


A Robust Lossless Data Compression Technique for Air Quality Time-Series Data using Genetic Algorithm


Banani Ghose

(Department of Information Technology, Haldia Institute of technology, Haldia, West Bengal, India

 <https://orcid.org/0000-0001-6558-1334>, bananighose@hithaldia.in)

Zeenat Rehena

(Department of Computer Science & Engg., Aliah University, Kolkata, West Bengal, India

 <https://orcid.org/0000-0002-0689-1903>, zeenatrehena@yahoo.co.in)

Abstract: Due to the advancement of industrialization and urbanization, air pollution become a serious issue in recent decades. To get rid of this problem Air Quality Monitoring Stations (AQMSs) are established that can assess the air quality and provide some measures to control it. These AQMSs capture the time series data through sensors and form an IoT based network to send the data to the cloud for further analysis. These IoT devices are paired with low capacity batteries and limited memory, transmission, and computational components. Transmitting the high volume data to the cloud for successive analytical purposes demands high energy dissipation and higher bandwidth. Moreover, for storing the big volumed data the system needs larger storage space. The single solution for all these constraints is data compression. Data compression reduces the volume of the datasets. The reduced volume of data saves energy and it is easier to transmit the compressed data through the limited bandwidth. In this paper, a lossless data compression algorithm using Genetic Algorithms is proposed. On successful implementation of the algorithm, the air quality time series data is compressed with absolutely no data loss. To evaluate the efficiency of the algorithm, its performance is compared with some classical and several state of the art compression schemes. The experimental results show that the proposed compression algorithm outperforms the classical as well as state of the art models concerning the performance evaluating parameters like Compression Ratio, Compression Factor, and most importantly Power Saving.

Keywords: Compression algorithm, Lossless algorithm, Power saving, Time series data, Robustness, Air quality

Categories: E.2, E.3, E.4, H.4

DOI: 10.3897/jucs.142860

1 Introduction

Air pollution monitoring and management are essential tasks that aim to provide the citizens of smart cities with a healthier environment. These tasks involve both controlling and preventing air pollution. By managing air pollution, we can protect people from harmful exposure to the polluted environment and strive to make the environment as pollution-free as possible. The air pollution monitoring system is deployed using IoT-enabled devices, sensors, and cloud computing. An IoT device (Microcontroller, like the ESP32 or ESP8266) collect the sensors data. After a fixed time interval, it transmits these collected data to clouds, and it demands a large storage capacity for storing and a high

bandwidth network infrastructure to transmit the dataset [Liu and Orban, 2008, Mohajer et al., 2024]. In the cloud, the time-series data are kept and analyzed for further work. Transmitting the high volumed time-series data from microcontroller to cloud mainly demands two facilities:

- A huge storage space, and
- A high network bandwidth to transmit the data to cloud.

Both these are the burdens of any system. Thus, a good infrastructure must attend these shortfalls and formulate a solution to lower down the extra burdens of the entire system.

Data compression may be thought of as a single solution for the problems encountered. Data compression is a technique of squeezing the size of a data file by lowering the number of bits required by the original data. It may be done by eliminating the redundant bits or by discarding unwanted data using some algorithms. The main objective of data compression is to reduce the data size before transmitting it through the network. It greatly takes care of the energy efficiency factor and the bandwidth demand during transmission [Shehabi et al., 2016, Yang and Mohajer, 2025]. Generally, this process is termed "encoding". Compression may be lossy or lossless. The lossy compression reduces the size of the data file by compromising some amount of data, i.e. at the receiver's end when the data file is received and it is decoded, then it is difficult for the human being to get back the original data. So, it is assumed that during compression some bits are missed, this kind of compression is termed "lossy". The lossless compression technique compresses the data in such a way that the information within stays intact. In comparison with the lossy compression technique, the lossless compression may have a lower compression speed, but it guarantees complete retrieval of original data after decompression. This compression scheme may be further divided into three broader categories:

- Dictionary based compression,
- Statistical compression, and
- Combinational compression.

The compression technique to be adopted whether lossy or lossless is application dependent. Lossy compression is used in applications where there is no harm in compromising the portion of data, such an example may be some multimedia files. A lossless compression scheme must be adopted by the applications where the sensor data needs to be accurate at the same time for the decision-making part, the data should not be tampered with or corrupted. Lossless compression technique [Zhang, 2015] is used where data loss cannot be allowed at all, like text files or any other application-sensitive data. The body area network mentioned in [Marcelloni and Vecchio, 2009] requires accurate high-quality undistorted data for proper decision-making. LZ77 (based on Lempel-Ziv 1977) [Hanumanthaiah et al., 2019] is a lossless compression technique.

The air pollution monitoring system is highly real-time and is a data-driven system. Actual data leads to actual forecasting or monitoring results. There is rarely any scope for re-analyzing the result all over again if any error is committed. So, the requirement is to go with a robust, lossless yet simple compression technique for the air pollution-related dataset, so that during the encoding process there will be absolutely zero data loss.

Generally, data compression techniques are so designed that they can take care of the storage saving and network bandwidth factor [Summers and Engineer, 2008, Harnik et al.,

2013]. Most of the compression algorithms do not pay any attention to energy saving and bandwidth saving factors. In some cases, they only take care of the compression ratio, which is one of the important metrics of a good compression algorithm. But it should not be the only factor that is to be taken care of. A good compression technique always minimizes the storage requirement, I/O bandwidth utilization, and most importantly the power saving factor. All these lead to good resource utilization and reduced overhead costs [Makatos et al., 2010, Nicolae, 2011].

This paper is aimed to formulate a lossless compression algorithm named Clustering and Shortest Path-based sensor data compression algorithm (CSP) to compress the air quality time-series data. Through this, the size of the dataset may be squeezed while being lossless and least complex in nature. The encoded dataset may be transmitted from microcontroller to cloud through the network efficiently, and at the decoding point, it will re-produce the very original dataset. The successful implementation of this compression technique will help in reducing the hardware cost. To implement this algorithm, the multivariate air pollution time-series dataset, containing multiple data points, one from each sensor is recorded and used. In this data compression technique, the K-means clustering algorithm is used and a genetic algorithm-based shortest path optimization technique is adopted to compress the multivariate air quality time-series dataset.

The rest of the paper is organized as follows. Section 2 describes the relevant research works that are already done in this field. In Section 3, the motivation of the work is briefed, the proposed algorithm is introduced, and a discussion about the working principle of the algorithm is elaborated. Section 4 describes the experiments carried out to validate the successful deployment of the proposed algorithm and the results yielded after experimentation are described in detail. The performance evaluation of the proposed algorithm is also illustrated concerning some existing classical and state-of-the-art compression techniques. Finally, Section 5 concludes this work and shows the future directions too.

2 Related Work

Nowadays, air quality monitoring and management is an active research field. Air pollution is increasing day by day due to the uncontrolled growth of urbanization and industrialization, and it causes a serious health hazard for livelihood. Air Quality Monitoring Stations form an IoT-based network to collect the time-series data from the sensors deployed in the environment and send the data to the cloud for further processing. Due to the limited resources (low battery power, low computational power, limited bandwidth, etc.) of the IoT devices, it is essential to optimally use these resources for better service. Data compression may be one of the solutions, and it is an integrated part of any air quality monitoring station.

Data compression is a technique to reduce the size of a data file by lowering the number of bits required by the original data. The primary objective of data compression is to abridge the size of the data without losing any information before transmitting it through the network. In recent times, a lot of research work has been conducted to compress the time-series data. Researchers surveyed the importance of data compression in [Srisooksai et al., 2012, Nassra and Capella, 2023, Chen et al., 2025]. In [Mathpal et al., 2017, Al-Kadhim and Al-Raweshidy, 2021a, Abdulzahra et al., 2021], the researchers revealed the importance of compression and described the types of compression. As time-series data is the need of a number of applications various researchers worked on

compressing the time-series data [Chiarot and Silvestri, 2023, Mochizuki and Komuro, 2024, Lykkegaard et al., 2025, Chen and Liu, 2024].

In [Mathpal et al., 2017, Silué et al., 2024] the authors compared various lossless compression schemes. A wavelet-based compression algorithm is formulated in [Wang et al., 2017, Pal et al., 2025]. But it seems to be lossy. In [Sangeetha et al., 2017, Lin et al., 2004, Wiseman, 2007], the researchers focused on a lossless data compression scheme, named LZW, but there are some limitations which are illustrated in [Le and Vo, 2018], the LZW algorithm requires a very large array, which is called a “dictionary”. If the data is of huge volume, then it is next to impossible to accommodate that dictionary portion. To overcome these limitations in [Le and Vo, 2018], the authors defined a modified lossless compression algorithm named D-LZW (Differential LZW), which is a modification of the LZW algorithm. In this D-LZW algorithm, the authors aimed to find the differences between two given data samples before feeding the data into the newly designed algorithm. In [Kolo et al., 2015], the authors lighted on the limitations of two state-of-the-art compression techniques, namely, LEC and S-LZW. Here the authors briefed that the LEC algorithm is a less robust technique that it cannot deal with the changing statistics of the source data sample. Whereas the S-LZW algorithm is described to be an algorithm with lesser complexity, it highly depends on the dictionary. The S-LZW algorithm divides the data sample into fixed-length blocks, and then the compression is performed on each block independently. Compared to FELAC, the authors concluded that S-LZW was much less robust to packet losses. S-LZW also suffers from a growing dictionary problem. These limitations of S-LZW and LEC are overcome by the FELAC [Kolo et al., 2015], which is a much low complex, very fast, robust, block-based compression algorithm which used Golomb-rice coding. It shows better performances on different parameters regarding compression. Another variation in the compression algorithm is proposed in [Liang and Li, 2014a]. Though it is fast and robust, then also there is room for improvement in terms of compression efficiency and energy saving. The researchers proposed an efficient compression technique known as Sequential-LEC (S-LEC). In this algorithm, the sequential context information is used to compress the data sample. In [Kolo et al., 2012], a spatiotemporal compression scheme is proposed, which is deployed in a PM sensing IoT Device. The compression scheme reduces the volume of data while reducing the amount of energy consumption and increasing the compression ratio. Adaptive Lossless Data Compression (ALDC) is proposed in [Chen et al., 2020, Liu et al., 2018]. This algorithm is smart enough to handle the changing statistics of the input data sample. But its compression performance needs to be improved, to make it more robust. The researchers proposed a compression scheme based on the delta coding approach in [Stojkoska and Nikolovski, 2017] for the temporally correlated data like temperature-related data received from different smart devices. They showed their work as energy-efficient one while showing a higher compression ratio. In [Jain et al., 2020], two compression approaches are introduced and the work is compared with CNN architecture. At first, the simple K-means-based training algorithm is proposed, and then as an extension of the work Symmetric K-means algorithm is formulated.

Reviewing various technologies adopted by earlier researchers in this field, the basic knowledge of the subject of data compression is gathered. In the last few decades lot of serious efforts have been made in data compression. Most of these data compression algorithms are generic, and they are not suitable for a particular application. Some of these algorithms are concentrated on improving the Compression Ratio and some of them focus on reducing energy consumption. So, there is a need for a compression algorithm for an IoT-based Air Quality Monitoring System, that can improve the Compression Ratio and also be energy efficient. In this article, a Clustering, and Shortest Path-based

sensor data compression algorithm (CSP) is proposed for this purpose. CSP divides the data sample into clusters depending on K-means clustering. Using a genetic algorithm, the shortest path among the data points in each cluster is calculated, and depending on the path difference the data sample is encoded. Thus, CSP is found to be a lossless, robust compression technique with a higher Compression Ratio and energy-saving outcome.

3 Proposed Approach

In this letter, a lossless compression technique, Clustering, and Shortest Path-based sensor data compression algorithm (CSP) are proposed. CSP is an efficient compressing technique. It is effective on multivariate time-series data captured by the sensors for monitoring air quality in smart cities. The working procedure of the proposed technique is depicted broadly in Figure 1.

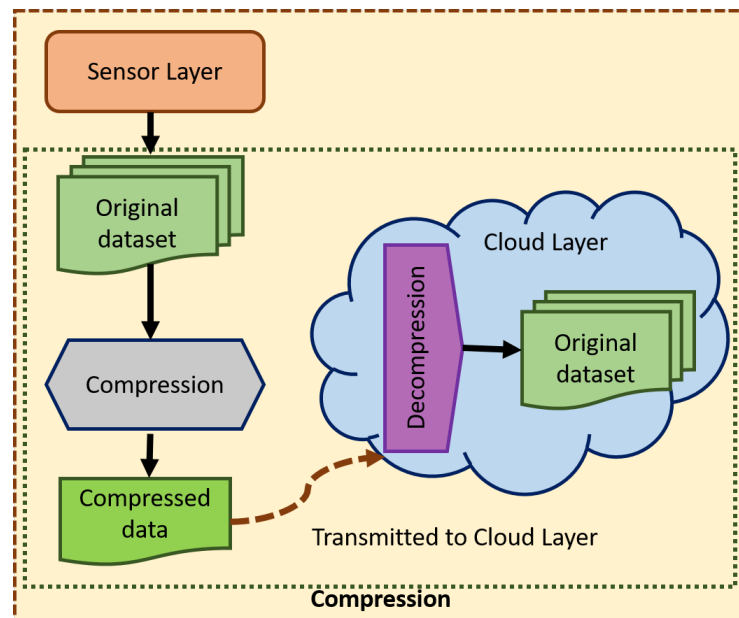


Figure 1: Basic block diagram of compression and decompression stages of CSP in an Air Pollution Monitoring System.

The block diagram of the proposed CSP algorithm for an Air Pollution Monitoring System is presented in the Figure 2. Here at each timestamp sensor captures the data and sends this data to an IoT (Microcontroller) device. Then after a fixed time interval, the IoT device compressed these sensor-captured big-volume data through the CSP algorithm. The compressed data is transmitted through the network to the cloud layer for further analysis and decision-making. To make the appropriate decision it is important to get back the actual data at the cloud layer. For that purpose, the compressed data is decompressed there. As the CSP is a lossless compression technique, the decompressed

data is as original as before compression. For further computation and analysis, the decompressed data is used.

CSP comprises of three major steps as shown in Figure 2.

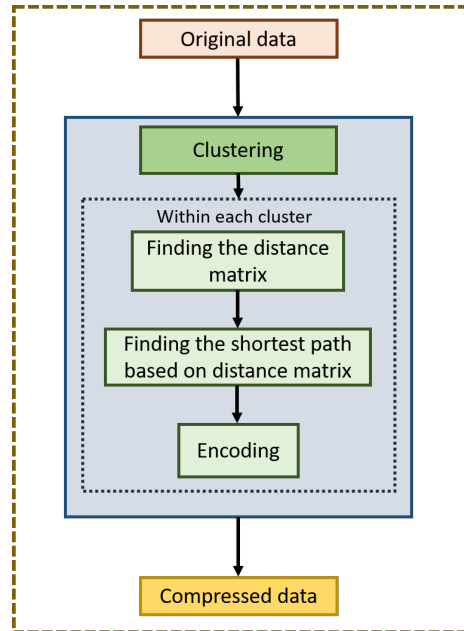


Figure 2: Flow diagram of Clustering and Shortest Path based sensor data compression algorithm (CSP).

1. Clustering,
2. Finding the distance matrix and the shortest path, and
3. Encoding.

In the assumed Air Quality Monitoring System, the IoT (Microcontroller) device stores and forwards sensor-captured data to the cloud on an hourly basis, and before forwarding, it compresses the data. So, in this environment, the microcontroller performs the data compression on an hourly basis. Maintaining the temporal sequence of the dataset is very important. For that, a data point representing the sequence number of each data sample is appended with each data sample. This sequence number helps in maintaining the temporal ordering of the time-series dataset during the decoding phase. This field is kept intact during the encoding. The rest of the operational details of CSP is described in Algorithm 1. In Table 1, the functions used in Algorithm 1 are described.

1. Clustering:

Clustering is the data analysis technique of dividing a set of data into some groups or clusters. The aim is to make homogeneous subgroups within a set of data. To be in the

Function	Purpose
Max(A)	For finding out the biggest value present in array A
Min_bit (B)	For finding out the minimum bits required to represent B
Encode(X,b)	For encoding every data point of vector X in b number of bits

Table 1: Descriptions of functions used in Algorithm 1.

Algorithm 1 Clustering and Shortest Path-based sensor data compression algorithm (CSP)

Input: Dataset $D_{1:M,1:N}$ (Where M is the number of data samples and N is the number of Pollutants.)

Output: Compressed Dataset CD

```

1: Select the number of cluster  $k$ 
2:  $C_{1:k} \leftarrow K - MEANS (D, k)$ 
3: for  $i \leftarrow 1$  to  $k$  do
4:   for  $p \leftarrow 1$  to  $n$  do
5:     for  $q \leftarrow p$  to  $n$  do
6:        $\delta_{p,q} \leftarrow FastDTW(DC_{i,p}, DC_{i,q})$  //find the FastDTW distance
        between the  $p^{th}$  and  $q^{th}$  data samples of cluster  $C_i$ 
7:     end for
8:   end for
9:    $\rho_{i,1:n} \leftarrow SHORTEST\_PATH\_GA(C_i, \delta)$  //shortest path in cluster  $C_i$ 
10:   $\phi_1 \leftarrow D_{\rho_{i,1}}$ 
11:  for  $j \leftarrow 2$  to  $n$  do
12:     $\phi_j \leftarrow D_{\rho^{(j-1)}} - D_{\rho_j}$ 
13:  end for
14:   $m1 \leftarrow Max(\phi_1)$ 
15:   $m2 \leftarrow Max(\phi_{2:n})$ 
16:   $b\_first = Min\_bit(m1)$ 
17:   $b\_remain = Min\_bit(m2)$ 
18:   $CD1 \leftarrow Encode(\phi_1, b\_first)$ 
19:  for  $j \leftarrow 2$  to  $n$  do
20:     $CD_j \leftarrow Encode(\phi_j, b\_remain)$ 
21:  end for
22: end for

```

same cluster, the data must share some common characteristics on which the grouping is done. The condition on which the clustering is done is application-specific. It may be Euclidean distance, or it may be correlation-based.

In this work, the entire multivariate time-series dataset is subdivided into several clusters by using the K-means clustering algorithm. The K-means clustering algorithm [Likas et al., 2003, Manchanda and Sharma, 2020] is one of the simplest and most popular algorithms widely used. It aims to cluster a set of data into a K number of pre-defined, non-overlapping clusters. The data points are allowed to be in a unique group. The data points inside a cluster are similar depending on a specific

feature. Here, by $K - MEANS(D, k)$, the dataset D is divided into k number of clusters. Since sensor data are collected in real-time, and constantly changes, it is dynamic in nature. Due to this dynamicity, it is not possible to use the same value of K (number of clusters) to compress the sensor data in each time interval. The success of the K-means clustering algorithm heavily depends on the value of the number of non-overlapping clusters K . The determination of the value of K for each time interval is a tedious job. As the air quality time-series data has daily and as well as weekly seasonality patterns [Bhanja et al., 2022, Ghose et al., 2022], these seasonality patterns are used to determine the value of K . The number of clusters (K) for the data compression of a particular hour's sensor data is determined from the previous week's same day's same hour's value of K . The K-means clustering procedure is described as $K - MEANS(D, k)$ below.

-
- 1: **procedure** K-means (D, k)
 - 2: Initialize the center points by shuffling the dataset primarily and then taking k data points for the center points without replacement.
 - 3: **repeat**
 - 4: Calculate the sum of the squared distance between the data points and all center points.
 - 5: Assign each data point to the closest center point within the cluster.
 - 6: By considering the average of all the data points that belong to a single cluster, calculate the center points for the clusters.
 - 7: **until** the center points are settled and no more change is taking place in them.
 - 8: Return (k number of clusters)
 - 9: **end procedure**
-

2. Finding the distance matrix and the shortest path:

In this phase, the shortest path between the inter-cluster data samples is ascertained. To find the shortest path among the data samples within a cluster the FastDTW distance is used as a distance metric.

DTW is used for calculating the distance between two dissimilar time-series data. It is also used to observe the similarity in two arrays or time-series data. DTW is generally used in the pre or post-processing steps in any experimentation. DTW is used majorly in recognizing speech, signature, and speaker. In forecasting works also this concept is majorly used. In stock market prediction, in pattern recognition, DTW is used widely. Many researchers have used DTW in their works. In [Li et al., 2020], DTW is used in clustering the time-series data for classifying car parks.

In this work, a variation of the DTW algorithm, named FastDTW [Salvador and Chan, 2007] is used. FastDTW is smart enough in handling spatio-temporal datasets. It is more accurate in producing the minimum-distance warp path between two time-series datasets. Next, to find the shortest path among all the data samples within each cluster, a genetic algorithm-based optimization technique is used. The genetic algorithm-based shortest path finding procedure is depicted in the procedure named

SHORTEST_PATH_GA(DS, Dist). The success of any Genetic Algorithm largely depends on properly selecting its hyperparameters, such as population size, mutation rate, convergence thresholds, etc. A large population size can reduce the algorithm's convergence speed, whereas a smaller size is not good enough for selecting the mating pool. So, it is essential to choose the population size optimally. Here the population size *pop_size* is selected as per the following formula.

$$pop_size = \lfloor n/2 \rfloor + 1, \quad (1)$$

here n is the number of data sample present in each cluster.

Mutation rate is another important parameter of the Genetic Algorithm. It restricts the algorithm to fasten into the local optima and adds heterogeneity into the population. If a lower mutation rate (0.01 or 0.001) is selected then only a small number of the population will be muted in each generation, and this leads to stuck the solution within a local optima of the search space. On the other hand, if the higher rate (>0.7) is selected then it explores the search space without cultivating promising solutions. In this proposed algorithm, the parameter tuning method is adopted to find the optimal value of the mutation rate. The convergence threshold is an essential parameter in Genetic Algorithms. It ensures that the algorithm reaches a stable state. In this proposed algorithm, the generation process is repeated until the stopping condition is reached, which is considered as the convergence threshold of the algorithm. If there is no change in the shortest path in two consecutive iterations (generations) then the generation process will be stopped.

3. Encoding:

In this phase, the inter-cluster data samples are encoded in bits. Here, within each of the clusters, the first data sample of the shortest path is kept unchanged and the difference among the consecutive data samples is calculated. Among all the first shortest paths' data samples of all the clusters, the biggest data point value is identified and the minimum number of bits required to represent it is formulated. All the data points of the first data samples of each cluster are represented in terms of that number of bits. Now, considering all the clusters, the biggest among all the difference values got for the data samples are found. The minimum number of bits to represent the difference is calculated. The different data samples present in each cluster are then represented in that number of bits. In this way, every cluster is encoded into a stream of bits where the first data sample of the shortest path is simply converted into bit streams without encoding. Now, the encoded clusters are ready to be transmitted through the network.

At the receiver's end, the compressed data received is to be decompressed to generate the original multivariate time-series data. All the clusters received are to be decompressed one by one to get the original data back. When the receiver gets the bitstream of an encoded cluster, he/she converts the first bit stream into its decimal equivalent to get the first data sample of that cluster, since the first data sample is not encoded. In the next step, when the next encoded data sample is got, that too is converted to decimal equivalent and the respective data points are added with the right data points with the first data sample got. This makes the second decoded data sample. To decode the third data sample of that cluster, the decimal equivalent of each data point of that encoded sample is added to the second decoded data sample. In this way, all the data samples of that cluster are decoded. This process is carried out on all the clusters received at the receiver's end to decompress the clusters. When

Parameter: Dataset $DS = DS_1, DS_2, \dots, DS_N$; where, DS_i is the i th. data sample in the dataset DS and N is the number of data samples. $Dist$ is the distance matrix containing distance between every pair of data samples.

Output: The shortest path.

```

1: procedure Shortest_path_GA( $DS, Dist$ )
2:   Select the number of path  $k$  (population size)
3:   Initialize each path  $P_i$  by randomly shuffling data samples of  $DS$ .
4:   do
5:     for each path  $P_i$  do
6:       Find the total path distance  $T_D$  by summing up all the inter sample
       path distances based on the distance matrix  $Dist$ 
7:       Select the shortest path  $short\_path$  whose path distance is minimum.
8:       Select  $M$  number of best paths  $B_p$  (Parents) from  $P$ ; where
        $M \leq (N - 2)$ .
9:       Generate  $O$  ( $O = N - M$ ) number of off springs from  $B_p$ .
10:      Generate  $O$  number of child by performing the mutation with mutation
       probability  $mp$ .
11:      Generate new path  $P$  (population) by accumulating  $M$  number of best
       paths and  $O$  number of child.
12:     end for
13:   while Stopping condition is reached //where Stopping condition is reached if
       there is no change in  $short\_path$ 
14:   Return ( $short\_path$ )
15: end procedure

```

all the clusters are decompressed, then based on the sequence number of the data samples they are rearranged to get the original multivariate time-series dataset. As there is no loss in data in this encoding, transmission, and decoding process, the compression technique followed is termed to be “lossless”.

4 Experiments & Results

This section describes the datasets used for the experimentation purposes, the details of experimental set up planned, the evaluation metrics with their equations, and the results along with the related discussions to justify the effectivity of the work.

4.1 Dataset Description

All the experiments are carried out on three different datasets. These three datasets are collected from three different sources. Three of the datasets contain pollutant concentration and meteorological factors as these are the two types of parameters which has a great influence on the air health of a particular place. The description of the datasets is depicted in Table 2.

Sl. No.	Source	Duration	Attributes	No. of rows	Freq- uency
1.	Delhi dataset (DS 1) [DelhiDataSet,]	01.01.2018 01:00 to 14.03.2019 15:00	CO,NO, NO ₂ , O ₃ , SO ₂ , PM10, PM2.5, HUMID, TEMP, WDR, WSP	10,502	Hourly
2.	Rozelli dataset (DS 2) [NSW,]	01.01.2018 13:00 to 31.12.2019 13:00	PM2.5, TEMP, HUMID, SD1, WDR, SOLAR, WSP, RAIN	17,497	Hourly
3.	Andhra Pradesh dataset (DS 3) [AndhraPradeshDataSet,]	01.07.2016 10:00 to 31.03.2023 23:00	PM10, NO NO _x , NH ₃ SO ₂ , CO O ₃ , TEMP WS, WD	59,150	Hourly

Table 2: Dataset description.

4.2 Experimental set up

All the experimentations are carried out on a PC with Intel(R) Core (TM) i3-8130U CPU 2.20 GHz, with a 64-bit operating system x64-based processor and memory is 4.00 GB. All the programs are done with the Python programming language.

To implement the CSP technique, each of the datasets is subdivided into several clusters. Let us consider, that each dataset is divided into k number of clusters. If the k is varied from 1 to l , where $l \leq$ number of data samples in the dataset, it is observed that the compression performance is varying with the varying k . By tuning the model, the value of k is ascertained for which the CSP technique is showing its best compression result for each of the dataset used. As all the datasets are comprising a large number of rows, for the sake of simplicity of the experiment k is varied as 5,10,15,20,... up to n , where n is the number of data samples.

4.3 Evaluation

This section focuses on an elaborated discussion about various parameters and in the later part the proposed compression technique is evaluated on those parameters.

4.3.1 Complexity analysis of the CSP

Complexity analysis is essential for any algorithm to establish its scalability and generability. In this section, the time and space complexity of the proposed CSP algorithm is analyzed. The complexity of each procedure of the proposed CSP is presented in Table 3.

The overall time and space complexity of the proposed CSP technique is as follows:

$$TimeComplexity = O \left(\frac{M^2 \cdot N}{k} + G \cdot P \cdot \frac{M^2}{k} + M \cdot N \right) \quad (2)$$

Procedure	Time Complexity	Space Complexity
K-means()	$O(k \cdot M \cdot N)$	$O(M \cdot N)$
FastDTW()	$O(M^2 \cdot N/k)$	$O(M^2/k)$
Shortest_path_GA()	$O(G \cdot P \cdot M^2/k)$	$O(G \cdot P \cdot M/k)$
Encode()	$O(M \cdot N)$	$O(M \cdot N)$
Parameters:		
M : Number of data samples		
N : Number of features (data value) per data sample		
k : Number of clusters		
G : Number of generations		
P : Number of populations		

Table 3: Time and space complexity of each functions of the proposed CSP.

$$SpaceComplexity = O\left(M \cdot N + \frac{M^2}{k} + G \cdot P \cdot \frac{M}{k}\right) \quad (3)$$

The compression algorithm CSP is primarily proposed to compress the air quality time-series data of the Air Quality Monitoring Systems. The number of features (pollutant, and meteorological factors) N are fixed for the Air Quality Monitoring Stations, and in general, it is less than 30. The IoT devices send the compressed data to the cloud every one or two hours, so the number of data samples M is not more than 120. Here M and N are constant terms, $k \leq M$, and the number of population $P \leq M$. So, for this particular application, the time and space complexity is a linear function of G , which is sufficiently good for this application.

In the case of time complexity, the most dominant term is $(G \cdot M^2)$, and in the worst case situation $G \geq M$. So, in the worst case, time complexity will be a polynomial of degree 3. On the other hand, the most dominant term of the space complexity is M^2 , so in the worst case, the space complexity will be quadratic. As per the time and space complexity analysis, the generality of CSP is acceptable.

4.3.2 Compression results

The performance of any compression algorithm is determined by several parameters, like

1. **Compression Factor (CF):** Compression Factor is the complement of Compression Ratio. It can be measured by taking the ratio of compressed sample size and uncompressed file size as mentioned in [Gopinath and Ravisankar, 2020]. So the formula for CF may be written as:

$$CF = \left(\frac{Compressed\ sample\ size}{Uncompressed\ sample\ size}\right) \quad (4)$$

2. **Compression Ratio (CR):** The most important parameter to evaluate the performance of a compression algorithm is its Compression Ratio. Which is generally measured in percentage (%). So, the Compression Ratio (CR) can be measured by

considering the ratio of the size of the compressed file and the original or uncompressed file size as in [Liang and Li, 2014a]. Then the percentage is calculated. The formula for CR is as follows:

$$CR = \left(1 - \frac{\text{Compressed sample size}}{\text{Uncompressed sample size}}\right) \times 100 \quad (5)$$

– Results & Discussion I:

As for the compression performance, CR% highly depends on the number of clusters k . The first and foremost work is to find the optimum k for which CSP shows the highest CR%. From Figure 3, it is clear that when the value of k is nearby 35, the Compression Ratio seems to be highest for all the three datasets.

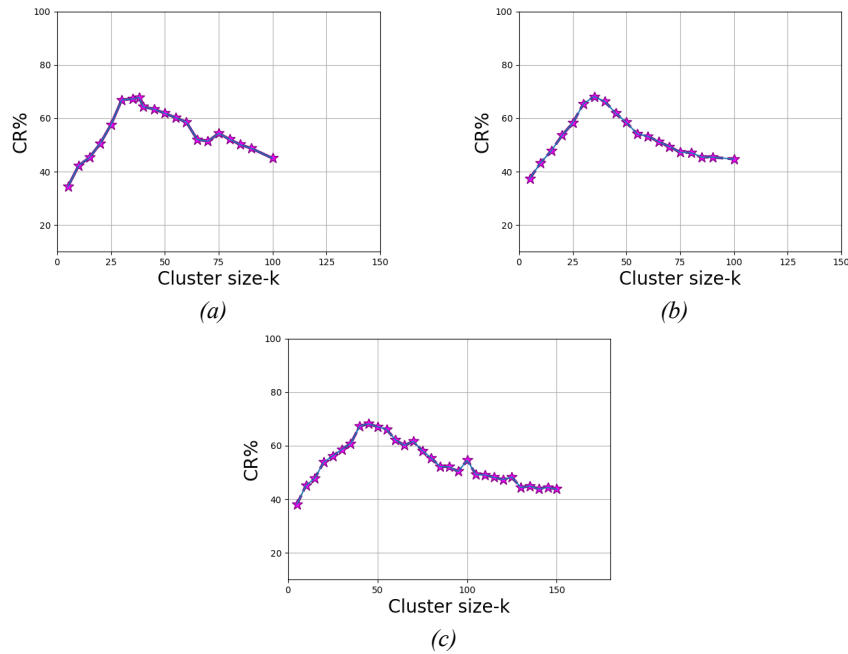


Figure 3: Performance of CSP with respect to number of clusters (a) Dataset DS 1 (b) Dataset DS 2, and (c) Dataset DS 3.

A compression technique needs to deliver a stable result irrespective of the dataset used. Its reliability, efficiency, and dependability increase with its stable and neutral attitude towards the variety of datasets on which it experiments. To validate this characteristic in the proposed CSP technique it is tested with three uneven datasets as depicted in Table 2. In Figure 4, it is observed that with the varying data sample size the CR% showcased by CSP is steady and stable for those three datasets. So, the proposed compression technique is robust and stable. Moreover, it is not only functioning at its best on any specific dataset, but its performance is stable and

reliable with the varying volume of the datasets. From this, it can be expected that CSP is capable of handling any given dataset and it will deliver consistent results with the same efficiency.

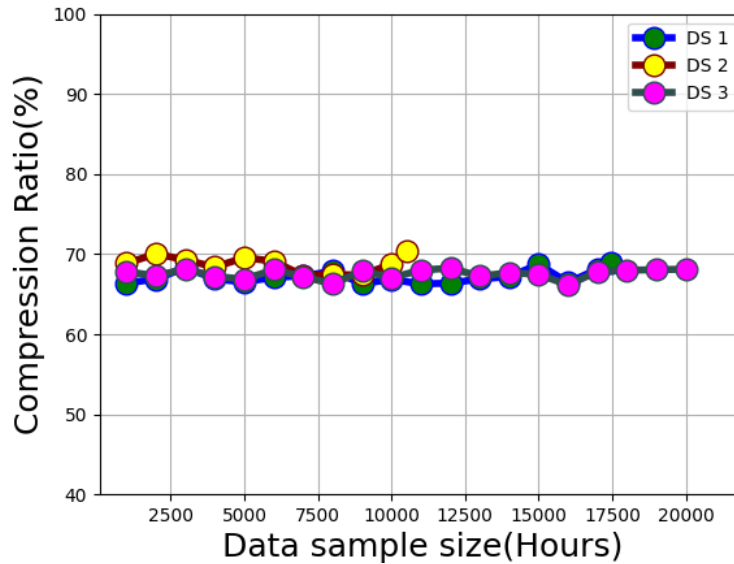


Figure 4: Performance of CSP with respect to different datasets.

To be effective in practice the compression algorithms must show satisfactory results with respect to the parameters like CR%, and CF. The applicability and efficiency of the algorithm highly depend on these metrics. In Table 4, the performance of the CSP technique is tabulated that validates the correctness of any newly evolved compression technique. The experiments are carried out on three different datasets mentioned in Table 2.

Dataset	Uncompressed sample size(bits)	Compressed sample size (bits)	CF	CR (%)
DS 1	2,31,044	74,373	0.32	67.81
DS 2	2,09,976	66,940	0.31	68.12
DS 3	1,04,10,400	33,16,753	0.319	68.14

Table 4: Performance of CSP compression technique.

- **Performance comparison with state-of-the-art models:**

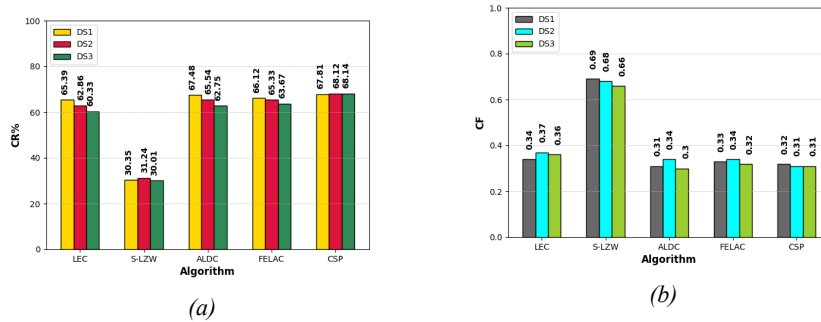


Figure 5: Performance assessment of CSP with other state-of-the-art model on Dataset 1 (DS 1), Dataset 2 (DS 2), and Dataset 3 (DS 3) with respect to (a) Compression Ratio (CR%) and (b) Compression Factor (CF).

To prove the applicability and effectiveness of the newly proposed compression technique, its performance must be compared with the already popular techniques. For that, its performances on different parameters on three datasets are compared to that of the state-of-the-art models like LEC, S-LZW, ALDC, and FELAC. The comparison is depicted in Table 5. The graphical representation of the comparison with the dataset DS 1, dataset DS 2, and dataset DS 3 are shown in Figures 5a and 5b respectively.

From Table 5, it can be easily concluded that among the various state-of-the-art compression models, the S-LZW seems to be showcasing the worst performance for all the three datasets. It is showing the poorest outcome in terms of Compression Ratio, and Compression Factor. On the other hand, the other models seem to deliver almost similar results on those two performance measuring parameters. The performance of the ALDC and CSP are very close to each other in CR%, but it differs remarkably in CF. Moreover, the difference lies in the implementation. ALDC requires additional memory to take care of the dictionary. For compression purposes, that dictionary is to be accessed frequently, which demands more time and loss of power too. Whereas the CSP does not need any extra memory or hardware for implementation, thus if the performance of the proposed CSP technique is observed, then it can be noticed that the proposed technique outruns the other compression models with respect to all the determining parameters while taking care of no extra hardware burden.

- **Performance comparison with the classical compression algorithms:**

For critically analyzing the performance and applicability of the proposed CSP compression technique, it is compared with some classical compression models. Some of the popular classical compression models are Gzip, Bzip, Rar, Huffman compression technique, Arithmetic compression model etc. Table 6 and Figure 6 depict the performance details of these models in terms of CR%.

By analyzing Table 6 and Figure 6, it can be visualized that two classical compression algorithms are delivering the worst performance. Those are the Huffman compression technique and the Arithmetic compression model. The other models are showing relatively better performances. But it is quite clear

DS 1 [DelhiDataSet,] (Sample size 10,502)				
Model	Uncompressed Data sample size (bits)	Compressed Data sample size (bits)	CF	CR%
S-LZW	2,31,044	1,60,922	0.69	30.35
ALDC	2,31,044	71,900	0.31	67.48
LEC	2,31,044	79,964	0.34	65.39
FELAC	2,31,044	78,277	0.33	66.12
CSP (Proposed)	2,31,044	74,373	0.32	67.81
DS 2 [NSW,](Sample size 17,498)				
Model	Uncompressed Data sample size (bits)	Compressed data sample size (bits)	CF	CR%
S-LZW	2,09,976	1,44,379	0.69	31.24
ALDC	2,09,976	72,357	0.34	65.54
LEC	2,09,976	77,985	0.37	62.86
FELAC	2,09,976	72,798	0.34	65.33
CSP (Proposed)	2,09,976	66,940	0.31	68.12
DS 3 [AndhraPradeshDataSet,](Sample size 59,150)				
Model	Uncompressed Data sample size (bits)	Compressed data sample size (bits)	CF	CR%
S-LZW	1,04,10,400	72,86,238	0.66	30.01
ALDC	1,04,10,400	38,77,874	0.30	62.75
LEC	1,04,10,400	41,29,805	0.36	60.33
FELAC	1,04,10,400	37,82,098	0.32	63.67
CSP (Proposed)	1,04,10,400	33,16,753	0.31	68.14

Table 5: Performance comparison of CSP with other state-of-the-art models.

from there that the proposed CSP technique outperforms the other compression techniques with respect to Compression Ratio (CR%).

4.3.3 Analyzing the power consumption of the network

Power consumption is one of the major factors of the IoT-based networks [Al-Kadhim and Al-Raweshidy, 2021b][Liang and Li, 2014b]. For evaluating the efficiency of the various compression algorithms regarding power consumption a network is designed as depicted in Figure 7. Several IoT devices are connected to a number of sensor nodes that are capturing air pollution-related data. IoT devices are such an important part of the network and are responsible for providing data to the cloud for

Model name	CR% on DS 1	CR% on DS 2	CR% on DS 3
Arithmetic	26.62	24.06	28.33
Huffman	24.69	23.19	25.91
Gzip	48.87	41.29	50.21
Bzip	57.82	56.22	60.39
Rar	59.12	54.56	62.01
CSP (proposed)	67.81	68.12	68.14

Table 6: Performance evaluation of CSP with other classical compression models.

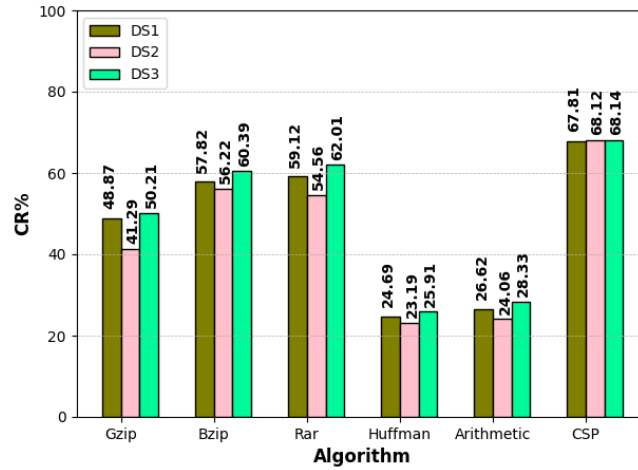


Figure 6: Performance evaluation of CSP with other classical compression models.

further analysis. The air pollution monitoring system deployed in the cloud receives requests regarding the air health of a place. The monitoring system analyses the data containing the records for various air pollutants and meteorological factors. After analyzing the air pollution-related data, the cloud sends back the result to the requesters. It is assumed that the compression scheme is applied in the lowermost layer, i.e. the IoT devices layer, after which the compressed data is forwarded to the cloud for further analysis.

The objective of the IoT-based network is to analyze the power consumption by different compression algorithms. Let, p (in bits) be the volume of the uncompressed data sample, and q (in bits) is the volume of the data sample after compression.

The mathematical representations of the proposed IoT-based network are as follows:

The power consumption of the micro-controller of the IoT devices in each clock cycle is,

$$EN_{clk} = \frac{V_{CC} \times I}{F_{clk}} \quad (6)$$

where V_{CC} = supply voltage of the microcontroller.

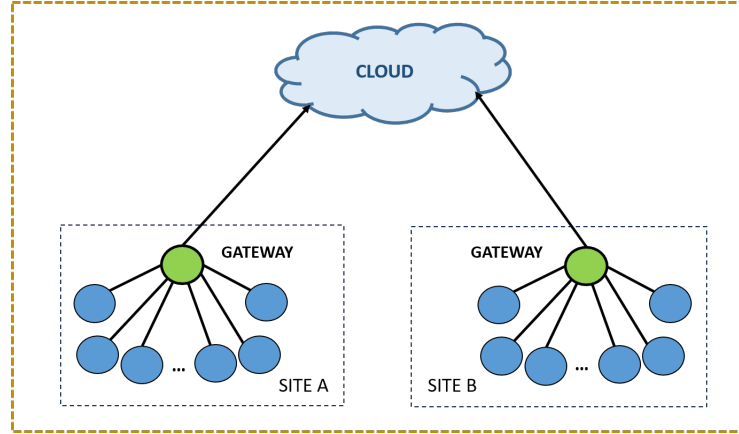


Figure 7: Overview of the IoT driven network.

I = current consumption of the microcontroller during active mode.

F_{clk} = clock rate of the microcontroller.

The energy consumed for transmitting and receiving by the IoT devices for each bit of data at the next-hop (single-hop transmission) is,

$$EN_{bit} = \frac{V_{CC} \times I_{TX} + V_{CC} \times I_{RX}}{D_{rate}} \quad (7)$$

where V_{CC} = supply voltage of the micro controller.

I_{TX} = current consumption during data transmission.

I_{RX} = current consumption during receiving data.

D_{rate} = effective data rate of the IoT device (in bps).

Transmission energy can be expressed as,

$$TrE = DT \times EN_{bit} \times q \quad (8)$$

where, DT = data traffic between an IoT device and the cloud layer before data compression.

Now, the compression power CP is calculated as,

$$CP = (N_{ADD} \times CL_{ADD} + N_{MUL} \times CL_{MUL} + N_{SUB} \times CL_{SUB} + N_{DIV} \times CL_{DIV} + N_{CMP} \times CL_{CMP}) \times EN_{clk} \quad (9)$$

where, N_{ADD} , N_{MUL} , N_{SUB} , N_{DIV} , and N_{CMP} are the total number of addition, multiplication, subtraction, division, and comparison operations required in the compression algorithm respectively. CL_{ADD} , CL_{MUL} , CL_{SUB} , CL_{DIV} , and CL_{CMP} are the number of clock cycles required by the micro-controller of the IoT devices to perform each addition, multiplication, subtraction, division, and comparison operation respectively. Finally, EN_{clk} is the energy required in each clock cycle.

Thus, the total power required by the network can be calculated as follows:

$$TP_{NET} = \sum_{i=1}^N (CP_i + TrE_i) \quad (10)$$

where, N is the total number of IoT devices in the network.

The evaluation parameters for the IoT based network is presented in Table 7: The

Parameters	Parameter values
Number of sites	2
Number of sensor nodes in each site	10
Number of gateway in each site	1
Capacity limit	10 Mbps
Radio communication standard	IEEE 802.11

Table 7: Evaluation parameters for the designed network.

proposed power consumption model is simulated by taking the real value of the parameters considering the widely used CC2420 [CC2420,] radio transceiver and MSP430 [Bierl, 2000] micro-controller of motes.

– Result & Discussion II:

Figure 8 is depicting a graph that is plotted with the total power consumptions of the compression algorithms with varying traffic loads of the network when the data rate of the network is 500 kbps. Likewise, when the data rate is set to 1000 kbps for each

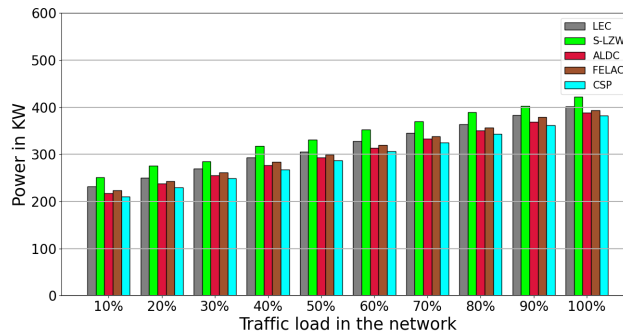


Figure 8: Total power consumption by the whole network when the link bit rate is 500kbps for each node of the network.

device of the network, the power consumption recorded is depicted by the Figure 9. From both the figures 8 and 9 it can be observed that, for every traffic load, the proposed CSP compression technique is consuming much lesser power than that

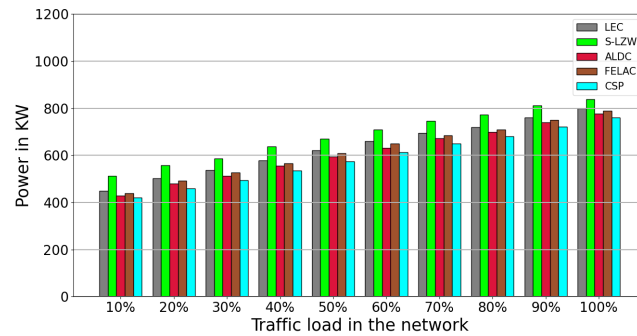


Figure 9: Total power consumption by the whole network when link bit rate is 1000kbps for each node of the network.

of the other compression schemes. Whereas, for both the cases the performance of S-LZW is the worst.

Compression scheme	Power consumption (in KW)	
	500 kbps	1000 kbps
S-LZW	340.26	686.35
ALDC	308.63	609.01
LEC	317.17	623.33
FELAC	309.66	621.69
CSP (proposed)	296.33	590.72

Table 8: Power consumption of different compression schemes in 500 kbps and 1000 kbps data rate.

In Table 8, the average power consumption of all the compression algorithms for data rates 500 kbps and 1000 kbps is depicted. From this table, it can be observed that the proposed CSP algorithm is consuming the lowest average power of 296.33 KW and 590.72 KW for a data rate of 500 kbps and 1000 kbps respectively. The proposed CSP algorithm is consuming 12.33 KW and 18.97 KW less power than its archival compression algorithm ALDC while the data rate of the network seems to be 500 kbps and 1000 kbps respectively.

5 Conclusion & Future Scope

In this paper, a lossless, power-saving, high-performance compression technique, CSP is proposed for an air pollution monitoring system. It has the highest Compression Factor and Compression Ratio too. The CSP technique is compared with some of the classical and state-of-the-art compression models, which proves that the proposed CSP technique is better in respect of every performance measuring parameter. The decompression is

also very simple and less expensive. As the CR% is much higher, the data transmission requires a lesser network burden in terms of bandwidth. Furthermore, from the Results & Discussion II it can be concluded that the power consumption of the proposed CSP technique is the least in comparison to the other compression schemes for both the data rates of 500 kbps and 1000 kbps. Thus, it is advantageous to compress data using CSP, as power-saving is maximum. As the data remains intact after decompression, CSP is a robust solution to the compression challenges. For this, CSP can be a reliable answer to the real-time applications of time-series data for Air Quality Monitoring Systems, where there is no scope for reconsideration of the analysis, and losslessness is the key constraint. The sensors become lesser liable for recharging or replacement just because of the extra shade off of power/energy. Its satisfactory performance in preserving the power is directly related to the longer lifetime of the network.

In the future, the proposed time-series data compression algorithm will be implemented on an Air Quality Monitoring Station to analyze its performance in a real-time environment. The CSP is mainly proposed for air quality time-series data. In the future, this research work will be extended to make this algorithm as a generic time-series data compression algorithm.

References

- [Abdulzahra et al., 2021] Abdulzahra, S. A., Al-Qurabat, A. K. M., and Idrees, A. K. (2021). Compression-based data reduction technique for iot sensor networks. *Baghdad Science Journal*, 18(1):0184–0184.
- [Al-Kadhimi and Al-Raweshidy, 2021a] Al-Kadhimi, H. M. and Al-Raweshidy, H. S. (2021a). Energy efficient data compression in cloud based iot. *IEEE Sensors Journal*, 21(10):12212–12219.
- [Al-Kadhimi and Al-Raweshidy, 2021b] Al-Kadhimi, H. M. and Al-Raweshidy, H. S. (2021b). Energy efficient data compression in cloud based iot. *IEEE Sensors Journal*, 21(10):12212–12219.
- [AndhraPradeshDataSet,] AndhraPradeshDataSet. Andhra pradesh data set. url<https://www.kaggle.com/datasets/abhisheksjha/time-series-air-quality-data-of-india-2010-2023>. [Andhra Pradesh Data Set].
- [Bhanja et al., 2022] Bhanja, S., Metia, S., and Das, A. (2022). A hybrid neuro-fuzzy prediction system with butterfly optimization algorithm for pm2. 5 forecasting. *Microsystem Technologies*, 28(12):2577–2592.
- [Bierl, 2000] Bierl, L. (2000). *MSP430 family mixed-signal microcontroller application reports*. Citeseer.
- [CC2420,] CC2420. Data-sheet of cc2420. <https://datasheetspdf.com/pdf/521406/Chipcon/CC2420/1>. [Data-sheet of CC2420].
- [Chen and Liu, 2024] Chen, H. and Liu, L. (2024). Acd: A lossless compression based on dimensionality reduction for multi-dimensional time series data. In *2024 6th International Conference on Frontier Technologies of Information and Computer (ICFTIC)*, pages 929–932. IEEE.
- [Chen et al., 2020] Chen, H.-C., Putra, K. T., Tseng, S.-S., Chen, C.-L., and Lin, J. C.-W. (2020). A spatiotemporal data compression approach with low transmission cost and high data fidelity for an air quality monitoring system. *Future Generation Computer Systems*, 108:488–500.
- [Chen et al., 2025] Chen, J., Fang, Y., Khisti, A., Özgür, A., and Shlezinger, N. (2025). Information compression in the ai era: Recent advances and future challenges. *IEEE Journal on Selected Areas in Communications*.
- [Chiarot and Silvestri, 2023] Chiarot, G. and Silvestri, C. (2023). Time series compression survey. *ACM Computing Surveys*, 55(10):1–32.

- [DelhiDataSet,] DelhiDataSet. Delhi data set. url<https://www.kaggle.com/rohanrao/air-quality-data-in-india>. Delhi Data Set.
- [Ghose et al., 2022] Ghose, B., Rehena, Z., and Anthopoulos, L. (2022). A deep learning based air quality prediction technique using influencing pollutants of neighboring locations in smart city. *Journal of Universal Computer Science*, 28(8):799.
- [Gopinath and Ravisankar, 2020] Gopinath, A. and Ravisankar, M. (2020). Comparison of lossless data compression techniques. In *2020 International Conference on Inventive Computation Technologies (ICICT)*, pages 628–633. IEEE.
- [Hanumanthaiah et al., 2019] Hanumanthaiah, A., Gopinath, A., Arun, C., Hariharan, B., and Murugan, R. (2019). Comparison of lossless data compression techniques in low-cost low-power (lclp) iot systems. In *2019 9th International Symposium on Embedded Computing and System Design (ISED)*, pages 1–5. IEEE.
- [Harnik et al., 2013] Harnik, D., Kat, R., Sotnikov, D., Traeger, A., and Margalit, O. (2013). To zip or not to zip: Effective resource usage for real-time compression. In *11th {USENIX} Conference on File and Storage Technologies ({FAST} 13)*, pages 229–241.
- [Jain et al., 2020] Jain, A., Goel, P., Aggarwal, S., Fell, A., and Anand, S. (2020). Symmetric k -means for deep neural network compression and hardware acceleration on fpgas. *IEEE Journal of Selected Topics in Signal Processing*, 14(4):737–749.
- [Kolo et al., 2015] Kolo, J. G., Shanmugam, S. A., Lim, D. W. G., and Ang, L.-M. (2015). Fast and efficient lossless adaptive compression scheme for wireless sensor networks. *Computers & Electrical Engineering*, 41:275–287.
- [Kolo et al., 2012] Kolo, J. G., Shanmugam, S. A., Lim, D. W. G., Ang, L.-M., and Seng, K. P. (2012). An adaptive lossless data compression scheme for wireless sensor networks. *Journal of Sensors*, 2012.
- [Le and Vo, 2018] Le, T. L. and Vo, M.-H. (2018). Lossless data compression algorithm to save energy in wireless sensor network. In *2018 4th International Conference on Green Technology and Sustainable Development (GTSD)*, pages 597–600. IEEE.
- [Li et al., 2020] Li, T., Wu, X., and Zhang, J. (2020). Time series clustering model based on dtw for classifying car parks. *Algorithms*, 13(3):57.
- [Liang and Li, 2014a] Liang, Y. and Li, Y. (2014a). An efficient and robust data compression algorithm in wireless sensor networks. *IEEE Communications Letters*, 18(3):439–442.
- [Liang and Li, 2014b] Liang, Y. and Li, Y. (2014b). An efficient and robust data compression algorithm in wireless sensor networks. *IEEE Communications Letters*, 18(3):439–442.
- [Likas et al., 2003] Likas, A., Vlassis, N., and Verbeek, J. J. (2003). The global k -means clustering algorithm. *Pattern recognition*, 36(2):451–461.
- [Lin et al., 2004] Lin, C. H., Xie, Y., and Wolf, W. (2004). Lzw-based code compression for vliw embedded systems. In *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, volume 3, pages 76–81. IEEE.
- [Liu and Orban, 2008] Liu, H. and Orban, D. (2008). Gridbatch: Cloud computing for large-scale data-intensive batch applications. In *2008 Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*, pages 295–305. IEEE.
- [Liu et al., 2018] Liu, W., Mei, F., Wang, C., O’Neill, M., and Swartzlander, E. E. (2018). Data compression device based on modified lz4 algorithm. *IEEE Transactions on Consumer Electronics*, 64(1):110–117.
- [Lykkegaard et al., 2025] Lykkegaard, M. B., Nielsen, S. V., Upadhyay, A., Copeland, M. B., and Trénell, P. (2025). Data compression for time series modelling: A case study of smart grid demand forecasting. *arXiv preprint arXiv:2505.02606*.

- [Makatos et al., 2010] Makatos, T., Klonatos, Y., Marazakis, M., Flouris, M. D., and Bilas, A. (2010). Zbd: Using transparent compression at the block level to increase storage space efficiency. In *2010 International Workshop on Storage Network Architecture and Parallel I/Os*, pages 61–70. IEEE.
- [Manchanda and Sharma, 2020] Manchanda, R. and Sharma, K. (2020). Energy efficient compression sensing-based clustering framework for iot-based heterogeneous wsn. *Telecommunication Systems*, pages 1–20.
- [Marcelloni and Vecchio, 2009] Marcelloni, F. and Vecchio, M. (2009). An efficient lossless compression algorithm for tiny nodes of monitoring wireless sensor networks. *the computer journal*, 52(8):969–987.
- [Mathpal et al., 2017] Mathpal, D., Darji, M., and Mehta, S. (2017). A research paper on lossless data compression techniques. *IJRST-International Journal for Innovative Research in Science & Technology!*, 4(1).
- [Mochizuki and Komuro, 2024] Mochizuki, S. and Komuro, N. (2024). Power-efficient wireless sensor network using distributed compressed sensing for time-series environmental monitoring. *Journal of Communications*, 19(4).
- [Mohajer et al., 2024] Mohajer, A., Hajipour, J., and Leung, V. C. (2024). Dynamic offloading in mobile edge computing with traffic-aware network slicing and adaptive td3 strategy. *IEEE Communications Letters*.
- [Nassra and Capella, 2023] Nassra, I. and Capella, J. V. (2023). Data compression techniques in iot-enabled wireless body sensor networks: A systematic literature review and research trends for qos improvement. *Internet of Things*, 23:100806.
- [Nicolae, 2011] Nicolae, B. (2011). On the benefits of transparent compression for cost-effective cloud data storage. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems III*, pages 167–184. Springer.
- [NSW,] NSW. Rozelli data set. <https://www.dpie.nsw.gov.au/air-quality/search-for-and-download-air-quality-data>. [Rozelli Data Set].
- [Pal et al., 2025] Pal, H. S., Kumar, A., Vishwakarma, A., and Singh, G. K. (2025). A 2d electrocardiogram signal compression algorithm using 1d discrete wavelet transform. *Physical and Engineering Sciences in Medicine*, pages 1–12.
- [Salvador and Chan, 2007] Salvador, S. and Chan, P. (2007). Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580.
- [Sangeetha et al., 2017] Sangeetha, M., Betty, P., and Kumar, G. N. (2017). A biometric iris image compression using lzw and hybrid lzw coding algorithm. In *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, pages 1–6. IEEE.
- [Shehabi et al., 2016] Shehabi, A., Smith, S., Sartor, D., Brown, R., Herrlin, M., Koomey, J., Masanet, E., Horner, N., Azevedo, I., and Lintner, W. (2016). United states data center energy usage report.
- [Silué et al., 2024] Silué, N., Ouattara, S., Dosso, M., and Clément, A. (2024). Quantitative comparative study of the performance of lossless compression methods based on a text data model. *Open Journal of Applied Sciences*, 14(07):1944–1962.
- [Srisooksai et al., 2012] Srisooksai, T., Keamarungsi, K., Lamsrichan, P., and Araki, K. (2012). Practical data compression in wireless sensor networks: A survey. *Journal of network and computer applications*, 35(1):37–59.
- [Stojkoska and Nikolovski, 2017] Stojkoska, B. R. and Nikolovski, Z. (2017). Data compression for energy efficient iot solutions. In *2017 25th Telecommunication Forum (TELFOR)*, pages 1–4. IEEE.

[Summers and Engineer, 2008] Summers, T. and Engineer, S. A. (2008). Hardware based gzip compression, benefits and applications. *CORPUS*, 3(2.75):2–68.

[Wang et al., 2017] Wang, S.-S., Lin, P., Tsao, Y., Hung, J.-W., and Su, B. (2017). Suppression by selecting wavelets for feature compression in distributed speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(3):564–579.

[Wiseman, 2007] Wiseman, Y. (2007). The relative efficiency of data compression by lzw and lzss. *Data Science Journal*, 6:1–6.

[Yang and Mohajer, 2025] Yang, J. and Mohajer, A. (2025). Multi objective constellation optimization and dynamic link utilization for sustainable information delivery using pd-noma deep reinforcement learning. *Wireless Networks*, 31(2):1839–1859.

[Zhang, 2015] Zhang, J. (2015). Real-time lossless compression of soc trace data.