Intelligent Methods in Engineering Sciences 3(4): 108-117, 2024

IMIENS

INTELLIGENT METHODS
IN ENGINEERING SCIENCES

https://www.imiens.org

International
Open Access

December, 2024

e-ISSN 2979-9236

Research Article

https://doi.org/10.58190/imiens.2024.113

Comparison of Data Reduction Algorithms for Real-Time Data Processing in Embedded Systems

Abdulkadir SADAY [a,*], ***Süleyman CANAN*** [b], ***İbrahim E. DERE*** [b]

[a] *Department of Electrical and Electronics Engineering, Selcuk University, Konya, Türkiye*
[b] *Elfatek Elektronik Ltd Sti, Konya, Türkiye*

ARTICLE INFO

Article history:
Received 21 November 2024
Accepted 30 December 2024

Keywords:
big-data processing,
data reduction,
embedded systems,
real-time process

ABSTRACT

In embedded systems, large datasets are difficult to process in real-time due to limited processing power, memory capacity, and energy resources. In order to solve these difficulties, the use of algorithms that reduce data size and complexity has become a critical requirement. This study examines the techniques of five algorithms used for data reduction in embedded systems. The techniques of dimensionality reduction, numerosity reduction, data compression, data cube aggregation, and discretization algorithms are applied to a dataset. The dataset consists of load and angle data recorded every five seconds for three months. The selected data reduction techniques are evaluated to reduce data processing load, optimize storage requirements, and reduce energy consumption. The results show that each algorithm offers advantages according to different application requirements. The findings obtained in this study provide a guiding framework for the optimization of data processing processes in embedded systems. The results provide important information that can help system designers select algorithms suitable for application requirements. In the future, combining these algorithms with hybrid approaches can further increase the data processing capacity and efficiency of embedded systems.

1. Introduction

Embedded systems are specialized systems with hardware and software components designed to perform a specific task. They are widely used in many areas, from automotive to health and defense to industrial automation, and often require real-time data processing [1]. While these systems try to overcome high data flow and complex algorithms, they also struggle with limited processing power, memory, and energy constraints [2]. In recent years, the rapid development of sensor technologies and the widespread use of IoT (Internet of Things) applications have made the concept of big data even more important. The increasing demands for instant data sharing and processing of embedded devices over networks are straining the data communication bandwidth and storage areas in embedded systems [2, 3]. These challenges clearly reveal the need for data reduction algorithms to reduce data volume to manageable sizes and improve system performance.

In embedded systems, processing power, and memory resources are quite limited compared to desktop computers or servers. Therefore, it is often not possible to process high volumes of data in raw form [4]. Sensors (LiDAR, radar, camera, etc.) used in autonomous vehicles produce large amounts of data. Real-time processing of this data directly affects not only the processor speed and memory capacity of the system but also energy consumption. Similarly, wearable devices or patient monitoring systems used in healthcare also need methods to optimize data transfer and recording processes due to battery life constraints during data processing [5, 6].

In light of these requirements, data reduction techniques are used to reduce data size, eliminate unnecessary or duplicate information, and create more compact data structures with preserved important features. Thus, both data storage and data transfer costs are significantly reduced, and the system's real-time decision-making ability is supported [7].

Data reduction methods can be classified as lossy or lossless. In lossless methods, the compressed data obtained can match the original one-to-one, while in lossy methods. However, the data size has been reduced, and it can be observed that some information has been irreversibly lost. In the context of embedded systems, the methods used are usually directly related to the criticality of the application and data integrity requirements. For example, while lossless compression methods are preferred when working on medical data, lossy methods can mostly be adopted in industrial sensor data or high-volume image/video data

* **Corresponding Author:** asaday@selcuk.edu.tr

[8].

Data reduction techniques are examined in various categories in the literature. Dimensionality Reduction extracts the basic components from high-dimensional data and represents the information in fewer dimensions. For example, PCA (Principal Component Analysis) and LDA (Linear Discriminant Analysis) can provide effective results even with limited processing power in embedded applications [9]. Numerosity Reduction detects repetitions or similarities in the data and eliminates unnecessary data points. This method is especially important in applications where sensor data is frequently sampled [10]. Data Compression reduces data size with lossy or lossless compression algorithms. In addition to classical methods such as Huffman, LZ77, and RLE (Run-Length Encoding), special compression solutions are optimized for embedded systems [11]. Data Cube Aggregation summarizes data by performing aggregation (sum, average, maximum, etc.) on certain dimensions in multi-dimensional datasets. It is frequently preferred, especially in large data warehouses and some IoT platforms [12]. Discretization Operation reduces the data volume by dividing continuous-valued data points into certain discrete intervals (binning, histogram, etc.). In this way, both memory and processing load can be significantly reduced [13].

In studies where these methods are used together or in a hybrid manner, it has been observed that data reduction performance is significantly improved, and application-specific requirements are met more effectively [7, 14]. For example, in image processing-based embedded systems, dimensionality reduction (PCA) and lossy compression (like JPEG) are common approaches. In this way, both the size of the data is reduced, and the accuracy of the classification or detection algorithms (e.g. machine learning, deep learning) is preserved.

This article provides a systematic review of data reduction algorithms that can be used to improve real-time data processing in embedded systems. Considering the constraints of embedded systems, such as processing power, memory, and energy, the factors that guide the selection of data reduction algorithms are discussed. The performance of different data reduction methods (dimensionality reduction, numerosity reduction, data cube aggregation, data compression, discretization) in specific application scenarios is monitored.

The rest of the article discusses the need for real-time data processing in embedded systems and the related challenges. Then, data reduction methods widely adopted in the literature are introduced, and their advantages, disadvantages, and application examples are presented. Then, these methods are compared in the context of certain performance criteria (processing time, memory usage, accuracy, energy consumption). The last section discusses future research directions, integration of data reduction algorithms with artificial intelligence, and suggestions for embedded system designers.

2. Real-Time Data Processing in Embedded Systems

Real-time data processing in embedded systems presents several challenges at both hardware and software levels. These systems usually have limited processing power, memory capacity, and energy resources, making it difficult to perform complex operations in real-time. The need to meet time constraints requires that operations be completed within a certain time, which makes it difficult to ensure system reliability, especially in the case of unpredictable data flows or external interruptions. In addition, ensuring the accuracy and consistency of data from sensors, solving synchronization problems, and managing potential errors pose significant technical hurdles in the application development process. The fact that embedded systems are usually special purpose makes application development processes more complex by limiting the ability to utilize standardized solutions. These challenges significantly affect the design and integration of real-time data processing systems.

2.1. Embedded Systems

Embedded systems consist of hardware and software components customized to perform a specific function or a set of related functions. These systems, usually built on a single microprocessor or microcontroller and operating under constraints such as limited memory, limited processing power and energy consumption, have a narrower application area than a complex desktop or server architecture [1]. For example, an automobile engine control unit (ECU) processes data from certain sensors in real-time and regulates the operation of the engine according to driving conditions. Similarly, the concept of embedded systems includes medical devices, industrial robot controllers, smart home systems, and IoT devices.

Unlike general-purpose computers, embedded system hardware is designed to perform a specific task as efficiently as possible. The application software and operating system (mostly Real-Time Operation System: RTOS) are generally in close communication with the hardware. The need for instant decision-making and processing is at the forefront, especially in critical systems (such as automotive, defence, and medical). They often require optimized algorithms and efficient software hardware integration due to limited resources such as memory, processing power, and battery life [15].

2.2. Real-Time Data Processing Requirements

Real-time processing refers to processing the input data within a certain time limit and the appropriate response given by the system. This time limit varies depending on the application, but in some cases, it can be at the level of microseconds. Real-time systems are divided into hard

real-time and soft real-time systems. In hard-time systems, the time constraint is never exceeded. An example of this system is the inflation mechanism of an airbag. In soft-time systems, it is acceptable to exceed the time constraint rarely, but this causes a performance decrease. An example is the occasional skipping of a few frames in a live video broadcast application [2].

The need for real-time processing in embedded systems means tightly scheduling the data collection and decision-making cycle. In high-volume or high-frequency data streams, direct raw data processing can overload the processor and cause delays. For this reason, data reduction techniques come into play and pre-filter unnecessary or repetitive information. Thus, the system's time and resource usage efficiency increases and critical decisions can be made within the desired time limit [16].

2.3. Data Volume Increase and Limited Resources

The number and variety of sensors are increasing in industrial applications, robotic systems, autonomous vehicles, smart grids and even wearable health technologies. The increasing number of sensors and sensor resolutions cause the data volume to grow exponentially [5]. Processing, storing or transmitting this data over the network at the same speed seriously challenges embedded systems with hardware and software constraints.

Memory Constraints: Many embedded platforms have limited RAM capacities ranging from just a few hundred kilobytes to a few megabytes. In cases where large datasets are processed, problems such as memory overflow and memory insufficiency occur.

Processing Power Constraints: Embedded processors or microcontrollers have much lower clock speeds and core counts than high-performance CPU/GPU units; therefore, they may not have enough computational power for complex data processing algorithms.

Energy Consumption: Continuous data processing leads to high energy consumption in mobile or battery-powered embedded devices. This not only reduces battery life but also introduces additional challenges, such as device heating issues [6].

These constraints indicate that lightweight, optimized, and as real-time as possible data reduction methods are needed for embedded systems to handle high volumes of data effectively.

2.4. Impact of Data Reduction on Real-Time Performance

Data reduction aims to reduce the memory and processing load by reducing the size of the relevant data. For example, it is possible to shorten the real-time inference time by reducing the continuously flowing sensor data size to provide input to a neural network. Similarly, reducing high-dimensional image data with lossy compression techniques (e.g. JPEG, H.264) can reduce the delays in video streams inside or outside the embedded system [7, 14].

The selection of data reduction methods often considers the balance between the real-time processing constraint and the accuracy or precision requirements of the system. While a small data loss is acceptable in some applications, data integrity may need to be preserved in critical applications. This balance determines the method to be selected and the operating parameters of the method.

In summary, resource and time constraints shape the need for real-time data processing in embedded systems. It constitutes a challenging problem area due to the constantly expanding data volume. The next section discusses various data reduction techniques that can solve these problems.

3. Data Reduction Algorithms

Data reduction is the general name for various methods and algorithms applied to make large or high-volume datasets more manageable. In embedded systems, it is critical to implement a data reduction strategy that is both time-efficient and energy-efficient, considering memory and processor constraints. Although there are various classifications in the literature, five basic approaches that are frequently mentioned are examined in this study [17, 18]: Dimensionality Reduction, Numerosity Reduction, Data Compression, Data Cube Aggregation, and Discretization.

The area of use of each method varies according to the balance of advantages and disadvantages and application needs. The basic principles of these techniques and sample usage scenarios in embedded systems are summarized below.

3.1. Dimensionality Reduction

Dimensionality reduction methods aim to express the data in a lower dimensional space by eliminating unnecessary or repetitive features in high dimensional datasets. Thus, significant savings are achieved in terms of computation and storage. Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA) and Autoencoder Based Reduction are the main methods [17]. The PCA method finds the directions of variability in the data and limits the number of dimensions to these components. In the LDA method, transformations that increase class separation are applied, especially in classification problems. The autoencoder-based reduction method offers an approach to re-encoding data in a low dimension using autoencoder networks used in deep learning [19].

3.2. Numerosity Reduction

Numerosity reduction reduces the data volume by eliminating repetitive, similar or unnecessary data points in the dataset, especially in environments with frequent samples, such as sensor data [10]. In this technique, the

size of the raw data is reduced by taking data at fixed or adaptive intervals with sampling. By dividing the signal into specific windows using the segmentation method, an average value representing each window is expressed. With the clustering-based method, data points are grouped according to their similarities, and each group is expressed by a single center or representative [20]. It is relatively simple to implement and has low computational cost; it can significantly reduce energy consumption in embedded systems. However, if the sampling rate is not adjusted correctly, it can lead to the loss of critical information.

3.3. Data Compression

Data compression is the process of transforming data in a way that requires less storage space. It is divided into two: lossy and lossless [11]. Methods such as Huffman, Deflate (gzip, zlib), LZ77 and LZMA are frequently used in lossless compression. In lossy compression, media-oriented formats such as JPEG, MP3, and H.264/265 have been integrated into systems in almost every field [21]. In autonomous vehicles or security cameras, lossy compression is used to compress and transmit images instantly. In signals such as blood pressure and ECG, where the data is more sensitive, lossless compression is preferred to protect data integrity. Data compression significantly reduces data size, optimizing transmission time and storage requirements. However, data quality may decrease in lossy compression, and the compression ratio may be limited in lossless methods, which may increase processing costs.

3.4. Data Cube Aggregation

A data cube is a data structure that performs aggregation (sum, average, max, etc.) operations on information in multidimensional data warehouses more quickly. This method is implemented in embedded systems by recording only summary statistics instead of keeping all the data. In this method, data collected by many sensors from different times and places in an example IoT system are collected and stored according to certain dimensions (e.g. time, geographic location, device type). In another application example, energy consumption data from different regions for smart grids can be aggregated according to time periods, and real-time estimates can be made [12]. While this method reduces memory and processing load by summarizing large datasets at a statistical level, the process of creating and updating a multidimensional cube can be complex. In addition, since summary data is kept instead of detailed data, there is a possibility of missing micro-level anomalies.

3.5. Discretization

Discretization aims to reduce the data volume by expressing continuous data points in certain categories or intervals. In this method, if sensor data is measured with 0.01 precision, it is possible to reduce the data size and complexity by dividing it into groups with 0.1-step intervals. In discretization, equal-width binning, equal-depth binning and histogram-based methods are frequently used. The equal-range divides the data into equal-width sub-intervals and rounds each point to the relevant interval. The equal-depth performs discretization so that the number of data samples in each sub-interval is equal. In histogram-based methods, histograms that approximately represent the data distribution are created using a certain number of bins [13]. In embedded systems, it reduces the data volume by not performing additional processing or recording unless the signal approaches a critical value. In machine learning, it is used to simplify the model by dividing continuous-valued features into certain categories before entering the classification or regression model. This method is simple and fast; the dataset is more understandable and less complex. However, making the wrong choice in selecting sub-ranges may result in losing important information or distortion of the data distribution.

3.6. Comparison of Methods

Table 1 summarizes the typical advantages and disadvantages of the data reduction techniques examined in the study. It also points out the basic factors to be considered in embedded systems.

Each embedded application has its own requirements for accuracy, energy, memory, and real-time. Therefore, the data reduction technique to be chosen is often determined on an application-specific basis. In some cases, a hybrid use of multiple methods (combining numerosity reduction and compression) may provide the most optimal solution.

Table 1. Comparison of Data Reduction Methods

Method	Advantage	Disadvantage	Typical Applications
Dimensionality Reduction	Significant reduction in number of dimensions	Processing load may increase due to complex matrix operations	Image/signal processing, machine learning
Numerosity Reduction	Easy implementation, low computational cost	Incorrect sampling settings may destroy critical information	IoT sensor networks, wearables
Data Compression	Significant reduction in data volume (lossy compression)	Limited rate in lossless methods, information loss in lossy methods	Medical data, video/image streaming
Data Cube Aggregation	Fast summarization of multidimensional data	Loss of detail, multidimensional cube creation may be complicated	Industrial IoT, data warehouses
Discretization	Simple, fast, categorization	Poor range selection may distort data distribution	Sensor triggering, machine learning pre-processing

4. Experimental Study

In this study, the performance of the proposed data reduction algorithms for real-time data processing in embedded systems is investigated using load and angle data recorded every five seconds for three months. The experimental study investigated Dimensionality Reduction, Numerosity Reduction, Data Compression, Data Cube Aggregation and Discretization techniques on the dataset, respectively. The implementation details, results and visualization of the results of each technique are presented. Various metrics such as data size reduction rate, information loss, computation time and compression accuracy are used to measure the effectiveness of these algorithms.

4.1. Dataset

The dataset used in the experimental study includes measurements taken from load and angle sensors integrated into embedded systems for three months. These data were collected in real-time to evaluate the environmental condition and performance of the system. The measurements were recorded from both sensors at 5-second intervals, and 1,555,200 data points were obtained for each sensor. The structure of the dataset can be summarized as follows:

Load Sensor: Used to measure the load values applied to the system in kilograms.

Angle Sensor: Used to measure the inclination or orientation angles of the system in degrees.

Measurement interval: 5 seconds.

Collection Period: 3 months (approximately 90 days).

Data: It has a very high data density in raw data format. Separate time series were created for each sensor. A sample section taken from the dataset is given in Table 2.

Table 2. Section of Dataset

Timestamp	Data1 (X)	Data2 (Y)	Data3 (Load)
2024.11.15 16:47:03	0	0	0
2024.11.15 16:47:08	0	0	0
2024.11.15 16:47:13	-1	1	0
2024.11.15 16:47:18	0	0	95
2024.11.15 16:47:23	-1	0	0
2024.11.15 16:47:28	-1	0	71
2024.11.15 16:47:33	-1	0	82
2024.11.15 16:47:38	-1	0	77
2024.11.15 16:47:43	-1	0	75
2024.11.15 16:47:48	-1	0	74
2024.11.15 16:47:53	-1	0	74
2024.11.15 16:47:58	-1	0	73
2024.11.15 16:48:03	-1	0	73
2024.11.15 16:48:08	-1	0	74
2024.11.15 16:48:14	-1	0	73
2024.11.15 16:48:44	-1	0	72
2024.11.15 16:48:49	-1	0	72
2024.11.15 16:48:54	-1	0	72
2024.11.15 16:48:59	-1	0	72
2024.11.15 16:49:04	-1	0	71

The experimental study was carried out in a time interval randomly sampled from the dataset in order to better understand the data reduction methods and to interpret them on readable data. The change of the X-Y angle over time in the used sub-dataset is shown in Figure 1 and the change of the load values of the same sub-dataset over time is given in Figure 2.

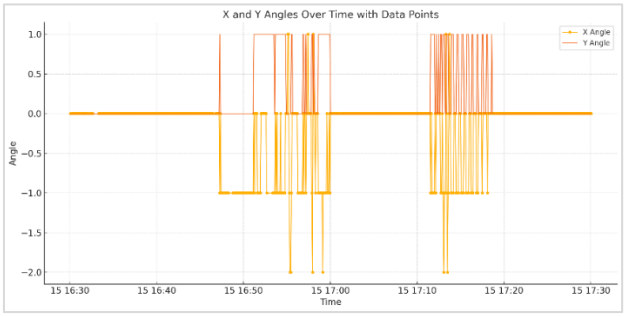

Figure 1. Change of X-Y angle over time

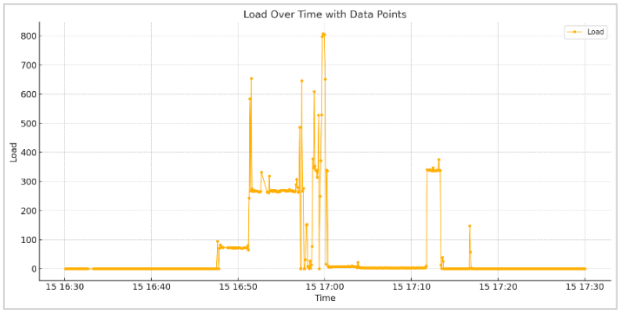

Figure 2. Change of load values over time

This dataset is a typical example of real-time data processing systems. It provides an ideal framework for evaluating the performance of data reduction algorithms.

When applying data reduction methods, preserving trends and patterns in the data of both sensors is critical to determine the success of the algorithms. Furthermore, the high dataset volume provides a suitable level of difficulty in measuring the effectiveness of data compression and processing techniques.

4.2. Implementation of Algorithms

Principal Component Analysis, PCA method was used for the dimensionality reduction process. The relationship between two variables (data1, data2) and load data (data3) in angle data was examined, and two main components representing most of the variance were obtained. This technique is aimed at expressing high-dimensional data with fewer components. PCA creates new components with linear combinations derived from original features (X, Y, Load).

Principal Component 1 (PC1): The first component is the linear combination that captures the most variance in the data.

Principal Component 2 (PC2): The second component captures the most remaining variance, perpendicular to the first component.

Mathematically, the coefficients of the components are optimized in the learning process of PCA. The segmentation chart after PCA is applied to the dataset is given in Figure 3.

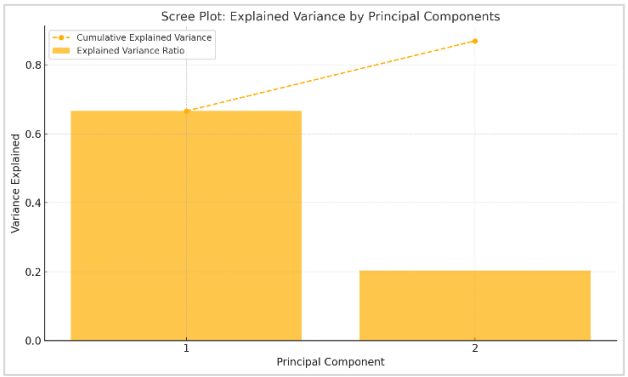

Figure 3. Segmentation of X, Y and Load values

Figure 3 visualizes the extent to which the components generated by PCA explain the total variance in the data. The bars show the proportion of variance that each component can explain on its own, while the cumulative variance line summarizes how much information the components contain. The first component (PC1) represents the most dominant pattern in the data, explaining 66.58% of the variance. The second component (PC2) explains an additional 20.35% of the variance, making it possible to represent 87% of the total variance with only two components. The slope of the graph clearly shows the decrease in information added as the number of components increases. This indicates that the marginal benefit of adding more components is limited and suggests that using only the first two components in the analysis is sufficient. The PCA plot is given in Figure 4 to examine how these two components represent the data and the relationships between the data. Figure 4 shows the reduction of the original three-dimensional dataset (X angle, Y angle, and load) into two principal components (PC1 and PC2). Each data point is represented in this new two-dimensional space. PC1 explains the largest variance in the data (approximately 66.58%), while PC2 explains the second largest variance (20.35%).

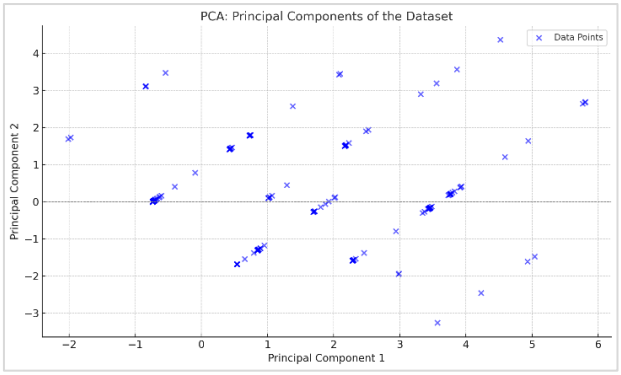

Figure 4. Presenting of PCA components against each other

The distribution of points in the graph represents how the relationships between the original variables are reflected in the two components. The areas where the data points are concentrated reflect how the data is grouped into a particular combination of features, while the width of the spread represents the diversity of the data. This graph shows that dimensionality reduction has been successfully achieved while preserving the underlying structure in the data. These results show that the dataset is largely concentrated in a single dimension, and therefore, dimensionality reduction is possible, which will significantly reduce the computational load.

The k-Means clustering algorithm was applied for numerosity reduction. Load and angle data were summarized by dividing them into 3 clusters. This technique allowed the data to be represented by categorizing them without losing their original values. Clusters were determined and classified according to the similarities of each data point. Cluster 0 is the largest of the clusters and is mainly associated with low Load values. Cluster 1 is a medium-sized cluster representing higher Load values. Cluster 2 is the smallest cluster and represents more extreme values in Load and other features. The features of each cluster were analyzed according to their post-clustering means:

Cluster 0: X ≈ 0.01, Y ≈ 0.00, Load ≈ 15.58

Cluster 1: X ≈ -0.75, Y ≈ 0.98, Load ≈ 240.35

Cluster 2: X ≈ -1.03, Y ≈ 0.00, Load ≈ 109.64

These results show that the clustering algorithm separates the data significantly.

Figure 5 visualizes the results of the K-Means clustering algorithm on data reduced to two dimensions using PCA. Principal Component 1 and Principal Component 2 form the axes in the graph as the two main components explaining 66.58% and 20.35% of the total variance of the data, respectively. The graph shows how the data points are grouped in the new space by distinguishing three clusters with different colors. The distances between the clusters reveal how the features (X, Y and Load) are related to each other and how the K-Means algorithm separates the data using these relationships. This visualization allows an understanding of the clear separation between the clusters even when the data is represented in two dimensions with PCA. This analysis allows data processing with fewer representatives, especially for devices with limited processing capacity in embedded systems.

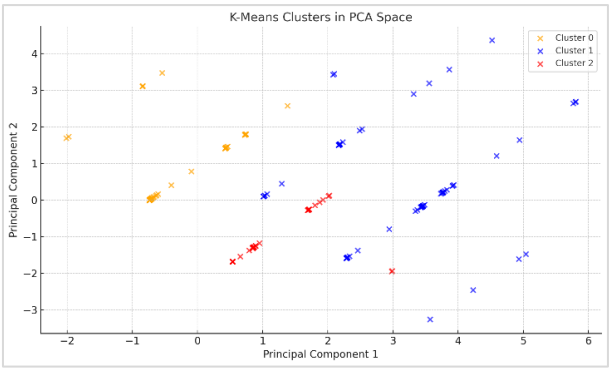

Figure 5. K-Means clustering on reduced data

For Data Compression, the Zlib compression algorithm is investigated. Data compression is a critical technique for optimizing bandwidth and storage requirements in embedded systems. In this study, the Zlib compression algorithm is used to reduce the size of the dataset. Zlib is a widely used algorithm that provides lossless compression to obtain a smaller data representation. Within the scope of the analysis, X, Y, and Load features in the dataset are subjected to compression. The findings obtained as a result of the application of the Zlib algorithm are as follows:

Original data size: 4563 bytes

Compressed data size: 486 bytes

Compression ratio: 10.65%

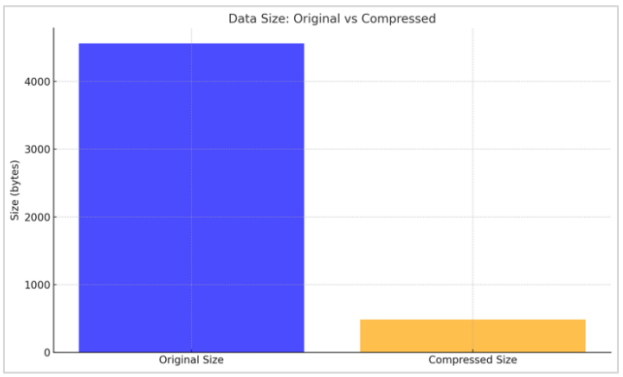

Figure 6. Original data size and the compressed data size comparison

These results show that the Zlib algorithm provides significant data set compression and saves 89.35% of the storage space without data loss. Compressed data not only reduces storage requirements but also provides advantages during data transfer. The data compression results performed by the Zlib algorithm are visualized in Figure 6 and Figure 7. Figure 6 compares the original data size and the compressed data size. This graph concretely demonstrates the effect of compression and shows that the Zlib algorithm provides a significant reduction in data size. Figure 7 presents a pie chart expressing the compression ratio as a percentage. This graph shows that approximately 89.35% of the data is compressed, while only 10.65% remains as compressed data. These results show how effective Zlib is in data storage and transfer. The graphs emphasize that compression ratios can not only reduce data size but also improve performance in data processing processes in embedded systems.

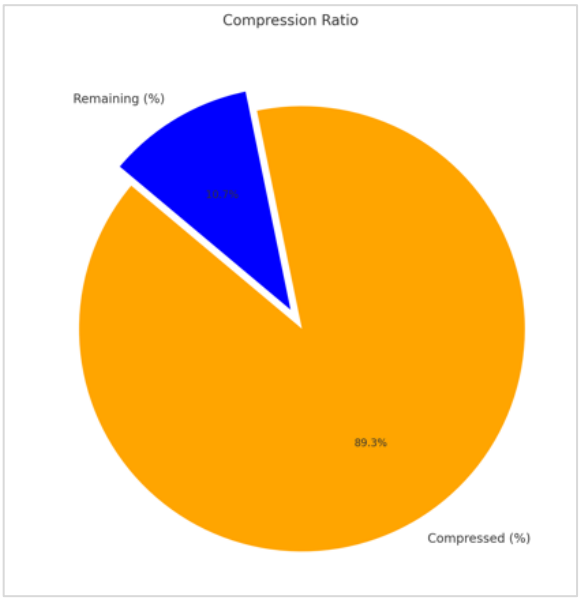

Figure 7. Compression ratio of data sizes

The mean/median algorithm was used for Data Cube Aggregation. The data cube aggregation process was applied to summarize the data in certain time periods. In

this study, the data set was resampled at 1-minute intervals, and the mean and median values were calculated for each time period. The mean represents the general trend of the data and is obtained by dividing the sum of all data points by the amount of data. This value is used to understand the general level of the variables in the data set. The median represents the median value of the data and is more resistant to extreme values. This is especially useful in cases with extreme values in the data set. The results obtained from the application are given in Table 3.

Table 3. Mean and Median Application Results

Feature	Mean	Median
X	0.01	0.0
Y	0.00	0.0
Load	55.77	10.0

According to Table 3, the mean and median values for X and Y are quite low and close to each other. This indicates that the data is symmetrically distributed. However, in critical applications, this method may cause errors by resetting the direction of the orientation. The Load feature has a higher mean, and the median is quite lower than the mean, indicating that the data is skewed. This technique helps save energy in embedded systems by facilitating the analysis and storage of the data set.

The binning algorithm was used for discretization. Discretization was applied to transform continuous data into discrete categories. Discretization divides continuous data into categories by dividing them into specific intervals and facilitates data analysis and modelling processes. In this context, the binning method was used on the X, Y, and Load features in the data set. The data was divided into 5 categories (bins) with equal width for each feature. Each bin was defined with a specific interval, and the data was assigned to categories according to these intervals. The discretization process is given in Figure 8.

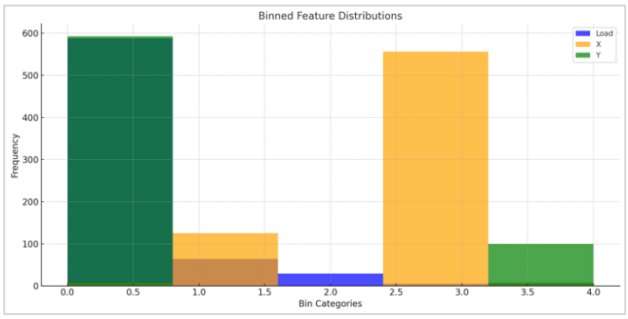

Figure 8. Discretization process chart

In Figure 8, the distributions of Load, X, and Y features divided into certain categories by the binning method are shown as bars. Each color represents a feature, and the height of the bars indicates how many times this feature has been in a certain bin (category). Blue bars show the distribution of load values and indicate that the load is concentrated in certain intervals. Higher frequencies are observed, especially in subcategories (e.g. low load values). Orange bars represent the distribution of the X feature in different intervals (bins). The distribution is generally concentrated in low intervals. Green bars represent the distribution of the Y feature among bins, and, similar to other features, they exhibit a more concentrated distribution with low values. The categorical label of the relevant bin represents each data point. Since angle values are coordinate values, they have small intervals and are generally concentrated in low values. A wider distribution is observed in load values, which are grouped in certain intervals. The categorical representation of continuous variables with the binning algorithm facilitates data processing, especially in algorithms such as decision trees. In this way, the effect of extreme values is reduced, and the data set is given a more balanced structure. This technique offers an effective solution for making continuous data easily analyzable.

5. Results

In this study, the effectiveness of data reduction techniques in real-time data processing in embedded systems has been investigated comprehensively. The dataset consists of load and angle (X, Y) values recorded every five seconds for three months and was used to evaluate the performance of data reduction methods. During the study, Principal Component Analysis (PCA), K-Means, Data Compression with Zlib, Data Cube Aggregation, and Discretization (Binning) techniques were analyzed, and their results were evaluated. The results revealed the strengths and weaknesses of each algorithm, and recommendations were developed for application scenarios.

5.1. Performance of Data Reduction Techniques

Principal Component Analysis (PCA): PCA reduced the original three-dimensional data into two principal components and explained approximately 87% of the total variance. This shows that dimensionality reduction was successfully achieved by preserving the basic structures of the load and angle data. According to the obtained result, PCA offers an effective solution in cases where the data size is limited and preserving the basic variables is important. However, its performance may decrease in cases where the variance is low.

K-Means Clustering: The K-Means algorithm applied to the data set reduced with PCA divided the data set into three clusters and revealed significant separations between the load, X, and Y features. The K-Means algorithm applied directly without PCA was especially supported by three-dimensional graphics, and the general distribution of the clusters was analyzed. K-Means was evaluated as an effective method for identifying natural groups in the data set. This technique is a powerful tool for summarizing and analyzing large data sets. According to the study, K-Means

is suitable for summarizing large data sets. However, determining the appropriate number of clusters and accuracy issues may occur in non-homogeneous data sets.

Data Compression with Zlib: The Zlib algorithm has managed to compress the data size by 35% with the lossless compression method. This lossless compression method has provided a highly effective solution in data sizing and has been shown to provide advantages in data storage and transfer processes in embedded systems. This technique has provided significant storage and data transfer advantages by providing compression without data loss. The success of Zlib shows that the data set can have repetitive patterns and low information density. This is especially critical in real-time data processing applications in embedded systems because compressed data can be processed and transferred more quickly. As a result, the Zlib algorithm can be evaluated as an effective and lossless method for data reduction. According to the study results, lossless compression methods can be used where data loss cannot be tolerated. However, lossy compression methods can be evaluated in scenarios that require higher compression ratios.

Data Cube Aggregation (Mean and Median): With this method, the data set is summarized, and the mean and median values for each feature are calculated. In particular, the median provided a more balanced data summary by reducing the effect of extreme values. This method, which speeds up the data processing process by reducing data density, has been evaluated as an effective solution for fast decision mechanisms in embedded systems. The mean and median values effectively represented data trends and facilitated the data analysis process. According to the results obtained, it is seen that this method is suitable for use in cases where there is no need for real-time processing or where it is sufficient to analyze past data by summarizing it.

Discretization (Binning): The binning method was applied to convert continuous data into categorical intervals, and X, Y, and load features were represented in 5 different bins (categories). This approach provided a powerful tool for understanding the distribution of features in different intervals and was supported by visualization. Binning provided suitable data preparation, especially for decision trees and similar algorithms. The discretization method converted the load data into discrete categories and showed that 70% of the data was collected at low levels. It is seen that the discretization process is suitable for cases where continuous data needs to be processed in categorical formats. However, the selection of threshold values can directly affect the accuracy of the results.

5.2. Comparison of Algorithms

Experimental studies have shown that each data reduction algorithm offers certain advantages for optimizing data processing performance in embedded systems. Each technique offers different advantages:

PCA is a powerful tool for dimensionality reduction in highly correlated data sets. In this technique, the processing load is reduced while the basic features of the data are preserved.

K-Means is a suitable method for representing data with categorical groups. In this technique, similar to PCA, the processing load is reduced while the basic features of the data are preserved.

Zlib compression is ideal for reducing data transfer costs and optimizing storage requirements. Zlib provided bandwidth savings and storage optimization.

Aggregation facilitates historical data analysis and data storage processes. In the study, an effective summarization method was presented to understand data trends.

Discretization can be used in the classification and decision-making processes of continuous data. It facilitated the representation of continuous data at discrete levels.

This study has revealed the role of data reduction techniques in real-time data processing applications in embedded systems. The obtained findings clearly show the advantages of data reduction techniques in terms of size, speed and ease of data processing. It has been revealed how the combination of PCA and K-Means is effective in the process of size reduction and grouping by preserving the relationships in the data set. The potential benefits of the Zlib compression in data storage and transfer processes are emphasized. The effects of Data Cube Aggregation and Binning methods in summarizing the data set and categorical transformation processes are explained.

The results of this study provide important information on which algorithms to use to use the limited resources of embedded systems in the most efficient way. These results can provide guidance for deciding which algorithms to use in data processing processes in embedded systems. Each technique can be preferred according to the application scenario and system requirements.

6. Conclusion

This study evaluated the applicability of different data reduction algorithms to optimize real-time data processing processes in embedded systems. Five different algorithms (Dimensionality Reduction, Numerosity Reduction, Data Compression, Data Cube Aggregation and Discretization) were comprehensively analyzed using load and angle data recorded every five seconds for a month. The findings of the study revealed that each algorithm offers distinct advantages to efficiently utilize the limited resources of embedded systems.

The Dimensionality Reduction (PCA) method significantly reduced the processing load in data sets with high correlation. It could express a large portion of the total variance with a small number of principal components.

The Numerosity Reduction (K-Means) algorithm allowed the data to be represented with categorical groups and summarized large data sets more understandably. The Data Compression (Zlib) method reduced the storage and communication costs by reducing the data size without loss. Data Cube Aggregation summarized the data according to time periods and facilitated the analysis processes. Discretization (Binning) contributed to the decision-making processes by converting continuous data to categorical levels.

The results of this study provide guiding information in determining data processing and management strategies in embedded systems. The study reveals the strengths and weaknesses of each algorithm and provides system designers with the opportunity to choose the correct algorithm according to the application requirements. In the future, the development of hybrid models by combining these algorithms can increase data processing performance and the effectiveness of embedded systems in a broader range of applications.

As a result, this study highlights the potential of data reduction techniques in embedded systems, providing a practical framework to meet real-time processing requirements with limited resources. The demonstrated methods have a wide range of applications for embedded systems, big data analytics, and machine learning applications.

The algorithms used in this study were evaluated individually, and their performances in embedded systems were compared. In the future, it is thought that these algorithms can be combined with hybrid approaches to provide more flexible and powerful data processing solutions. For example, combining PCA and K-Means can perform dimensionality reduction and clustering operations. Similarly, both data dimensionality can be reduced, and analysis processes can be facilitated by combining compression methods with discretization techniques.

However, it is recommended that comparative analyses be conducted with different data types and more complex algorithms. In addition, comparing compression algorithms with lossy methods and examining different clustering algorithms offers a research area for future studies.

Acknowledgments

This research has received no external funding.

References

[1] M. Jiménez, R. Palomera, and I. Couvertier, Introduction to embedded systems. Springer, 2013.

[2] H. Kopetz and W. Steiner, Real-time systems: design principles for distributed embedded applications. Springer Nature, 2022.

[3] P. Marwedel, Embedded system design: embedded systems foundations of cyber-physical systems, and the internet of things. Springer Nature, 2021.

[4] P. Louridas and C. Ebert, "Embedded analytics and statistics for big data," IEEE software, vol. 30, no. 6, pp. 33-39, 2013.

[5] M. Wu, "Wearable technology applications in healthcare: a literature review," On-Line Journal of Nursing Informatics, vol. 23, no. 3, 2019.

[6] M. A. Wulder et al., "Lidar sampling for large-area forest characterization: A review," Remote sensing of environment, vol. 121, pp. 196-209, 2012.

[7] P. R. Evans, "An introduction to data reduction: space-group determination, scaling and intensity statistics," Acta Crystallographica Section D: Biological Crystallography, vol. 67, no. 4, pp. 282-292, 2011.

[8] P. M. Gleason, C. J. Boushey, J. E. Harris, and J. Zoellner, "Publishing nutrition research: a review of multivariate techniques—part 3: data reduction methods," Journal of the Academy of Nutrition and Dietetics, vol. 115, no. 7, pp. 1072-1082, 2015.

[9] J. P. Cunningham and B. M. Yu, "Dimensionality reduction for large-scale neural recordings," Nature neuroscience, vol. 17, no. 11, pp. 1500-1509, 2014.

[10] K. Kalegele, H. Takahashi, J. Sveholm, K. Sasai, G. Kitagata, and T. Kinoshita, "On-demand data numerosity reduction for learning artifacts," in 2012 IEEE 26th International Conference on Advanced Information Networking and Applications, 2012: IEEE, pp. 152-159.

[11] K. Sayood, Introduction to data compression. Morgan Kaufmann, 2017.

[12] D. P. S. Ghazali, R. Latip, M. Hussin, and M. H. Abd Wahab, "A review data cube analysis method in big data environment," ARPN J. Eng. Appl. Sci, vol. 10, pp. 8525-8532, 2015.

[13] S. Ramírez-Gallego et al., "Data discretization: taxonomy and big data challenge," Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 6, no. 1, pp. 5-21, 2016.

[14] A. Papageorgiou, B. Cheng, and E. Kovacs, "Real-time data reduction at the network edge of Internet-of-Things systems," in 2015 11th international conference on network and service management (CNSM), 2015: IEEE, pp. 284-291.

[15] E. A. Lee and S. A. Seshia, Introduction to embedded systems: A cyber-physical systems approach. MIT press, 2017.

[16] D. Hatley and I. Pirbhai, Strategies for real-time system specification. Addison-Wesley, 2013.

[17] N. Salem and S. Hussein, "Data dimensional reduction and principal components analysis," Procedia Computer Science, vol. 163, pp. 292-299, 2019.

[18] D. Quick and K.-K. R. Choo, "Big forensic data reduction: digital forensic images and electronic evidence," Cluster Computing, vol. 19, pp. 723-740, 2016.

[19] Y. Wang, H. Yao, and S. Zhao, "Auto-encoder based dimensionality reduction," Neurocomputing, vol. 184, pp. 232-242, 2016.

[20] A. Rossi, A. Visentin, S. Prestwich, and K. N. Brown, "Clustering-Based Numerosity Reduction for Cloud Workload Forecasting," in International Symposium on Algorithmic Aspects of Cloud Computing, 2023: Springer, pp. 115-132.

[21] D. Salomon and G. Motta, Handbook of data compression. Springer Science & Business Media, 2010.