \quad (16)$$

denote the remapped residual row, where N_a is the number of values in one row of measurement-variable blocks a . The entries of $\mathbf{z}_t$ are remapped residuals when signed-value remapping is used, and residual magnitudes otherwise. The row bit width is

$$m_t = \left\lceil \log_2 \left(\max_{1 \leq j \leq N_a} z_{t,j} + 1 \right) \right\rceil. \quad (17)$$

In this implementation, $M = 8$, and m_t is stored as one unsigned 8-bit header before each residual row. The maximum supported bit width when signed-value remapping is used is 32 bits, which is sufficient for the investigated dataset. An error is raised before remapping if any residual is outside the signed 32-bit range. Let $\mathbf{s}_t$ denote the sign-bit sequence for row t , whose length is S_t . For row t , the encoded bitstream is defined as

$$\mathcal{C}_t = [m_t]_M \parallel [z_{t,1}]_{m_t} \parallel \cdots \parallel [z_{t,N_a}]_{m_t} \parallel \mathbf{s}_t, \quad (18)$$

where $[z_{t,j}]_{m_t}$ denotes the m_t -bit binary representation of $z_{t,j}$, and $\parallel$ denotes bit-string concatenation. For each measurement-variable block, both ends know the number of values in each residual row and the number of residual rows. After reading the header m_t , m_t bits are read for each value in row t , followed by S_t sign bits when signed-value remapping is not used. If all entries in row t are zero, $m_t = 0$, the complete row is reconstructed as zeros. In this case, only the zero-width header is stored when signed-value remapping is used. Otherwise, the S_t sign bits are also stored. The rows are concatenated without row padding. Zero bits are added only after the last residual row of the measurement-variable block to complete the final byte. After the agreed number of residual rows has been reconstructed, these padding bits are discarded.

The adaptive bit-width (adaptive-BW) packed bitstream is obtained by concatenating all row bitstreams:

$$P_{\text{adp}} = C_1 \parallel C_2 \parallel \cdots \parallel C_{T_r}. \quad (19)$$

where T_r is the number of residual rows. The total number of packed bits is

$$B_{\text{adp}} = \sum_{t=1}^{T_r} (M + N_a m_t + S_t). \quad (20)$$

where S_t denotes the number of sign bits stored for row t . When signed-value remapping is used, $S_t = 0$. When signed-value remapping is not used, the packed values are residual magnitudes, and the corresponding sign bits are stored within the measurement payload. In this case, $S_t = N_a$ in our implementation. The row bit-width headers and sign bits are stored with the compressed payload and are included in the reported compressed size. Equations (15) and (20) give the packed size of one measurement-variable block. In the combined-variable setting, they are applied independently to the SoC, SoH, temperature, and voltage blocks. Let $B_{\text{fix}}^{(a)}$ and $B_{\text{adp}}^{(a)}$ denote the sizes obtained for block a . The total packed sizes are

$$B_{\text{fix,total}} = \sum_{a \in \mathcal{A}} B_{\text{fix}}^{(a)}, \quad B_{\text{adp,total}} = \sum_{a \in \mathcal{A}} B_{\text{adp}}^{(a)}. \quad (21)$$

Here, $\mathcal{A}$ denotes the set of measurement variables. The packed streams of all blocks are concatenated with the required side information before FLC.

3.5. Lossless Reconstruction

The decompression process follows the inverse order of compression. After FLC, the proposed method recovers the packed bitstream. Bit-width unpacking then reconstructs the non-negative remapped residuals. Inverse signed-value remapping converts these values back to signed residuals. Inverse differencing reconstructs the integer working matrix. The stored scaling factors convert the recovered integers back to the original finite-precision measurement values. Finally, the stored column order and missing-value mask are used to restore missing entries at their original positions. Thus, the proposed method reconstructs the original raw monitoring table exactly.

4. Evaluation

We evaluate the proposed method on the AMD and Raspberry Pi platforms, as summarized in Table 4. The AMD platform represents cloud-side processing, while the Raspberry Pi is selected for edge-side processing, considering the requirements for lightweight deployment and low power consumption [34]. In real-time compression, each newly collected time record is compressed as one block. In block-based compression, the records are compressed in 5 min or 60 min blocks, or all 59,452 records are compressed as one global block. In this paper, real-time refers to the one-record block used in the experiment. Because the dataset

comes from one industrial BESS deployment, the evaluation is limited to the investigated sampling configuration.

Table 4. Platform specifications for the AMD and Raspberry Pi platforms.

Specification	AMD Platform	Raspberry Pi Platform
Model	AMD Ryzen 5 7500F 6-Core Processor	Raspberry Pi 4 Model B Rev 1.5
Processor	AMD Zen 4, 6 cores/12 threads	Broadcom BCM2711, Cortex-A72, 4 cores
Maximum Frequency	3.701 GHz	1.800 GHz
Architecture	x86	aarch64
Memory	32 GB RAM	2 GiB available RAM
Operating System	Windows	Debian GNU/Linux 13

We consider two input settings. In the combined-variable setting, SoC, SoH, temperature, and voltage form separate measurement-variable blocks and are compressed in one operation. Each block uses its own scaling factor and bit-width parameters. Temporal differencing is applied along the time dimension. Spatial differencing is confined to columns within the same block. The variable-wise streams are concatenated before FLC. The combined-variable compression ratio is calculated from the total compressed size and the total original size of the four blocks. It is not the sum or average of the individual-variable ratios. In the individual-variable setting, each measurement variable is compressed independently.

In this evaluation, the FLC is implemented with zlib. The method labels specify the operations performed before FLC. The labels TD, SD, and TSD denote temporal differencing, spatial differencing, and cascaded temporal–spatial differencing, respectively. The TSD operation applies TD followed by SD. SRM denotes signed-value remapping implemented using ZigZag encoding. The labels fixBW and adpBW denote fixed and adaptive bit-width packing, respectively. The FLC baseline applies FLC directly to the integerized data. For packed variants, FLC is omitted from the method label for brevity but is still applied after bit-width packing.

Two compression models are considered for different deployment requirement. In the compression ratio evaluation, side information refers to all information concatenated with the measurement payload before FLC. For the transmission model, it includes the missing-data mask, boundary values, and preprocessing information, e.g., the bit-width headers and sign bits. For the storage model, it further includes other necessary information, timestamps, and cell IDs. The compression ratio for the transmission model is

$$R_{tr} = \frac{\text{FLC}(\text{measurement payload} + \text{side information})}{\text{uncompressed measurement}} \times 100\%. \quad (22)$$

The compression ratio for the storage model is

$$R_{st} = \frac{\text{FLC}(\text{measurement payload} + \text{side information})}{\text{uncompressed measurement} + \text{timestamps} + \text{cell IDs}} \times 100\%. \quad (23)$$

For both ratios, the measurement payload and the corresponding side information are concatenated and processed together by FLC. The final FLC output forms the numerator. For the transmission model, the denominator contains only the uncompressed measurement representation. Timestamps are excluded because we assume that the same timestamp sequence is stored at both ends under transmission model. For the storage model, the denominator contains the uncompressed measurement representation, timestamps, and cell IDs. Other necessary information is included in the side information and therefore in

the numerator, but it is not included in the denominator. Table 5 reports both ratios relative to the integerized measurements, Float32 values, and CSV text. The results indicate that different data representations affect the reported compression ratios but do not change which methods give lower compression ratios. Unless otherwise stated, the compression ratio refers to transmission ratios with the integerized measurement representation.

Table 5. Compression ratios for three data representations.

Method	Compression Ratio R_{tr} Transmission Model (%)			Compression Ratio R_{st} Storage Model (%)		
	Integerized	Float32	CSV	Integerized	Float32	CSV
FLC	1.354	0.677	0.345	1.631	0.818	0.624
TD-FLC	1.128	0.564	0.287	1.406	0.705	0.538
SD-FLC	1.768	0.884	0.450	2.044	1.024	0.782
TSD-FLC	1.172	0.586	0.298	1.450	0.727	0.555
TD-SRM-fixBW	0.990	0.495	0.252	1.270	0.636	0.486
TD-SRM-adpBW	0.779	0.389	0.198	1.059	0.531	0.405
SD-SRM-fixBW	4.269	2.135	1.087	4.532	2.272	1.734
SD-SRM-adpBW	2.051	1.025	0.522	2.324	1.165	0.889
TSD-SRM-fixBW	0.973	0.487	0.248	1.253	0.628	0.479
TSD-SRM-adpBW	0.898	0.449	0.229	1.178	0.591	0.451
TSD-fixBW	1.114	0.557	0.284	1.392	0.698	0.533
TSD-adpBW	1.135	0.568	0.289	1.413	0.708	0.541

Table 6 compares six final lossless coders under the same input settings. Each coder is applied directly to the integerized measurements and to the packed bitstream after TD-SRM-adpBW. The proposed processing reduces the ratio for every tested final lossless coder. Brotli leads the smallest compression ratio 0.770%, after TD-SRM-adpBW for storage model and the corresponding compression ratio of transmission model is 0.516%. In the following, zlib is retained as the reference implementation of the FLC stage.

Table 6. Compression ratios using different final lossless coders.

Final Lossless Coder	Input to Final Lossless Coding	Compression Ratio R_{tr} Transmission Model (%)	Compression Ratio R_{st} Storage Model (%)
zlib	Preprocessed Measurements	1.354	1.631
gzip	Preprocessed Measurements	1.354	1.631
zstd	Preprocessed Measurements	1.115	1.419
LZ4	Preprocessed Measurements	3.691	4.124
Brotli	Preprocessed Measurements	1.062	1.312
bzip2	Preprocessed Measurements	0.848	1.222
zlib	Packed Bitstream after TD-SRM-adpBW	0.779	1.059
gzip	Packed Bitstream after TD-SRM-adpBW	0.779	1.059
zstd	Packed Bitstream after TD-SRM-adpBW	0.593	0.899
LZ4	Packed Bitstream after TD-SRM-adpBW	1.409	1.853
Brotli	Packed Bitstream after TD-SRM-adpBW	0.516	0.770
bzip2	Packed Bitstream after TD-SRM-adpBW	0.720	1.086

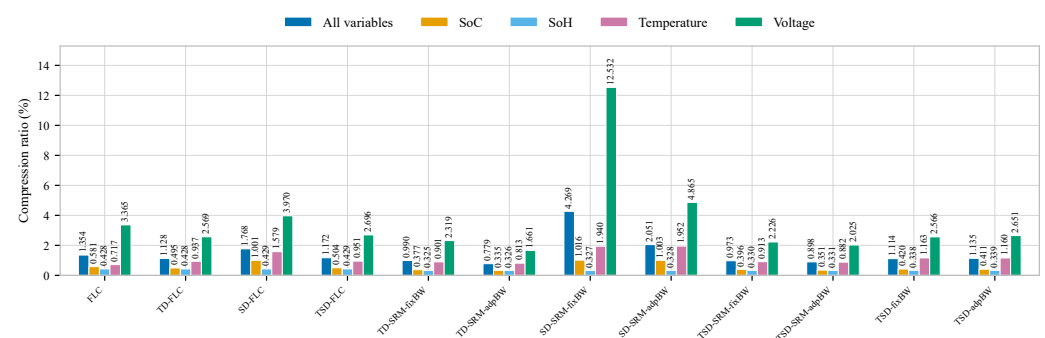
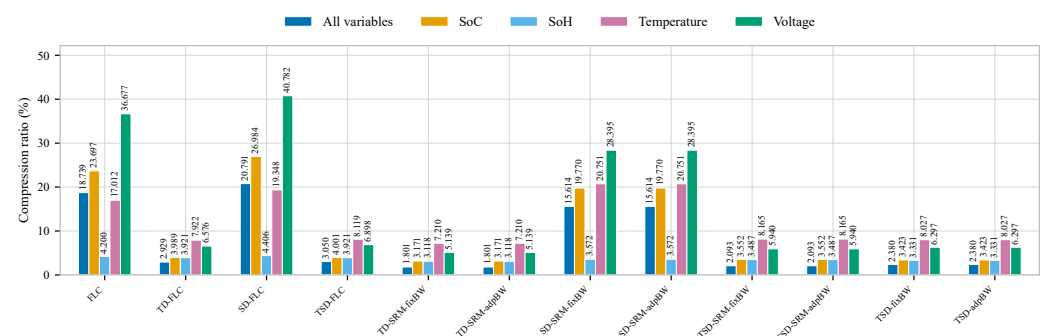
Table 7 shows compression ratios in the transmission model of 0.869%, 0.773%, and 0.779% for 50, 200, and 240 cells, respectively. The compression ratio in the storage model decreases from 2.004% to 1.059% as the number of cells increases from 50 to 240. The compression ratios also depend on the measurement variable.

Table 7. Compression ratios for different numbers of cells and measurement variables.

Measurement Variables	Number of Cells	Compression Ratio R_{tr} Transmission Model (%)	Compression Ratio R_{st} Storage Model (%)
SoC + SoH + Temperature + Voltage	50	0.869	2.004
SoC + SoH + Temperature + Voltage	200	0.773	1.100
SoC + SoH + Temperature + Voltage	240	0.779	1.059
SoC	240	0.335	1.300
SoH	240	0.326	1.291
Temperature	240	0.813	2.861
Voltage	240	1.661	2.604

4.1. Block-Based Compression Results

Figures 3a and 4 present the block-based compression results for the global block using all 59,452 records and for the 60 min and 5 min blocks in the transmission model. In the global setting, all records are compressed as one block. In the 5 and 60 min settings, the same record sequence is processed in shorter blocks. Each violin in Figure 4 shows the distribution of compression ratios in the transmission model across the corresponding blocks. The horizontal line denotes the median.

**(a)** Global block-based compression using all 59,452 records.**(b)** Real-time compression.**Figure 3.** Compression performance comparison.

For the global block, FLC gives a compression ratio in the transmission model of 1.354% when all measurement variables are compressed together. TD-FLC and SD-FLC give compression ratios of 1.128% and 1.768% in the transmission model, respectively. TD-SRM-adpBW gives the lowest compression ratio in the transmission model of 0.779%, which is 42.5% lower than the FLC result for the same integerized measurement representation. The corresponding compression ratio in the storage model is 1.059%. TSD-SRM-adpBW gives a compression ratio of 0.898% in the transmission model and 1.178% in the storage model. Similar measurement values across cells may not imply similar temporal incre-

ments. Therefore, temporal–spatial differencing would not provide an additional gain over temporal differencing in this setting.

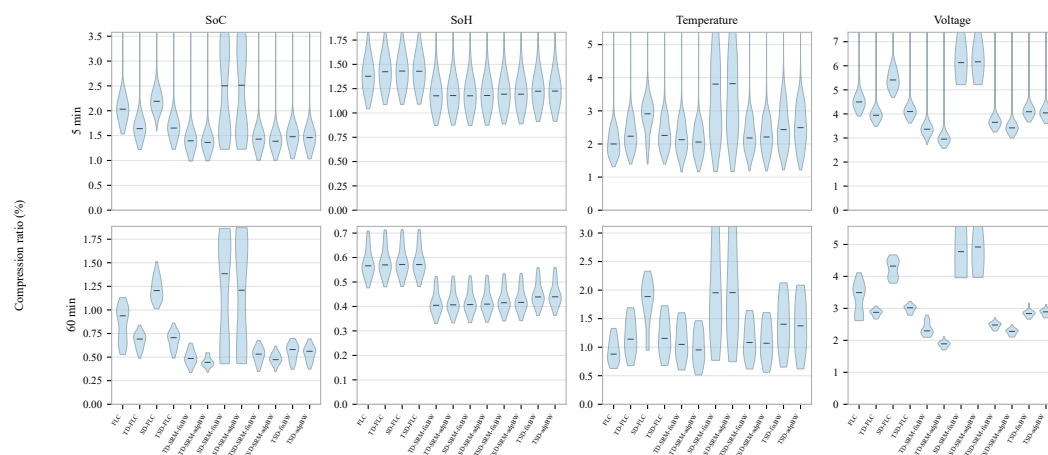


Figure 4. Block-based compression results for 5 and 60 min blocks.

The global results also differ across measurement variables. For SoC and voltage, TD-SRM-adpBW gives compression ratios of 0.335% and 1.661% in the transmission model, respectively. For SoH, TD-SRM-fixBW and TD-SRM-adpBW give compression ratios of 0.325% and 0.326% in the transmission model, respectively. The nearly constant SoH values produce many zero-temporal residuals. Temperature shows a different trend. In the transmission model, FLC gives a compression ratio of 0.717%, which is lower than the 0.813% obtained by TD-SRM-adpBW. This may be because the storage saved by reducing the residual size is less than the storage required for the bit-width headers and reconstruction side information.

The 5 and 60 min results show that longer blocks generally reduce the compression ratio. A longer block spreads the block-level side information over more time records. It also provides a longer input sequence for FLC. Figure 4 shows lower median compression ratio for SoC and voltage at 60 min than at 5 min in the transmission model. The distributions also become narrower for most configurations. The block-length effect remains dependent on the measurement variable. SoH gives the lowest compression ratios because it changes little over the studied sequence. Figure 4 shows a lower best median compression ratio for SoH at 60 min than at 5 min in the transmission model. For temperature, FLC gives the lowest median compression ratio in both block settings. Adaptive-BW packing is effective when the required width changes across residual rows. However, it requires a width header for each row and measurement-variable block. Fixed-BW packing can therefore be comparable when the residual range is stable. Overall, temporal differencing performs consistently well for SoC and voltage, while FLC could be sufficient for temperature.

Table 8 presents the components of the side information and compares their sizes. The missing-data mask requires 1,216,056 bytes for all 12 methods before FLC. Temporal differencing requires 1660 bytes of boundary values, whereas spatial and temporal–spatial differencing require more boundary values. These quantities are included in the reported ratios. Table 9 reports the average side information for the 5 min and real-time compression. The real-time missing-data mask and total values are averages per time record across all 59,452 records. In transmission mode, the timestamp sequence is assumed to be stored at both ends and is not transmitted.

Table 10 evaluates the contribution of each operation of the proposed method. Removing adaptive bit-width packing increases the compression ratio from 1.059% to 1.403% in the storage model. Removing SRM increases it to 1.184%, while removing temporal differencing increases it to 1.958%. Without FLC, it increases to 3.336%. Thus, all four

operations reduce the final compressed size, although their effects differ. Specifically, in the current TD-SRM-adpBW + FLC configuration, all four operations help reduce the final compression ratio. With FLC retained, removing temporal differencing gives the largest increase in the ratio.

Table 8. Storage model: side information for the 12 methods using all 59,452 records.

Method	Preprocessing Information (Bytes)	Missing-Data Mask (Bytes)	Boundary Values (Bytes)	Other Necessary Information (Bytes)	Total (Bytes)
FLC	10,751	1,216,056	0	484,367	1,711,174
TD-FLC	10,907	1,216,056	1660	484,370	1,712,993
SD-FLC	10,907	1,216,056	475,616	484,370	2,186,949
TSD-FLC	10,911	1,216,056	1,904,092	484,371	3,615,430
TD-SRM-fixBW	10,907	1,216,056	1660	484,376	1,712,999
TD-SRM-adpBW	10,907	1,216,056	1660	484,376	1,712,999
SD-SRM-fixBW	10,907	1,216,056	475,616	484,376	2,186,955
SD-SRM-adpBW	10,907	1,216,056	475,616	484,376	2,186,955
TSD-SRM-fixBW	10,911	1,216,056	1,904,092	484,377	3,615,436
TSD-SRM-adpBW	10,911	1,216,056	1,904,092	484,377	3,615,436
TSD-fixBW	10,911	1,216,056	1,904,092	484,373	3,615,432
TSD-adpBW	10,911	1,216,056	1,904,092	484,373	3,615,432

Note: Preprocessing information includes scaling factors, column names and order, integer formats, and the differencing method. Other necessary information includes the timestamps and cell IDs.

Table 9. Storage model: Average side-information composition for a 5 min window and real-time compression.

Method	Preprocessing Information (Bytes)	Missing-Data Mask (Bytes)	Boundary Values (Bytes)	Other Necessary Information (Bytes)	Total (Bytes)
5 min: TD-SRM-adpBW	10,899	3848	1660	10,352	26,759
5 min: TSD-SRM-adpBW	10,903	3848	8092	10,352	33,195
Real-Time: TD-SRM-adpBW	0	36.454	0	8	44.454
Real-Time: TSD-SRM-adpBW	0	36.454	8	8	52.454

Table 10. Ablation experiment for TD-SRM-adpBW followed by FLC.

Configuration	Compression Ratio R_{tr} Transmission Model (%)	Compression Ratio R_{st} Storage Model (%)
TD-SRM-adpBW + FLC	0.779	1.059
TD-SRM-FLC	1.125	1.403
TD-adpBW + FLC	0.905	1.184
SRM-adpBW + FLC	1.682	1.958
TD-SRM-adpBW without FLC	2.870	3.336

Note: Each subsequent row removes one operation from the complete configuration in the first row.

The dataset columns are ordered by the database cell IDs, from battery 1 to battery 240. The public dataset does not provide physical positions, module assignments, or electrical topology. Therefore, spatial differencing in this study follows the database column order and does not imply strict physical adjacency. Table 11 shows that shuffling the cell order has little effect on TD-SRM-adpBW, but increases the self-contained storage ratio of SD-SRM-adpBW. The TSD-SRM-adpBW result changes only slightly because temporal differencing

is applied first. Overall, although the physical positions of the cells are unavailable, the results show that spatial differencing depends strongly on cell order. Shuffling the cell order has a smaller effect when temporal differencing is applied.

Table 11. Storage model: results after cell-order shuffling.

Method	Cell Order	Compressed Size (Bytes)	Self-Contained Storage Ratio (%)
TD-SRM-adpBW	Original	1,050,186	1.059
TD-SRM-adpBW	Shuffle 1	1,065,505	1.074
TD-SRM-adpBW	Shuffle 2	1,065,195	1.074
SD-SRM-adpBW	Original	2,305,108	2.324
SD-SRM-adpBW	Shuffle 1	3,774,896	3.806
SD-SRM-adpBW	Shuffle 2	4,211,024	4.246
TSD-SRM-adpBW	Original	1,168,601	1.178
TSD-SRM-adpBW	Shuffle 1	1,198,041	1.208
TSD-SRM-adpBW	Shuffle 2	1,196,577	1.207

4.2. Real-Time Compression Results

Figure 3b presents the real-time results in the transmission model. Each compression block contains one time record. The previous sample is retained as a stateful reference for temporal differencing. However, FLC processes each time record separately and cannot exploit a long temporal context within a block. Its combined-variable compression ratio therefore increased.

Temporal differencing substantially improves real-time compression. The TD-FLC variant reduces the combined-variable ratio to 2.929%. Adding signed-value remapping and adaptive-BW packing further reduces it to 1.801%. This is the lowest ratio among the evaluated real-time variants and represents a 90.4% reduction in compressed size relative to FLC. Adaptive-BW and fixed-BW packing achieve the same result after temporal differencing. The ratios of TD-SRM-adpBW and TD-SRM-fixBW are both 1.801%. Because each real-time block contains only one time record, fixBW and adpBW select the same bit width and therefore produce the same compressed size. Spatial differencing is less effective in the real-time setting. The SD-FLC variant gives 20.791%, which is higher than the FLC result. The TSD variants reduce the ratio by using the retained temporal reference, but they do not outperform temporal differencing alone. For example, TSD-SRM-adpBW gives 2.093%, compared with 1.801% for TD-SRM-adpBW. The same trend is observed for most individual variables. TD-SRM-fixBW and TD-SRM-adpBW give the same ratios for SoC, temperature, and voltage, with values of 3.171%, 7.210%, and 5.139%, respectively. For SoH, both variants give 3.118%. These results confirm that temporal differencing with signed-value remapping and bit-width packing is the most effective real-time configuration for the combined-variable input.

4.3. Compression Time and Memory Requirement

Table 12 reports the compression time, decompression time, and peak memory requirement for storage mode using all 59,452 records. The Raspberry Pi experiment measures compression only as it deployed on the edge side.

In the real-time setting, all variants complete within 0.250 ms per time record on the AMD platform and within 1.554 ms on the Raspberry Pi platform. TD-SRM-adpBW requires 0.198 ms on AMD and 1.252 ms on Raspberry Pi. These values correspond to approximately 5062 and 798 records/s, respectively. Both values are substantially higher than the measured average acquisition rate. This comparison is limited to the sampling configuration of the investigated BESS. The global storage-mode results provide a separate all-record benchmark. TD-SRM-adpBW compresses all 59,452 records in 10.317 s and decompresses them in

2.754 s on AMD. These values correspond to approximately 5762 records/s for compression and 21,588 records/s for decompression. Its peak memory requirement is 1105.191 MiB. On Raspberry Pi, compression requires 61.515 s, corresponding to approximately 966 records/s, and peak memory requirement is 927.875 MiB.

Table 12. Compression time, decompression time, and peak memory required for all 59,452 records in storage mode.

Method	AMD Platform			Raspberry Pi Platform	
	Compression Time (ms)	Decompression Time (ms)	Peak Memory (MiB)	Compression Time (ms)	Peak Memory (MiB)
FLC	2085.139	281.402	1117.938	17,234.854	961.629
TD-FLC	2353.539	621.466	1209.035	19,174.616	962.531
SD-FLC	2501.418	399.079	1195.305	19,743.375	962.781
TSD-FLC	2429.793	722.718	1213.266	20,041.926	886.812
TD-SRM-fixBW	9944.685	1023.129	1086.203	49,259.688	895.578
TD-SRM-adpBW	10,317.484	2753.906	1105.191	61,514.554	927.875
SD-SRM-fixBW	9941.343	691.102	1131.469	51,532.826	915.098
SD-SRM-adpBW	12,378.389	3214.460	1045.719	72,950.801	928.805
TSD-SRM-fixBW	9272.840	1101.316	1133.074	49,673.890	941.613
TSD-SRM-adpBW	10,353.406	2861.855	1122.383	61,954.198	914.527
TSD-fixBW	2574.889	791.838	1135.785	21,666.838	872.355
TSD-adpBW	4919.362	2659.904	1127.633	39,917.867	863.922

4.4. Discussion

The results reveal several findings. First, temporal differencing is more effective than temporal–spatial differencing in several settings. The investigated BESS data exhibit strong temporal sparsity after integerization. Most temporal residuals are therefore zero or close to zero. Applying spatial differencing to these residuals does not always reduce their range further. Similar measurement values across cells do not necessarily indicate similar temporal increments. Temporal–spatial differencing is beneficial only when consecutive columns in database order have similar temporal changes. Otherwise, it introduces additional side information and packing overhead.

Second, compression performance strongly depends on the block length. FLC leads to 1.354% in the global block-based setting but 18.739% in the real-time setting in the transmission model, where each block contains one time record. Such a short block provides little context for FLC and gives a larger relative side-information cost. Temporal differencing reduces this dependence by using the previous sample as a reference. Increasing the buffering window from 5 to 60 min also lowers the ratio for most variables. Longer blocks distribute the block-level side information across more records and provide more input data for FLC.

Third, the preferred representation depends on the measurement variable. Temporal differencing with adaptive-BW packing performs well for SoC and voltage because their residual ranges vary over time. SoH changes little within the studied sequence. Its residuals are highly concentrated, which makes fixed-BW packing competitive with adaptive-BW packing. Temperature shows a different pattern. For this variable, FLC often outperforms the differencing and packing variants because their additional side information exceeds the reduction in residual size. A single compression configuration is therefore not optimal for all measurement variables.

Finally, compression efficiency, compression time, and peak memory consumption show a clear tradeoff. Signed-value remapping and bit-width packing improve the compression ratio but require additional processing. All tested variants complete real-time com-

pression within 0.32 ms on the AMD platform and 1.64 ms on the Raspberry Pi platform. In the transmission model, TD-SRM-adpBW gives the lowest in the global combined-variable setting, whereas TD-SRM-fixBW and TD-SRM-adpBW give the lowest real-time ratio. These conclusions are limited to the investigated BESS dataset and sampling configuration.

5. Conclusions

This paper presented a structure-aware lossless compression method for cell-level BESS monitoring data in cloud-edge systems and released an industrial cell-level dataset collected from a 215 kWh BESS. The dataset was analyzed to identify compression-relevant properties, including finite decimal precision, slow temporal changes, spatial similarity in database column order, and different value ranges across measurement variables. Based on these properties, the proposed method groups the data into measurement-variable blocks and reversibly converts finite-precision values into integers. It then applies differencing, signed-value remapping, bit-width packing, and FLC. The compressed output retains the side information required to reconstruct the measurement values and missing-value indicators exactly. The results show that temporal differencing provides the main compression gain in the investigated dataset. When all measurement variables and all 59,452 records are compressed together, TD-SRM-adpBW leads to 0.779% in the transmission model and 1.059% in the storage model. Its real-time transmission ratio is 1.801%. The block-based results further show that longer blocks generally improve compression. However, temporal-spatial differencing does not consistently improve temporal differencing, and the preferred configuration depends on the measurement variable.

The present evaluation uses 59,452 records from one 215 kWh industrial BESS deployment. It does not establish performance across longer periods, different BESSs, battery chemistries, load profiles, operating conditions, sensor architectures, BMS sampling configurations, etc. Power consumption was not measured because the required test equipment was unavailable. Further work would include multi-day and multi-system validation, different missing-data rates, power measurement, and the CPU-utilization of the platforms.

Author Contributions: Conceptualization, T.L. (Tianle Liu) and C.Y.; methodology, T.L. (Tianle Liu) and C.Y.; software, T.L. (Tengjiao Lu); validation, T.L. (Tengjiao Lu), T.L. (Tianle Liu) and C.Y.; formal analysis, T.L. (Tengjiao Lu); investigation, T.L. (Tengjiao Lu); resources, C.Y.; data curation, C.Y.; writing—original draft preparation, T.L. (Tengjiao Lu) and T.L. (Tianle Liu); writing—review and editing, P.P., T.H. and C.L.; visualization, T.L. (Tengjiao Lu); supervision, C.Y.; project administration, T.L. (Tianle Liu) and C.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Huzhou Key Research and Development Program grant number 2024ZD2029.

Data Availability Statement: The dataset used in this study is publicly available at the repository specified in the manuscript. The source code is available from the corresponding author upon reasonable request.

Conflicts of Interest: Author Changlin Yang was employed by the Energy AI Department, Zhejiang Tianneng Energy Storage Technology Development Co., Ltd., Huzhou, China. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Sebbagh, T.; Manfo, T.A.; Şahin, M.E. Advances in Battery Technologies for Next-Generation Energy Storage Systems. *Electronics* **2026**, *15*, 690. [[CrossRef](#)]
2. Nyamathulla, S.; Dhanamjayulu, C. A Review of Battery Energy Storage Systems and Advanced Battery Management System for Different Applications: Challenges and Recommendations. *J. Energy Storage* **2024**, *86*, 111179. [[CrossRef](#)]

3. Elalfy, D.A.; Gouda, E.; Kotb, M.F.; Bureš, V.; Sedhom, B.E. Comprehensive Review of Energy Storage Systems Technologies, Objectives, Challenges, and Future Trends. *Energy Strategy Rev.* **2024**, *54*, 101482. [[CrossRef](#)]
4. Laila, N. Modeling and Simulation of a PV-Battery-Hydrogen Hybrid Energy Storage System for Stable Renewable Power Supply. Master's Thesis, University of Vaasa, Vaasa, Finland, 2025.
5. Mo, Y.; Lin, Q.; Chen, M.; Qin, S.J. Optimal Peak-Minimizing Online Algorithms for Large-Load Users with Energy Storage. In Proceedings of the IEEE INFOCOM 2021—IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Vancouver, BC, Canada, 10–13 May 2021; pp. 1–2. [[CrossRef](#)]
6. Tang, G.; Yuan, H.; Guo, D.; Wu, K.; Wang, Y. Reusing Backup Batteries as BESS for Power Demand Reshaping in 5G and Beyond. In Proceedings of the IEEE INFOCOM 2021—IEEE Conference on Computer Communications, Vancouver, BC, Canada, 10–13 May 2021; pp. 1–10. [[CrossRef](#)]
7. Krishna, T.N.V.; Kumar, S.V.S.V.P.D.; Srinivasa Rao, S.; Chang, L. Powering the Future: Advanced Battery Management Systems (BMS) for Electric Vehicles. *Energies* **2024**, *17*, 3360. [[CrossRef](#)]
8. Yildiz, B.; Stringer, N.; Klymenko, T.; Samhan, M.S.; Abramowitz, G.; Bruce, A.; MacGill, I.; Egan, R.; Sproul, A.B. Real-World Data Analysis of Distributed PV and Battery Energy Storage System Curtailment in Low Voltage Networks. *Renew. Sustain. Energy Rev.* **2023**, *186*, 113696. [[CrossRef](#)]
9. Kharlamova, N.; Hashemi, S. Evaluating Machine-Learning-Based Methods for Modeling a Digital Twin of Battery Systems Providing Frequency Regulation. *IEEE Syst. J.* **2023**, *17*, 2698–2708. [[CrossRef](#)]
10. Afrose, K.R. Safety Challenges and Mitigation Strategies in Grid-Scale Stationary Lithium-Ion Battery Energy Storage Systems. Master's Thesis, University of Vaasa, Vaasa, Finland, 2026.
11. Urquizo, J.; Singh, P. A Review of Health Estimation Methods for Lithium-Ion Batteries in Electric Vehicles and Their Relevance for Battery Energy Storage Systems. *J. Energy Storage* **2023**, *73*, 109194. [[CrossRef](#)]
12. Lu, J.; Xiong, R.; Tian, J.; Wang, C.; Sun, F. Deep Learning to Estimate Lithium-Ion Battery State of Health Without Additional Degradation Experiments. *Nat. Commun.* **2023**, *14*, 2760. . [[CrossRef](#)] [[PubMed](#)]
13. Al-Kadhim, H.M.; Al-Raweshidy, H.S. Energy Efficient Data Compression in Cloud Based IoT. *IEEE Sens. J.* **2021**, *21*, 12212–12219. [[CrossRef](#)]
14. Kong, X.; Zheng, Y.; Ouyang, M.; Li, X.; Lu, L.; Li, J.; Zhang, Z. Signal Synchronization for Massive Data Storage in Modular Battery Management System with Controller Area Network. *Appl. Energy* **2017**, *197*, 52–62. . [[CrossRef](#)]
15. Zeng, Y.; Chen, M.; Meng, J.; Zhao, H.; Bai, Z. Compressing and Reconstructing the Voltage Measurement of Lithium-Ion Battery Using Compressed Sensing. *J. Energy Storage* **2025**, *121*, 116613. [[CrossRef](#)]
16. Hu, H.; Dong, Y.; Jiang, Y.; Chen, Q.; Zhang, J. On the Age of Information and Energy Efficiency in Cellular IoT Networks with Data Compression. *IEEE Internet Things J.* **2023**, *10*, 5226–5239. [[CrossRef](#)]
17. Liu, Q.; Ma, J.; Zhao, X.; Zhang, K.; Meng, D.; Jiao, Z. Fault Diagnosis of Early Internal Short Circuit for Power Battery Systems Based on the Evolution of the Cell Charging Voltage Slope in Variable Voltage Window. *Appl. Energy* **2024**, *376*, 124310. [[CrossRef](#)]
18. Chen, C.; Wang, W.; Yin, H.; Zhan, L.; Liu, Y. Real-Time Lossless Compression for Ultrahigh-Density Synchrophasor and Point-on-Wave Data. *IEEE Trans. Ind. Electron.* **2022**, *69*, 2012–2021. [[CrossRef](#)]
19. Unterweger, A.; Engel, D. Lossless compression of high-frequency voltage and current data in smart grids. In Proceedings of the 2016 IEEE International Conference on Big Data (Big Data), Washington, DC, USA, 5–8 December 2016; pp. 3131–3139. [[CrossRef](#)]
20. Jeong, H.; Seo, G.; Hwang, E. SPDFlow: Lightweight Normalizing Flows With Range Asymmetric Numeral Systems for Online Compression of Large-Scale Smart Metering Data. *IEEE Internet Things J.* **2026**, *13*, 7695–7713. [[CrossRef](#)]
21. Chiarot, G.; Silvestri, C. Time Series Compression Survey. *ACM Comput. Surv.* **2023**, *55*, 1–32. [[CrossRef](#)]
22. Tan, H.; Xia, W.; Zou, X.; Deng, C.; Liao, Q.; Gu, Z. The Design of Fast Delta Encoding for Delta Compression Based Storage Systems. *ACM Trans. Storage* **2024**, *20*, 1–30. [[CrossRef](#)]
23. Kuldeep, G. Lightweight Time-Series Signal Compression Period Extraction and Multiresolution Using Difference Sequences. *IEEE Internet Things J.* **2022**, *9*, 7043–7050. [[CrossRef](#)]
24. Barr, K.C.; Asanović, K. Energy-Aware Lossless Data Compression. *ACM Trans. Comput. Syst.* **2006**, *24*, 250–291. [[CrossRef](#)]
25. Harnik, D.; Khaitzin, E.; Sotnikov, D.; Taharlev, S. A Fast Implementation of Deflate. In *Proceedings of the 2014 Data Compression Conference*; IEEE: Piscataway, NJ, USA, 2014; pp. 223–232. . [[CrossRef](#)]
26. Almeida, S.; Oliveira, V.; Pina, A.; Melle-Franco, M. Two High-Performance Alternatives to ZLIB Scientific-Data Compression. In *Proceedings of the International Conference on Computational Science and Its Applications*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 623–638.
27. Ward, J.R.; Barker, S.K. Powerstrip: High-Performance Compression for Energy Data. In Proceedings of the Eleventh ACM International Conference on Future Energy Systems, e-Energy '20, Virtual Event, 22–26 June 2020; pp. 242–252. [[CrossRef](#)]
28. Lu, T.; Xia, W.; Zou, X.; Xia, Q. Adaptively Compressing IoT Data on the Resource-Constrained Edge. In Proceedings of the USENIX Workshop on Hot Topics in Edge Computing, Virtual Event, 25–26 June 2020.

29. Giorgi, G. Lightweight Lossless Compression for N -Dimensional Data in Multi-Sensor Systems. *IEEE Sens. J.* **2019**, *19*, 8895–8903. [CrossRef]
30. Vestergaard, R.; Lucani, D.E.; Zhang, Q. Generalized Deduplication: Lossless Compression for Large Amounts of Small IoT Data. In Proceedings of the European Wireless 2019; 25th European Wireless Conference, Aarhus, Denmark, 2–4 May 2019; pp. 1–5.
31. Zhou, L.; He, L.; Zheng, Y.; Lai, X.; Ouyang, M.; Lu, L. Massive Battery Pack Data Compression and Reconstruction Using a Frequency Division Model in Battery Management Systems. *J. Energy Storage* **2020**, *28*, 101252. [CrossRef]
32. Zheng, Y.; Ouyang, M.; Li, X.; Lu, L.; Li, J.; Zhou, L.; Zhang, Z. Recording Frequency Optimization for Massive Battery Data Storage in Battery Management Systems. *Appl. Energy* **2016**, *183*, 380–389. [CrossRef]
33. BESS Dataset. 2026. Available online: <https://github.com/yueshangmotian/battery-datasets.git> (accessed on 15 September 2026).
34. Bekaroo, G.; Santokhee, A. Power consumption of the Raspberry Pi: A comparative analysis. In Proceedings of the 2016 IEEE International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies (EmergiTech), Balaclava, Mauritius, 3–6 August 2016; pp. 361–366. [CrossRef]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.