Scientific African 23 (2024) e02008

Contents lists available at ScienceDirect

Scientific African

journal homepage: www.elsevier.com/locate/sciaf

Energy-efficient algorithms for lossless data compression schemes in wireless sensor networks

Lucia K. Ketshabetswe, Adamu Murtala Zungeru *, Caspar K. Lebekwe, Bokani Mtengi

Department of Electrical, Computer and Telecommunications, Faculty of Engineering & Technology, Botswana International University of Science and Technology, Botswana

ARTICLE INFO

Editor: DR B Gyampoh

Keywords:
Wireless sensor networks
Data compression
Compression ratio
Decompression
Energy efficiency

ABSTRACT

Wireless sensor networks *(WSNs)* are reliant on limited power resources, primarily provided by small batteries within sensor nodes. Inefficient energy management within these networks can lead to premature battery depletion during data transmission between sensor nodes, significantly impacting network longevity. Data compression emerges as a viable strategy to mitigate energy consumption by reducing data size before transmission and employing various compression and decompression techniques. This work presents a comparative analysis of data compression algorithms tailored for *WSNs*. It studies and enhances two adaptive lossless data compression techniques, namely 'Adaptive Lossless Data Compression' *(ALDC)* and 'Fast and Efficient, Lossless Adaptive Compression System' *(FELACS)*, as means to effectively manage energy consumption in wireless sensor networks. *ALDC* and *FELACS* algorithms encode differences between consecutive data readings, thereby reducing the number of bits required for encoding. *ALDC* employs Huffman coding, while *FELACS* leverages the Golomb-Rice coding method. Encoding data samples by using three Huffman tables interchangeably as an enhancement of the *ALDC* algorithm, resulted in an improvement in energy saving from 73 % to 77 %. Analysis of *FELACS* unveiled the impact of natural phenomena-induced anomalies on measured data, identified as outliers. The outliers disrupt data patterns and ranges, subsequently altering the optimal coding parameters for data samples, resulting in encoding and decoding errors. This study proposes a robust method for identifying and replacing outliers within sensor data, significantly enhancing compression performance. A reduction of variations in dataset patterns facilitated more accurate sampling and encoding of data. Consequently, fewer bits are required to encode data samples, rendering the algorithm energy-efficient and suitable for applications demanding error-free data recovery or meticulous error analysis. The proposed method was successfully applied to the modified *ALDC* algorithm, exhibiting efficient performance. An optimum block size of sampled data was discovered for Fishnet relative humidity deployment ensuring efficient transmission of environmental data real-world sensor network deployments like Fishnet, Lucerne, and Le Genepi. These findings underscore the potential for significant energy savings and improved data accuracy through adaptive lossless data compression techniques, making them valuable assets for applications with stringent energy constraints or demanding data integrity.

* Corresponding author.
E-mail address: zungerum@biust.ac.bw (A.M. Zungeru).

https://doi.org/10.1016/j.sciaf.2023.e02008
Received 25 September 2023; Received in revised form 17 November 2023; Accepted 29 November 2023
Available online 30 November 2023

Nomenclature

ADC	Analogue to Digital Converter.
ALDC	An Adaptive Lossless Data Compression Scheme.
ALEC	Adaptive Lossless Entropy Compression.
AREDaCoT	Adaptive Rate Energy-saving Data Collecting Technique
BLE	Low Power Bluetooth
CoC	Computational Complexity.
CR	Compression Ratio.
CS	Compressed/Compressive Sensing.
CT	Compression Technique.
DA	Data Aggregation.
DC	Data Compression.
DSC	Distributed Source Coding.
DSM	Distributed Source Modelling.
DTC	Distributed Transform Coding.
DWT	Distributed Wavelet Transform.
ETDTR	Energy efficient Two-layer Data Transmission Reduction
FELACS	Fast and Efficient Lossless Adaptive Compression Scheme.
iid	independently and identically distributed.
ID	Identification.
IoP	Internet of People.
IoT	Internet of Things.
IP	Internet Protocol.
JPEG	Joint Photographic Experts Group.
LEC	Lossless Entropy Compression.
LED	Local Emergency Detection.
LoRa	Long Range.
LZW	Lempel Ziv.
MATLAB	Matrix Laboratory.
MP3	MPEG Audio Layer 3.
MPEG	Moving Pictures Expert Group.
MSPT	Minimum Spanning Tree Projection.
NB_IoT	Narrow Band Internet of Things.
PSNs	Periodic Sensor Networks.
QoI	Quality of Information.
QoS	Quality of Service.
RA	Routing Algorithm.
RIP	Restricted Isometry Property.
RSS	Received Signal Strength.
SNR	Signal to Noise Ratio.
SR	Sampling Ratio.
S-LZW	LZW foe Sensor Nodes.
TMT	Two Modal Transmission.
TTL	Time to Live.
WAN	Wide Area Network.
WBSNs	Wireless Body Sensor Networks.
WSN	Wireless Sensor Network.
WSS	Wireless Sensor System.

Introduction

Continuous monitoring in wireless sensor network deployments facilitates the collection of vast datasets, serving diverse purposes in wireless sensor systems. These applications span ecological surveillance, healthcare monitoring, industrial operations, seismic activity analysis, structural integrity assessments, and more [1–5]. Wireless sensor networks (*WSNs*) comprise mobile sensors and wireless technology-enabled networks. These sensor nodes autonomously self-organize in a randomized manner, collectively monitoring the deployed field. They employ radio communication to capture and transmit event information to a central point or destination. Each node incorporates essential components: a battery for power, a sensor unit equipped with analog-to-digital converters and various sensors for data collection, a processing unit housing a microcontroller and memory, and a communications unit responsible

for data exchange within the network. Advancements in technology have led to the miniaturization of sensor nodes, imposing constraints on memory, bandwidth, battery capacity, and data processing capabilities.

Utilizing short-range, low-power wireless communication technologies such as Bluetooth, Wireless Hart, and Zigbee fosters connectivity and communication within *WSNs*. Notably, the longevity of these networks, hinges on the capacity of the sensor node's battery. Given the compact size of these batteries, strategic power management becomes essential for extending the network's operational lifespan [2]. Fig. 1*(a and b)* depict common node and network architectures employed in wireless sensor networks.

Effective techniques to manage power in wireless sensor networks (*WSNs*) with the intention to sustain or extend their lifespan, should address the different ways in which energy is consumed in these networks. Energy consumption in *WSNs* is primarily attributed to data sensing, processing, and transmission activities. Notably, data transmission, both sending and receiving, incurs more substantial energy costs than data processing [6]. Addressing potential data conflicts during sensing is essential to mitigate data loss. Considering the impracticality of recharging or replacing batteries in numerous and often hard-to-reach sensor nodes, effective energy monitoring and management in *WSNs* assume paramount importance. This research endeavors to confront the pressing challenge of energy consumption, which jeopardizes the longevity of *WSNs*. The core objective is to minimize data size before its transmission across the network. To this end, we explore and employ efficient data transmission methods, focusing on adaptive lossless data compression techniques. The need to develop adaptive lossless compression schemes was motivated by the ability of these schemes to adjust their compression methods based on the characteristics of the data. The original content of data is preserved while the size of data is reduced. They are useful in real time applications, where bandwidth is limited, processing power and memory are constrained and where errors can be recovered in data transmission. These schemes ensure quick and efficient transmission of data. They lead to savings in bandwidth and costs of data storage and efficient use of memory and processing power. Investigating algorithms helps to discover efficient methods that save time and energy as well as gain insight in problem solving and optimization. Algorithms help to effectively organize and manipulate data, making it easy to conduct complex simulations. The contributions of this study encompass the following key aspects:

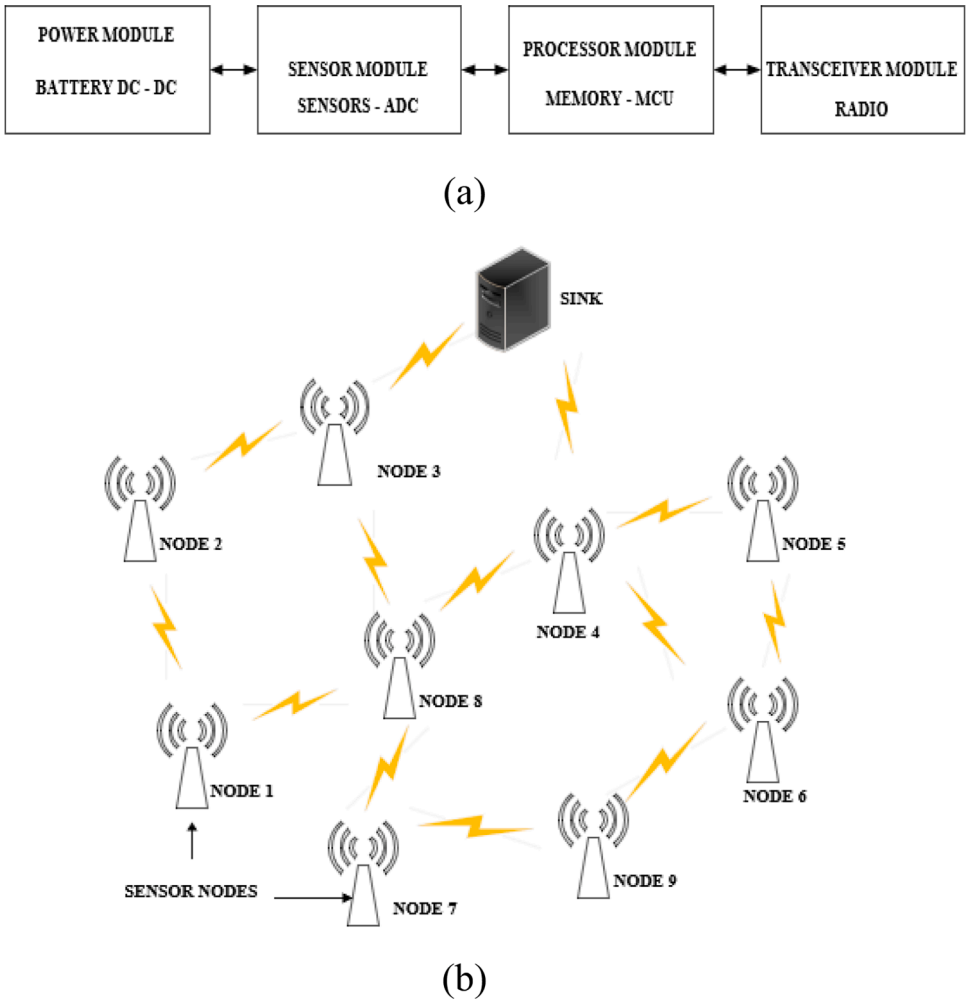

Fig. 1. (a) A wireless sensor node structure. (b) A wireless sensor network.

- Proposal of a novel method for detecting and replacing outliers within *WSN* data sets, specifically designed for *WSN* data applications.
- Study and enhancement of 'An Adaptive Lossless Data Compression Scheme' (*ALDC)* [2] and 'Fast and Efficient Lossless Adaptive Compression Scheme' *(FELACS)* [3].
- A comparative analysis of existing data compression algorithms tailored for wireless sensor networks.
- Discovery of the optimum block size of a dataset to ensure and provide efficient transmission of environmental data that is collected from diverse deployments for *WSN* applications.

The subsequent sections outline the organization of the remaining paper: *Section II* provides a general overview of data compression algorithms and conducts a comparative analysis of select algorithms. Furthermore, it delves into the detailed examination and simulation of the *ALDC* and *FELACS* algorithms using *MATLAB*. *Section III* focuses on algorithm modification and enhancement, presenting a novel algorithm and discussing its intricacies. *Section IV* offers an evaluation of the experimental results pertaining to all investigated and modified data compression algorithms. Finally, *Section V* provides a summary and discussion of the research's conclusions. This research underscores the critical importance of energy-efficient data management in *WSNs,* aiming to extend their operational lifespan and enhance their overall effectiveness in various applications.

Data compression techniques

Data compression stands out as an effective energy-saving strategy, demonstrating its efficiency through the reduction of data size before transmission across the network. Importantly, this compression process should achieve data size reduction while optimizing resource utilization. Furthermore, it should account for user-specific characteristics, considering the user's ability to utilize the data effectively [7]. In addition to data compression, several other energy-saving techniques are integral to wireless sensor networks. These include data aggregation and the optimization of data routing. Data packets can be aggregated using various routing methods, capitalizing on the inherent characteristics and statistics of data sets from sensor nodes, such as maximum, minimum, and average values. Prior to transmission to the destination node, these characteristics are extracted and manipulated. Effective utilization of data aggregation necessitates the seamless integration of routing methods and data compression strategies [6,8]. Fig. 2 provides a visual representation of the interplay among these techniques. This research underscores the significance of considering various energy-saving approaches in wireless sensor networks, ultimately contributing to the network's efficiency, and prolonging its operational lifespan.

As depicted in Fig. 2*,* during the data aggregation process, sensor nodes extract and aggregate minimum, maximum, and average values from their data. This is represented by the set *DA*, situated above the routing algorithm *RA*. Through this aggregation, redundant

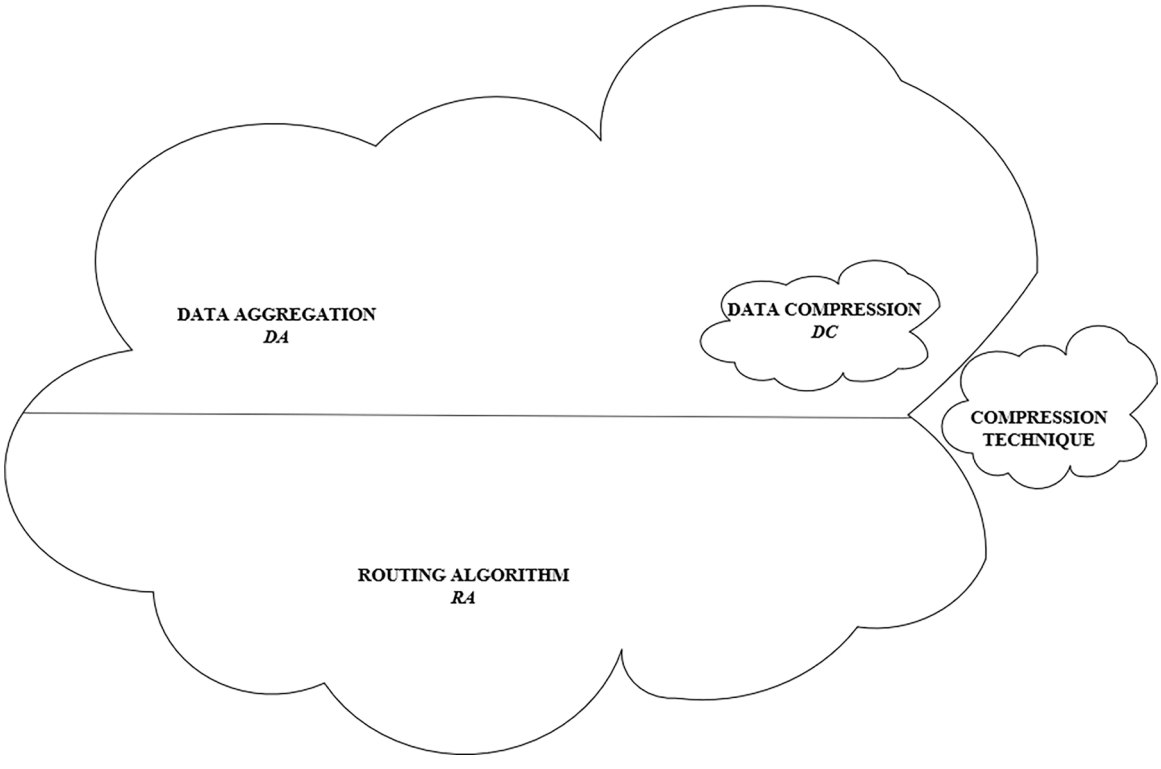

Fig. 2. Routing, aggregation, and compression – The relationship.

data is effectively eliminated, resulting in a reduced volume of data for network transmission. However, this compression of data may alter the original data arrangement, a challenge that can be addressed through the application of data compression algorithms, as denoted by set *DC* in Fig. 2. These algorithms extend the data compression techniques, dispersing compression processes throughout the network. Specifically, set *CT* autonomously compresses data at each local sensor node without relying on a routing method or a densely interconnected network [6]. Data compression involves transforming data from its original representation into a more compact form, from which the original or near-approximate data can be reconstructed. This reconstruction process is termed decompression. Data compression yields several benefits, including enhanced processor efficiency by reducing data storage requirements, faster internet download speeds, and reduced resource costs. The reduction in data size is achieved by eliminating data redundancy, making it more convenient for file transfer and storage. Additionally, data compression simplifies and economizes hardware systems. The processes of data compression and decompression are collectively known as encoding and decoding [9].

Fig. 3 provides a visual representation of the data compression process. In this illustration, raw source data is encoded into compressed codes, eliminating redundant data, and subsequently reducing the data's storage or transmission size. The reverse of this process, decompression, involves reconstructing the original source data from the data codes.

Diverse data formats can undergo encoding and decoding processes utilizing a variety of compression and decompression algorithms. The key to efficient data compression lies in the exploration and analysis of data characteristics to identify suitable patterns that enable the representation of raw data in more manageable sizes. In the design of data compression algorithms, it is crucial to consider both their effectiveness and efficiency. Data compression can be applied to various data formats, including video, text, and audio. The process involves data modelling, which aims to recognize, extract, and describe inherent data redundancies. A comparison between the data and its model is conducted, and the resulting residuals or differences are subjected to a coding phase where they are encoded in binary form. This coding can be executed using various statistical methods, such as variable-sized codes, prefix codes, Golomb codes, Huffman codes, and numerous others. These methods play a pivotal role in achieving efficient data compression across a wide range of data formats.

Classification of data compression techniques

Data compression algorithms are typically categorized into two main classes: Lossy and Lossless data compression algorithms. These categories can be further subdivided into Data Aggregation [7,10-15], Local or Regional Data Compression, and Distributed or Across-the-Board Data Compression algorithms or protocols, as outlined in Table 1. This research specifically centers on lossless data compression algorithms, which find prominent applications in compressing text, programs, or strings. In the realm of lossless compression, the original data can be faithfully reconstructed through the application of decompression algorithms, ensuring data integrity and accuracy.

Data aggregation compression techniques involve the use of aggregator nodes to selectively gather crucial data from neighboring nodes [10,11]. Within this approach, sensor nodes compute essential statistics such as maximum, minimum, and average values and subsequently apply aggregation methods [12,13]. In contrast, local data compression techniques capitalize on the temporal correlation inherent in the collected data, enabling compression at each local sensor node, either in a lossless or lossy manner. Distributed data compression techniques leverage the spatial correlation of data collected from sensor nodes in densely interconnected networks [6,7, 32].

Related works

Table 2 is a discussion of literature that relates to lossless data compression techniques dating back from 2006 to 2023. The description of the research papers has been provided as well as highlights on their merits, demerits, and unique features. Recent related works recommended by reviewer [33–44] were reviewed and some [33,34,41-43]have been included in the table for comparison with other existing works. The proposed approach is included at the bottom of the list in Table 2 for comparisons with existing related works.

A variety of data compression algorithms have been developed, tailored to the nature or structure of the data being compressed. This research investigates and compares two of these algorithms to identify a superior strategy for transmitting data with minimal

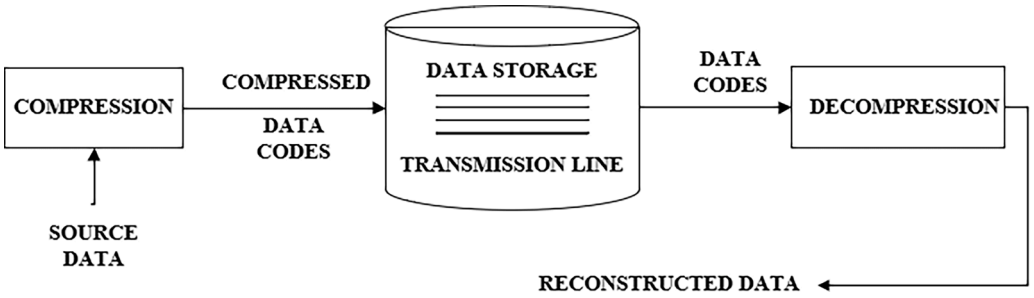

Fig. 3. Illustration of compression and decompression of data.

Table 1
Data compression techniques classification.

FEATURE	DISTRIBUTED DATA COMPRESSION			LOCAL/REGIONAL DATA COMPRESSION					DATA AGGREGATION		
	DSC. DSC-Multirate. [16].	DSM [17].	CS [18 Adaptive CS [19] Bayesian CS [20,21].	DCT KLT [22]. DWT-lifting [23]. DWT Harr [24].	TMT	LEC [25]	ALDC [2]	FELACS [3].	PEGASIS [26, 27] .	LEACH [28, 29].	PEDAP [30].
1. Compression rate.	NA	NA	Depends on data sparsity.	Depends on SNR.	@ 40 %.	@ 45–75 %	52.8 to 73.9 %	41 to 73.8 %	NA	NA	NA
2. Energy saved: minimized transmission	NA	YES	YES	YES	@ 40 %.	@ 53 %			NA	NA	NA
3. Processing Complexity	Low cost	Low cost	Low cost	Low cost	4 additions. 2 integer multiplications. 2 shifts., 2 comparisons.	12 instructions per saved bit.	Simple.	Lightweight.	High	High	High
4. Net Energy Saving	50.7 %	YES	YES	YES	@ 36 %	@ 32 %	NA	More than 55 %	NA	NA	NA
4. Structure/Type	Source Modelling.	Source Modelling.	Compressive Sensing.	Image		Text	Text.	Text	Chain based.	Cluster based.	Tree based.
5. Limitations	Precise knowledge of correlation among nodes needed [31].	Restricted to specific applications [31].	Needs suitable transformation to improve sparsity for real world data [31].	Blocks artifacts. Low scalability[7].	No spatial correlations exploitation [6].	No spatial correlations exploitation [6].	No spatial correlations exploitation [6].	No spatial correlations exploitation [31].	High overhead and the levels of scalability and robustness are low. [7].	Assumes nodes are homogenous [7].	Operation is centralized and requires prior global knowledge of sensor nodes [7].

Table 2
Comparative analysis of related works.

CITATION	Paper	Year	Description	Merits	De-merits	Uniqueness
1. [11]	Data aggregation techniques in sensor networks: A survey.	2006	A study of efficient strategies of electing aggregation nodes with the aim of reducing transmission data in WSNs.	The size of sensed data is significantly reduced before it is sent to the sink.	Data that has been compressed through data aggregation techniques is typically irretrievable.	Compares different algorithms and highlights the trade-offs between latency, lifetime of the network and accuracy of data.
2. [10]	Data compression techniques in wireless sensor networks.	2012	Explores and compares string based, image based, distributed source coding, compressed sensing, and data aggregation techniques.	A thorough examination of techniques for data compression. Lossless data compression provided by string based, *CS, DSC* so that data can be recovered at the sink.	Image based techniques may experience minor losses of sensed data. Data that has been compressed through data aggregation techniques is typically irretrievable.	A comprehensive study of data compression techniques, showing classification and comparisons.
3. [2]	An adaptive lossless data compression scheme for wireless sensor networks.	2012	A data compression algorithm that is lossless and can acclimatize to alterations in the properties of the source data, compressing blocks of data by employing two code options using Huffman coding.	Significantly reduce the size of data by compressing the differences between data samples, instead of compressing the actual samples.	Does not leverage spatial correlations within the data.	A straightforward and lightweight compression technique, appropriate for real-time applications and communications that tolerate delay.
4. [6]	Practical data compression in wireless sensor networks: A survey.	2012	Provides a thorough review of data compression protocols in *WSNs.* Classifies and defines compression schemes and unveil what real-world *WSN* data compression is supposed to be. Comparison of the performance, limitation and well-suited deployments of the schemes is presented.	Presents typical limitations in *WSN* that need to be considered when designing data compression algorithms as well as distinct criteria associated with the design tailored for specific real-world application.	Further exploration of the efficiency of data compression algorithms for specific *WSN* applications is still needed.	Review and analysis of data compression approaches was based on the requirements observed from practical *WSN* applications.
5. [7]	Compression in wireless sensor networks: A survey and Comparative evaluation.	2013	Discovered that the benefits of data aggregation are determined by the distance among the sources aggregated data, contrasting the distance between the sink and the sources, and considering the volume of summarized data in relation to original data.	Prompted consideration of aggregation structures and exploration of optimal methods of integrating data.	Data that has been compressed through data aggregation techniques is typically irretrievable.	Presents a holistic view of a survey of literature on data compression approaches and frameworks.
6. [3]	Fast and efficient lossless adaptive compression scheme for wireless sensor networks.	2015	A fast data compression algorithm that is lossless and can acclimatize to alterations in the properties of the source data, compressing blocks of data by employing Golomb-Rice coding.	The algorithm minimizes network load resulting in fewer collisions and repeated communications. It is well suited for applications that are highly rated, where sensed data is of high fidelity.	Does not leverage spatial correlations within the data. Does not consider appearance of outliers in sensed data.	A fast algorithm that demonstrates high robustness to loss of data packets.
7. [33]	Data Transmission protocol for reducing the energy consumption in wireless sensor networks.	2018	Proposes a data transmission algorithm that minimizes energy consumption in wireless sensor networks by dividing the network into distinct periods. It is applied locally within sensor nodes. It uses Modified *k*-Nearest Neighbour techniques	Improved accuracy in data that is received at the sink node.	Does not leverage spatial correlations within the data.	Partitioning of the network into distinct periods.

(*continued on next page*)

Table 2 (*continued*)

CITATION	Paper	Year	Description	Merits	De-merits	Uniqueness
			for enhancement of *WSNs* lifetime.			
8. [34]	Energy-efficient two-layer data transmission reduction protocol in periodic sensor networks of IoTs	2020	Proposed a two-layer data reduction scheme that uses less energy by leveraging on temporal and spatial redundancy within sensed data in periodic sensor networks (*PSN)*. It implements clustering in a bottom-up hierarchy at cluster heads to minimize the amount of data reaching aggregator nodes before being sent to the base station.	Rapid grouping method selects representation of entire data to send to the base station and minimizes transmission load.	There is need to adjust the period size dynamically so that the data reduction feature of the *ETDTR* protocol is optimized. There is need to implement a scheduling approach for sensor nodes with similar readings.	Leverages both spatial and temporal correlations of sensed data.
9. [41]	Adaptive rate energy-saving data collecting technique (AREDaCoT) for health monitoring in wireless body sensor networks	2020	A method that intends to minimize the volume of sensed data in wireless body sensor networks (*WBSNs*) and additionally allow sensor nodes to alter their frequency of sampling considering the changing nature of the level of the danger faced by a patient within an observed field.	Decrease in pointless data that come because of unwarranted sampling results in the preservation of sensor node energy.	Demonstration of the performance approach looking at energy consumption, sustaining integrity of data and improving decision can be attained by gathering data using several different sensors.	Provides Local Emergency Detection (LED) and evaluates the level of severity of danger to the patient.
10. [31]	Data Compression Algorithms for Wireless Sensor Networks: A Review and Comparison.	2021	A survey and comparison of data compression techniques. Modification of the use of Huffman tables used for coding data in [2] to further reduce energy consumption in wireless sensor networks.	Compression performance was improved further. The approach can be used to compress real word data sets.	Does not leverage spatial correlations within the data.	A straightforward and lightweight compression technique, appropriate for real-time applications and communications that tolerate delay.
11. [42]	Distributed Energy efficient Data Reduction Approach based on Prediction and Compression to reduce data transmission in IoT networks.	2022	Presents an approach that relies on prediction and data reduction in IoT networks.	It partitions time into periods and predicts data for subsequent periods employing a prediction approach to determine whether to send the data for the current period.	Not purely for *WSNs*. Strategies of predicting missing data can be explored to increase data accuracy.	The ability to send data only when it is necessary contributes to minimized energy consumption.
12. [43]	A distributed prediction compression-based mechanism for energy saving in IoT networks.	2023	Proposes a distributed prediction compression-based mechanism (DiPCoM) for energy saving in IoT networks. It employs ARIMA (Autoregressive Integrated Moving Average) prediction technique for each period to forecast data for the next and determine whether the data at hand should be transmitted to the gateway, thereby eliminating redundancy.	Algorithm works in periods to determine the necessity to send data. It uses less energy and is more accurate than other algorithms compared against it.	Size of data between the gateway and cloud can be minimized by application of Machine and Deep learning approaches. This will also improve the quality of data.	It integrates various data transmission compression strategies
13.	The Proposed Algorithm	2023	A comparative analysis of data compression algorithms tailored for *WSNS*. Also studies and enhances *ALDC* and	Improvement on energy saving because of reduction of number of bits used to encode data samples.	The proposed approach does not leverage on sparsity of measured sensor data. There is need to explore	Can be applied across different adaptive lossless data compression algorithms depending on the required or desired

(*continued on next page*)

Table 2 (*continued*)

CITATION	Paper	Year	Description	Merits	De-merits	Uniqueness
			FELACS with the intention to minimize energy consumption and extend sensor network lifetime. Proposes a robust approach for identifying and replacing outliers within sensor data significantly enhancing compression performance. Provides efficient transmission of environmental data that is collected from diverse deployments for *WSN*.	More accurate sampling and encoding of data accomplished from minimization of variations in data patterns brought about by replacing outliers. Facilitates data error detection and promotes accurate transmission of data.	methods for independently encoding outlier bits, apart from the cleaned compressed data, and subsequently integrating them with the cleaned compressed data before decompression. This will facilitate the seamless restoration of sensor network data to its original form while maintaining energy efficiency.	applications. Well suited for applications demanding error free data recovery. Ensures efficient transmission of environmental data as it can determine optimum block sizes for encoding and decoding data from WSN deployments.

energy consumption. The following section, *Section B,* delves into a detailed exploration of these techniques through coding, mathematical analysis, and simulation performed using *MATLAB*.

Study and review of lossless data compression schemes

1. Analyzing 'An Adaptive Lossless Data Compression Scheme' (ALDC) [2].

The *ALDC* algorithm, as introduced by [2], is designed with the purpose of enhancing adaptability to changes in data properties collected from sensors, ultimately contributing to the prolonged lifetime of sensor networks. This algorithm exhibits versatility by accommodating various code options simultaneously and supporting diverse types of data. It employs Huffman coding for data sample encoding. *ALDC* operates through two Adaptive Lossless Entropy Compression *(ALEC)* options, each employing distinct sets of Huffman tables. These options are known as 2-Huffman Table *ALEC* and 3-Huffman *ALEC*. Utilizing entropy coding, *ALDC* achieves notable energy savings, resulting in a higher compression ratio. To establish the Huffman tables, practical datasets with diverse data property measurements were employed. Notably, the *ALDC* algorithm under investigation adopts the Decision Region method from the two *ALEC* code options, as opposed to the Brute Force method. This choice is motivated by the former's lower memory requirements, reduced complexity, and decreased energy consumption. The *ALDC* algorithm also incorporates prediction coding, a technique that efficiently extracts critical information from data samples. This prediction method relies on the difference between consecutive environmental data samples within a block, simplifying the algorithm and data processing. The anticipated data sample is determined based on the sample observed at the end of the previous observation.

These features collectively contribute to the effectiveness and energy efficiency of the *ALDC* algorithm in handling sensor data with changing properties, ultimately enhancing the sustainability of sensor networks.

$$\text{Anticipated sample } \widehat{n}_i \;=\; \text{Previous sample } n_i - -1 \tag{1}$$

The successive sample differences (*diffs)*are obtained from the differences between the Present Reading n_i and the Anticipated sample $n_i - 1$, which is represented by,

$$diffs = n_i - n_i - 1, \tag{2}$$

To be able to obtain the difference at the start *diff0,* it is assumed that,

$$diff0 = 2^{DYRange-1} \tag{3}$$

The dynamic range (*DYRange)* of symbols is determined by the shift from maximum to minimum values within the incoming data, signifying the variations in symbol values that serve as input to the encoder.

In the Decision Region method, the sum of all the differences in sample values within the data, for a block of a samples, denoted as *DiffSum,* is computed. This sum is then compared against three predefined decision levels to identify the boundary region, denoted as H. The selection of the most suitable Huffman *ALEC* code option is based on this boundary region determination.

The calculation of the sum of differences is expressed as follows:

$$DiffSum = \sum_{i=1}^{a} |diffs| \tag{4}$$

The class region H that falls within the decision boundary can be determined from the following:

1. 2-Huffman Table *ALEC* code option is selected for region $H \leq 3a$.
2. 3-Huffman Table *ALEC* code option is selected for region $3a < H \leq 21a$.

3. 2-Huffman Table *ALEC* code option is selected for region $12a < H$.

The selected class regions are computed to determine the best code option to encode differences. Its associated identifier *ID* will be generated as either 0 or 1 for either 2-Huffman Table *ALEC* or 3-Huffman Table *ALEC*. Pseudocode 3 is used to generate the encoded bitstream.

The encoded output is made up of

I. The code identifier *ID*, indicating the Table A*LEC* code option used (either 2-Huffman Table *ALEC* or 3-Huffman Table *ALEC*).
II. A table identifier *ID* indicating the Huffman coding Table used (either Tables 1–3) to encode the sample blocks with the selected code option.
III. The Huffman group code for the difference sample.
IV. The binary representation of the difference sample. This representation is not needed when the differences value is zero.

This information in the output bitstream is needed by the encoder so that it can recreate the raw bitstream out of the code. *Pseudocode I* is used to calculate the differences *diffs between* the consecutive readings. *Pseudocode II* calculates the sum of absolute values of differences, determines the boundary region and generates the code option *ID*.

Pseudocode I: Function **diffs (**data, DYRange, diff0, **diff)**
//data >> the stream of raw data.
//DYRange >> symbolic diversity of source data.
Calculating the first difference.
//diff0 >> the initial residue
diff(n) → source(n) – xo.
//Calculate the differences for entire bitstream.
OUTPUT diff.

Pseudocode II: Function **Adaptive** (F, K, **ID)**.
//Function Adaptive sets the decision region limit.
//Calculate the sum of absolute values for differences F.
//*K* >> the residue length.
Call Function Residual (); Calculate K.
//Set the F region.
IF $F< 3\,K$
//ASSIGN code ID to use of Huffman Tables A and B
ID → ‘0′.
ELSE
IF $3K< F <= 12\,K$
//ASSIGN code ID to use of Huffman tables A, B, and C
ID → ‘1′.
CLOSE IF
OUTPUT ID.

Analyzing ‘Fast and efficient lossless adaptive compression scheme’ (FELACS) [3]

Another noteworthy lossless adaptive compression technique, introduced by [3], operates on sample blocks and is known as the ’Fast and Efficient Lossless Adaptive Compression Scheme (*FELACS)*.’ *FELACS* employs Golomb-Rice coding for both data encoding and decoding processes and is designed to prioritize lightweight implementation with reduced complexity. One of *FELACS* ’s key strengths lies in its resilience against data packet loss. It achieves this by independently compressing source information in discrete blocks. The utilization of Golomb-Rice coding contributes to its speed, enabling faster data encoding. These codes are derived from a set of prefix codes with exponential growth, a concept originally devised by Golomb and later independently rediscovered by Rice, hence the name ’Golomb-Rice’ [45]. *FELACS* stands out as an efficient and lightweight solution for lossless data compression, making it

Table 3
ALDC Selection of Tables for encoding.

Sample	Table	No. of bits
10	3	2
0	1	2
0	1	2
−1	1	2
1	1	2
0	1	2
0	1	2
6	3	2

particularly suitable for applications where complexity and speed are critical considerations.

For a parameter ∂ that is a non-negative integer, and a known value s, the Golomb codes encode the non-negative integer ∂ in unary as $\lfloor \partial/s \rfloor$. The split part is encoded as the modulus of ∂ and s. The Golomb-Rice codes utilized by *FELACS* are particularly well-suited for sources that exhibit geometric distributions, where they yield optimal compression results. In the case of *FELACS*, to maintain minimal overhead costs for the *ID* sequence, the options for Golomb-Rice codes are restricted to a set of 8 codes, each represented using just 3 bits. This limitation contributes to the efficiency and effectiveness of *FELACS* in handling various data sources with geometric characteristics.

This sets s to 2^t for a non-negative integer t. When t is known, the term that precedes the integer's code ∂ is made up of representation in unary of $\lfloor \partial/2^t \rfloor$ and translates to $\lfloor \partial/2^t \rfloor$ number of zeros (0) with a one (1) at the end.

The word ending suffix is made up of binary representation of ∂, particularly its t least significant bits, meaning that the modulus of ∂ and 2^t uses t bits binary representation of ∂. A block of samples is encoded by sending an identity (*ID*) bit sequence along with the encoded bit stream as an indication to the decoder, the option of the code that was used to encode the samples. This *ID* pattern represents t in binary representation form, using $\lceil \log_2 R \rceil$, where R is a sequence of R-bits symbols to be encoded. Rice coding accommodates and handles changes in source data statistics and as such it can allow different code options to be used for every sample block. The Golomb-Rice code length of t is denoted by,

$$l_t = \lfloor \partial/2^t \rfloor + 1 + t \tag{5}$$

This code length l_t was used to compute the code length minima and t, the associated optimum coding parameter, that were eventually used to draw a table of code options for selected non-negative integers. The sum of samples $\partial_i = \ \delta_1 \ \delta_2 \ldots \partial_q$ in a block of Q samples is calculated as

$$Q_{sum} = \sum_{i=1}^{Q} \partial_i \tag{6}$$

In the *FELACS* approach, the computed sum is compared to pre-calculated decision regions to determine the optimal parameter 't.' Notably, in *FELACS,* instead of relying on tabulated information, bit shift operations are employed, enhancing the algorithm's speed. This efficiency is facilitated by the presence of powers of 2 that multiply the Q samples, aligning with the decision region boundaries. *FELACS* incorporates predictive coding to ensure that the original data sources exhibit geometric distributions. It employs a linear prediction model that operates on the differences between successive sampled data points. This simplifies the processing, making it well-suited for sensor nodes with limited computational capacity.

The anticipated sample is defined the same way as realized in *ALDC* by Eq. (1). In the same way, the successive sample differences *diffs* are as in Eq. (2). Unlike in *ALDC*, to obtain the first difference value, it is assumed that the initial difference is determined by,

$$diff0 = 0 \tag{7}$$

In the development of the FELACS algorithm, it was presumed that each sample value falls within the range [0, 2^N-1], where '*N*'' corresponds to the resolution of the analog-to-digital converter (*ADC)* utilized by the sensor node. To adapt Golomb-Rice coding for implementation within wireless sensor networks (*WSNs*), certain modifications were made to align the coding scheme with the specific requirements and constraints of *WSNs*.

The encoding procedure

The coding options in *FELACS* are deliberately limited to eight (*8*) Golomb-Rice family codes, where *'t'* takes on values from *0* to *7*. This constraint ensures that the bit sequence required for the *ID* remains at a minimum of *3* bits, calculated as $\lceil \log_2 8 \rceil$. This design choice effectively enables compression for data sets with source entropy ranging from *1.5* to *9.5*. Importantly, data blocks are independently encoded to enhance data packet robustness and mitigate packet loss. To transform the differences of the samples, computed using Eq. (2), from a Laplacian distribution to a geometric distribution (∂_i) a Rice mapping function is employed. The transformation is defined as follows:

$$\partial_i = \begin{cases} 2\ diffs & 0 \le \ \ diffs \le \theta \\ 2|diffs| \ - 1 & -\theta \ \le diffs < 0 \ [45] \\ \theta + \ |diffs| & otherwise, \end{cases} \tag{8}$$

Here, θ is determined as the minimum value between the anticipated sample and 2^R-1 minus the anticipated sample. The resulting samples form a sequence of mapped differences. The optimal code value *'t'* is determined by evaluating the sum of these mapped differences, excluding the first sample, against predefined decision levels.

The first sample is directly encoded in its natural R-bits binary form, serving as the reference sample. Subsequently, the remaining samples are encoded using both Unary and Split-part representations. The resulting codewords are concatenated. The codeword for the reference sample is appended to the resultant codeword of the remaining samples, forming the encoded bitstream. The optimal code value *'t'* is converted to binary and appended as the first *3* bits to the encoded bitstream. This comprehensive encoding process in *FELACS* ensures efficient data compression while accommodating a range of data properties and facilitating robust data packet transmission.

The decoding procedure

To decode the incoming data stream, the decoder requires information about the sample number, the *ADC* resolution, and the *3*-bit sequence number of the *ID*. The decoding process unfolds as follows: a) The first 3 bits from the bitstream are extracted and converted into decimal form, representing the optimal coding option *'t.'* b) The subsequent *R*-bits represent the reference sample and are converted into decimal form. c) The remaining samples are decoded using the Golomb-Rice coding method, following these steps: i. To determine the sample that follows the reference sample, the decoder counts the number of consecutive zeros that precede a one (*1*). This count is converted into its binary representation, and the subsequent *'t'* bits are appended to this binary representation. ii. The resulting binary number is then converted into decimal form. iii. The above procedure is repeated for the remaining samples until the entire data packet is fully decoded.

The mapped differences obtained through this decoding process are subsequently reverse mapped into sample differences using the inverse of Eq. (8), leading to the formulation of Eq. (9).

This decoding method ensures the accurate reconstruction of the original data samples, effectively reversing the compression process.

$$d_i = \begin{cases} \left[\frac{\partial_i}{2}\right] & 0 \leq \partial_i \leq \theta \\ \frac{|\partial_i|}{2} + 1 & -\theta \leq \partial_i < 0 \\ \theta - |\partial_i| & otherwise \end{cases} \tag{9}$$

The reverse mapping function yields the differences of the original data stream. To generate the original samples, these differences are added to the previously read samples using (Eqs. 2 and 7). These resultant values are then appended to the reference decimal, forming the decoded stream, which effectively represents the reconstructed original bitstream. The entire encoding and decoding procedures are systematically illustrated in the accompanying flowchart, as depicted in Fig. 4. This flowchart serves as a visual

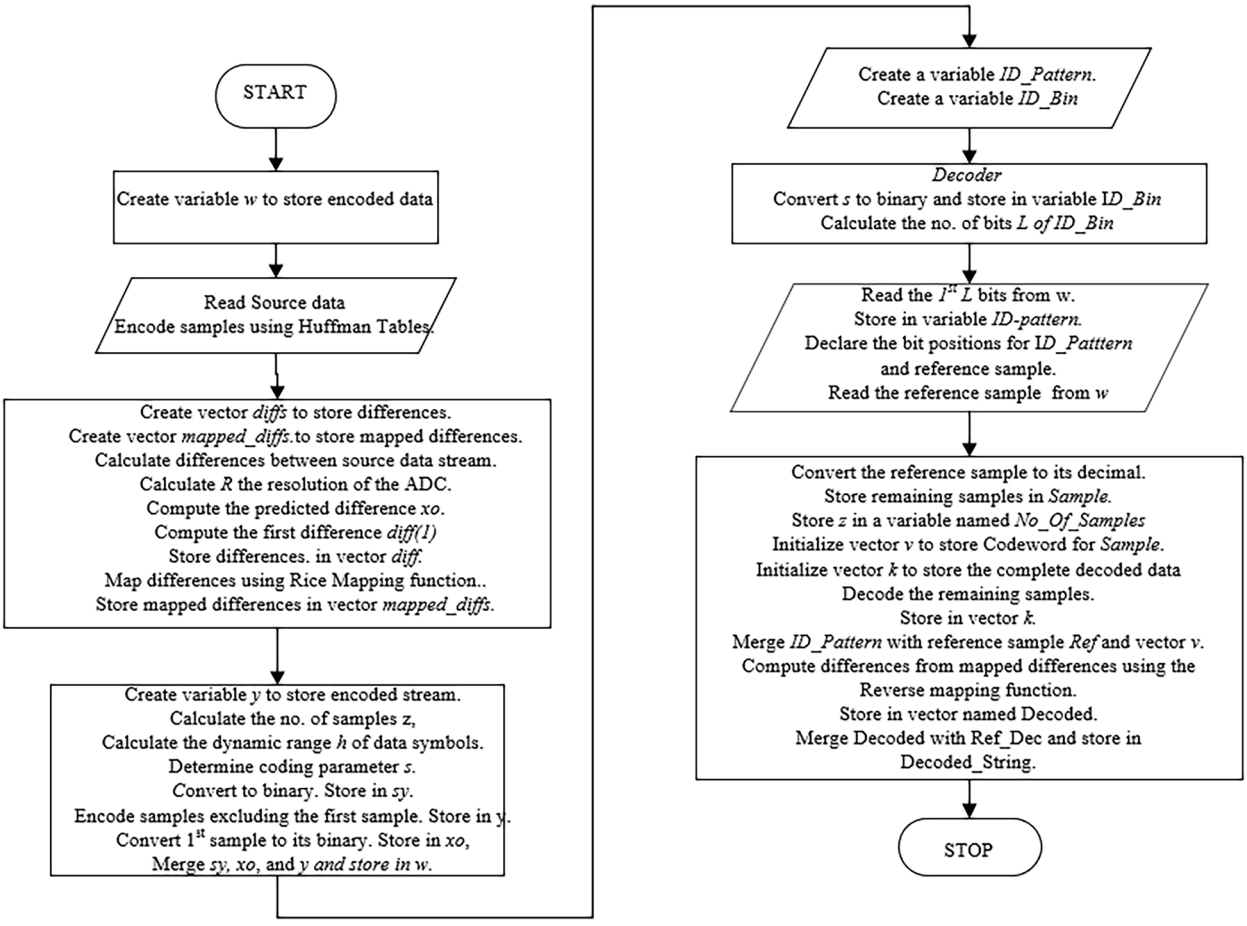

Fig. 4. FELACS flowchart – Encoding and decoding.

representation of the step-by-step processes involved in both encoding and decoding, ensuring the accuracy and integrity of the data throughout the compression and decompression cycles.

The proposed algorithm

As emphasized by the authors in [32], the hallmark of an efficient lossless compression technique lies in its ability to utilize an average number of bits that align with the information entropy of the source to encode the source's output. Information entropy represents the inherent uncertainty or unpredictability of the source's data. The difference between the information entropy and the average length of the source's encoding is referred to as information redundancy. This metric serves as a measure of the efficiency of the encoding scheme. In essence, the smaller the information redundancy, the more efficient the code, as it minimizes the additional bits required to represent the source's data accurately. This underscores the importance of achieving high compression efficiency while preserving data integrity in lossless compression techniques.

Description of the modified ALDC scheme

This research endeavor presents an enhancement to the *ALDC* algorithm, focusing on the modification of the application of Huffman coding tables, which has led to a noteworthy reduction in energy consumption. In contrast to the previous work in [2], which utilized two code options involving three Huffman coding tables, our proposed algorithm adopts a strategy that selects shorter codes from these three tables for encoding data samples. This approach of choosing tables with shorter codes significantly diminishes the number of bits required for encoding data samples. The outcome is a more energy-efficient algorithm, which aligns with the goal of optimizing energy consumption in wireless sensor networks. Modification is made by addition of *Pseudocode III*, '*Huff_4*' which uses a modified encoder function '*My_Encoder',* in *Pseudocode IV*.

Pseudocode III, Function **Huff_4** (diff_vector, T-A, T-B, T-C, codeword_1)
//**My_Encoder()** >> function to encode.
//codeword_1 >> the output bitstream of N diff_vector.
//* indicates chain or concatenation
//Encoding a block of N diff_vector utilizing all three Huffman Tables.
CALL Function **My_Encoder()** to obtain Tables A, B and C and a block of N diff_vector and outputting ci1. ciA → ci1
// append encoded bitstream ciA to codeword_1
codeword_1 → codeword_1* ciA
OUTPUT codeword_1

Pseudocode IV: Function **My_encoder(**diff_value**, TABLE,**ci_code**)** [2]**.**
// TABLE >> Encoding Huffman table.
// bi >> number showing set of diff_value and the minimum number of bits that are required in encoding the diff_value.
// ci_code >> output bitstream.
// hi >> Huffman code to encode the set of diff_value.
// li >> integer code to encode the position of the pointer for the set of diff_value.
// * indicates chain or concatenation
// (index)| bi indicates the binary form of index over bi bits
// calculate diff_value class
IF diff_value = 0
ASSIGN bi TO 0
ELSE
ASSIGN bi TO _log2 (|diff_value|)_
CLOSE IF
// Obtain code hi from TABLE
ASSIGN hi TO TABLE [bi]
// Create ci
IF bi = 0 THEN
// li is not needed
ASSIGN ci TO hi
ELSE
// Create li
ASSIGN li TO (index)| bi
// Create ci
ASSIGN ci TO hi * li
CLOSE IF
OUTPUT ci

Table 3 demonstrates how the different samples were selected from the three Huffman tables for every sample, indicating the bit numbers which were used for encoding the samples.

Description of the proposed FELACS algorithm

Environmental data collected from sensor networks is vulnerable to the influence of natural phenomena, which can result in substantial variations in data patterns. These variations often manifest as values that fall well outside the typical range of the dataset and are commonly referred to as outliers. These outliers can have a detrimental impact on the compressibility of data, potentially leading to misrepresentations or difficulties in restoring the data to its original form.

The proposed *FELACS* algorithm addresses this issue by introducing a method for detecting and replacing outliers. It operates under the assumption that temperature readings measured at a given time exhibit uniformity and should not deviate significantly from one another, unless an environmental disturbance occurs, causing spikes or altering the data's range and uniformity. The analysis involves a stream of data, and it examines the presence of outliers at various points within the data stream, including the beginning (first data point), the end (last data point), and intermediate positions (between the first and last data points). For instance:

term = [1930 1801 1807 1806 1804 1910].

To assess whether the first term ($i = 1930$) is an outlier, we evaluate whether *10* times the absolute difference between term *1807* and term 1*801* is less than the absolute difference between term *1930* and term *1801*. If the former is less than the latter, term *1930* is deemed an outlier. In such cases, it is replaced by the sum of the subsequent term (*1801*) and a random number between -5 and *5,* which represents noise or an error that may either amplify or attenuate the term. If the former is greater, term *i* retains its value of *1930* and is not classified as an outlier.

For assessing middle terms as outliers, the algorithm resets term *i* and assigns $k = 0$. It checks for a condition where term $i + 1$, now representing the new term *i (1801),* is less than or equal to the number of terms in the stream, excluding one term, and greater than *1*. If the absolute value of the difference between term *1801* and the preceding term *i-1* is greater than ten times the absolute difference between term *1807* and the previous term *1801*, and if *k* is equal to *0, k* takes on the value of term i *(1801)*. Term *1801* is then classified as an outlier and replaced by the median value of the terms on either side of it (term *i-1* and term $i + 1$), with the addition of a random value between -5 and *5*. If the previous term *i-1* equals *k*, term *i* retains its value of *1801,* indicating it is not an outlier. The variable *k* is subsequently reset to *0*. If *i-1* does not equal *k*, term *i* again retains its value of *1801* and is not considered an outlier.

To assess whether the last term (*1910*) is an outlier, a comprehensive evaluation is conducted across the entire dataset. It checks if the absolute difference between *1910* and its preceding term *(1804)* exceeds ten times the absolute difference between term *1804* and its previous term i-2 *(1806)*. If this condition is met, *1910* is identified as an outlier and replaced by its preceding term *(1804)* added to a random value between -5 and *5*. If the condition is not met, *1910* remains unchanged and is not classified as an outlier.

These outlier detection and replacement processes are implemented using *MATLAB* through *Pseudocode V,* ensuring the robustness and accuracy of the data.

***Pseudocode V: Function Remove_Outlier** (term, k, **Cterm**)*
//term >> a vector of terms of data.
//k >> value that resets term i when the length of the term is reduced by 1 and the terms after the first term are tested.
//Cterm >> output showing replaced outliers.
//Testing the first term.
Initialize a vector *Cterm* to store output.
FOR *i being* the entire length of term
IF *i is an outier*
IF 10 x abs*(*diff*(term i + 2 and term i + 1))* is less than abs*(*diff*(term i* and *term i + 1)).*
ASSIGN *Cterm i* TO term *i + 1* plus random number between −5 and 5.
ELSE
ASSIGN *Cterm i* TO term *i.*
END
END
RESET *k* TO 0.
//Testing middle terms.
IF *i* is less or equal to number of terms excluding *1* term AND *i* is greater than *1*.
IF abs(diffs(*term i* and *term i-1*)) is greater than10 x abs(diffs(*term i + 1*) and *term i-1*)) AND *k* is *0*
ASSIGN *k* TO term *i.*
ASSIGN *Cterm i* TO round(median(*term i-1* and *term i + 1*) plus a random number between −5 and 5).
ELSEIF
term i-1 is *k*
ASSIGN *Cterm i* TO *term i.*
RESET *k*
ELSE
ASSIGN *Cterm i* TO *term i.*
END
END.
//Testing the last term.
IF *i* is all terms.
IF abs(diffs(*term i* and *term i-1))* is greater than 10 x abs(diffs(*term i-1* and *term i-2*))

ASSIGN *Cterm i* TO *term i-1* plus a random number between −5 and 5.
ELSE
ASSIGN *Cterm i* TO term *i*.
END
END
END
END.

Fig. 5 provides a visual representation of the workflow employed to implement the modified algorithm. As depicted in the diagram, the process begins by subjecting the data term to outlier detection. Any identified outliers are subsequently replaced using the method outlined, resulting in the formation of *Cterm*. Encoding operations are then carried out on various blocks of *Cterm* to ascertain the optimal coding parameters for each block size. Finally, the decoding phase is executed to restore the cleaned original data, which has been purged of outliers. This workflow encapsulates the key steps involved in the algorithm's operation, from outlier identification and replacement to encoding and, ultimately, data restoration.

In *FELACS*, the Golomb-Rice coding method is employed to encode data samples. During this coding process, the unary code is determined based on the mapped differences and the optimal coding parameter. Specifically, if the optimal coding parameter is zero, the unary code is equal to the value of the mapped differences. This condition holds true even when the unary code value itself is zero. Notably, the original algorithm did not account for these conditions in the encoding and decoding of data samples. To rectify this oversight, the proposed algorithm introduces enhancements through *Pseudocode V1*, ensuring that these specific conditions are appropriately addressed during the encoding and decoding phases.

Pseudocode V1: (*mapped_diffs, Option_k,* ***unaryCoding(u,m,y)***).
//u is the codeword for unary representation in Golomb-Rice coding method.
//m is the position of u, where u is equal to 0.
//y is the number of bits for values of m.
CREATE zero vectors to store values of u, m and y.
// Compute unary representation code
FOR *i* number of *mapped_diffs*
IF $u == 0$ THEN
SET *m(i)* TO *i*
ELSE
SET *m(i)* TO *0*
ENDIF

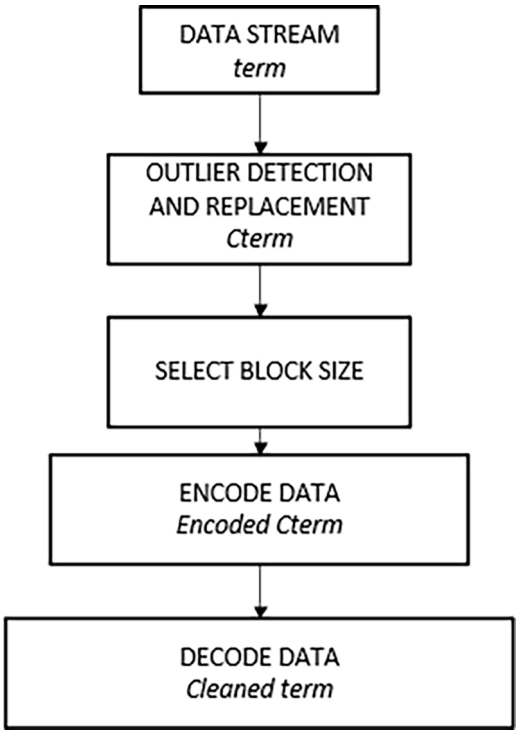

Fig. 5. Modified *FELACS* CODEC workflow.

SET *y* TO *mapped_diffs*
END
RETURN

Fig. 6 illustrates the encoding stage of the modified algorithm.

At the decoding stage, the algorithm also checks for these conditions and addresses them as illustrated by the decoder flow chart in Fig. 7.

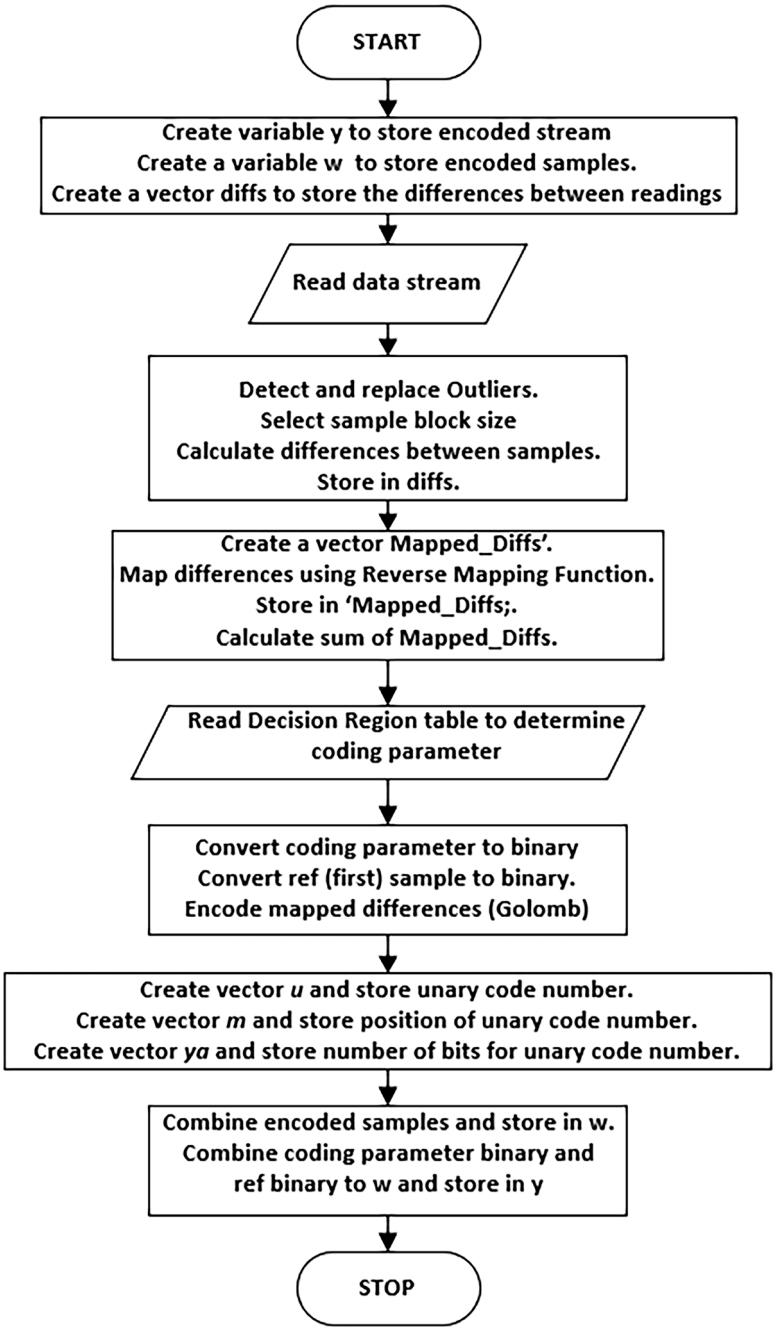

Fig. 6. Modified FELACS encoder flowchart.

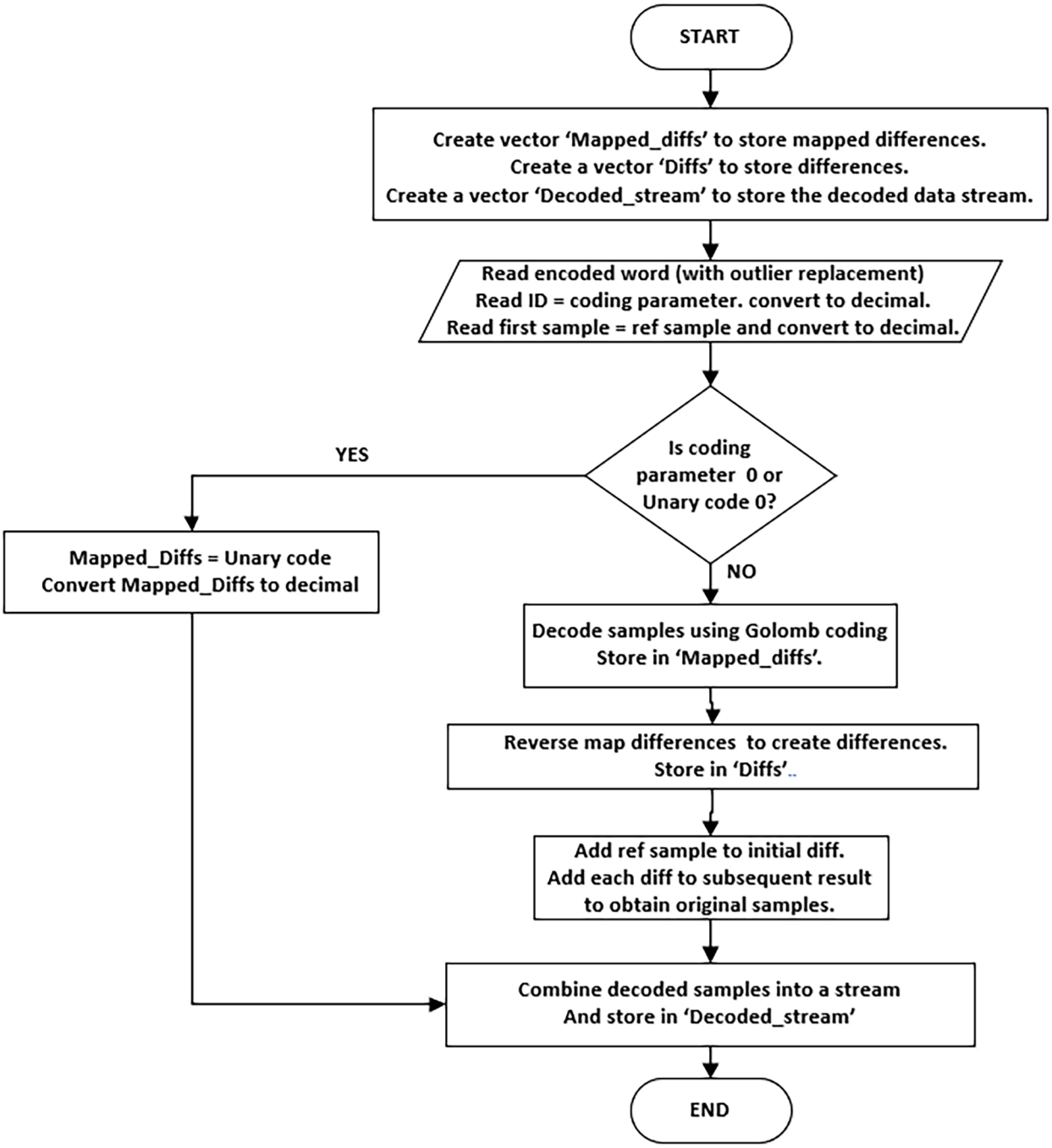

Fig. 7. Modified FELACS decoder flowchart.

Table 4
Main properties of datasets.

Dataset	Fishnet_101	Luce_84	LG_20
FELACS			
Date	09/08/2007 to 31/08/2007.	23/11/2006 to 17/12/2006.	04/09/2007 to 03/10/2007.
Number of Samples	12,652	64,913.	21,523.
Proposed			
Date	22/11/2006 to 09/05/2007.	06/08/2007 to 02/09/2007.	27/08/2007 to 31/10/2007.
Number of Samples.	14,721.	447,772.	43,059.

Evaluation and experimental results

Datasets

The datasets employed in this research were sourced from three distinct deployments: HES-SO *Fishnet, Le Genepi*, and *LUCE,* which form part of the Sensorscope practical environmental monitoring wireless sensor network. These deployments feature TinyNode nodes equipped with an MSP430 microcontroller from Texas Instruments, an *XE1205* radio transmitter from Xemics, and an *SHT75* sensor module manufactured by Sensirion. The sensors are integrated with a *12*-bit analogue-to-digital converter for relative humidity measurements and a *14*-bit converter for temperature measurements. For reference, Table 4 provides a summary of the key properties of these datasets along with their respective acquisition dates.

The parameters under examination in this study included Relative Humidity, Ambient Temperature, and Surface Temperature. The research involved a comparative analysis of the algorithm's performance proposed in this work with that of the algorithm presented in [3]. This comparative assessment allowed for an evaluation of the effectiveness and efficiency of the proposed algorithm in relation to the existing one.

Evaluation metrics used

To gauge the efficacy of the *ALDC* algorithm, two key metrics were employed: Compression Ratio (*CR*) and Energy Saving. The parameter setting for Relative Humidity, *Fishnet_101* dataset is as shown in Table 5.

Energy Saving quantifies the amount of energy conserved by encoding the differences between data samples, as opposed to encoding the raw data. It is quantitatively expressed as follows:

$$Energy\ Saving = \frac{Energy\ saved\ through\ Compression\ E_{comp}}{Energy\ consumed\ without\ Compression\ E_{Uncomp}}\%.[46] \tag{10}$$

If we were to utilize binary representations for encoding these samples, each sample would necessitate 14 bits, as demonstrated below:

$$Energy\ Saving = \frac{Average\ number\ of\ bits\ saved\ per\ sample\ N_b}{Number\ of\ bits\ per\ sample\ of\ original\ data\ N}\%. \tag{11}$$

Energy saving can also be expressed as

$$Compression\ Ratio\ CR = \left(1 - \frac{Compressed\ data}{raw\ data}\right) x\ 100. \tag{12}$$

[2]

To evaluate the effectiveness of the proposed *ALDC* algorithm, these expressions were utilized to calculate compression ratios for varying sizes of data blocks. The results were then visualized through graphical representations to unveil the relationship between them.

Similarly, for the *FELACS* scheme, the assessment was based on three key metrics: Compression Rate, Energy Saving, and Entropy. Table 6 demonstrates the parameter settings for Relative Humidity measurements from *Fishnet_101* dataset.

The Compression Rate signifies the rate at which data is compressed and is computed as the ratio of the number of compressed data bits to the number of original data samples. This metric is expressed as follows:

$$Compression\ rate = \frac{number\ of\ bits\ of\ compressed\ data}{number\ of\ samples\ of\ the\ original\ data.} \tag{13}$$

Entropy serves as a metric for quantifying the average number of binary symbols necessary to encode the output of a data source. It characterizes the typical self-information contained within random experiments and hinges on the sequential arrangement of data samples. The *FELACS* algorithm gauges the optimal lossless compression of source data by computing information entropy for both the original data and pre-processed data. The formula for entropy is articulated as follows:

$$H = \sum_{i=1}^{R} p(x_i)\ .\ log2(p(x_i)), \tag{14}$$

Table 5
Parameter settings-modified ALDC.

PARAMETER	VALUE
Number of samples	20
Sample Block Size	$N = 1$ to 20.
ADC Resolution	DR = 14
First difference	2^{DR-1}
Decision Region	F >12N

Table 6
Parameter settings - Proposed FELACS.

Parameter	Value
Number of samples	14,720
Sample Block Size	20, 100, 500, 1000, 5000, 14,720.
ADC Resolution	15
Coding Parameter	
First difference	0
Entropy (source)	9.2781

where R represents a count of the *ADC*'s potential values x_i and $p(x_i)$ is the probabilistic mass function of x_i.

Experimental results

When the data remains unaffected by natural phenomena, the disparity between consecutive data points tends to be minimal and diminishes even further as the sampling rate of the data increases. In such cases, the *ALDC* algorithm transmits the binary representation of these differences instead of conveying the binary representation of the actual data points. This approach effectively conserves energy. To illustrate this, consider the following stream of Relative Humidity samples from the *Fishnet_101* deployment in *Example 1* below:

data = [9910, 9519, 9518, 9518, 9518, 9518, 9518, 9517, 9517, 9517, 9517, 9517, 9517, 9517, 9517, 9517, 9516, 9516, 9516, 9516].

If we were to utilize binary representations for encoding these samples, each sample would necessitate *14* bits, as demonstrated below:

Encoding Bits = ['10011010110110''10010100101111''10010100101110''10010100101110''10010100101110', ...''10010100101100'].

Consequently, we would end up with a lengthy *280*-bit binary number, demanding substantial memory space and considerable energy for transmission. Assuming the use of a *14*-bit *ADC* for encoding, the initial data sample is configured as $2^{14-1} = 2^{13} = 8192$. *ALDC*, on the other hand, encodes the disparities between consecutive differences, which can be enumerated as follows:

ALDC data differences =

[1718, −391, −1, 0, 0, 0, 0, −1, 0, 0, 0, 0, 0, 0, 0, 0, −1, 0, 0, 0].

These differences represent smaller values, which in turn require shorter bit lengths for encoding. Consequently, the *ALDC*-encoded word is significantly more compact, comprising just *71* bits, as shown below:

Encoded word = {'01101110101110101101101101110110010000000000100000000000000000010000000'}

In contrast to the original 280-bit number, transmitting the encoded word will demand significantly less energy, and it will also occupy less storage space. In total, *209* bits have been conserved.

The *ALDC* baseline algorithm was subjected to simulation using the numerical example provided in [2]. The necessary number of bits for encoding the temperature samples was curtailed from *112* bits to *30* bits. The alterations made to *ALDC*, as described in *Section A*, further reduced the bit count to *26* bits. The results have been visualized in Fig. 8.

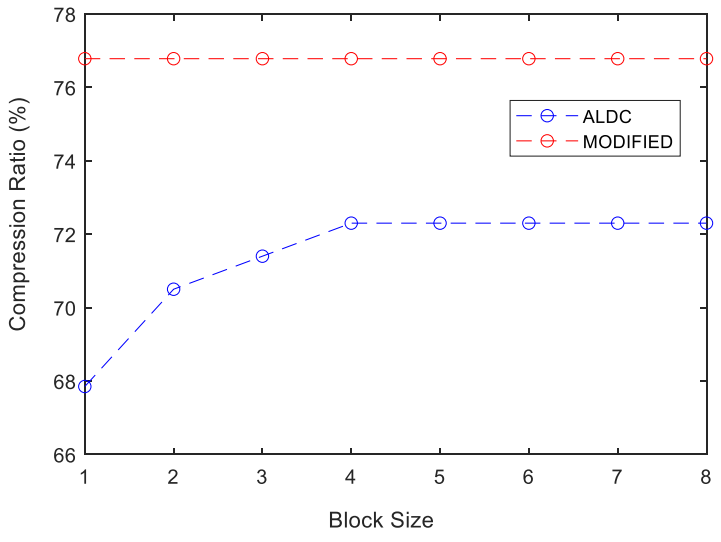

Fig. 8. Compression ratio vs block sizes for improved ALDC [31].

As indicated in Table 3, only *2* bits were necessary to encode each symbol. Figs. 9 and 10 visually demonstrate the reduced bit count achieved with the modified *ALDC* and the corresponding energy savings resulting from this reduction compared to the original scheme.

Smaller block sizes result in lower measurement precision, a smaller number of potential codewords, and lower code option values. Smaller block sizes lead to less complex processing, faster operations, and reduced susceptibility to data errors or loss. On the other hand, as block length increases, there is more room to identify data patterns and redundancies. Increased block length also expands the dynamic range of data and results in higher code option values. While this can improve precision and maintain a steady compression performance, it also introduces a higher risk of data loss or distortion and increases processing complexity.

Using the first example as reference, when compressing a block of *20* samples, the dynamic range was *14*, the code option region H was *2112,* which falls within *ALDC* region $12a < H$, and the algorithm utilized the *2*-Huffman coding table to encode the block of data. The achieved compression ratio was *74.6 %.* In contrast, when compressing a block of *8* samples, the dynamic range remained at *14*, but the code option region decreased to *2111*, still within the code option region $12a < H$. Larger block sizes resulted in a higher number of bits required, leading to lower compression ratios compared to smaller block sizes.

Table 7 presents a comprehensive view of the compression performance, including the processing time for various block lengths.

In the *FELACS* scheme, datasets underwent an outlier detection and cleaning process using the proposed method before being encoded and decoded with different block sizes. The results for both the baseline and modified algorithms are presented in Tables 8 and 9, for block sizes of the *Fishnet_101* Relative Humidity dataset.

The results demonstrate that the encoded word size for different block sizes was larger for the baseline *FELACS* algorithm compared to the modified version. This indicates that the outlier replacement process reduced the size of the original data and improved the compression rate. According to Eq. (13), a smaller number of compressed bits leads to a better compression rate. When outliers are not identified and cleaned from the data, the code option changes according to the decision region determined by the Rice Mapping function in Eq. (8). This function utilizes the sum of the absolute values of the mapped differences to determine the optimal code for the data block. Cleaning outliers smoothens the mapped differences and maintains the code option at a consistent level. Larger block sizes provide more time to observe data patterns, improving precision. The processing time for the modified algorithm is slightly higher than that of the baseline due to the additional time required for outlier processing.

Entropy, as expressed in Eq. (14), measures the amount of randomness in data. Raw data contains more information compared to compressed data, as indicated by the lower entropy demonstrated by the *modified FELACS*. Compression eliminates data redundancy, reducing entropy. Larger compressed block sizes exhibit higher entropy than smaller ones, indicating more information content in larger blocks. The relationship between compression rate and different block sizes is illustrated in Fig. 11 to compare the two algorithms.

The comparison between the baseline algorithm and the modified algorithm with outlier replacement for encoding the given Relative Humidity samples clearly demonstrates the impact of outlier removal on the encoding process. In the baseline algorithm, as the block size increased from *2* to *10* samples, the number of bits required for encoding also increased. This was due to the changing optimum coding parameter, which accommodated larger block sizes by selecting a different code parameter. However, the baseline coding did not correctly handle the condition where the value of the unary code was 0, resulting in codewords that did not accurately represent the original data.

On the other hand, the modified algorithm, which included outlier replacement, significantly reduced the number of bits required for encoding. By replacing the first data point *(9910)* with *9522* and reducing the large variation between successive terms, the codeword size decreased from 93 to *43* bits. The baseline 93-bit codeword:

'11001001101011011000000000000010011011000001100000010000001000000100000010000001100000001000000'.

The modified algorithm 43 bits codeword:

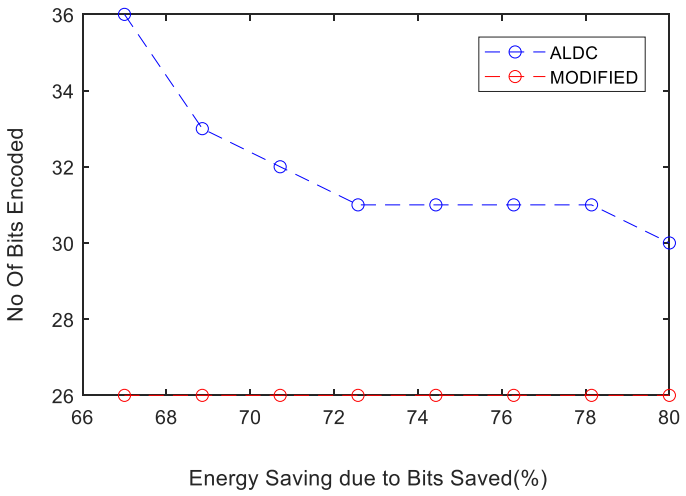

Fig. 9. Energy saving vs no. of bits encoded.

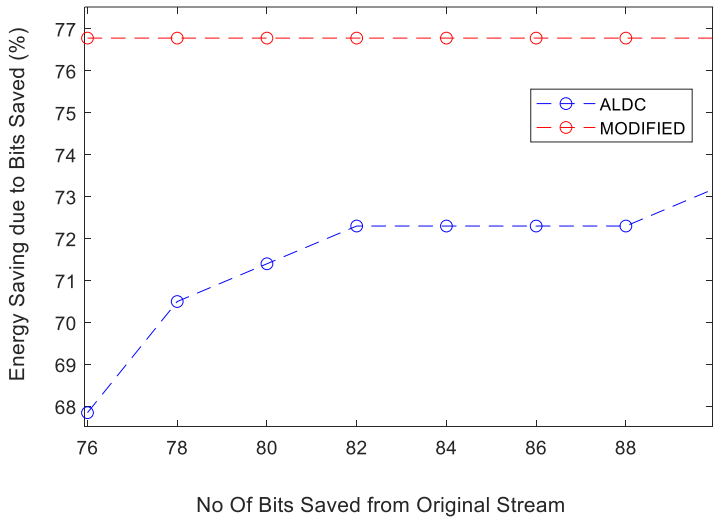

Fig. 10. Number of bits saved vs energy saving. [31].

Table 7
Improved ALDC Results for different block lengths.

Block Size *a*	No. of bits Compressed	Compression Ratio	*Code Option H*	Processing time (μs)
1	22	92.1	1718	32.7
2	32	88.6	2109	33.3
3	35	87.5	2110	38.2
4	37	86.8	2110	37.7
5	39	86.1	2110	32.9
6	41	85.4	2110	45.2
7	43	84.6	2110	39.5
8	46	83.6	2111	43
9	48	82.9	2111	35.2
10	50	82.1	2111	37.1
11	52	81.4	2111	35.4
12	54	80.7	2111	37.1
13	56	80	2111	37
14	58	79.3	2111	34.1
15	60	78.6	2111	35.5
16	62	77.9	2111	39.7
17	65	76.8	2112	35.3
18	67	76.1	2112	37.2
19	69	75.4	2112	40.3
20	71	74.6	2112	39.5

Table 8
Fishnet_101 relative humidity baseline FELACS.

Block Size	No. of Bits Coded word	Processing Time (s)	Compression Rate	Code Option	Entropy
20	156	1.07	7.8	5	1.92
40	261	1.19	6.53	4	1.91
60	351	1.22	5.85	3	1.88
80	431	0.98	5.39	3	2.07
100	510	1.23	5.1	2	2.16
120	570	1.21	4.75	2	2.23
140	630	1.32	4.5	2	2.34
160	690	1.1	4.31	2	2.36
180	750	1.15	4.17	1	2.4
200	806	1.16	4.03	1	2.54
220	846	1.17	3.85	1	2.59
240	886	1.34	3.69	1	2.74
260	926	0.99	3.56	1	2.89
280	966	1.12	3.45	1	3.02
300	1006	1.5	3.35	1	3.14

Table 9
Fishnet_101 relative humidity *modified FELACS*.

Block Size	No. of Bits Coded word	Processing Time (s)	Compression Rate	Code Option	Entropy
20	60	2.86	3	0	1.92
40	108	3.12	2.7	0	1.91
60	144	3.63	2.4	0	1.88
80	182	5.27	2.28	0	2.07
100	230	6.26	2.3	0	2.16
120	271	3.34	2.26	0	2.23
140	310	6.49	2.21	0	2.34
160	351	4.54	2.19	0	2.36
180	392	3.09	2.18	0	2.4
200	423	2.96	2.12	0	2.55
220	469	5.7	2.13	0	2.59
240	512	2.88	2.13	0	2.77
260	547	5.34	2.1	0	2.9
280	595	3.19	2.13	0	3.02
300	631	3.11	2.1	0	3.14

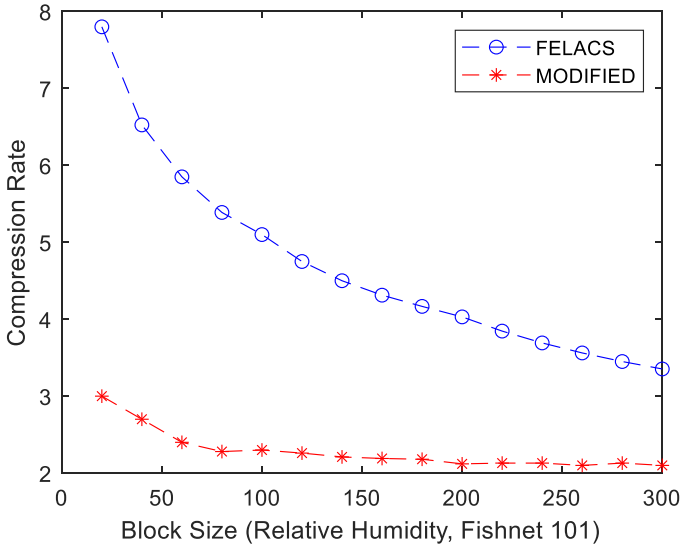

Fig. 11. Fishnet_101 relative humidity compression rate vs block sizes.

'00001001010011001000000100101010101001010101'

The binary numbers highlighted in green identify the binary representation of the selected optimum coding parameter. The ones highlighted in blue are the binary representation of the reference sample, while the rest represent the sample codes. This reduction was achieved because the modified coding addressed the change in the decision region to an optimum code option of zero for the block size of *10*, satisfying the equations below.

Unary Code $= \lfloor \partial / 2^0 \rfloor = \partial$

Split Part Code $= modulus(\ \partial,\ 2^0) = \partial.$

This demonstrates the effectiveness of outlier removal in improving compression efficiency and ensuring that the encoded data more accurately represents the original data. The proposed algorithm, which includes outlier removal and addresses optimum parameter conditions in its coding process, demonstrates its effectiveness in accurately representing the original data while achieving reduced compression rates and improved compression efficiency. This is evident in the results for *Fishnet_101* Ambient Temperature and *Le Genepi_20* Surface Temperature datasets as displayed in Tables 10-13.

In these tables, the modified algorithm consistently outperforms the baseline algorithm in terms of compression rate, energy saving, and entropy. The reduced number of bits required for encoding and improved entropy values indicate that the modified algorithm provides a more efficient compression process, resulting in a better representation of the original data. Figs. 12 and 13 further illustrate the performance of both algorithms, with the modified algorithm consistently showing better results across different block sizes. Overall, the proposed algorithm's ability to handle outliers and adapt to changing coding conditions leads to improved compression efficiency, making it a valuable approach for data compression in wireless sensor networks.

Table 10
Fishnet ambient temperature baseline FELACS.

Block Size	No. of Bits Coded word	Processing Time (s)	Compression Rate	Code Option	Entropy
20	101	1.19	5.05	3	3.88
40	177	2.14	4.43	2	4.9
60	242	2.95	4.03	2	5.43
80	300	1.24	3.75	1	5.74
100	348	4	3.48	1	6.03
120	406	1.68	3.38	1	6.38
140	454	1.01	3.24	1	6.59
160	497	1.35	3.11	1	6.68
180	538	1.29	2.99	1	6.73
200	579	1.34	2.9	1	6.83
220	620	1.29	2.82	1	6.84
240	969	0.94	4.04	0	6.86
260	1016	1.42	3.91	0	6.91
280	1062	1.04	3.79	0	6.97
300	1107	1.27	3.69	0	7

Table 11
Fishnet ambient temperature modified FELACS.

Block Size	No. of Bits Coded word	Processing Time (s)	Compression Rate	Code Option	Entropy
20	101	3	5.05	3	3.88
40	177	3.53	4.43	2	4.9
60	242	3.34	4.03	2	5.43
80	300	3.54	3.75	1	5.74
100	348	5.82	3.48	1	6.03
120	406	3.64	3.38	1	6.38
140	454	4.31	3.24	1	6.59
160	497	5.36	3.11	1	6.68
180	538	4.99	2.99	1	6.73
200	579	6.98	2.9	1	6.83
220	620	5.1	2.82	1	6.84
240	969	3.51	4.04	0	6.86
260	1016	4.63	3.91	0	6.91
280	1062	3.53	3.79	0	6.97
300	1107	4.65	3.69	0	7

Table 12
Le Genepi surface temperature baseline FELACS.

Block Size	No. of Bits Coded word	Processing Time (s)	Compression Rate	Code Option	Entropy
20	89	3.13	4.45	2	2.76
40	142	3.89	3.55	1	2.996
60	192	2.79	3.2	1	3.31
80	248	3.05	3.1	1	3.59
100	398	4.68	3.98	0	3.83
120	473	4.67	3.94	0	4.07
140	513	4.82	3.66	0	4.02
160	589	4.39	3.68	0	4.16
180	640	4.87	3.56	0	4.23
200	704	3.97	3.52	0	4.37
220	755	4.15	3.43	0	4.41
240	795	4.63	3.31	0	4.36
260	846	3.39	3.25	0	4.39
280	897	7.04	3.2	0	4.45
300	950	3.13	3.17	0	4.5

The two datasets exhibit a consistent compression rate across both algorithms, primarily due to the dataset's stable patterns within the tested block sizes, signifying the absence of outliers in the dataset. This underscores the proposed method's utility in data error validation.

Further investigations were made for block sizes of 20, 100, 500, 1000 and 5000 using relative humidity dataset of *Fishnet_101*

Table 13
Le Genepi surface temperature modified FELACS.

Block Size	No. of Bits Coded word	Processing Time (s)	Compression Rate	Code Option	Entropy
20	89	2.83	4.45	2	2.76
40	142	2.78	3.55	1	2.996
60	192	3.17	3.2	1	3.31
80	248	2.12	3.1	1	3.59
100	398	2.76	3.98	0	3.83
120	473	4.63	3.94	0	4.07
140	513	5.77	3.66	0	4.02
160	589	5.67	3.68	0	4.16
180	640	6.74	3.56	0	4.23
200	704	7.47	3.52	0	4.37
220	755	3.66	3.43	0	4.41
240	795	5.29	3.31	0	4.36
260	846	5.16	3.25	0	4.39
280	897	5.68	3.2	0	4.45
300	950	3.64	3.17	0	4.5

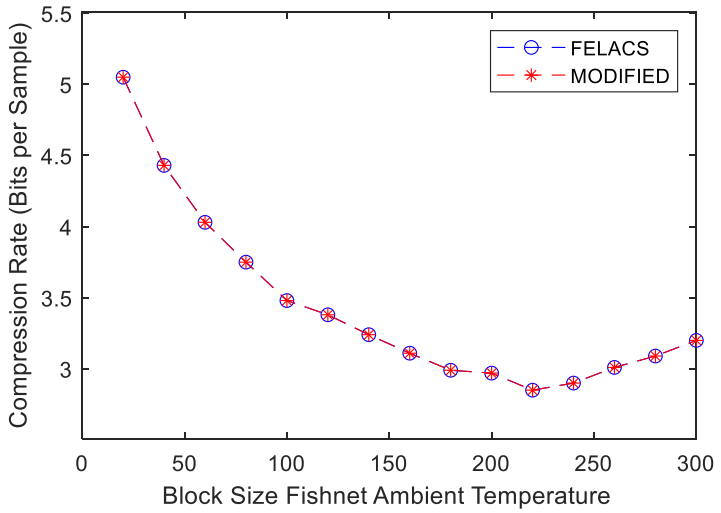

Fig. 12. Fishnet ambient temperature compression rate vs block sizes.

deployment. The compression performance for the different block sizes were plotted as shown in Figs. 14-16 to determine the optimum block size for this dataset. Results indicate that the optimum block size is *1000*. Compared to other block sizes. It demonstrates lower compression rates, adequate sampling rates and reasonable processing time. The processing time increases from *4* to *12 s* as the number of samples increase from *1000* to *14,000*. This confirms that it takes longer to sense and process a bigger block of data than it takes for a smaller one. A bigger block allows observation of patterns of data and is most likely to suffer natural phenomena disturbances. Although block sizes of *100* have higher sampling rates, their compression rates are higher and begin to fall when the number of samples reach *1000* and above. The block size of 5000 exhibit lower compression rates, but the sampling rate is very low for *14,720* samples and compromises the precision for data encoding and decoding.

The proposed method additionally holds the potential to augment the modified *ALDC* algorithm's performance by proactively detecting outliers before the encoding process. This capability was demonstrated when compressing ten *Fishnet_101* Relative Humidity samples (Source), resulting in an encoded word length of 50 bits using the modified *ALDC* algorithm.

Source = [9910 9519 9518 9518 9518 9518 9518 9517 9517 9517].

Codeword=

'01,101,110,101,110,101,101,101,101,110,110,010,000,000,000,100,000'

Upon implementing the proposed outlier replacement method on the same source stream, it successfully identified the first source term, *9910,* as an outlier and subsequently replaced it. The resulting cleaned stream was as follows:

ALDC_Cterm = [9522 9519 9518 9518 9518 9518 9518 9517 9517 9517]

The encoded output was a 43-bit codeword =

'0100000001110100110010110010000000000100000'.

Substituting outliers with more representative data diminishes the compression load since it narrows the variations between data

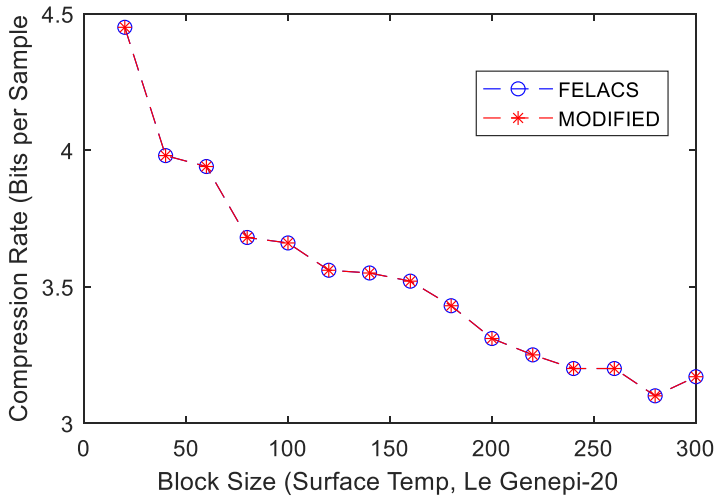

Fig. 13. LeGenepi_20 surface temperature compression rate vs block sizes.

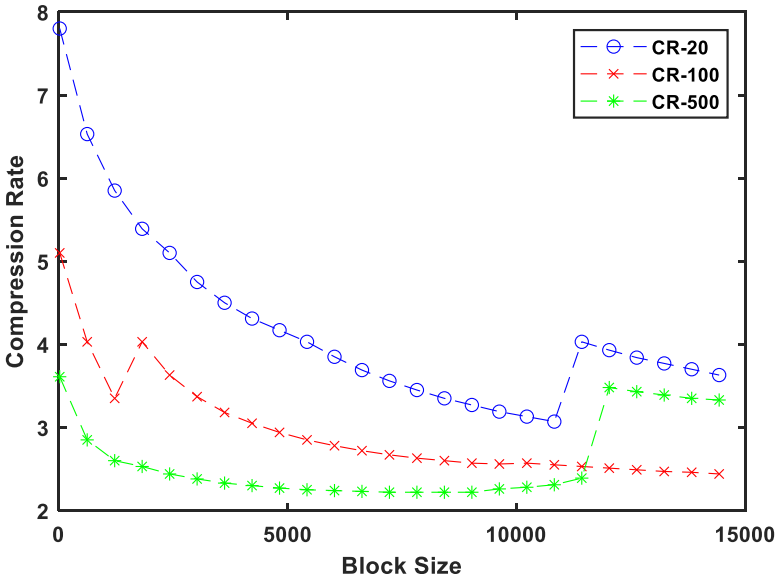

Fig. 14. Performance comparison for block size of 20, 100 and 500 samples.

readings, thereby necessitating fewer bits for encoding. Utilizing the *ALDC* Eq. (3) with a *14*-bit resolution analogue-to-digital converter, a sample block containing *10* relative humidity dataset readings attained compression ratios of $CR = \left(1 - \frac{50}{140}\right) \times 100\% = 64.2\,\%$ for compression without detection and cleaning of outliers, and $CR = \left(1 - \frac{43}{140}\right) \times 100\,\% = 69.2\,\%$, for compression with outlier cleaning. Compression performance was improved by outlier cleaning. Fig. 17 demonstrates the reduced number of bits for the encoded word with outlier cleaning on the different block sizes.

Experimental results show that the proposed method outperformed the two algorithms that were analyzed. On the *Modified ALDC* it demonstrated improved compression ratio from *64.2 % to 69.2 %* using the experimental example shown on *page 20*. The same was evidenced with the *FELACS* algorithm where compression rate was significantly reduced from a range of *7.8* to *3.35* for *300* samples that were sampled in block sizes of *20* as shown in Table 8 on *page 17* and Fig. 11 on *page 18*, showing better compression performance. The proposed approach was not experimented on other recent literature that are available under Table 2 of related works because it was more focused on the two adaptive lossless data compression schemes under analysis. However, enhancement of both *ALDC* and *FELACS* is an indication that this robust approach can be universally applied across other data compression schemes to further reduce the size of data and improve energy efficiency. Identification of optimum block sizes for encoding and decoding data can also ensure efficient transmission of environmental data from various WSN deployments.

It was noted that when data is sampled at an adequate rate, consecutive data readings tend to follow a consistent pattern. However,

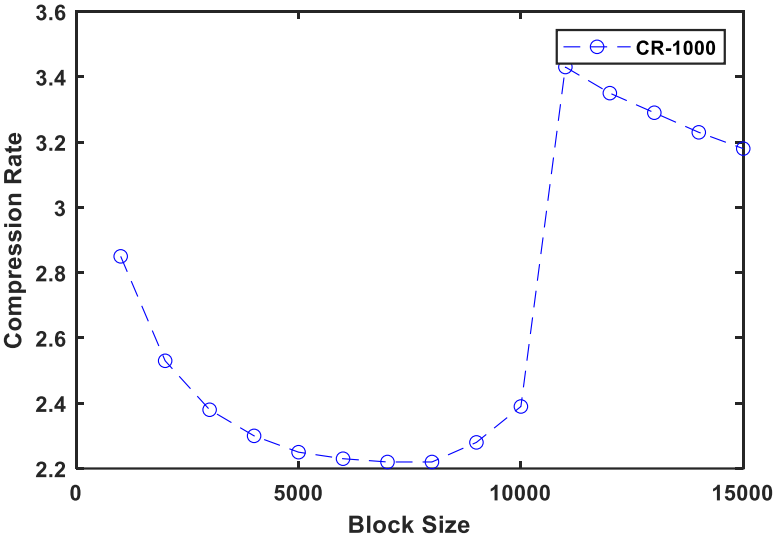

Fig. 15. Performance comparison for block size of 1000 samples.

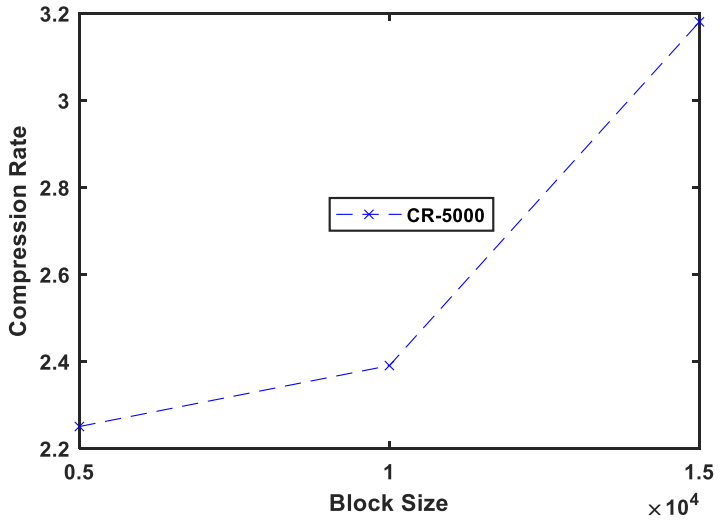

Fig. 16. Performance comparison for block size of 5000 samples.

longer intervals between data sampling result in significant fluctuations between data readings. The presence of outliers in the data introduces these large variations in the data sequence, increasing data size, reducing compressibility, and distorting the original data representation. Additionally, outliers elevate data entropy, which poses challenges for compression algorithms in effectively encoding the data. The outlier replacement technique proves valuable in scenarios where clean, compressed data is required without an emphasis on error detection. Conversely, for applications that demand the identification of errors or faults in data, such as seismic data applications, the outlier bits can be transmitted independently with minimal bit overhead. These outlier bits can then be reintegrated during the decompression stage to recover the true raw data. This approach offers a more energy-efficient solution compared to transmitting the entire data, including the outliers, which would result in a heavier data load.

Conclusions

In conclusion, this study has addressed a critical concern in wireless sensor networks—energy efficiency—by examining and enhancing two adaptive lossless data compression algorithms, *ALDC* and *FELACS*. The primary objective was to extend the operational lifetime of wireless sensor networks while ensuring the efficient transmission of environmental data, specifically temperature and relative humidity, collected from diverse field deployments, including *Fishnet, Lucerne*, and *Le Genepi.* Both *ALDC* and *FELACS* have demonstrated their effectiveness in reducing the data payload by encoding variations between consecutive data readings, leading to a reduction in the number of bits required for transmission. In particular, the modified *ALDC* algorithm, which employed optimized

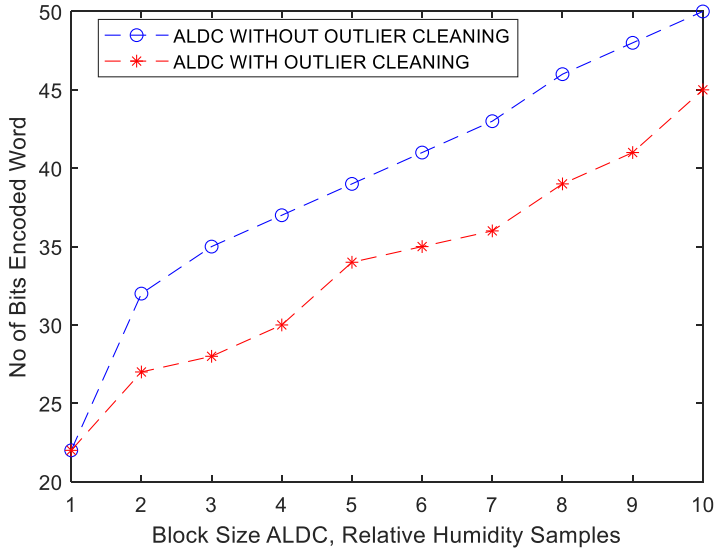

Fig. 17. *ALDC* outlier cleaning vs no outlier cleaning.

Huffman coding tables, achieved an impressive 77 % improvement in energy efficiency. The analysis of *FELACS* revealed the impact of natural phenomena-induced anomalies on sensor data, necessitating a novel approach. We introduced a robust method for identifying and replacing outliers within the sensor data, significantly enhancing compression performance. This approach not only reduced the compression overhead but also improved the precision of data encoding and decoding. To ensure the accurate encoding and decoding of data samples, we incorporated adjustments to the selection of the optimum coding parameters using *MATLAB* coding, thereby further enhancing the energy efficiency of both *ALDC* and *FELACS*. The study has also identified *1000* as the optimum block size for transmission of relative humidity data from *Fishnet_101* deployment.

The applicability of these proposed algorithms can be adapted to a range of wireless sensor network applications, depending on the specific data patterns and nature of the deployed sensors. Our research contributes to the efficient management of energy resources in wireless sensor networks. As a direction for future research, we suggest exploring methods for independently encoding outlier bits, apart from the cleaned compressed data, and subsequently integrating them with the cleaned compressed data before decompression. This approach aims to facilitate the seamless restoration of sensor network data to its original form while maintaining energy efficiency.

We believe that the findings and methodologies presented in this study hold significant promise for advancing the field of wireless sensor networks and data compression, contributing to the development of more sustainable and effective sensor network deployments.

Author contributions

All co-authors of this manuscript have equally contributed to the content of this manuscript. **Lucia K Ketshabetswe** – Conceived idea, designed and performed the simulation work, and took the lead in writing the manuscript. **Adamu Murtala Zungeru** – Senior author who conceived the idea, supervised the project and co-written the manuscript. **Caspar K. Lebekwe** – Also, conceived the idea, supervised the project and co-written the manuscript, and provided critical feedback and helped shape the research, analysis. **Bokani Mtengi**– Also, conceived the idea, supervised the project and co-written the manuscript, and provided critical feedback and helped shape the research, analysis.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors thank Botswana International University of Science and Technology for the assistance, resources and opportunities that were made available to make this research work a success. Many thanks to the Department of Electrical, Computer, and Telecommunications for the continuous sustenance, patience and motivation offered from time to time when needed.

References

[1] L.K. Ketshabetswe, A.M. Zungeru, M. Mangwala, J.M. Chuma, B. Sigweni, Communication protocols for wireless sensor networks: a survey and comparison, Heliyon 5 (5) (2019) e01591.
[2] J.G. Kolo, S.A. Shanmugam, D.W.G. Lim, L.M. Ang, K.P. Seng, An adaptive lossless data compression scheme for wireless sensor networks, J. Sensors 2012 (2012).
[3] J.G. Kolo, S.A. Shanmugam, D.W.G. Lim, L.M. Ang, Fast and efficient lossless adaptive compression scheme for wireless sensor networks, Comput. Electr. Eng. 41 (C) (2015) 275–287.
[4] F. Marcelloni, M. Vecchio, A simple algorithm for data compression in wireless sensor networks, IEEE Commun. Lett. 12 (6) (2008) 411–413.
[5] N. Kimura, S. Latifi, A survey on data compression in wireless sensor networks, Int. Conf. Inf. Technol. Coding Comput. ITCC 2 (2005) 8–13.
[6] T. Srisooksai, K. Keamarungsi, P. Lamsrichan, K. Araki, Practical data compression in wireless sensor networks: a survey, J. Netw. Comput. Appl. 35 (1) (2012) 37–59.
[7] M.A. Razzaque, C. Bleakley, S. Dobson, Compression in wireless sensor networks: a survey and comparative evaluation, ACM Trans. Sens. Netw. 10 (1) (2013).
[8] (University of Nebraska), *Introduction to*, Third Edit. 2006.
[9] Y. Arora, "Literature Survey on Image and Text Compression Techniques," vol. 3, no. 09, pp. 626–630, 2017.
[10] Y.C. Wang, Data compression techniques in wireless sensor networks, Pervasive Comput. 6 (1) (2012) 61–86.
[11] R. R, P. Varshney, Data-aggregation techniques in sensor networks: a survey, *IEEE Xplore*, no. IEEE Commun. Surv. Tutorials 8 (4) (2007) 48–63.
[12] S. Ozdemir, Y. Xiao, Secure data aggregation in wireless sensor networks: a comprehensive overview, Comput. Netw. 53 (12) (2009) 2022–2037.
[13] H. Karl, A. Willig, Protocols and Architectures for Wireless Sensor Networks, Protoc. Archit. Wirel. Sens. Netw. (2006) 1–497.
[14] N. Sadagopan, B. Krishnamachari, Maximizing data extraction in energy-limited sensor networks, IEEE Infocom 3 (2004) 1717–1727.
[15] F. Ordóñez, B. Krishnamachari, Optimal information extraction in energy-limited wireless sensor networks, IEEE J. Sel. Areas Commun. 22 (6) (2004) 1121–1129.
[16] W. Wang, D. Peng, H. Wang, H. Sharif, H.H. Chen, Cross-layer multirate interaction with distributed source coding in wireless sensor networks, IEEE Trans. Wirel. Commun. 8 (2) (2009) 787–795.
[17] R.K. S, H.V. Joel B Predd, A collaborative training algorithm for distributed learning, IEEE Trans. Inf. THEORY. 55 (4) (2009).
[18] G. Quer, R. Masiero, D. Munaretto, M. Rossi, J. Widmer, M. Zorzi, On the interplay between routing and signal representation for Compressive Sensing in wireless sensor networks, Inf. Theory Appl. Work. ITA (2009) 206–215. March2009.
[19] C.T. Chou, R. Rana, W. Hu, Energy efficient information collection in wireless sensor networks using adaptive compressive sensing, in: *Proc. - Conf. Local Comput. Networks, LCN*, no. October, 2009, pp. 443–450.
[20] S. Ji, Y. Xue, L. Carin, Bayesian compressive sensing, IEEE Trans. Signal Process. 56 (6) (2008) 2346–2356.
[21] J. Kho, L. Tran-Thanh, A. Rogers, N.R. Jennings, Decentralised control of adaptive sampling and routing in wireless visual sensor networks, Proc. Int. Jt. Conf. Auton. Agents Multiagent Syst. AAMAS 2 (212) (2009) 1208–1209.
[22] A. Amar, A. Leshem, M. Gastpar, Recursive implementation of the distributed Karhunen-Love transform, IEEE Trans. Signal Process. 58 (10) (2010) 5320–5330.
[23] A. Ciancio, A. Ortega, A distributed wavelet compression algorithm for wireless multihop sensor networks using lifting, ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc. IV (2005) 825–828.
[24] G. Shen, A. Ortega, Transform-based distributed data gathering, IEEE Trans. Signal Process. 58 (7) (2010) 3802–3815.
[25] P. Elias, Predictive coding-part, IRE Trans Inf Theory (1) (1955) 16–24.
[26] S. Lindsey, C.S. Raghavendra, PEGASIS: power-efficient gathering in sensor information systems, IEEE Aerosp. Conf. Proc. 3 (2002) 1125–1130.
[27] S. Lindsey, C. Raghavendra, K. Sivalingam, P.O. Box, Data gathering in sensor networks using the energy * delay metric, Washington State University 00 (C) (2001) 0–7.
[28] W.R. Heinzelman, A. Chandrakasan, H. Balakrishnan, Energy-efficient communication protocol for wireless microsensor networks, Proc. 33rd Annu. Hawaii Int. Conf. Syst. Sci. (2000) 10, vol.1c.
[29] W.B. Heinzelman, A.P. Chandrakasan, H. Balakrishnan, An application-specific protocol architecture for wireless microsensor networks, IEEE Trans. Wirel. Commun. 1 (4) (2002) 660–670.
[30] H.Ö. Tan, I. Körpeoğlu, Power efficient data gathering and aggregation in wireless sensor networks, SIGMOD Rec. 32 (4) (2003) 66–71.
[31] K.L. Ketshabetswe, A.M. Zungeru, B. Mtengi, C.K. Lebekwe, S.R.S. Prabaharan, Data Compression Algorithms for Wireless Sensor Networks: a Review and Comparison, IEEE Access 9 (2021) 136872–136891.
[32] K. Sayood, Introduction to Data Compression, 3rd Edit, Univ. Nebraska.: Morgan Kaufmann Series in Multimedia Information and Systems, Nebraska, USA, 2006.
[33] R. Alhussaini, A.K. Idrees, M.A. Salman, Data transmission protocol for reducing the energy consumption in wireless sensor networks, Commun. Comput. Inf. Sci. 938 (2018) 35–49. October.
[34] A.K. Idrees, R. Alhussaini, M.A. Salman, Energy-efficient two-layer data transmission reduction protocol in periodic sensor networks of IoTs, Pers. Ubiquitous Comput. 27 (2) (2023) 139–158.
[35] A.S. Jaber, A.K. Idrees, Energy-saving multisensor data sampling and fusion with decision-making for monitoring health risk using WBSNs, Softw. - Pract. Exp. 51 (2) (2021) 271–293.
[36] Z. Boulouard, M. Ouaissa, M. Ouaissa, and S. El Himer, *AI and IoT for sustainable development in emerging countries*, vol. 105, no. October. 2022.
[37] A.K. IDREES, M.S. Khlief, Efficient compression technique for reducing transmitted EEG data without loss in IoMT networks based on fog computing, SSRN Electron. J. (2022).
[38] K.K. Al-Nassrawy, D. Al-Shammary, A.K. Idrees, High performance fractal compression for EEG health network traffic, Procedia Comput. Sci. 167 (2019) (2020) 1240–1249.
[39] A.K. Idrees et al., "An edge-fog computing enabled lossless EEG data compression with epileptic seizure detection in IoMT networks to cite this version : HAL Id : hal-04257358," 2023.
[40] A. Kadhum Idrees, M. Saieed Khlief, A new lossless electroencephalogram compression technique for fog computing-based IoHT networks, Int. J. Commun. Syst. 36 (15) (2023) 1–21.
[41] A. Shawqi Jaber, A. Kadhum Idrees, Adaptive rate energy-saving data collecting technique for health monitoring in wireless body sensor networks, Int. J. Commun. Syst. 33 (17) (2020) 1–16.
[42] A.M. Hussein, A.K. Idrees, R. Couturier, Distributed energy-efficient data reduction approach based on prediction and compression to reduce data transmission in IoT networks, Int. J. Commun. Syst. (2022) 1–23. March.
[43] A.M. Hussein, A.K. Idrees, R. Couturier, A Distributed Prediction–Compression-Based Mechanism For Energy Saving in IoT Networks, 79, Springer US, 2023.
[44] A.K. Idrees, L.W. jawad, Energy-efficient data processing protocol in edge-based IoT networks, Ann. des Telecommun. Telecommun. 78 (5–6) (2023) 347–362.
[45] R.F. Rice, P.S. Yeh, W. Miller, Algorithms for a very high speed universal noiseless coding module, JPL Publ. Lab. 91 (1) (1991) 1–30.