Instituto Superior
de Engenharia
Politécnico de Coimbra

DEPARTMENT OF SYSTEMS AND COMPUTER ENGINEERING

Data Reduction Techniques for Edge Environments

Dissertation to fulfil the Master's degree in Informatics Engineering

Specialization Area of Software Engineering

Author

Vítor Hugo Silva Fernandes

Supervisors

Prof. Dr. Jorge Bernardino, ISEC, Polytechnic University of Coimbra

Co-Supervisors

Prof. Dr. Vasco Pereira, CISUC, University of Coimbra
Prof. Gonçalo Carvalho, CISUC, University of Coimbra

INSTITUTO POLITÉCNICO
DE COIMBRA

INSTITUTO SUPERIOR
DE ENGENHARIA
DE COIMBRA

Coimbra, December 2025

ABSTRACT

The rapid growth in data production has made managing, storing, and transmitting large volumes of information increasingly challenging, especially in environments with limited resources. This work presents a comparison between different data reduction techniques. A comprehensive review of the state of the art was conducted to identify and categorize algorithms focused on their characteristics, forming the basis for a structured taxonomy. The pipeline used for tests operates on a Raspberry Pi device and performs data preparation, reduction and metric collection using commonly datasets encompassing different data types, including temporal, numeric, and textual data. The evaluation results showed that JPEG and Block Averaging are effective at reducing image data, while MP3 excels at compressing audio, as for numeric data and temporal data BZip2 had the best results. This work aims to analyse various data reduction algorithms and assess their suitability for different types of data.

Keywords: resource-constrained environments, data reduction, data types.

RESUMO

O rápido crescimento na produção de dados tornou cada vez mais desafiante a gestão, o armazenamento e a transmissão de grandes volumes de informação, especialmente em ambientes com recursos limitados. Este trabalho apresenta uma comparação entre diferentes técnicas de redução de dados. Foi realizada uma revisão abrangente do estado da arte para identificar e categorizar algoritmos com foco nas suas características, formando a base para uma taxonomia estruturada. O pipeline utilizado para os testes opera num dispositivo Raspberry Pi e realiza a preparação, redução e recolha de métricas de dados utilizando conjuntos de dados comuns que abrangem diferentes tipos de dados, incluindo dados temporais, numéricos e textuais. Os resultados da avaliação mostraram que JPEG e Block Averaging são eficazes na redução de dados de imagem, enquanto MP3 se destaca na compressão de áudio. Quanto aos dados numéricos e temporais, o BZip2 apresentou os melhores resultados. O objetivo deste trabalho é analisar diferentes algoritmos de redução de dados e avaliar a adequação dos mesmos a diferentes tipos de dados.

Palavras-chave: ambientes com recursos limitados, redução de dados, tipos de dados

ACKNOWLEDGEMENTS

This master's project marks the culmination of a long and rewarding journey. I am deeply grateful to Professor Jorge Bernardino, Professor Vasco Pereira, and Professor Gonçalo Carvalho for their invaluable guidance, constant encouragement, and steadfast belief in my potential. Their support was essential to the successful completion of this work.

I would also like to thank the Instituto Superior de Engenharia de Coimbra, and in particular, the professors, for providing the resources, knowledge, and opportunities that made this research possible. I am equally grateful to my master's colleagues for their collaboration, encouragement, and companionship throughout this journey.

Finally, I extend my heartfelt thanks to my friends and family for their patience, understanding, and emotional support, which gave me the strength to overcome challenges and reach this important milestone.

"The future belongs to those who believe in the beauty of their dreams."

Eleanor Roosevelt

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Acronyms

EtC	Encryption-then-compression
IDC	International Data Corporation
IoT	Internet of Things
LSD	Log-Spectral Distortion
LZW	Lempel–Ziv–Welch
PCA	Principal Component Analysis
PSVM	Polynomial Support Vector Machines
SAX	Symbolic Aggregate Approximation
SE	Spectral Embedding
UMAP	Uniform Manifold Approximation and Projection
ViT	Vision Transformer
WSN	Wireless Sensor Network

1 INTRODUCTION

Over the past decade, technological advancements such as smartphones, smart homes, and connectivity technologies have revolutionized how society interacts with the world, leading to a massive volume of data. Smartphones have become essential to our daily lives, capturing moments, tracking activities, and facilitating communication. Until 2003, humans had produced 5 exabytes of data, but today, the same amount is generated in just two days [1]. The International Data Corporation (IDC) predicts that by 2025 there will be 41.6 billion IoT devices creating 79.4 ZB of data [2].

Besides all the benefits data brings to our society, it also presents some challenges. The challenges we are currently facing relate to the five Vs of Big Data: volume, velocity, value, variety and veracity [3]. This document will focus on reducing the volume of data while applying different data reduction techniques, ensuring that both data value and veracity are maintained.

The earliest studies in data reduction can be traced back to the field of information theory, which was pioneered by Claude Shannon in the mid-20th century. His work established the basis for our understanding of the fundamental limits of data compression and the trade-offs between data size and information content.

Data reduction techniques are essential for optimizing storage, processing, bandwidth consumption, and analysis. The process of data reduction involves minimizing the size or complexity of data while preserving its essential characteristics and minimizing information loss.

Strategically employing data reduction techniques streamlines data transfer and storage processes. The most effective way to perform this task is to implement these techniques in middle servers or gateways, where data can be compressed or aggregated before being sent to the cloud. This approach significantly reduces bandwidth usage and improves efficiency. Alternatively, if energy and computation power are not an issue, data reduction should be applied directly at the sensor level using techniques such as compression or filtering. These techniques offer valuable ways to optimize data handling and improve overall performance.

This research project involves analysing and experimenting with various data reduction techniques across different datasets in order to evaluate their performance. As part of this study, we introduce a new taxonomy for these techniques, categorizing them as either lossy or lossless. Categorizing them based on data loss characteristics provides insight into the trade-offs between reducing data and preserving critical features. Ultimately, our goal is to contribute to the advancement of data reduction techniques by offering practical evidence and recommendations for their implementation.

1.1 Motivation

As discussed in the previous section, the production of data has undoubtedly brought numerous benefits to modern society. However, this expansion of data also presents inherent challenges and costs. The massive volume of data generated can limit the extraction of relevant features, hinder the seamless exchange of data across different locations and make it difficult to store this information efficiently in the cloud. These challenges are further complicated by other complexities, making data management and utilization more difficult.

To address the exponential growth in data generated by distributed devices such as Internet of Things (IoT) sensors, autonomous systems, and mobile platforms, researchers have been investigating new techniques to reduce the volume of data transmitted to centralized cloud servers. One such approach is edge computing, a distributed computing paradigm in which data processing and storage are performed closer to the data source, at the "edge" of the network. This architecture typically involves a hierarchy of nodes, ranging from low-power edge devices (e.g., sensors, gateways) to more capable edge servers located within local or regional facilities. By offloading computation from the cloud to edge nodes, edge computing reduces network congestion, lowers communication latency, improves energy efficiency, and enhances data privacy and security. Moreover, this decentralized model enables real-time or near-real-time analytics, which is critical for time-sensitive applications such as autonomous driving, industrial monitoring, and smart healthcare [4], [5].

Although edge computing has many advantages, edge devices often have limited processing power, memory, energy and storage capacity. These limitations make it challenging to perform complex computations or store large volumes of data locally. Additionally, while edge computing reduces the need for constant communication with the cloud, periodic data transmission is still necessary for tasks such as model updates, long-term storage, or large-scale analytics. As a result, there is a growing need for efficient data reduction techniques, such as data compression, filtering, feature extraction, and summarization, that can minimize the volume of data without compromising the quality or utility of the information. These techniques not only alleviate resource constraints on edge devices but also further reduce bandwidth consumption and improve overall system responsiveness.

To address this issue, we examine various algorithms across different datasets containing various data types. We evaluate the accuracy and computational efficiency of these algorithms. Through experiments involving selected datasets and analysis of the performance metrics of the algorithms, we aim to provide valuable insights into their effectiveness in reducing data volume while preserving essential information. This comprehensive approach provides a thorough understanding of the strengths and limitations of each algorithm, enabling informed decision-making in real-world data reduction scenarios.

1.2 Objectives

The growing volume and variety of data have rendered data reduction an essential phase in contemporary data processing pipelines, especially in resource-limited contexts. The goal of this thesis's is to facilitate well-informed algorithm selection by providing helpful advice for selecting appropriate approaches in a range of application scenarios.

1.3 Contributions

This work presents several key contributions to the field of data reduction. Firstly, it offers a comprehensive analysis of the current state-of-the-art techniques, providing a solid foundation for understanding existing approaches. Secondly, it introduces a structured categorization of data reduction methods, organized according to their algorithmic characteristics. Furthermore, the study includes an extensive experimental evaluation, detailing the experimental setup, proposing a data reduction pipeline, and thoroughly analyzing the obtained results. The research findings were published in a peer-reviewed article in the journal Applied Sciences, which highlights their academic importance [6].

Lastly, the work was presented at the national conference INForum 2023, both through an oral communication and a poster session, enhancing visibility and fostering engagement and collaboration within the research community [7].

1.4 Structure of the Document

This document is structured to provide a clear and logical progression through the various stages of the project. The rest of the document is structured as follows:

Section 2 provides a review of the state of the art, examining recent advancements in data reduction and discussing the algorithms and techniques currently available.

Section 3, presents the proposed taxonomy, offering a new way to categorize data reduction methods based on insights gathered from the literature.

Section 4 presents the algorithms chosen for experimental evaluation, along with practical examples that illustrate their application.

Section 5 describes the experimental setup in detail, including the data reduction pipeline, the evaluation metrics, the datasets used, the results obtained, and an analysis of which algorithms are best suited to each scenario.

Finally, Section 6 concludes the document by summarizing the main findings and suggesting possible directions for future work.

2 RELATED WORK

Data reduction is a major research topic in many areas, including big data and the IoT. The primary objective of data reduction strategies is to minimize the storage requirements of a dataset. This section summarizes some of the existing work in this area.

Obaise, Salman, and Lafta [8] explored a solution in the IoT gateways that converts time series domains to frequency domains to extract patterns or trends in the streamed data. The solution presented in the paper could reduce cloud storage and bandwidth consumption and prevent I/O bottlenecks. However, the paper did not present the accuracy and reduction percentage, and there is not enough data to calculate it.

Mahmoud, Moussa, and Badr [9] introduced a domain-independent IoT-based spatiotemporal data reduction approach for IoT-structured data, which consists of treating spatial data with the K-Means algorithm to preserve location information and using a data similarity technique on a time basis to preserve temporal data information. The presented solution achieves an average data reduction of 54% with an average accuracy of 95%.

Fathy, Barnaghi, and Tafazolli [10] presented a data reduction technique based on data compression. To reduce the energy consumption when transmitting data to the gateway, the authors proposed an approach in the sensor nodes. The proposed approach consists of two stages: the first stage involves a lossy SAX quantization stage to minimize the dynamic range of the sensor readings. The final stage is a lossless LZW compression to compress the output of the first stage. The results obtained are a reduction of more than 90% of the produced data, with only 4.74% not reaching the gateway in the worst case. It remains an open possibility to propose a dynamic compression algorithm that can convert from lossless to lossy-based parameters.

Habib et al. [11] present a set of algorithms and techniques to reduce high-volume and high-velocity data in Big Data environments. This survey shows some dimension-reduction approaches to reduce the heterogeneity and massively variable data in-to manageable data and indicates deduplication techniques and redundancy techniques. It presents a taxonomy of the best and most-used data compression algorithms in terms of decompression overhead. The authors conclude that many efficient data reduction techniques achieve good performance in reduction and accuracy, but they still need more focus from researchers.

Dias, Bellalta, and Oechsner [12] analysed several prediction-based data reduction approaches to decide which technique is the best to reduce energy consumption in wireless sensor networks. The main contribution was to emphasize not only the introduction of the prediction techniques, but also the methods used in data reduction solutions for WSNs. The authors conclude that the IoT path will depend

on the scalability of the sensor networks and their ability to self-access the wireless medium. In addition, the reduction in transmissions based on prediction-based data reduction algorithms depends on the sensed phenomena, the user requirements, and the architecture used to make the predictions.

Chhikara et al. [13] proposed a taxonomy to show the different data dimensionality reduction techniques. The results gathered by the authors with feature extraction techniques show that the best algorithm may vary from data singularities PCA, SE, and UMAP were the extraction techniques that gathered the best results in more than one dataset. Among the feature selection techniques, Random Forest and backward feature elimination achieved a reduction percentage of 60%, and the best feature selection technique is forward feature selection, with a reduction percentage of 68%. It was possible to conclude that data dimensionality reduction techniques make it much easier and faster to process and analyze data by machine learning algorithms and human input.

Azar et al. [14] proposed an energy-efficient approach for IoT data collection and analysis. The proposed technique consists of a data compression technique based on a fast error-bounded lossy compressor conducted on the collected data before sending them to the edge layer. Then, after sending the data to the edge, a supervised machine learning technique is applied. The authors' results proved that the data can be reduced up to 103 times without affecting the data quality. Energy savings are up to 27% after 4 h, and the data accuracy is 98%. The algorithm is better suited for multisensory reading compression. It remains open to try to increase the energy reduction to increase the lifetime of the IoT network.

Papageorgiou, Cheng, and Kovacs [15] created a solution that addresses the two main bottlenecks of the analyzed solutions. The first bottleneck is the inability of time series network-reduction data reduction techniques to make decisions on an item-by-item basis. The other bottleneck is the lack of systems that can apply any data reduction method without heavy reconfiguration or significant delays. Despite the ability of the proposed solution to handle streamed data, it has inferior results when compared to other a posteriori data reduction techniques. Its accuracy may depend on the domain in which it is applied. It remains to further investigate the "streamification" of data reduction techniques and adaptation of data reduction techniques to specific use cases. In addition, the authors do not present the original size of the studied data in the paper.

Hanumanthaiah et al. [16] gather information about compression techniques and design a solution within a node that compresses data using two lossless compression techniques. The author's concern in designing the method was to create a system that can compress data at a minimal energy cost in low-power devices. The system achieved a compression ratio of 52.67% and a maximum space-saving (free space after compression) of 46.1%. However, the paper does not provide detailed information on the type, size, and accuracy of the data examined.

Mcfarlane J, Chakravarthi B [17] proposed a machine learning-based method for classifying MP3 compression levels using audio analysis statistics, aiming to improve audio quality assessment in lossy formats. Their approach involved training classifiers such as PSVM and XGBoost on features derived from statistical measures, spectral indices, and structural similarity metrics. The PSVM and XGBoost models achieved high accuracies of 98.3% and 97.0%, respectively, in identifying original compression levels across various bitrates (128-320 kbps). While the models demonstrated strong performance on original files, challenges remained in detecting transcoded samples. The study highlights the potential of automated, feature-driven methods for audio quality evaluation and calls for further enhancements through advanced statistical measures and expanded datasets.

Hagihara et al. [18] examined the effects of JPEG compression on ViT image classification within an EtC framework. Their findings demonstrate that JPEG compression significantly reduces the size of encrypted images while maintaining high classification accuracy - exceeding 98% at a quality factor of 85 with over 80% data reduction. Compared to linear quantization, JPEG compression achieved superior classification and compression performance. The study evaluated two schemes: one using uncompressed training images (Scheme A) and another using JPEG-compressed training data (Scheme B), both yielding minimal accuracy degradation (maximum of 1.22%). Furthermore, compression-induced noise, particularly in high-frequency components, did not hinder accuracy and may enhance robustness. The results support JPEG compression as a practical solution for efficient, privacy-preserving image classification.

Table 1 compares the different tests done with different algorithms in the studied papers.

In summary, this reviewed literature highlights significant advancements in data reduction techniques across various domains. While existing approaches show promising results in terms of compression ratios, accuracy, and energy efficiency, many solutions are still constrained by their reliance on specific data types, lack of adaptability, and inability to process data in real-time. Furthermore, numerous studies overlook critical metrics such as reduction percentages, computational overhead, and scalability performance. These gaps highlight the need for further research focused on summarize and experiment different data reduction techniques in different data types.

Table 1 - Related work data reduction techniques: comparison.

Algorithm	Accuracy	Dataset Size	Reduction Percentage	Reference
Subtractive Clustering Algorithm	NA [1]	NA [1]	~97%	[8]
Spatiotemporal Data Reduction	95%	843 KB, 960 KB	~54%	[9]
Perceptually Important Points, Sampling, Piecewise Approximation	76.3% to 93.8%	NA [1]	~66%	[15]
Run Length Encoding and Delta lossless compression	NA [1]	NA [1]	~52%	[16]
Pattern System	NA [1]	2880 lines	NA [1]	[19]
DCT and mixing operation of scrambled image	100%	NA [1]	44.16%	[20]
CCSDS and Scalable encryption scheme	100%	NA [1]	12.19%	[21]
JPEG XR compression and Block Scrambling	100%	NA [1]	60.77%	[22]

[1] Not Available.

3 PROPOSED TAXONOMY

Data reduction techniques cover a range of approaches that can vary in their data loss characteristics. For some organizations, the amount of data loss can be a critical factor in their algorithm selection and decision-making processes.

We believe that the taxonomy presented in this section addresses a gap in the reviewed literature by explicitly considering data loss in data reduction strategies [23], [24].

The proposed taxonomy, illustrated in Figure 1, provides a structured framework for categorizing data reduction techniques. There are two fundamental levels: the first level focuses on the existence of data loss after the application of data reduction algorithms, and the second level focuses on algorithmic properties. This taxonomy provides a methodological approach to organize and understand the various data reduction techniques. The visual representation of the taxonomy makes it easier for researchers and practitioners to navigate and understand the techniques' different categories and subcategories. This taxonomy ultimately improves the clarity and accessibility of data reduction techniques, making them easier to evaluate and select for specific data management and analysis purposes.

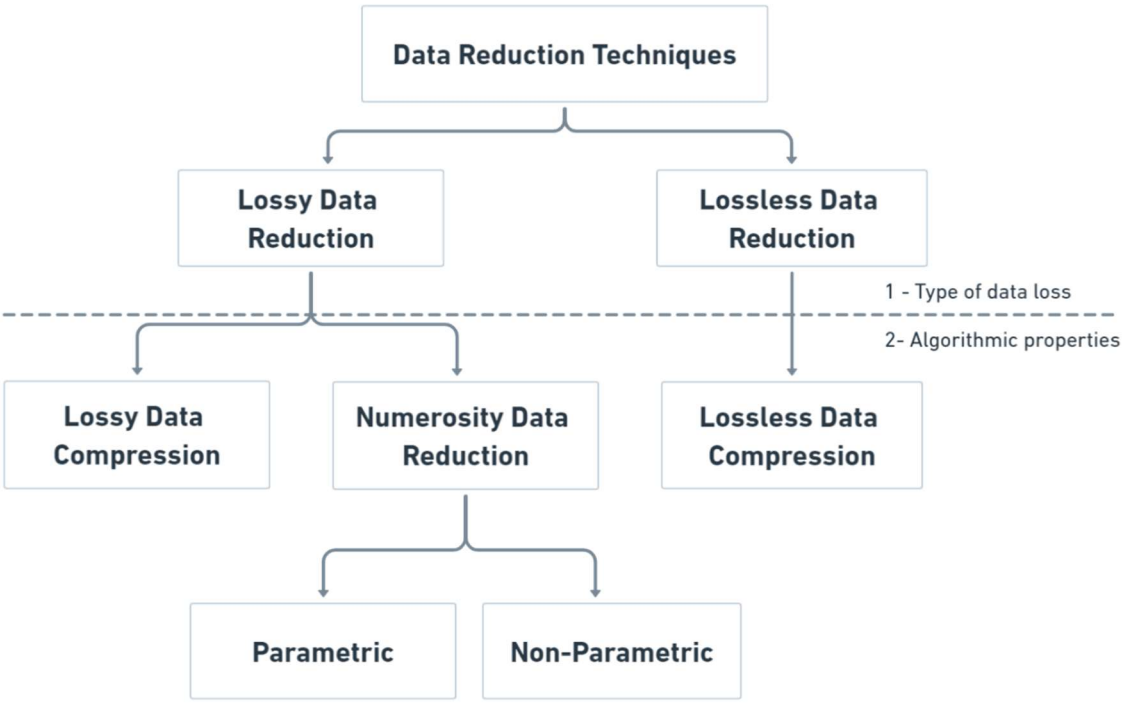


Figure 1 - Data Reduction Techniques Taxonomy.

In the proposed taxonomy, data reduction approaches are divided into two main categories:

- **Lossy data reduction techniques** [12], [25], [26], [27]**:** these focus on reducing data by discarding some details considered less relevant to the analysis or less noticeable to human perception, and thus achieve higher compression ratios than other methods. Some examples of these techniques are transform encoding, Discrete Cosine Transform, Random Projection, or Fractal Compression [28], [29], [30]. Numerosity data reduction techniques [12], [25], [26] reduce the amount of data by capturing the overall trend or patterns of the data. These techniques aim to represent concise and summarized data while preserving its essential characteristics and patterns. Unlike lossy compression, numerosity data reduction techniques do not intentionally discard data details but summarize them to make them more manageable and efficient. There are two main types of numerosity data reduction techniques: parametric and non-parametric.
    - **Parametric techniques** [31] rely on pre-existing data models, such as Linear Regression and Log-Linear, to estimate the data and reduce its quantity.
    - **Non-parametric techniques** [32], [33], on the other hand, focus on creating compressed versions of the data while preserving essential characteristics and patterns. Examples of non-parametric algorithms include Simple Random Sampling, K-Means, and data aggregation [11], [25], [26], [32].
- **Lossless data reduction techniques** [16], [34], [35]**:** these focus on identifying redundancies, patterns, and other inherent characteristics within the data to eliminate or minimize unnecessary and repetitive information. Reducing the data size without any loss is remarkable. However, these algorithms often depend on the type of data they are analysing. These perform better on repetitive single-sensor data, such as temperature readings. For large-scale datasets, these techniques tend to require a significant processing time to complete the compression, which can be challenging in real-time environments. When time is a primary feature and the workload is enormous and complex, using lossy data reduction algorithms is a viable option. These algorithms include various methods for reducing the data size by selectively discarding or approximating information from the original dataset. Their main advantages include faster processing speeds and the ability to handle a variety of data formats. All these lossless algorithms compress data and can decode the reduced algorithm back to its original size. Some examples of lossless algorithms include BZip2, Delta Encoding, LZ77, LZ78, Huffman coding, and Run Length Encoding [36], [37].

Table 2 - Examples of Data Reduction algorithms by taxonomic category.

Data reduction techniques categories	**Examples of Data reduction algorithms**
Lossless Data Compression Algorithms	**Delta Encoding**, **BZip2**, Huffman Encoding, Run-Length Encoding, Lempel-Ziv Compression (LZ77, LZ78, LZW)
Lossy Data Compression Algorithms	**Random Projection**, **Quantization**, Discrete Cosine Transform, Wavelet Compression, Cartesian Perceptual Compression, Fractal Compression
Parametric Numerosity Data Reduction Algorithms	**Linear Regression**, **Principal Component Analysis**, Random Forest, Support Vector Machines, Gaussian Regression, Log-Linear Models
Non-Parametric Numerosity Data Reduction Algorithms	**K-Means**, **Simple Random Sampling**, DBSCAN, Mean-Shift, OPTICS

4 DATA REDUCTION ALGORITHMS

The experimental evaluation used all the different data reduction techniques identified in the taxonomy illustrated in Figure 1. From the examples in Table 2, we selected: JPEG, MP3, Quantization, Down Sampling, Sample Reduction, Block Averaging, Discrete Cosine Transform, Delta Encoding, and BZip2. This choice was based on their popularity and ease of implementation.

4.1 Lossy Data Reduction Algorithms

We evaluated two types of lossy data reduction algorithms: lossy data compression techniques (JPEG, MP3, Quantization) and numerosity data reduction algorithms. Numerosity data reduction algorithms are divided into parametric (Block Averaging, Discrete Cosine Transform) and non-parametric (Down Sampling, Sample Reduction).

JPEG (Joint Photographic Experts Group) is a widely used lossy image compression algorithm designed to significantly reduce file size while maintaining acceptable visual quality [18]. It operates by transforming image data from the spatial domain to the frequency domain using the Discrete Cosine Transform (DCT), which separates the image into parts of differing importance based on visual perception. High-frequency components, which correspond to fine details and are less noticeable to the human eye, are typically compressed more aggressively or discarded. The resulting frequency coefficients are then quantized, introducing controlled loss of information to achieve compression. Finally, the quantized data is encoded using entropy coding techniques such as Huffman coding. While JPEG compression introduces some level of quality degradation, it offers a good trade-off between compression ratio and perceptual quality, making it suitable for a wide range of applications, including digital photography, web images, and multimedia systems.

MP3 (MPEG-1 Audio Layer III) is a widely adopted lossy audio compression algorithm designed to reduce audio file size while preserving perceptual sound quality [17]. It leverages psychoacoustic models to identify and eliminate audio components that are less audible to the human ear, such as frequencies masked by louder sounds. The compression process begins by dividing the audio signal into small frames and applying a filter bank along with a Modified Discrete Cosine Transform (MDCT) to convert the signal into the frequency domain. Irrelevant or redundant frequency components are then quantized and encoded using Huffman coding. This selective data removal enables substantial compression ratios while maintaining a listening experience that is subjectively close to the original. Although some fidelity is lost during compression, MP3 strikes an effective balance between file size and perceived audio quality, making it suitable for applications such as music streaming, digital storage, and portable audio devices.

Quantization is another lossy data compression algorithm studied, used in signal processing and data compression to reduce the precision of numerical data representation [38]. In signal processing, the algorithm involves mapping continuous values to discrete levels, typically by dividing a range into intervals and assigning specific values to each interval. This results in a representation with a bit lower depth. In data compression, quantization reduces the number of bits needed to represent numerical values, thereby reducing storage or transmission requirements. However, this reduction introduces quantization error as an approximation to the original values. Achieving a balance between efficient data representation and minimizing quantization error is critical in several applications, including image and audio compression, where optimization is essential to maintain an acceptable level of quality.

Downsampling is a lossy data reduction technique commonly used in signal and image processing to decrease the resolution or sampling rate of data [39], [40]. In the context of audio, it involves reducing the sampling frequency, thereby decreasing the number of samples used to represent the signal over time. For images or multidimensional data, downsampling reduces the spatial resolution by averaging or selecting specific pixels or blocks from the original data. This process leads to a smaller dataset and reduced storage or transmission requirements. However, downsampling inherently discards information, potentially leading to a loss of detail and fidelity, especially in high-frequency components. The key challenge in downsampling is to preserve as much relevant information as possible while significantly lowering data volume, making it particularly useful in scenarios where bandwidth or storage is limited, and exact reproduction of the original data is not critical.

Sample Reduction is a lossy data reduction technique that reduces the amount of data by selectively discarding or skipping samples from the original dataset [41]. In signal processing, this typically involves removing samples based on a fixed pattern or threshold criteria. The technique is widely used in image and audio compression, where it may involve discarding pixel values or audio samples that contribute less to perceptual quality. By retaining only a subset of the original data, the algorithm decreases the total number of data points, leading to reduced memory usage and faster transmission. However, this simplification comes at the cost of losing fine-grained details, which can affect the accuracy and quality of the reconstructed data. The effectiveness of sample reduction depends heavily on the nature of the signal and the sampling strategy used. It is especially useful in applications where approximate representations are acceptable, such as preliminary data analysis, multimedia compression, or embedded systems with limited resources.

Block Averaging is a lossy data compression technique commonly used in image and video processing to reduce data size by simplifying pixel information [42]. The method works by dividing an image into non-overlapping blocks of fixed size and replacing all pixels with their average intensity or colour value. This reduces the

amount of unique data points in the image, leading to smaller file sizes and faster processing or transmission. While the compression is efficient, the trade-off is a loss of fine detail and visual sharpness, as local variations within blocks are removed. Block Averaging is particularly effective in scenarios where low resolution is acceptable, such as in thumbnail generation, low-bandwidth image transmission, or embedded systems with limited processing and storage capabilities. Its simplicity and low computational cost make it a practical solution for basic image reduction tasks.

Discrete Cosine Transform is a widely used signal processing and data compression technique that transforms a signal or image from the spatial or time domain into the frequency domain [43]. The DCT represents data as a sum of cosine functions oscillating at different frequencies, effectively concentrating the signal's energy into some significant coefficients. This property allows for efficient data compression by enabling the selective retention of important frequency components while discarding less perceptually significant ones. In image and audio compression, the DCT helps reduce redundancy and achieve high compression ratios with minimal perceptible loss of quality. The transformation and its inverse form the basis for many standard compression formats such as JPEG and MP3. Balancing the number of retained coefficients against compression efficiency and reconstruction quality is a key consideration in the application of DCT-based methods.

Table 3 - Lossy Data Reduction Algorithms main use cases and applications.

Algorithm	Use Case	Application
JPEG	Image compression, web graphics, digital photography	JPEG is a widely used lossy image compression standard that significantly reduces file sizes by discarding perceptually less important image information. It uses techniques like color space transformation, downsampling, and quantization of frequency components (via the Discrete Cosine Transform). JPEG is the default format for storing and transmitting images on the web and in consumer devices due to its balance of compression efficiency and acceptable visual quality loss.
MP3	Audio compression, streaming, digital music storage	MP3 is a popular audio coding format designed for lossy compression. It works by analyzing and removing audio components that are less perceptible to human hearing, then encoding the remaining data efficiently. MP3 enables compact storage and fast transmission of audio files, making it the standard format for music

Algorithm	Use Case	Application
		distribution, podcasts, and online streaming platforms.
Quantization	Image and audio compression, signal processing	Quantization is commonly used in image and audio compression algorithms such as JPEG and MP3. It involves reducing the precision of the data representation by mapping continuous values to a finite set of discrete values. This results in lossy compression, where some information is lost, but for many applications it allows a significant reduction in file size with no noticeable loss in quality.
Block Averaging	Image compression, data simplification, low-resolution preview generation	Block averaging divides an image into fixed size blocks and replaces each block with the average color or intensity of its pixels. This simple lossy technique dramatically reduces the amount of data by preserving only coarse visual information. It's useful in applications requiring lightweight image previews, low-bandwidth transmission, or where high detail is not necessary, such as thumbnail generation or background subtraction.
Discrete Cosine Transform	Image and audio compression, signal transformation, feature extraction	The Discrete Cosine Transform converts signals from the time or spatial domain into the frequency domain. It concentrates signal energy into some low-frequency components, making it ideal for compression. DCT is a core component in compression standards like JPEG and MP3, where it helps isolate perceptually important information. It's also used in image processing tasks like denoising and in machine learning for feature extraction.
Downsampling	Signal compression, image resizing, bandwidth optimization	Down sampling reduces the sampling rate of a signal by retaining fewer data points over time or space. In image processing, it involves lowering the resolution by selecting a subset of pixels, while in audio, it means decreasing the sampling frequency. This technique is widely used in multimedia compression, preview

Algorithm	Use Case	Application
		generation, and transmission over limited-bandwidth channels. Although it reduces data size significantly, it can lead to loss of detail or aliasing if not properly filtered beforehand.
Sample Reduction	Data compression, signal processing, resource-constrained transmission	Sample reduction involves decreasing the number of data points or samples in a signal or dataset while preserving its essential structure. In audio and image processing, this means removing intermediate samples or pixels, effectively reducing resolution or temporal detail. It's commonly applied in scenarios where storage, bandwidth, or processing power is limited - such as embedded systems, real-time monitoring, or mobile applications - allowing faster transmission and lower memory usage at the cost of some data fidelity.

4.2 Lossless Data Reduction Algorithms

We evaluated two algorithms for lossless data compression techniques: Delta Encoding and BZip2.

Delta Encoding is a technique used in data compression to represent or transmit data more efficiently by encoding the differences (or deltas) between values rather than the actual values themselves [44]. The basic idea is to store or transmit only the changes between successive data elements, which is particularly useful in scenarios of similarity between adjacent elements. This method is appropriate in scenarios where sequential data has some degree of correlation or where redundancy is present. By subtracting each data element from its predecessor, delta encoding reduces the amount of information that must be stored or transmitted, resulting in more efficient data representation and compression.

BZip2 is the acronym for Burrows-Wheeler Block Sorting Huffman Coding, a widely used lossless data compression algorithm [45]. BZip2, developed by Julian Seward, combines Burrows-Wheeler Transform (BWT), Run-Length Encoding (RLE), and Huffman coding to achieve efficient compression. The Burrows-Wheeler Trans-form rearranges the input data to increase redundancy and make it more amenable to subsequent compression techniques. Run-length encoding is then applied to compress repeated sequences. Finally, Huffman coding assigns variable-length codes to different symbols based on their frequencies. Known for its high

compression ratios, BZip2 is often used to compress large files or archives, providing a balance between compression speed and efficiency.

The primary use cases and applications for Delta Encoding and BZip2 algorithms are listed in Table 4

Table 4 - Lossless Data Reduction key use cases and applications.

Algorithm	Use Case	Application
Delta Encoding	Data storage optimization, version control systems	Delta encoding is commonly used in scenarios where data changes over time, such as in version control systems like Git. Instead of storing entire files, only the differences (delta) between versions are stored. It is also used in data compression techniques to reduce redundancy. Applications include minimizing storage requirements for historical versions of files, efficient data transfer over network protocols, and optimizing database storage for incremental backups.
BZip2	File compression, data transmission over networks	BZip2 is used to compress files and data streams. It provides a high compression ratio and is particularly effective for compressing text files, XML, and large datasets. Applications include compressing software distributions for faster downloads, reducing storage requirements for archival data, and optimizing data transmission over limited bandwidth networks. It is often used in conjunction with formats such as TAR to create compressed archives (tar.bz2 files).

5 EXPERIMENTAL EVALUATION

This section presents the experimental evaluation. The following subsections explain the datasets and metrics used, present the results obtained for each algorithm employed in the experiments, and compare these results to determine the most suitable algorithms for different data types.

The experimental setup is based on a Raspberry Pi 4 Model B equipped with 8 GB of memory, running Raspberry Pi OS, which is based on Debian 12 and uses the Linux kernel version 6.6. This device was selected due to its widespread adoption in embedded and prototyping applications, as well as its ease of integration with various tools and languages. Python, version 3.12.3, was chosen as the primary programming language for implementation. Its intuitive syntax, extensive standard library, and wide range of third-party packages - particularly for algorithm development and data processing - made it well-suited for the objectives of this project.

The experiment involves executing each algorithm five times across multiple datasets. Depending on the characteristics of the dataset, a preprocessing phase may be required, where data is either processed in bulk or split into individual lines or objects for analysis. Following this, a metrics evaluation phase is carried out, in which the original and reduced datasets are compared. This comparison focuses on both the accuracy of the transformation and the execution performance of each algorithm-dataset combination. The overall pipeline of this process is illustrated in the following diagram.

Figure 2 shows the data reduction pipeline diagram. The pipeline begins by acquiring the dataset objects from the datasets. Once gathered, the data enters a preparation phase where it is cleaned, normalized, and formatted to ensure consistency and suitability for analysis. Following preparation, a data reduction algorithm is applied to reduce the dataset's size or complexity while preserving key information. The resulting reduced dataset is then saved to a designated storage location for future use. Finally, performance metrics are collected to assess the reduction's effectiveness, including measures like information retention or compression efficiency.

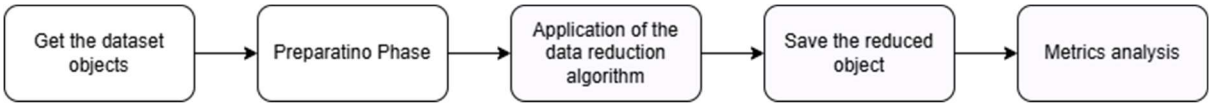


Figure 2 - Data Reduction Pipeline Steps

The selection of datasets was intended to reflect a variety of data types and structures, simulating the diverse contexts in which these algorithms might be applied. This diversity helps to provide a more robust and generalizable evaluation of their performance. The experiment consisted of running five times each algorithm and then aggregating the data to analyze its consistency and the mean results from it.

The categorization of the data reduction algorithms is according to the proposed taxonomy. The selected lossy data reduction algorithms are JPEG, Quantization, Down Sampling, Sample Reduction, Block Averaging, Discrete Cosine Transform and MP3. The selected lossless data reduction algorithms are Delta Encoding and BZip2.

In the following tables, all metric values are presented with four decimal places to facilitate precise comparisons between algorithms. Since the datasets used are relatively lightweight, highlighting marginal differences becomes important for drawing meaningful conclusions - especially when extrapolating these results to larger datasets of similar nature. All the code used for the experimental evaluation is available and accessible on GitHub [46].

5.1 Datasets

To evaluate the performance and suitability of the data reduction algorithms across various contexts, we selected a range of datasets encompassing different data types, including temporal, numeric, and textual data. All datasets were obtained from Kaggle [47]. Table 5 summarizes the key characteristics of the selected datasets.

The temporal, numeric datasets, such as *Daily Minimum Temperatures in Melbourne*, *Electric Production*, and *Monthly Beer Production in Austria*, contain time-series data where numerical values are recorded over specific time intervals. These datasets, with a relatively small number of records, are well-suited for time-based trend analysis, forecasting, and pattern recognition.

The numeric datasets, like the *Accelerometer* and *Smoke Detection* datasets, focus on pure numerical data. The *Accelerometer* dataset provides sensor data, while the *Smoke Detection* dataset contains sensor readings related to smoke detection. These datasets have a larger number of records and columns, making them suitable for detailed statistical analysis, anomaly detection, and machine learning applications.

The temporal, text dataset, *IoT Temperatures*, combines time-series data with textual readings, making it ideal for applications that require analyzing both temporal patterns and textual information for more nuanced insights.

The audio dataset, *Musical Instrument Chord Classification (Audio)*, is based on audio recordings used for classifying musical chords. Audio datasets like this are typically analyzed for pattern recognition and classification tasks and are central to the field of speech and sound analysis.

Finally, the image dataset, *Body Parts X-Ray Images in PNG*, consists of medical X-ray images. Image datasets like this one are commonly used in computer vision and image classification tasks, including applications such as medical diagnosis, where precision is critical.

Together, these diverse datasets provide a comprehensive foundation for evaluating different data reduction techniques across a range of data types.

Table 5 - Overview of dataset characteristics.

Data Type	Dataset Name	Dataset Size (MB)	N. Records	N. Columns	Reference
Temporal, Numeric	Daily Minimum Temperatures in Melbourne	0.0558	3650 Lines	2	[48]
	Electric Production	0.0730	397 Lines	2	[49]
	Monthly Beer Production in Austria	0.0690	476 Lines	2	[50]
Numeric	Accelerometer	3.7000	153,000 Lines	5	[51]
	Smoke Detection	5.8000	62,631 Lines	15	[52]
Temporal, Text	IoT Temperatures	6.7000	97,606 Lines	5	[53]
Audio	Musical Instrument Chord Classification (Audio)	169.67	859 Files	-	[54]
Image	Body Parts X-Ray Images in PNG	211	2482 Files	-	[55]
	Weather Image Recognition	636.73	9862 Files	-	[56]

5.2 Metrics

In the experimental evaluation, we used a set of metrics to assess the execution performance of each algorithm on the different datasets. These metrics include total

dataset reduction time, mean execution time per file, total dataset reduction in MB, total dataset reduction in percentage, mean file size difference in MB, and mean file size difference in percentage.

The **Total Dataset Reduction Time** is measured from the start of the run until the end of the run and corresponds to the subtraction between the end time and the start time of the run.

$$Total\ dataset\ reduction\ time\ (ms) = end\ time\ (ms) - start\ time\ (ms) \quad (5.2.1)$$

The **Mean Execution Time per File** is the average value of the time it takes to reduce each individual file. This is applied to all media files datasets (audio and images).

$$Mean\ execution\ time\ per\ file\ (ms) = \frac{\sum file\ end\ time\ (ms) - file\ start\ time\ (ms)}{number\ of\ files} \quad (5.2.2)$$

Total Dataset Reduction quantifies the overall decrease in storage size achieved by a data reduction algorithm across an entire dataset. It is typically reported in megabytes (MB) and is calculated as the difference between the cumulative size of the original files and the cumulative size of the reduced files:

$$Total\ dataset\ reduction\ (MB) = \sum_{i=1}^{n} Size_{original,i} - \sum_{i=1}^{n} Size_{reduced,i} \quad (5.2.3)$$

This metric provides a direct measure of the total storage savings resulting from the reduction process. It is particularly useful for evaluating the scalability and impact of an algorithm when applied to large datasets.

In this study, Total Dataset Reduction is used to assess the overall efficiency of each algorithm in minimizing storage requirements without compromising data integrity or quality. This metric is presented in both absolute terms (megabytes saved) and relative terms (percentage reduction), providing a clearer understanding of each algorithm's effectiveness across datasets of different sizes.

$$Total\ dataset\ reduction\ (\%) = \left(\frac{\sum_{i=1}^{n} Size_{original,i} - \sum_{i=1}^{n} Size_{reduced,i}}{\sum_{i=1}^{n} Size_{original,i}}\right) \times 100 \quad (5.2.4)$$

Mean File Size Difference measures the average change in file size resulting from the application of a data reduction algorithm. It is typically reported in megabytes

(MB) and calculated by subtracting the size of each reduced file from its corresponding original file, then averaging the result across all files in the dataset:

$$Mean\ File\ Size\ Difference = \frac{\sum_{i=1}^{n}(Size_{original,i} - Size_{reduced,i})}{n} \quad (5.2.5)$$

This metric provides insight into the effectiveness of a reduction technique in terms of storage savings. Larger values indicate more substantial reductions in file size, which may be beneficial for compression and storage purposes.

In this evaluation, the Mean File Size Difference is used to compare the efficiency of each algorithm across different datasets. It provides a quantitative measure of the average storage savings achieved per file. To offer a more comprehensive view, this metric is reported in both absolute terms (total space saved in megabytes) and relative terms (percentage reduction), allowing for more meaningful comparisons across datasets of varying sizes.

$$Mean\ File\ Size\ Difference\ (\%) = \frac{\sum_{i=1}^{n}\left(\frac{Size_{original,i} - Size_{reduced,i}}{Size_{original,i}}\right)}{n} \times 100 \quad (5.2.6)$$

To evaluate the impact of the data reduction algorithms on data quality, we additionally considered the following metrics: log-spectral distortion (LSD), signal-to-noise ratio (SNR), loudness, sampling rate (SR), and duration. A detailed description of these metrics is presented below.

Duration refers to the total playback time of an audio file, typically measured in seconds. It represents the length of the audio signal from start to end and is a fundamental attribute in audio analysis.

For this experiment, duration serves as a check to ensure that the data reduction algorithms do not alter the temporal integrity of the audio. Any change in duration between the original and reduced files may indicate trimming, padding, or unintended modifications, which could affect the completeness or usability of the resulting audio.

Log-Spectral Distortion is a metric used to quantify the difference between two audio signals in the frequency domain. It measures how much the logarithmic power spectrum of a processed or reduced audio signal deviates from that of the original. LSD is typically expressed in decibels (dB) and is particularly sensitive to perceptual distortions that may not be evident in time-domain measures.

The metric is calculated by first applying a Short-Time Fourier Transform (STFT) to both the original and processed signals, converting them into spectral representations. The log power spectra are then compared frame by frame, and the root-mean-square error is computed across frequency bins and time frames.

LSD is especially valuable for evaluating the perceptual fidelity of audio compression or reduction algorithms, as it aligns more closely with how humans perceive changes in spectral content. In this study, LSD is used to detect subtle spectral differences introduced by data reduction algorithms, providing insight into the preservation of audio quality beyond basic waveform similarity.

Signal-to-Noise Ratio (SNR) is a measure used to quantify the relationship between the level of the desired signal and the level of background noise. It is typically expressed in decibels (dB). A higher SNR indicates a cleaner signal with less noise interference, which generally implies better audio quality.

Mathematically, SNR is calculated as the ratio of the power of the signal to the power of the noise. In the context of audio processing, it helps determine how much the original sound content stands out from any unwanted distortion, background interference, or artifacts introduced during compression or transformation.

Within this evaluation, SNR is used to assess whether the data reduction algorithms preserve the quality of the original audio. A significant drop in SNR after reduction may suggest a loss of clarity or the introduction of noise, which could impact the usefulness of the processed audio in real-world applications.

Loudness refers to the perceived intensity of a sound by the human ear. While it is related to the physical amplitude of the audio signal, loudness is not measured directly by amplitude or volume alone - it is a psychoacoustic property, meaning it considers how humans hear and interpret sound.

In this study, loudness is used as a metric to assess whether the data reduction algorithms significantly alter the perceptual quality of the audio. A notable change in loudness may indicate loss of important audio information or distortion introduced during the reduction process.

The **sampling rate** (or sampling frequency) refers to the number of samples of an audio signal taken per second during the process of digital recording. It is measured in Hertz (Hz). A higher sampling rate captures more detail from the original analog signal, allowing for more accurate digital representation. This metric is extracted from the audio file metadata.

It is essential to note that distinct evaluation metrics were employed for audio and image datasets, considering their unique data structures and perceptual characteristics. Audio signals were assessed with Log-Spectral Distance, Signal-to-Noise Ratio, and Loudness, which effectively measure changes in spectral content, noise levels, and perceived intensity. Sampling Rate and Duration were also included to identify any temporal or resolution variations caused by compression or transformation. In contrast, image datasets were evaluated using Structural Similarity Index Measure, Peak Signal-to-Noise Ratio, and Gradient Magnitude Similarity Deviation, which capture differences in luminance, contrast, and structural information between original and processed images. These metrics offer a more accurate reflection of visual quality and perceptual degradation in images. By

tailoring metric selection to each data modality's intrinsic properties, the evaluation criteria ensure a meaningful and domain-specific assessment of compression performance and data integrity.

5.3 Lossy Data Reduction Algorithms Experiment Results

In this subsection, we explain the implementation of the algorithms and provide an analysis of the results. We also present a detailed analysis of our findings through tabulated metrics.

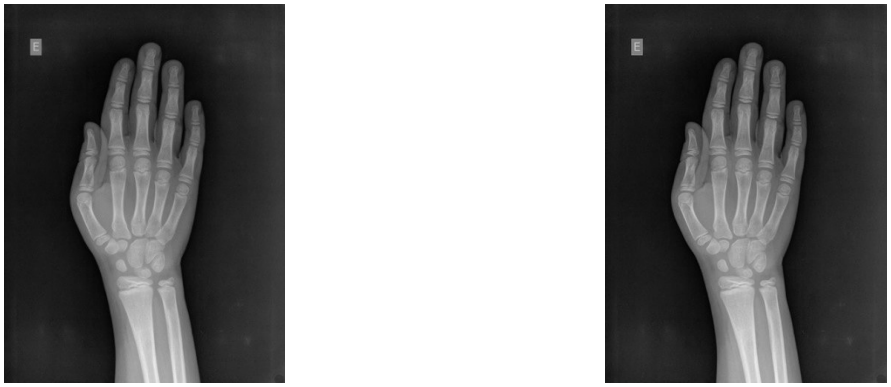
5.3.1 JPEG

JPEG is an algorithm that is only possible to be applied to images, due to this on this experiment the algorithm was applied to the *Body Parts X-Ray Images on PNG* and Weather *Image Recognition* datasets.

In our pipeline, we employed the Pillow library (version 11.2.1), a maintained fork of the original Python Imaging Library (PIL), to perform lossy image compression using the JPEG format. Pillow allows for straightforward image manipulation, including resizing, format conversion, and compression-key operations for evaluating data reduction techniques.

To reduce file size, the images were saved in the JPEG format with a controlled ***quality*** parameter, in our case we used 85 as the value. The quality argument in Pillow's ***save()*** method specifies the compression level, where lower values result in higher compression (and lower visual quality), and higher values preserve more detail at the cost of larger file size. Additionally, we set the flag ***optimize=True*** , which instructs Pillow to perform additional optimizations to reduce file size without affecting the specified quality level.

Upon visual inspection, no major perceptual differences were observed between the original and the reduced images. Figure 3 and Figure 4 illustrate examples of both the original and the compressed images from each dataset, demonstrating that the lossy compression applied using JPEG preserved the overall visual quality despite the reduction in file size.



a) b)

Figure 3 - JPEG data reduction on the *Body Parts X-Ray Images on PNG* dataset: a) original image; b) reduced image

a) b)

Figure 4 – JPEG data reduction on the *Weather Image Recognition* dataset: a) original image; b) reduced image

Regarding the results, the JPEG algorithm demonstrated strong effectiveness in reducing dataset size, particularly for the *Body Parts X-Ray Images in PNG* dataset, which achieved a total reduction of approximately 73.57%. Similarly, the *Weather Image Recognition* dataset experienced a reduction of around 43.31%. These values reflect substantial storage savings while maintaining acceptable visual quality, as discussed earlier. A detailed summary of these results can be found in Table 6.

Table 6 - JPEG dataset reduction results

Dataset Name	Total Dataset Reduction Time	Mean Execution Time per file	Total Dataset Reduction (MB | %)	Mean File Size Difference (MB | %)
Body Parts X-Ray Images in PNG	33.9723	0.0137	146.5311 MB ---- 73.5651 %	0.0591 MB ---- 73.0093 %

Dataset Name	Total Dataset Reduction Time	Mean Execution Time per file	Total Dataset Reduction (MB | %)	Mean File Size Difference (MB | %)
Weather Image Recognition	112.1252	0.0163	262.9912 MB ---- 43.3096 %	0.0038 MB ---- 13.2641 %

JPEG compression was applied to the image datasets to evaluate its impact on visual quality and data integrity. As shown in Table 7, for the Body Parts X-Ray Images in PNG dataset, SSIM was 0.9804, PSNR was 44.9187, and GMSD was 0.2684, indicating a small reduction in structural similarity and minor visual distortion compared to the original images. For the Weather Image Recognition dataset, SSIM was 0.9754, PSNR was 41.3598, and GMSD was 0.1572, reflecting only a minimal decrease in visual fidelity. These results demonstrate that JPEG compression preserves the majority of structural and visual content while providing effective reduction in storage size. Although minor distortions are introduced, the high SSIM and PSNR values indicate that image quality remains largely intact, making JPEG a practical option for applications where slight loss of detail is acceptable in exchange for efficient storage.

Table 7 - JPEG quality metrics on images datasets

Algorithm	SSIM	PSNR	GMSD
Body Parts X-Ray Images in PNG	0.9804	44.9187	0.2684
Weather Image Recognition	0.9754	41.3598	0.1572

5.3.2 MP3

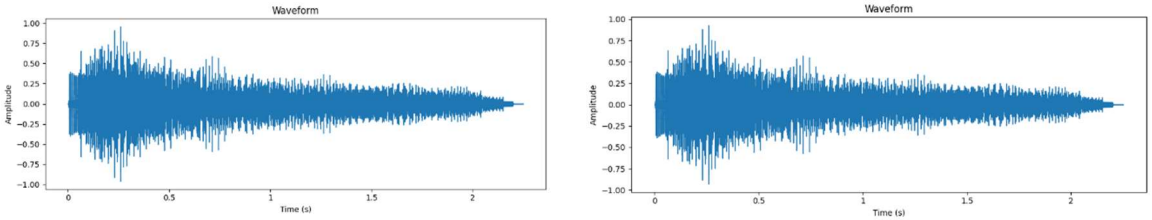
MP3 is an algorithm that is only possible to be applied to audio files, due to this on this experiment the algorithm was applied to the *Musical Instrument Chord Classification (Audio)* dataset.

To apply MP3 compression in our data reduction pipeline, we used the pydub library (version 0.25.1), a high-level audio processing package in Python that leverages FFmpeg for encoding and decoding various audio formats. The MP3 format was chosen due to its widespread adoption and its ability to significantly reduce file size by discarding audio information that is less perceptible to human hearing - a key principle of lossy compression.

The compression was performed by converting uncompressed or lossless audio files (e.g., WAV) into MP3 format using a fixed bitrate setting. The ***bitrate*** parameter controls the quality and size of the resulting file - lower bitrates result in smaller files, but may also introduce noticeable degradation in audio quality.

In our setup, we used a bitrate of 128 kbps, which offers a common trade-off between file size and acceptable audio fidelity for general-purpose use. This process was applied uniformly across the audio files in the datasets to ensure consistency in compression and evaluation.

Figure 5 shows an example of the sound curve of an audio file before and after the reduction.



a) b)

Figure 5 – MP3 data reduction on the *Musical Instrument Chord Classification (Audio)*: a) original soundwave image; b) reduced soundwave image

Regarding the results, the MP3 algorithm achieved a total reduction of approximately 71.8691%. These values reflect substantial storage savings while maintaining acceptable audio quality, as discussed earlier. A detailed summary of these results can be found in Table 8.

Table 8 - MP3 dataset reduction results

Dataset Name	Total Dataset Reduction Time	Mean Execution Time per file	Total Dataset Reduction (MB | %)	Mean File Size Difference (MB | %)
Musical Instrument Chord Classification	112.1252	0.0163	116.2903 MB ---- 71.8691 %	0.1354 MB ---- 71.8692 %

MP3 compression was applied to the Musical Instrument Chord Classification (Audio) dataset to evaluate its effectiveness in reducing data size and the associated computational performance. As presented in Table 9, the total dataset reduction required 112.1252 seconds, with a mean execution time per file of 0.0163 seconds, indicating that the compression process is computationally efficient. The overall dataset size decreased by 116.2903 MB, corresponding to a 71.8691% reduction,

while the mean file size per audio file decreased by 0.1354 MB, also representing a 71.8692% reduction. These results demonstrate that MP3 provides substantial storage savings while maintaining fast processing times, making it a practical method for audio dataset compression. However, unlike lossless algorithms, MP3 introduces lossy compression, which may affect audio fidelity, although the metric does not capture perceptual quality changes.

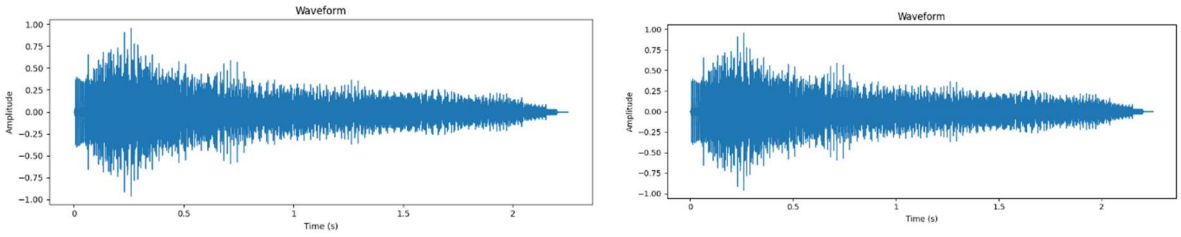
Table 9 - MP3 quality metrics on audio datasets

Dataset Name	Total Dataset Reduction Time	Mean Execution Time per file	Total Dataset Reduction (MB | %)	Mean File Size Difference (MB | %)
Musical Instrument Chord Classification (Audio)	112.1252	0.0163	116.2903 MB ---- 71.8691 %	0.1354 MB ---- 71.8692 %

5.3.3 Quantization

Quantization is a data reduction technique applicable to both image and audio files, as it simplifies the data by reducing the precision of values while preserving essential information. In this experiment, quantization was applied to two image datasets - *Body Parts X-Ray Images in PNG* and *Weather Image Recognition* - as well as to the *Musical Instrument Chord Classification (Audio)* dataset. This approach allowed us to evaluate the effectiveness of quantization across different data types and assess its impact on file size and perceptual quality.

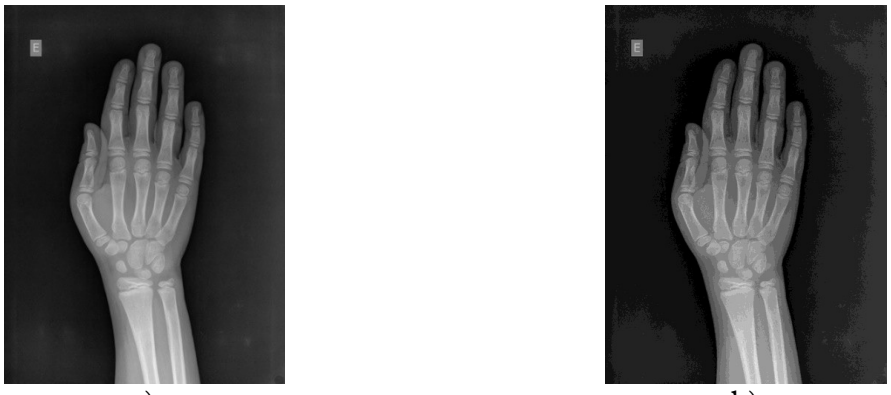
Quantization for audio involves reducing the precision of an audio signal by mapping its continuous amplitude values to a smaller set of discrete levels. This technique compresses data by lowering the bit depth - for instance, converting a 16-bit signal to 8-bit - thereby reducing the number of bits required to store each sample. In this study, quantization was implemented in Python by first normalizing the audio signal (using NumPy library), then scaling it to fit the desired bit depth. The values were rounded to the nearest quantization level and rescaled back to the original amplitude range. This approach simulates reduced-resolution audio while maintaining the waveform's overall structure. Although some fidelity is lost in the process, the perceptual quality of the audio remains sufficient for many practical use cases, and the resulting files occupy significantly less storage space. At the end looking at the soundwave image present on Figure 6, we can't notice any major differences, but while listening we notice some distortion, that values are shown in Section 5.5.1.



a) b)

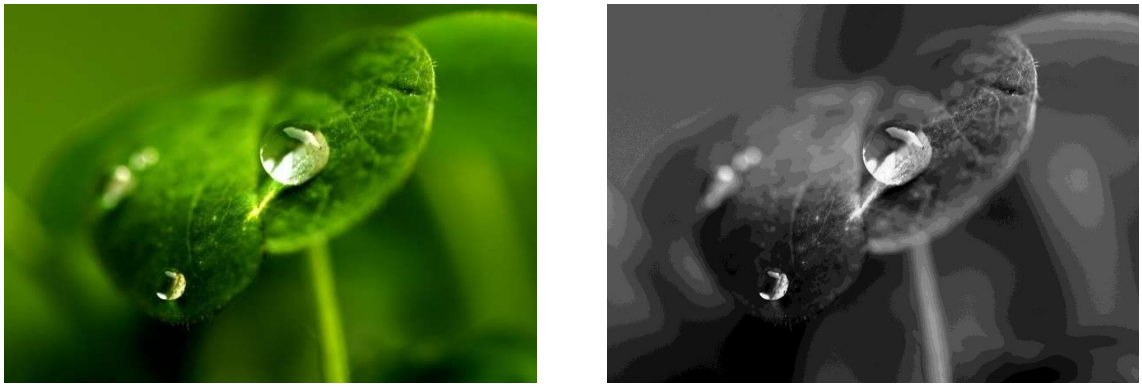
Figure 6 - Quantization data reduction on the *Musical Instrument Chord Classification (Audio)*: a) original soundwave image; b) reduced soundwave image

As for images, Quantization involves reducing the number of distinct color values used to represent an image, effectively compressing the data by limiting color precision. In this study, quantization was applied to color images by converting them to black and white, significantly reducing the color space from millions of possible RGB combinations to just two levels - black and white (binary). In Python, this can be implemented using the Pillow library by first converting the image to grayscale and then applying a threshold operation to binarize the image. This process simplifies the image by representing pixel values as either 0 or 255, depending on whether they fall below or above a defined threshold. While this results in a substantial loss of color information, the essential structure and contrast of the image are preserved, enabling effective data reduction with minimal impact on the visual representation of key features. Figure 7 and Figure 8 illustrate the effects of quantization on two image examples. In the first case, a black and white image, the reduced version exhibits a noticeable loss in quality, with some background distortion becoming apparent. In the second example, a color image, the quantization process removes all color information, resulting in a black and white image. Despite this significant reduction in color detail, the main content remains perceptible, and the subject of the image is still easily identifiable.



a) b)

Figure 7 - Quantization data reduction on the *Body Parts X-Ray Images on PNG* dataset: a) original image; b) reduced image



a) b)

Figure 8 – Quantization data reduction on the *Weather Image Recognition* dataset: a) original image; b) reduced image

As shown in Table 10, quantization proved most effective on the *Body Parts X-Ray Images in PNG* dataset, which consists of black and white images. Although the compression was substantial, some visual distortion was observed when compared to the original images. For the *Musical Instrument Chord Classification (Audio)* dataset, quantization nearly halved the total size of the dataset while maintaining a low variance ratio of just 0.0346, indicating minimal alteration in the signal. In the case of the *Weather Image Recognition* dataset, the impact was more visually pronounced, as the color information was completely removed during quantization. Nevertheless, this transformation resulted in a significant dataset reduction of approximately 58%.

Table 10 - Quantization dataset reduction results

Dataset Name	**Total Dataset Reduction Time**	**Mean Execution Time per file**	**Total Dataset Reduction (MB | %)**	**Mean File Size Difference (MB | %)**
Musical Instrument Chord Classification	3.2707	0.0038	80.8863 MB ---- 49.9889 %	0.0942 MB ---- 49.9889 %
Body Parts X-Ray Images in PNG	27.0395	0.0109	146.5190 MB ---- 73.5590 %	0.0591 MB ---- 73.2658 %
Weather Image Recognition	164.0191	0.0239	352.2181 MB **----** 58.0036 %	0.0505 MB ---- 34.4946 %

The Quantization algorithm was applied to the Musical Instrument Chord Classification (Audio) dataset to evaluate its impact on signal quality and integrity. As shown in Table 11, the original dataset exhibited an LSD of 1.4084, SNR of 0.3197, loudness of - 17.5661 LUFS, a sampling rate of 44,100 Hz, and a duration

of 2.2389 s. After Quantization, LSD decreased to 0.3193, while SNR remained nearly constant at 0.3193, indicating minimal changes in signal-to-noise characteristics. Loudness increased slightly to - 17.8325 LUFS, suggesting a minor alteration in amplitude, whereas the sampling rate and duration remained unchanged at 44,100 Hz and 2.2389 s, respectively. These results demonstrate that Quantization preserves the overall temporal structure and sampling resolution of the audio while introducing only slight changes in amplitude and spectral characteristics. Consequently, while it can reduce data representation size, care must be taken in applications that rely on precise signal fidelity or fine spectral detail.

Table 11 - Quantization quality metrics on audio datasets

Algorithm	Dataset Analyzed	LSD	SNR	Loudness	SR	Duration
Musical Instrument Chord Classification (Audio)	Original	1.4084	0.3197	-17.5661	44100.0000	2.2389
	Reduced		0.3193	-17.8325	44100.0000	2.2389

The Quantization algorithm was applied to the image datasets to assess its impact on visual quality and structural integrity. As shown in Table 12, for the Body Parts X-Ray Images in PNG dataset, SSIM was 0.8545, PSNR was 28.9457, and GMSD was 0.4036, indicating a significant reduction in structural similarity and the introduction of noticeable visual distortion compared to the original images. For the Weather Image Recognition dataset, SSIM decreased to 0.8915, PSNR dropped to 22.3749, and GMSD increased to 0.3245, reflecting a marked decline in image fidelity and higher levels of structural degradation. These results demonstrate that Quantization introduces measurable distortions in image quality, with varying impact depending on the complexity and content of the dataset. While the algorithm can reduce storage requirements, the decreases in SSIM and PSNR and the increases in GMSD indicate a trade-off between compression efficiency and visual fidelity, which must be considered for applications requiring accurate image representation and detailed structural information.

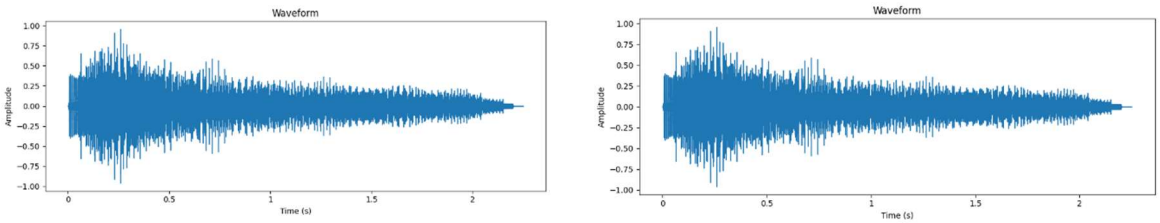
Table 12 - Quantization quality metrics on images datasets

Algorithm	SSIM	PSNR	GMSD
Body Parts X-Ray Images in PNG	0.8545	28.9457	0.4036
Weather Image Recognition	0.8915	22.3749	0.3245

5.3.4 Down Sampling

Down Sampling reduces data volume by lowering the resolution of images or the sampling rate of audio files, effectively decreasing the number of data points while preserving the essential content. In this study, down sampling was used to compress two image datasets - *Body Parts X-Ray Images in PNG* and *Weather Image Recognition* - as well as the *Musical Instrument Chord Classification (Audio)* dataset. This approach enabled a cross-domain evaluation of how down sampling impacts both storage efficiency and the perceptual quality of visual and audio data.

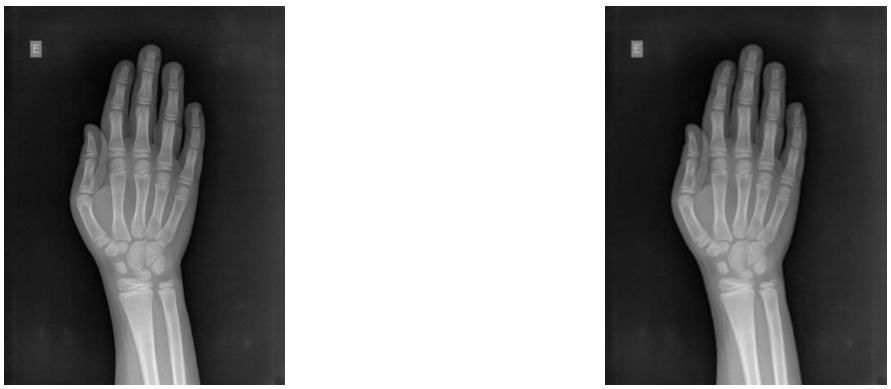
In this study, Down sampling of audio files was implemented using the *librosa* library in Python. The original audio signals were first loaded with ***librosa.load()***, which retrieves both the waveform and its native sampling rate. To reduce the sampling rate, we applied ***librosa.resample()***, specifying a lower target rate - such as reducing from 44,100 Hz to 22,050 Hz. This process effectively compresses the audio by reducing the number of samples per second, while preserving the perceptual quality of the sound. Librosa's built-in antialiasing ensures that the down sampled signals remain clean and free of distortion, making it a reliable method for efficient audio data reduction. Looking at Figure 9, we can observe slight differences between the two images. The reduced image appears slightly more filled in the soundwaves, suggesting the presence of noise.



a) b)

Figure 9 – Down Sampling data reduction on the *Musical Instrument Chord Classification (Audio)*: a) original soundwave image; b) reduced soundwave image

For image data, Down Sampling was implemented using the Pillow library in Python, which provides efficient tools for image resizing. The original images were first loaded using ***Image.open()*** and then resized to smaller dimensions using the ***resize()*** method. This method reduces the total number of pixels, effectively decreasing the image resolution and file size. During the resizing process, interpolation methods such as bilinear or bicubic interpolation can be specified to smooth the results and preserve as much visual quality as possible. In this study, image dimensions were typically reduced by a fixed ratio, in this study was 50%, allowing us to evaluate the impact of spatial resolution reduction on data size and perceptual quality. This approach enabled significant storage savings while maintaining the recognizability of key image features. Figure 10 and Figure 11 show examples of both image dataset final results after applying Down Sampling.



a) b)

Figure 10 – Down Sampling data reduction on the *Body Parts X-Ray Images on PNG* dataset: a) original image; b) reduced image

a) b)

Figure 11 – Down Sampling data reduction on the *Weather Image Recognition* dataset: a) original image; b) reduced image

The results from applying the Down Sampling algorithm across three datasets, Table 13, show varying degrees of dataset reduction and processing efficiency.

For the *Musical Instrument Chord Classification* dataset, the algorithm achieved a significant reduction of 63.7%, resulting in a total reduction of 103.08 MB. The mean execution time per file was 0.1376 seconds, which suggests that while the algorithm is effective in reducing the dataset size, it operates with a relatively moderate processing time.

The *Body Parts X-Ray Images in PNG* dataset also showed a substantial reduction of 73.1%, with 145.68 MB of data being compressed. The mean execution time per file, at 0.0366 seconds, was much lower than the previous dataset, indicating that Down Sampling is relatively quick when applied to X-ray images.

Finally, the *Weather Image Recognition* dataset experienced the highest reduction, with 528.78 MB reduced, equating to an 87.1% reduction in size. This dataset also exhibited a slightly lower mean execution time per file (0.0225 seconds), reflecting the algorithm's efficiency in compressing large image datasets quickly.

In summary, the Down Sampling algorithm proves to be highly effective in reducing dataset size across all three datasets, with the most notable reduction seen in the Weather Image Recognition dataset. It also demonstrates a consistent performance in execution time, making it suitable for large-scale data compression, especially for image-based datasets like weather and medical images.

Table 13 - Down Sampling dataset reduction results

Dataset Name	Total Dataset Reduction Time	Mean Execution Time per file	Total Dataset Reduction (MB | %)	Mean File Size Difference (MB | %)
Musical Instrument Chord Classification	118.1707	0.1376	103.0794 MB ---- 63.7046 %	0.1200 MB ---- 63.7045 %
Body Parts X-Ray Images in PNG	90.8964	0.0366	145.6754 MB ---- 73.1355 %	0.0587 MB ---- 72.7920 %
Weather Image Recognition	154.7181	0.0225	528.7792 MB ---- 87.0798 %	0.0768 MB ---- 77.8794 %

The Down Sampling algorithm was applied to the Musical Instrument Chord Classification (Audio) dataset to evaluate its impact on signal quality and integrity. As presented in Table 14, the original dataset showed an LSD of 2.8519, SNR of 0.3197, loudness of -17.5661 LUFS, a sampling rate of 44,100 Hz, and a duration of 2.2389 s. After DS, LSD decreased to 0.3197, while SNR remained unchanged at 0.3197, indicating no measurable variation in the signal-to-noise ratio. Loudness also remained nearly constant at -17.5672 LUFS, demonstrating that the overall amplitude and dynamic range were preserved. However, the sampling rate was reduced from 44,100 Hz to 16,000 Hz, resulting in a significant decrease in temporal resolution and a potential loss of high-frequency components. The duration remained identical at 2.2389 s, confirming that the temporal structure was maintained. These results show that DS effectively reduces data size by lowering the sampling rate while preserving amplitude and timing consistency. Nevertheless, the reduction in sampling frequency implies a loss of fine spectral detail, which may affect the performance of applications requiring high-frequency audio information.

Table 14 - Down Sampling quality metrics on audio datasets

Algorithm	Dataset Analysed	LSD	SNR	Loudness	SR	Duration
Musical Instrument Chord Classification (Audio)	Original	2.8519	0.3197	-17.5661	44100.0000	2.2389
	Reduced		0.3197	-17.5672	16000.0000	2.2389

The Down Sampling algorithm was applied to the image datasets to assess its impact on visual quality and structural integrity. As shown in Table 15, for the Body Parts X-Ray Images in PNG dataset, SSIM was 0.9694, PSNR was 40.8765, and GMSD was 0.2490, indicating a small reduction in structural similarity and a slight increase in distortion compared to the original images. For the Weather Image Recognition dataset, SSIM decreased to 0.8412, PSNR dropped to 28.5014, and GMSD increased to 0.2850, reflecting a more noticeable loss of detail and higher visual distortion. These results demonstrate that DS introduces measurable but controlled degradation in image quality, with the degree of distortion being more pronounced in datasets containing greater texture variation and visual complexity. While DS effectively reduces the dataset's spatial resolution and storage size, the observed decreases in SSIM and PSNR and the increase in GMSD values indicate a trade-off between compression efficiency and visual fidelity, which should be considered in applications requiring precise image detail preservation.

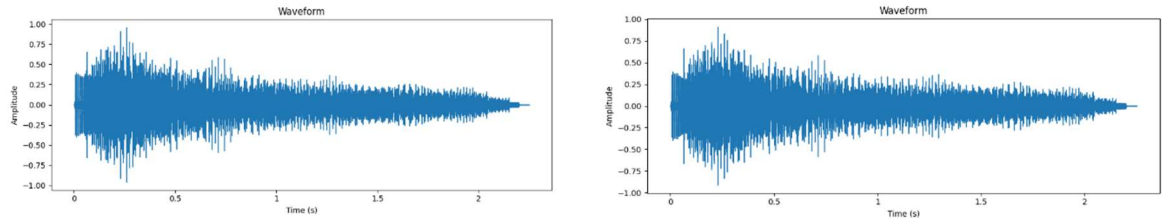
Table 15 – Down Sampling quality metrics on image datasets

Algorithm	SSIM	PSNR	GMSD
Body Parts X-Ray Images in PNG	0.9694	40.8765	0.2490
Weather Image Recognition	0.8412	28.5014	0.2850

5.3.5 Sample Reduction

Sample reduction involves decreasing the number of data instances or frames in a dataset, thereby reducing its overall size while aiming to retain the most relevant information. In this experiment, sample reduction was applied to two image datasets - *Body Parts X-Ray Images in PNG* and *Weather Image Recognition* - as well as the *Musical Instrument Chord Classification (Audio)* dataset. This technique allowed us to explore its effectiveness in streamlining datasets without significantly compromising the quality or representativeness of the original content.

Sample reduction for audio involves decreasing the number of audio samples in a dataset, which reduces the overall file size while aiming to retain the most important auditory features. In this study, sample reduction was applied using Python's librosa library. The audio files were first loaded with ***librosa.load()***, which retrieves both the waveform and the sampling rate. To reduce the number of samples, the audio signal was downsampled by selecting every nth sample or by applying decimation techniques. This process effectively reduces the data density, keeping only the most significant portions of the audio signal. Although this reduction can lead to some loss of detail in the sound, it still allows the overall content to remain recognizable, and the reduced files take up less storage. Additionally, when using *librosa*, antialiasing filters can be applied during the reduction process to mitigate distortion. Figure 12 show the shape of the soundwave after reducing a sound when compared to its original.



a) b)

Figure 12 – Sample Reduction data reduction on the *Musical Instrument Chord Classification (Audio)*: a) original soundwave image; b) reduced soundwave image

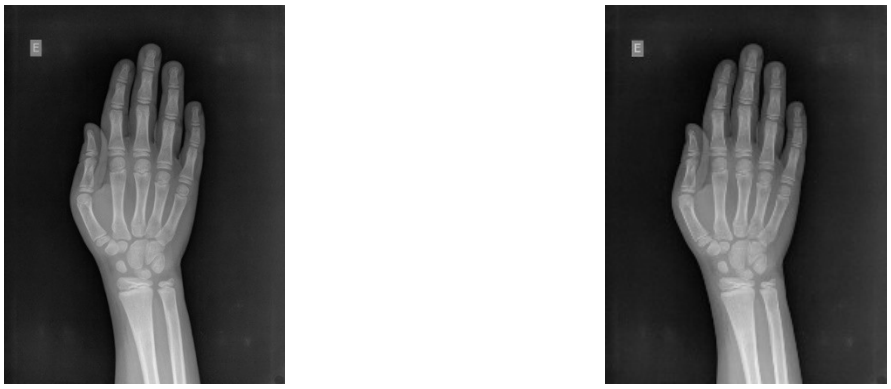
The sample reduction algorithm, for images, reduces the resolution of the image by skipping rows and columns at fixed intervals. Initially, the image is loaded using the ***Image.open()*** function from the Pillow library, and it is converted to a NumPy array for easier manipulation. If the image is in grayscale, it is reshaped to ensure consistent handling across all image types (grayscale, RGB, and RGBA).

The core of the sample reduction is performed by selecting every nth pixel from both rows and columns, where the step determines the interval. Specifically, the operation ***image_array[::step, ::step]*** skips pixels in both dimensions, effectively reducing the image's resolution. This method decreases the total number of pixels, leading to a lower file size and a reduction in the image's overall detail.

After the resolution is reduced, if the image was originally grayscale, the extra dimension added for compatibility is removed. The reduced array is then converted back into an image using ***Image.fromarray()*** and saved to the specified output path.

This approach provides an efficient way to reduce image file size by simplifying its structure, although it comes with a loss in image detail, especially as the step value increases. The algorithm is particularly useful when the goal is to reduce image size significantly while retaining basic content and features, although at a lower resolution. Figure 13 and Figure 14 illustrate the final images after applying sample

reduction, presenting a bit of blurriness after the algorithm is applied, particularly in the X-ray image.



a) b)

Figure 13 – Sample Reduction data reduction on the *Body Parts X-Ray Images on PNG* dataset: a) original image; b) reduced image

a) b)

Figure 14 – Sample Reduction data reduction on the *Weather Image Recognition* dataset: a) original image; b) reduced image

Sample reduction demonstrated impressive results across all three datasets in terms of dataset size reduction, Table 16.

For the *Musical Instrument Chord Classification* dataset, sample reduction achieved a remarkable reduction of 153.68 MB, which represents 94.98% of the original dataset size. The mean file size difference was 0.1789 MB, with a total dataset reduction time of 97.61 seconds and a mean execution time per file of 0.1136 seconds. This indicates that sample reduction was highly efficient in significantly compressing the dataset.

For the *Body Parts X-Ray Images in PNG* dataset, the reduction was 140.72 MB, or 70.65% of the original size. The mean file size difference per image was 0.0567 MB, and the total dataset reduction time was 76.85 seconds, with an average execution time per file of 0.0310 seconds. This shows that sample reduction was effective in compressing medical images while maintaining a moderate reduction time.

In the *Weather Image Recognition* dataset, sample reduction achieved a reduction of 520.97 MB, or 85.79% of the total dataset. The mean file size difference per image was 0.0757 MB, with a total dataset reduction time of 104.71 seconds and a mean execution time per file of 0.0153 seconds. The high reduction percentage and relatively low processing time demonstrate the effectiveness of this technique in compressing large image datasets.

Overall, sample reduction proved to be highly effective for all three datasets, particularly in terms of achieving large reductions in file size. It was especially beneficial for the *Musical Instrument Chord Classification* and *Weather Image Recognition* datasets, where it provided substantial size reductions without significant computational cost.

Table 16 - Sample Reduction dataset reduction results

Dataset Name	Total Dataset Reduction Time	Mean Execution Time per file	Total Dataset Reduction (MB | %)	Mean File Size Difference (MB | %)
Musical Instrument Chord Classification	97.6148	0.1136	153.6831 MB ---- 94.9785 %	0.1789 MB ---- 94.9784 %
Body Parts X-Ray Images in PNG	76.8451	0.0310	140.7173 MB ---- 70.6463 %	0.0567 MB ---- 70.5031 %
Weather Image Recognition	104.7057	0.0153	520.9664 MB ---- 85.7932 %	0.0757 MB ---- 75.9824 %

The SR algorithm was applied to the Musical Instrument Chord Classification (Audio) dataset to evaluate its impact on signal quality and integrity. As presented in Table 17, the original dataset showed an LSD of 3.4861, SNR of 0.3197, loudness of -17.5661 LUFS, a sampling rate of 44,100 Hz, and a duration of 2.2389 s. After SR, LSD decreased to 0.3196, and SNR remained nearly unchanged at 0.3196, indicating no significant variation in the signal-to-noise characteristics. Loudness also remained effectively constant at -17.5662 LUFS, suggesting that amplitude and dynamic range were preserved. However, the sampling rate decreased substantially from 44,100 Hz to 2,205 Hz, resulting in a considerable reduction in temporal resolution and potential loss of high-frequency content. The duration remained almost identical at 2.2391 s, confirming that the temporal structure was maintained. These results demonstrate that SR effectively reduces the dataset's size by lowering the sampling rate while preserving overall amplitude and timing, although the

decrease in frequency resolution may limit its suitability for tasks requiring fine spectral detail.

Table 17 - Sample Reduction quality metrics on audio datasets

Algorithm	Dataset Analyzed	LSD	SNR	Loudness	SR	Duration
Musical Instrument Chord Classification (Audio)	Original	3.4861	0.3197	-17.5661	44100.0000	2.2389
	Reduced		0.3196	-17.5662	2205.0000	2.2391

The Quantization algorithm was applied to the image datasets to evaluate its impact on visual quality and data integrity. As presented in Table 18, for the Body Parts X-Ray Images in PNG dataset, SSIM was 0.9565, PSNR was 37.6494, and GMSD was 0.2256, indicating a slight reduction in structural similarity and the introduction of mild visual distortion compared to the original images. For the Weather Image Recognition dataset, SSIM decreased to 0.8223, PSNR dropped to 27.3996, and GMSD increased to 0.2848, suggesting a more pronounced degradation in visual quality and structural details. These results demonstrate that Quantization introduces measurable distortion in image data, with a stronger impact on datasets containing higher visual complexity or variability. Although it effectively reduces data size, the observed decreases in SSIM and PSNR, along with the increase in GMSD, indicate that Quantization may compromise image fidelity, making it less suitable for applications requiring high visual accuracy or detailed structural preservation.

Table 18 – Sample Reduction quality metrics on image datasets

Algorithm	SSIM	PSNR	GMSD
Body Parts X-Ray Images in PNG	0.9565	37.6494	0.2256
Weather Image Recognition	0.8223	27.3996	0.2848

5.3.6 Block Averaging

Block averaging is a data reduction technique that divides images into fixed size segments or blocks and replaces each block with its average value. This process reduces detail while maintaining the overall structure and patterns of the original content. In this study, block averaging was applied to the *Body Parts X-Ray Images in PNG*, *Weather Image Recognition*. The aim was to evaluate how effectively this method reduces file size across different media types while preserving essential visual and auditory characteristics.

The block averaging algorithm reduces the resolution of an image by averaging pixel values within small blocks. The process begins by opening the image using the ***PIL*** library and converting it into a NumPy array for easier manipulation. If the image is in RGBA format, it is first converted to RGB, and grayscale images are given a dummy channel to ensure consistency in processing. The height and width of the image are then divided by the block size to calculate the new dimensions of the reduced image.

Once the new dimensions are determined, the algorithm reshapes the image array so that each block of pixels, defined by the block size, is grouped together. The mean value of the pixels in each block is computed along both the row and column axes. This results in a lower-resolution image where each block is represented by its average pixel value, effectively reducing the image size while preserving the overall structure of the image.

After the block wise averaging is performed, the reduced image is converted back into a standard format using ***PIL.Image.fromarray()*** and saved to the output path. If the image was grayscale, the dummy channel added earlier is removed to restore the image's original format. In the case of errors, such as the input file not being found, exceptions are caught, and appropriate error messages are displayed. This block averaging technique is useful for reducing image size and resolution while retaining broad features, which is especially beneficial for applications where detailed image quality is less critical. Figure 15 and Figure 16 shows the expected differences after applying block averaging algorithm to black and white and color images.

a) b)

Figure 15 – Block Averaging data reduction on the *Body Parts X-Ray Images on PNG* dataset: a) original image; b) reduced image

a) b)

Figure 16 – Block Averaging data reduction on the *Weather Image Recognition* dataset: a) original image; b) reduced image

The block averaging algorithm, Table 19, was effective in reducing dataset sizes across both the *Body Parts X-Ray Images in PNG* and *Weather Image Recognition* datasets. For the *Body Parts X-Ray Images in PNG*, block averaging achieved a notable reduction of 146.53 MB, which is 73.57% of the original dataset size. The mean file size difference was 0.0591 MB, reflecting a significant compression. The total dataset reduction time was 33.97 seconds, with an average execution time of 0.0137 seconds per file, indicating efficient processing.

For the *Weather Image Recognition* dataset, block averaging reduced the dataset by 262.99 MB, or 43.31%. The mean file size difference per image was 0.0038 MB, corresponding to a 13.26% reduction. The total dataset reduction time was 112.13 seconds, with a mean execution time per file of 0.0163 seconds.

While block averaging showed strong results in both cases, the *Body Parts X-Ray Images* dataset saw a more substantial reduction, likely due to the higher redundancy in X-ray images, which are well-suited for this compression technique. The *Weather Image Recognition* dataset experienced a more moderate reduction, possibly because the complex nature of weather images, including varied textures and patterns, may not compress as efficiently with block averaging. Nonetheless, block averaging proves to be a useful method for dataset reduction, especially for data types with considerable redundancy.

Table 19 - Block Averaging dataset reduction results

Dataset Name	Total Dataset Reduction Time	Mean Execution Time per file	Total Dataset Reduction (MB | %)	Mean File Size Difference (MB | %)
Body Parts X-Ray Images in PNG	33.9723	0.0137	146.5311 MB ---- 73.5651 %	0.0591 MB ---- 73.0093 %

Dataset Name	Total Dataset Reduction Time	Mean Execution Time per file	Total Dataset Reduction (MB | %)	Mean File Size Difference (MB | %)
Weather Image Recognition	112.1252	0.0163	262.9912 MB ---- 43.3096 %	0.0038 MB ---- 13.2641 %

Block Averaging was applied to the image datasets to evaluate its effect on visual quality and data integrity, all the metrics related with the image quality are presented in Table 20. For the Body Parts X-Ray Images in PNG dataset, SSIM decreased to 0.9658, PSNR was 39.8994, and GMSD increased to 0.2475, indicating some loss of structural similarity and the introduction of mild visual distortion compared to the original images. Similarly, for the Weather Image Recognition dataset, SSIM dropped to 0.8397, PSNR decreased to 27.9974, and GMSD increased to 0.2875, reflecting a more noticeable degradation in image quality and structural details. These results show that, unlike lossless compression methods, BA introduces measurable alterations in image fidelity, with greater effects observed in datasets containing more complex visual patterns. While BA can reduce data size effectively, the reductions in SSIM and PSNR, along with increased GMSD values, suggest that its use may compromise the accuracy of applications that rely on precise image structure or detail.

Table 20 - Block Averaging quality metrics on image datasets

Algorithm	SSIM	PSNR	GMSD
Body Parts X-Ray Images in PNG	0.9658	39.8994	0.2475
Weather Image Recognition	0.8397	27.9974	0.2875

5.3.7 Discrete Cosine Transform

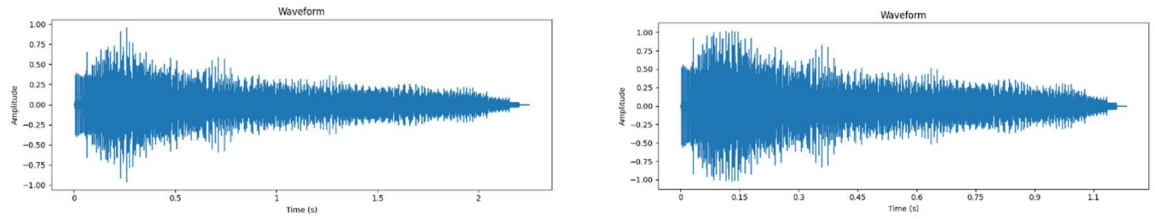
The Discrete Cosine Transform (DCT) algorithm is particularly well-suited for audio data, as it operates on continuous signal representations to efficiently capture frequency components. Although DCT can also be applied to image data, preliminary tests in this study did not yield satisfactory results for image compression. Therefore, its use was limited to the *Musical Instrument Chord Classification (Audio)* dataset, where it demonstrated more effective performance.

DCT is used in audio compression to convert a signal from the time domain into the frequency domain, allowing for the identification and reduction of less important

frequency components. By applying the DCT to segments of an audio file, the algorithm captures the most relevant sound information using fewer coefficients. Higher-frequency components, which are often less perceptible to the human ear, can be reduced or eliminated, resulting in a smaller file size. The original audio can later be reconstructed using the inverse DCT, with minimal perceptual loss.

In this study, DCT was applied using Python's ***scipy.fftpack*** module. The audio waveform was first loaded and then transformed using the ***dct()*** function with normalization to preserve energy. To simulate compression, only a portion of the most significant coefficients were retained, and the rest were set to zero. The signal was then reconstructed using the inverse function ***idct()***, producing a compressed version of the audio while maintaining acceptable sound quality.

Figure 17 show an example of the sound curve of an audio file before and after the reduction. With this image we can get that after the reduction the duration of the files tend to decrease even though the sound curve looks very similar to the original one. While listening to the file, we listen to a similar sound but performed faster.



a) b)

Figure 17 – Discrete Cosine Transform data reduction on the *Musical Instrument Chord Classification (Audio)*: a) original soundwave image; b) reduced soundwave image

The Discrete Cosine Transform algorithm achieved a total dataset reduction of approximately 49.35%, demonstrating a significant improvement in storage efficiency. This reduction was accomplished while preserving acceptable audio quality, as supported by both the quantitative evaluation metrics and the perceptual observations discussed earlier. A detailed summary of these results can be found in Table 21.

Table 21 - Discrete Cosine Transform dataset reduction results

Dataset Name	Total Dataset Reduction Time	Mean Execution Time per file	Total Dataset Reduction (MB | %)	Mean File Size Difference (MB | %)
Musical Instrument Chord Classification	120.6547	0.1405	79.8517 MB ---- 49.3495 %	0.0930 MB ---- 49.3345 %

The DCT was applied to the Musical Instrument Chord Classification audio dataset to evaluate its impact on signal quality and integrity. For the original dataset, LSD was 1.2626, SNR was 0.3197, loudness was –17.5661 LUFS, the sampling rate was 44,100 Hz, and the duration was 2.2389 s. After DCT-based reduction, LSD decreased to 0.3207, while SNR remained nearly unchanged at 0.3207, indicating only minor changes in signal-to-noise characteristics. However, loudness increased to –14.6301 LUFS, and the duration was reduced to 1.1338 s, reflecting noticeable alterations in amplitude and temporal structure. These results indicate that, unlike lossless compression methods, DCT reduction introduces measurable changes in audio quality and timing. While it can effectively reduce data size, the observed modifications in loudness and duration suggest that caution is required when using DCT in applications where precise signal fidelity and temporal accuracy are critical. These results are presented in Table 22.

Table 22 - Discrete Cosine Transform quality metrics on audio datasets

Algorithm	Dataset Analysed	LSD	SNR	Loudness	SR	Duration
Musical Instrument Chord Classification (Audio)	Original	1.2626	0.3197	-17.5661	44100.0000	2.2389
	Reduced		0.3207	-14.6301	44100.0000	1.1338

5.4 Lossless Data Reduction Algorithms Experiment Results

The next subsections will present an explanation of the lossless algorithms, in addition to showing the tabulated metrics with the experimental results.

5.4.1 Delta Encoding

The implementation of Delta Encoding faced the challenge of handling the diverse data forms within our dataset. To address this, we categorized the data into numeric and non-numeric types, allowing an adaptive approach to accommodate different structures. When processing string data, the algorithm carefully identified similarities between consecutive strings, recording different characters as delta strings and using a placeholder character ('\0') for unchanged positions. This analysis provided an understanding of string variations and ensured robust Delta Encoding.

Regarding numeric values, the algorithm began the process by storing the first numeric value and continued with iterative subtraction between successive values, storing the result as the current subtractor. This systematic approach continued until the last value, providing a seamless Delta Encoding solution for numeric data. The adaptability of our method shines through, highlighting a tailored and

comprehensive approach capable of handling the intricacies inherent in the diverse data structures and characteristics of our dataset.

As summarized in Table 23, Delta Encoding was applied to a variety of datasets differing in format and size, revealing consistent patterns in performance. The method proved to be fast and effective in reducing raw data volume, though its compression efficiency varied significantly depending on the dataset structure.

Delta Encoding performed best on the *IoT Temperatures* and *Smoke Detection* datasets, achieving compression ratios of 37.18% and 31.22%, respectively, with large absolute reductions of 2.46 MB and 1.74 MB. Despite a longer processing time for *IoT Temperatures* (22.48 seconds), these results highlight Delta Encoding's ability to remove substantial data when dealing with large, high-frequency time series.

In contrast, its efficiency dropped notably for smaller or highly numeric datasets. The *Accelerometer* dataset, for example, saw a low 5.41% reduction, though it still achieved a meaningful absolute reduction of 0.19 MB in just 0.0353 seconds. Similarly, datasets like *Monthly Beer Production* and *Electric Production* showed minimal percentage reductions (12.26% and 15.10%, respectively), though processing times remained under 0.01 seconds, highlighting the algorithm's speed.

Overall, Delta Encoding shows clear advantages in speed and absolute volume reduction, especially with large or temporally rich data. However, its lower percentage efficiency makes it less suitable when the priority is compact final size. Its use is therefore best aligned with applications that require quick processing and moderate space savings, rather than maximum compression efficiency.

Table 23 - Delta Encoding results.

Dataset Name	Total Dataset Reduction Time	Total Dataset Reduction (MB | %)
Daily Minimum Temperatures in Melbourne	0.0073	0.0175 MB ---- 32.8672 %
Electric Production	0.0056	0.0011 MB ---- 15.0998 %
Monthly Beer Production in Austria	0.0098	0.0008 MB ---- 12.2555 %
Accelerometer	0.0353	0.1927 MB ---- 5.4149 %

Dataset Name	Total Dataset Reduction Time	Total Dataset Reduction (MB | %)
Smoke Detection	0.1663	1.7372 MB ---- 31.2215 %
IoT Temperatures	22.4753	2.4636 MB ---- 37.1795 %

5.4.2 BZip2

The BZip2 algorithm, implemented using the bz2 Python library, encapsulates the underlying logic, consistent with usual practices in the Python ecosystem. This encapsulation simplifies reproduction and provides an accessible and user-friendly approach. In particular, the BZip2 algorithm excels in preserving data value after compression, ensuring complete data integrity. While it has a relatively slower compression speed, potentially less suitable for real-time data transfers, the algorithm's ability to maintain data integrity compensates for this trade-off. In addition, its data type agnosticism proves beneficial, making it a versatile choice for various applications.

BZip2 compression has been applied to various datasets to evaluate its efficiency in reducing file sizes and the time taken for compression. Table 24 shows that BZip2 performs well across different types of data, with notable reductions in file size for most datasets, though the extent of compression varies depending on the characteristics of the data.

BZ2 performed exceptionally well on datasets such as *IoT Temperatures*, *Accelerometer*, and *Smoke Detection*, achieving compression ratios of 86.94%, 84.77%, and 79.88%, respectively. These reductions were obtained in relatively short times considering dataset size - for instance, IoT Temperatures were processed in just 3.3964 seconds, while *Accelerometer* completed in 1.8317 seconds. Such results highlight BZ2's strength in compressing dense temporal and numeric data.

Similarly, smaller datasets like *Monthly Beer Production*, *Electric Production*, and *Daily Minimum Temperatures* showed consistent percentage reductions above 70%, confirming BZ2's effectiveness across a range of structured, low-volume datasets.

In contrast, BZ2 was less efficient with image-based datasets. *Weather Image Recognition, X-Ray Images*, and *Musical Instrument Chord Classification* showed much lower compression ratios - 5.52%, 0.85%, and 29.56%, respectively. These results indicate that for media-rich or already compressed formats, BZ2 yields diminishing returns in both absolute and percentage terms, despite longer processing times (e.g., over 273 seconds for the Weather dataset).

Overall, BZ2 proves highly efficient and consistent for compressing structured datasets, particularly those with temporal or numeric data. However, its performance drops considerably with unstructured or image-heavy content, making it less suitable when applied to such domains.

Table 24 - BZip2 results.

Dataset Name	Total Dataset Reduction Time	Mean Total Time per File	Total Dataset Reduction (MB | %)	Mean File Size Diff. (MB | %)
Daily Minimum Temperatures in Melbourne	0.0274	-	0.0441 MB ---- 82.7553 %	-
Electric Production	0.0137	-	0.0051 MB ---- 72.7658 %	-
Monthly Beer Production in Austria	0.0094	-	0.0051 MB ---- 77.1404 %	-
Accelerometer	1.8317	-	3.0163 MB ---- 84.7687 %	-
Smoke Detection	2.0489	-	4.4448 MB ---- 79.8844 %	-
IoT Temperatures	3.3964	-	5.7610 MB ---- 86.9425 %	-
Musical Instrument Chord Classification	14.3196	0.0167	47.8329 MB ---- 29.5615 %	0.0557 MB ---- 29.5783 %
Body Parts X-Ray Images in PNG	26.681	0.0107	1.6861 MB ---- 0.8465 %	0.0007 MB ---- 0.8998 %
Weather Image Recognition	273.6266	0.0401	33.4961 MB ---- 5.5162 %	0.0036 MB ---- 1.0566 %

After applying the bzip2 algorithm to the Musical Instrument Chord Classification dataset, evaluation metrics were gathered to assess its compression effectiveness and impact on audio quality and integrity. Table 25 summarizes these findings,

demonstrating that all measured values remained unchanged before and after compression, which confirms the lossless nature of the bzip2 algorithm. The LSD value of 0.0000 indicates a perfect spectral match between the original and reconstructed signals, while the consistent SNR of 0.3197 confirms that no additional noise or distortion was introduced. Additionally, the loudness level (–17.5661 LUFS), sampling rate (44,100 Hz), and duration (2.2389 s) were fully preserved. Overall, these results show that bzip2 compression maintains complete audio fidelity, enabling more efficient storage and transmission without compromising signal quality or analytical reliability.

Table 25 - Bzip2 quality metrics on audio datasets

Algorithm	Dataset Analysed	LSD	SNR	Loudness	SR	Duration
Musical Instrument Chord Classification (Audio)	Original	0.0000	0.3197	-17.5661	44100.0000	2.2389
	Reduced		0.3197	-17.5661	44100.0000	2.2389

The bzip2 algorithm was applied to image datasets to assess its impact on visual quality and data integrity. For the Body Parts X-Ray Images in PNG dataset, quality metrics - SSIM = 1.0000, PSNR = 100.0000, and GMSD = 0.0000 - indicate perfect reconstruction after compression and decompression. These results, presented on Table 26, confirm that bzip2 preserves all structural and visual details, including pixel values, contrast, and gradients, without introducing distortion. The Weather Image Recognition dataset showed identical metrics, further demonstrating that compression maintained complete visual fidelity and structural integrity. Overall, these findings confirm the lossless nature of bzip2: it reduces data size while fully preserving image content and quality, making it ideal for applications requiring exact reconstruction and analytical reliability.

Table 26 - Bzip2 quality metrics on image datasets

Algorithm	SSIM	PSNR	GMSD
Body Parts X-Ray Images in PNG	1.0000	100.0000	0.0000
Weather Image Recognition	1.0000	100.0000	0.0000

5.5 Discussion

Having presented the results, we now focus on the interpretation and implications of the results. Before the analysis, it is critical to address some of the limitations of our research. Because the datasets differ in size, this variation represents the main limitation of our study. This limitation required working with datasets of different sizes, which can introduce variability and potentially affect the comparison of results.

Since each algorithm has distinct characteristics, their performance can vary across different datasets. Therefore, the following subsections will focus on each dataset individually to analyse how each algorithm performs relative to the specific properties of that dataset.

5.5.1 Daily Minimum Temperatures in Melbourne

The *Daily Minimum Temperatures in Melbourne* dataset, which includes temporal and numeric information, was evaluated using BZ2 and Delta Encoding, as detailed in Table 27.

BZ2 achieved a strong compression ratio of 82.76%, reducing the dataset by 0.0441 MB in 0.0274 seconds. This result highlights BZ2's effectiveness in compressing structured time series data, offering both high reduction and low latency.

Delta Encoding, while faster at 0.0073 seconds, produced a lower compression ratio of 32.87%, corresponding to 0.0175 MB of data removed. Although its processing speed is advantageous, the lower efficiency limits its usefulness when high compression is needed.

In summary, BZ2 offers a better balance of speed and compression efficiency for this dataset type, while Delta Encoding may be preferred when rapid processing is critical, and a moderate reduction suffices.

Table 27 - Reduction results for Daily Minimum Temperatures in Melbourne

Algorithm	Total Dataset Reduction Time	Total Dataset Reduction (MB | %)
BZ2	0.0274	0.0441 MB ---- 82.7553 %
Delta Encoding	0.0073	0.0175 MB ---- 32.8672 %

5.5.2 Electricity Production

The *Electricity Production* dataset, characterized by temporal and numeric data, was compressed using BZ2 and Delta Encoding, as summarized in Table 28.

BZ2 reduced the dataset by 0.0051 MB, achieving a compression ratio of 72.77% in just 0.0137 seconds. This demonstrates BZ2's consistent efficiency, even with relatively small datasets, providing substantial reduction with minimal processing time.

Delta Encoding performed the compression in 0.0056 seconds, noticeably faster, but achieved only a 15.10% reduction, corresponding to 0.0011 MB. Despite its speed, the compression gain was limited.

Overall, BZ2 is the more effective choice for compressing small, structured datasets like Electric Production, where both reduction efficiency and processing time are important. Delta Encoding may still be useful where speed is prioritized over compression depth.

Table 28 - Reduction results for Electricity Production

Algorithm	Total Dataset Reduction Time	Total Dataset Reduction (MB | %)
BZ2	0.0137	0.0051 MB ---- 72.7658 %
Delta Encoding	0.0056	0.0011 MB ---- 15.0998 %

5.5.3 Monthly Beer Production in Austria

The *Monthly Beer Production* dataset, containing temporal and numeric information, was evaluated using BZ2 and Delta Encoding, with results shown in Table 29.

BZ2 achieved a notable compression ratio of 77.14%, reducing the dataset by 0.0051 MB in just 0.0094 seconds. This indicates that even for smaller datasets, BZ2 maintains strong compression performance with minimal processing time.

In contrast, Delta Encoding took a similar amount of time - 0.0098 seconds - but achieved only a 12.26% reduction, corresponding to 0.0008 MB. While fast, its compression efficiency was significantly lower, offering limited benefit in this context.

These results suggest that BZ2 is the preferred algorithm for small, numeric-temporal datasets where meaningful compression is still desired without compromising speed.

Table 29 - Reduction results for Monthly Beer Production

Algorithm	Total Dataset Reduction Time	Total Dataset Reduction (MB | %)
BZ2	0.0094	0.0051 MB ---- 77.1404 %
Delta Encoding	0.0098	0.0008 MB ---- 12.2555 %

5.5.4 Accelerometer

The Accelerometer dataset, composed entirely of numeric data, was compressed using both BZ2 and Delta Encoding, as shown in Table 30. The results reveal a significant contrast in compression effectiveness and speed between the two approaches.

BZ2 achieved a high compression ratio of 84.77%, reducing the dataset by 3.0163 MB in 1.8317 seconds. This demonstrates BZ2's strong ability to handle dense numeric data efficiently, balancing both speed and compression quality.

In comparison, Delta Encoding completed compression in just 0.0353 seconds, substantially faster than BZ2. However, it achieved only a 5.41% reduction, corresponding to 0.1927 MB. Despite its speed, the low compression ratio indicates limited effectiveness in significantly reducing dataset size.

These results suggest that BZ2 is far more effective for compressing numeric data where reduction efficiency is crucial, while Delta Encoding may be useful in scenarios where speed is prioritized but minimal compression is acceptable.

Table 30 - Reduction results for Accelerometer

Algorithm	Total Dataset Reduction Time	Total Dataset Reduction (MB | %)
BZ2	1.8317	3.0163 MB ---- 84.7687 %
Delta Encoding	0.0353	0.1927 MB ---- 5.4149 %

5.5.5 Smoke Detection

The *Smoke Detection* dataset, which consists entirely of numeric data, was assessed using BZ2 and Delta Encoding, with results shown in Table 31. This dataset provides insight into how each algorithm handles dense, structured numerical input.

BZ2 achieved a compression ratio of 79.88%, reducing the dataset by 4.4448 MB in 2.0489 seconds. This indicates a high level of compression efficiency, with a relatively short processing time that supports its suitability for time-sensitive applications.

On the other hand, Delta Encoding completed compression much faster, in just 0.1663 seconds, and reduced the dataset by 1.7372 MB, which corresponds to a 31.22% reduction. Although its absolute and percentage reductions were lower than BZ2's, the speed of execution highlights Delta Encoding's strength in scenarios where quick, lightweight compression is preferred.

Overall, BZ2 is the more effective choice when maximum compression is required, while Delta Encoding offers advantages in speed, making it a potential alternative when rapid processing is prioritized over compression ratio.

Table 31 - Reduction results for Smoke Detection

Algorithm	Total Dataset Reduction Time	Total Dataset Reduction (MB | %)
BZ2	2.0489	4.4448 MB ---- 79.8844 %
Delta Encoding	0.1663	1.7372 MB ---- 31.2215 %

5.5.6 IoT Temperatures

Looking for Table 32, the *IoT Temperatures* dataset was evaluated using two compression algorithms: BZ2 and Delta Encoding. This dataset contains both temporal and textual information, making it a suitable candidate for assessing compression performance across different data types.

The BZ2 algorithm achieved a substantial compression ratio of 86.94%, reducing the dataset by 5.7610 MB in just 3.3964 seconds. This reflects both high compression efficiency and reasonable processing time, positioning BZ2 as a strong choice for reducing storage costs in applications handling similar data.

In contrast, Delta Encoding required significantly more time - 22.4753 seconds - to complete the compression, yet achieved a lower reduction ratio of 37.18%, corresponding to 2.4636 MB of data removed. While this shows that Delta Encoding

can eliminate a large amount of raw data, its lower percentage efficiency and slower performance make it less suitable when both speed and compactness are prioritized.

These results underline a clear trade-off: BZ2 offers more effective and time-efficient compression for the IoT Temperatures dataset, while Delta Encoding, despite its larger absolute data removal in some contexts, falls short in both compression ratio and speed.

Table 32 - Reduction results for IoT Temperatures

Algorithm	Total Dataset Reduction Time	Total Dataset Reduction (MB | %)
BZ2	3.3964	5.7610 MB ---- 86.9425 %
Delta Encoding	22.4753	2.4636 MB ---- 37.1795 %

5.5.7 Musical Instrument Chord Classification (Audio)

The results show that Sample Reduction remains the most effective algorithm for compressing the Musical Instrument Chord Classification dataset, achieving a substantial 94.98% reduction, equating to over 153 MB saved. While its execution time is relatively high (97.61 seconds total), it offers an excellent trade-off between efficiency and compression.

MP3 compression, now reflecting a corrected reduction value of 71.87%, shows improved effectiveness compared to the previous version. It balances strong compression with low per-file execution time, making it a practical option for reducing audio data while preserving performance.

Down Sampling and Quantization also perform well, with reductions of 63.70% and 49.99% respectively. Down Sampling is slower but achieves higher reduction, whereas Quantization is faster with slightly lower savings. These results illustrate a trade-off between processing speed and compression depth.

Discrete Cosine Transform shows performance similar to Quantization but requires more processing time, suggesting it may be better suited for scenarios where compression quality is prioritized over speed.

BZip2, the general-purpose compression method, delivers the lowest reduction rate (29.56%) and smallest mean file size difference, though it is efficient in runtime. This makes it a lightweight but less effective option compared to domain-specific techniques.

Table 33 - Compression performance metrics for the *Musical Instrument Chord Classification (Audio)* dataset using different algorithms

Algorithm	Total Dataset Reduction Time	Mean Execution Time per file	Total Dataset Reduction (MB | %)	Mean File Size Difference (MB | %)
BZ2	14.3196	0.0167	47.8329 MB ---- 29.5615 %	0.0557 MB ---- 29.5783 %
MP3	112.1252	0.0163	116.2903 MB ---- 71.8691 %	0.1354 MB ---- 71.8692 %
Quantization	3.2707	0.0038	80.8863 MB ---- 49.9889 %	0.0942 MB ---- 49.9889 %
Down Sampling	118.1707	0.1376	103.0794 MB ---- 63.7046 %	0.1200 MB ---- 63.7045 %
Sample Reduction	97.6148	0.1136	153.6831 MB ---- 94.9785 %	0.1789 MB ---- 94.9784 %
Discrete Cosine Transform	120.6547	0.1405	79.8517 MB ---- 49.3495 %	0.0930 MB ---- 49.3345 %

The results in Table 34 illustrate how each compression technique affects the signal quality and structural properties of the audio in the *Musical Instrument Chord Classification* dataset.

BZip2, as expected from a lossless method, leaves all signal properties unchanged, confirming its preservation of original audio quality.

MP3 introduces minimal perceptual degradation, slightly increasing the LSD and decreasing SNR, while maintaining the original sampling rate and duration. This indicates that MP3 retains a high level of fidelity despite achieving significant compression.

Quantization and Discrete Cosine Transform show a moderate impact on audio quality. While their LSD values indicate some distortion, they maintain the sampling

rate and loudness within reasonable bounds. Notably, Discrete Cosine Transform also reduces the duration of the audio, suggesting a more aggressive transformation.

Down Sampling and Sample Reduction produce the most significant structural changes. Both techniques drastically lower the sampling rate - to 16,000 Hz and 2,205 Hz, respectively - which leads to more compact data at the cost of high-frequency content. Despite this, their LSD values remain moderate, indicating that much of the core audio information is preserved.

Overall, the results highlight the trade-offs between compression ratio and audio fidelity, with lossless methods preserving quality entirely and lossy techniques offering varying degrees of balance between size reduction and perceptual impact.

Based on both compression and quality metrics, Sample Reduction offers the highest size reduction (~95%) with acceptable quality loss, making it ideal for storage or machine learning tasks. MP3 provides a good balance between compression (~72%) and audio fidelity, suitable for scenarios where perceptual quality matters. Methods like Quantization and Down Sampling offer moderate trade-offs, while BZip2 achieves the least compression and is less effective for space-saving needs.

Table 34 - Audio quality and signal characteristics before and after compression of the *Musical Instrument Chord Classification* dataset

Algorithm	Dataset Analyzed	LSD	SNR	Loudness	SR	Duration
BZ2	Original	0.0000	0.3197	-17.5661	44100.0000	2.2389
	Reduced		0.3197	-17.5661	44100.0000	2.2389
MP3	Original	0.1990	0.3197	-17.5661	44100.0000	2.2389
	Reduced		0.3198	-17.8306	44100.0000	2.2389
Quantization	Original	1.4084	0.3197	-17.5661	44100.0000	2.2389
	Reduced		0.3193	-17.8325	44100.0000	2.2389
Down Sampling	Original	2.8519	0.3197	-17.5661	44100.0000	2.2389
	Reduced		0.3197	-17.5672	16000.0000	2.2389
Sample Reduction	Original	3.4861	0.3197	-17.5661	44100.0000	2.2389
	Reduced		0.3196	-17.5662	2205.0000	2.2391
Discrete Cosine Transform	Original	1.2626	0.3197	-17.5661	44100.0000	2.2389
	Reduced		0.3207	-14.6301	44100.0000	1.1338

5.5.8 Body Parts X-Ray Images in PNG

The compression performance on the Body Parts X-Ray Images dataset highlights a clear trade-off between file size reduction and image quality. From Table 35, JPEG, Quantization, and Block Averaging achieved the highest compression ratios, reducing the dataset by more than 73% with relatively low per-file processing time. Down Sampling and Sample Reduction followed closely, also providing significant reductions above 70%.

Table 35 - Compression performance metrics for the *Body Parts X-Ray Images in PNG* dataset using different algorithms

Algorithm	Total Dataset Reduction Time	Mean Execution Time per file	Total Dataset Reduction (MB | %)	Mean File Size Difference (MB | %)
BZ2	26.681	0.0107	1.6861 MB ---- 0.8465 %	0.0007 MB ---- 0.8998 %
JPEG	33.9723	0.0137	146.5311 MB ---- 73.5651 %	0.0591 MB ---- 73.0093 %
Quantization	27.0395	0.0109	146.5190 MB ---- 73.5590 %	0.0591 MB ---- 73.2658 %
Down Sampling	90.8964	0.0366	145.6754 MB ---- 73.1355 %	0.0587 MB ---- 72.7920 %
Sample Reduction	76.8451	0.0310	140.7173 MB ---- 70.6463 %	0.0567 MB ---- 70.5031 %
Block Averaging	33.9723	0.0137	146.5311 MB ---- 73.5651 %	0.0591 MB ---- 73.0093 %

However, Table 36 reveals how these reductions impact image fidelity. BZ2, a lossless algorithm, unsurprisingly preserved perfect image quality, with SSIM = 1.0, PSNR = 100 dB, and GMSD = 0, indicating a bit-for-bit reconstruction with no perceptual degradation.

In contrast, JPEG and Block Averaging offered a strong balance, achieving high compression rates while maintaining good perceptual quality. Both methods scored SSIM values around 0.96 - 0.98, meaning the compressed images are visually very

similar to the originals. Their PSNR values (≈ 40 - 45 dB) are above the typical acceptability threshold (~30 dB), suggesting low noise and little distortion. Additionally, GMSD values under 0.27 indicate minimal structural changes in edges and textures, key in maintaining diagnostic utility in medical images.

Down Sampling and Sample Reduction also performed well, with SSIM above 0.95, PSNR between 37 - 41 dB, and GMSD below 0.25, suggesting that these methods introduce moderate but generally acceptable visual changes.

Quantization, while effective in reducing file size, showed the most noticeable quality degradation: SSIM = 0.85, PSNR ≈ 29 dB, and GMSD = 0.40. An SSIM below 0.9 and PSNR under 30 dB generally indicate perceivable differences that may interfere with tasks requiring high visual accuracy, such as medical diagnostics.

Overall, JPEG and Block Averaging stand out as optimal choices, offering high compression with minimal perceptual impact. BZ2 guarantees perfect fidelity but provides negligible size reduction, making it less practical for large-scale storage or transmission.

Table 36 - Image quality evaluation metrics for different compression algorithms applied to the *Body Parts X-Ray Images in PNG* dataset

Algorithm	SSIM	PSNR	GMSD
BZ2	1.0000	100.0000	0.0000
JPEG	0.9804	44.9187	0.2684
Quantization	0.8545	28.9457	0.4036
Down Sampling	0.9694	40.8765	0.2490
Sample Reduction	0.9565	37.6494	0.2256
Block Averaging	0.9658	39.8994	0.2475

5.5.9 Weather Image Recognition

The compression results for the Weather Image Recognition dataset, as shown in Table 37, highlight the effectiveness of methods like Down Sampling and Sample Reduction. Down Sampling achieved the highest reduction, with an 87.08% reduction in dataset size, followed closely by Sample Reduction with 85.79%. These techniques strike a good balance between efficient compression and file size reduction, making them suitable for storage-limited applications. Quantization performed moderately, reducing the dataset by 58% but with noticeable trade-offs in quality. JPEG and Block Averaging provided moderate compression at around 43%, while BZ2, being lossless, achieved only a small reduction of 5.52%, making it less suited for scenarios demanding high compression.

Table 37 - Compression performance metrics for the *Weather Image Recognition* dataset using different algorithms

Algorithm	Total Dataset Reduction Time	Mean Execution Time per file	Total Dataset Reduction (MB | %)	Mean File Size Difference (MB | %)
BZ2	273.6266	0.0401	33.4961 MB ---- 5.5162 %	0.0036 MB ---- 1.0566 %
JPEG	112.1252	0.0163	262.9912 MB ---- 43.3096 %	0.0038 MB ---- 13.2641 %
Quantization	164.0191	0.0239	352.2181 MB **----** 58.0036 %	0.0505 MB ---- 34.4946 %
Down Sampling	154.7181	0.0225	528.7792 MB ---- 87.0798 %	0.0768 MB ---- 77.8794 %
Sample Reduction	104.7057	0.0153	520.9664 MB ---- 85.7932 %	0.0757 MB ---- 75.9824 %
Block Averaging	112.1252	0.0163	262.9912 MB ---- 43.3096 %	0.0038 MB ---- 13.2641 %

Table 38 reveals how these compression techniques impact image quality. As expected, BZ2, the lossless method, maintained perfect fidelity, with SSIM = 1.0, PSNR = 100 dB, and GMSD = 0, meaning no perceptible degradation occurred during compression. JPEG also performed admirably, achieving an SSIM of 0.9754, PSNR of 41.36 dB, and GMSD of 0.1572. These results suggest that the compression was highly efficient while still retaining excellent visual quality, with very little distortion or noise that would interfere with practical use. Quantization introduced more noticeable degradation, with an SSIM of 0.8915, PSNR of 22.37 dB, and GMSD of 0.3245, indicating perceptible differences in the compressed images that could affect tasks requiring fine details.

Similarly, Down Sampling and Sample Reduction showed good performance, with SSIM values above 0.82, PSNR between 28 - 41 dB, and GMSD below 0.29. These methods resulted in moderate loss of image quality, but the visual changes were still considered acceptable for most applications. Block Averaging showed similar results to Down Sampling, with an SSIM of 0.8397 and PSNR of 27.99 dB, making it an adequate option for compression where some loss of quality is acceptable.

In conclusion, BZ2 remains the best choice for scenarios where image fidelity is paramount, though it offers minimal compression benefits. JPEG stands out as the optimal method, offering a strong balance between compression and image quality. Down Sampling and Sample Reduction are also good alternatives for reducing file sizes while maintaining an acceptable level of image fidelity. However, Quantization should be used cautiously, as its significant quality degradation may interfere with tasks requiring precise image details.

Table 38 - Image quality evaluation metrics for different compression algorithms applied to the *Weather Image Recognition* dataset

Algorithm	**SSIM**	**PSNR**	**GMSD**
BZ2	1.0000	100.0000	0.0000
JPEG	0.9754	41.3598	0.1572
Quantization	0.8915	22.3749	0.3245
Down Sampling	0.8412	28.5014	0.2850
Sample Reduction	0.8223	27.3996	0.2848
Block Averaging	0.8397	27.9974	0.2875

6 CONCLUSIONS AND FUTURE WORK

In this work, we analysed variety of data reduction algorithms and evaluated their suitability for different data types. This enabled us to identify their strengths, limitations and optimal application contexts. As cloud storage becomes increasingly expensive and data transmission more resource intensive, efficient data reduction emerges as a crucial solution. To address this, we developed a pipeline designed to evaluate which algorithms, selected from a pre-classified and studied set, are best suited for reducing specific types of data. The algorithm selection was guided by a comprehensive review of the state of the art and the creation of a taxonomy that categorizes algorithms based on their characteristics, ensuring a structured and representative evaluation.

To make the pipeline more aligned with real world scenarios, we selected datasets containing the most common data types, numerical, temporal, audio and image. The pipeline reads dataset objects, performs a preparation phase that includes data cleaning and formatting, applies a chosen data reduction algorithm, stores the reduced data locally on a Raspberry Pi, and finally collects performance metrics for analysis. We selected the Raspberry Pi to run the data-reduction algorithms as its minimal hardware profile and near-zero background activity enable more accurate performance measurements.

Our evaluation showed that lossless data reduction algorithms generally achieve the highest accuracy by preserving data integrity after decompression, though they tend to be more computationally demanding. For image data, both in colour and grayscale, JPEG and Block Averaging proved to be the most effective. For audio data, MP3 emerged as the best overall algorithm. However, depending on the specific application, techniques such as Sample Reduction, Quantization, and Down Sampling may also be suitable.

Although the algorithms reduce the data volume while preserving critical information, it currently lacks mechanisms for tracking energy consumption and bandwidth usage. These are key factors in environments with limited power or connectivity, such as embedded or edge devices powered by small batteries. Another limitation of this study is that all experiments were conducted using batch processing. It would be valuable to investigate the use of stream processing to determine which approach is more suitable for real time and resource-constrained systems.

The contributions of this work are consolidated into two published papers. The first one shows a categorization of the different data reduction techniques in a form of a taxonomy [7]. On the second one we analysed some of the data reduction techniques from each different taxonomy category [6].

As future work, we plan to incorporate metrics for energy and bandwidth consumption to enable us to evaluate the efficiency of algorithms under constrained

conditions more effectively. We also intend to test the pipeline in practical, real-world scenarios, such as agricultural sensor networks and image systems. This will help us to validate its performance, adaptability and impact in operational environments even further.

REFERENCES

[1] S. Sagiroglu and D. Sinanc, "Big data: A review," Proceedings of the 2013 International Conference on Collaboration Technologies and Systems, CTS 2013, pp. 42–47, 2013.

[2] S. T. Siddiqui, M. R. Khan, Z. Khan, N. Rana, H. Khan, and M. I. Alam, "Significance of Internet-of-Things Edge and Fog Computing in Education Sector," in 2023 International Conference on Smart Computing and Application (ICSCA), IEEE, Feb. 2023, pp. 1–6. Accessed: Apr. 16, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10087582/

[3] F. Mostajabi, A. A. Safaei, and A. Sahafi, "A Systematic Review of Data Models for the Big Data Problem," IEEE Access, vol. 9, pp. 128889–128904, 2021, [Online]. Available: https://ieeexplore.ieee.org/document/9537765/

[4] D. gan Zhang et al., "A Novel Edge Computing Architecture Based on Adaptive Stratified Sampling," Comput Commun, vol. 183, pp. 121–135, Feb. 2022, doi: 10.1016/J.COMCOM.2021.11.012.

[5] K. Jain and S. Mohapatra, "Taxonomy of edge computing: Challenges, opportunities, and data reduction methods," EAI/Springer Innovations in Communication and Computing, pp. 51–69, 2019, doi: 10.1007/978-3-319-99061-3_4/COVER.

[6] V. Fernandes, G. Carvalho, V. Pereira, and J. Bernardino, "Analyzing Data Reduction Techniques: An Experimental Perspective," Applied Sciences 2024, Vol. 14, Page 3436, vol. 14, no. 8, p. 3436, Apr. 2024, doi: 10.3390/APP14083436.

[7] V. Fernandes, J. Bernardino, V. Pereira, and B. Cabral, "A Taxonomy for Data Reduction Techniques," Inforum 2023, 2023, doi: https://inforum2023-ylfd.vercel.app/Atas/paper_7965/7965-CM.pdf.

[8] R. M. Obaise, M. A. Salman, and H. A. Lafta, "Data reduction approach based on fog computing in iot environment," International Conference on Electrical Engineering, Computer Science and Informatics (EECSI), vol. 2020-October, pp. 65–70, Oct. 2020.

[9] D. F. Mahmoud, S. M. Moussa, and N. L. Badr, "The Spatiotemporal Data Reduction (STDR): An Adaptive IoT-based Data Reduction Approach," Proceedings - 2021 IEEE 10th International Conference on Intelligent Computing and Information Systems, ICICIS 2021, pp. 355–360, 2021.

[10] Y. Fathy, P. Barnaghi, and R. Tafazolli, "An adaptive method for data reduction in the Internet of Things," IEEE World Forum on Internet of Things, WF-IoT 2018 - Proceedings, vol. 2018-January, pp. 729–735, May 2018.

[11] Muhammad Habib, C. S. Liew, A. Abbas, P. P. Jayaraman, T. Y. Wah, and S. U. Khan, "Big Data Reduction Methods: A Survey," Data Sci Eng, vol. 1, no. 4, pp. 265–284, Dec. 2016, Accessed: Oct. 16, 2022. [Online]. Available: https://link.springer.com/article/10.1007/s41019-016-0022-0

[12] G. M. Dias, B. Bellalta, and S. Oechsner, "A Survey About Prediction-Based Data Reduction in Wireless Sensor Networks," ACM Computing Surveys (CSUR), vol. 49, no. 3, Nov. 2016, Accessed: Oct. 16, 2022. [Online]. Available: https://dl.acm.org/doi/10.1145/2996356

[13] P. Chhikara, N. Jain, R. Tekchandani, and N. Kumar, "Data dimensionality reduction techniques for Industry 4.0: Research results, challenges, and future research directions," Softw Pract Exp, vol. 52, no. 3, pp. 658–688, Mar. 2022, Accessed: Oct. 16, 2022. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/spe.2876

[14] J. Azar, A. Makhoul, M. Barhamgi, and R. Couturier, "An energy efficient IoT data compression approach for edge machine learning," Future Generation Computer Systems, vol. 96, pp. 168–175, Jul. 2019.

[15] A. Papageorgiou, B. Cheng, and E. Kovacs, "Real-time data reduction at the network edge of Internet-of-Things systems," Proceedings of the 11th International Conference on Network and Service Management, CNSM 2015, pp. 284–291, Dec. 2015.

[16] A. Hanumanthaiah, A. Gopinath, C. Arun, B. Hariharan, and R. Murugan, "Comparison of Lossless Data Compression Techniques in Low-Cost Low-Power (LCLP) IoT Systems," Proceedings of the 2019 International Symposium on Embedded Computing and System Design, ISED 2019, pp. 63–67, Dec. 2019.

[17] J. Mcfarlane and B. R. Chakravarthi, "MP3 compression classification through audio analysis statistics", Accessed: May 24, 2025. [Online]. Available: http://www.aes.org/e-lib

[18] G. Hamano, S. Imaizumi, and H. Kiya, "Effects of JPEG Compression on Vision Transformer Image Classification for Encryption-then-Compression Images," Sensors 2023, Vol. 23, Page 3400, vol. 23, no. 7, p. 3400, Mar. 2023, doi: 10.3390/S23073400.

[19] A. Chen, F. H. Liu, and S. De Wang, "Data reduction for real-time bridge vibration data on edge," Proceedings - 2019 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2019, pp. 602–603, Oct. 2019.

[20] M. I. D. M. V.Radha, "Secured Compound Image Compression Using Encryption Techniques," in International Conference on Computational Intelligence and Computing Research, 2010.

[21] M. Li, X. Yi, and H. Ma, "A scalable encryption scheme for CCSDS image data compression standard," Proceedings 2010 IEEE International Conference on Information Theory and Information Security, ICITIS 2010, pp. 646–649, 2010.

[22] S. Shunmugan and P. A. J. Rani, "Encryption-then-compression techniques: A survey," 2016 International Conference on Control Instrumentation Communication and Computational Technologies, ICCICCT 2016, pp. 675–679, Jul. 2017.

[23] Abdulwahab H.M., S. Ajitha, and M. A. N. Saif, "Feature selection techniques in the context of big data: taxonomy and analysis," Applied Intelligence 2022 52:12, vol. 52, no. 12, pp. 13568–13613, Jan. 2022, Accessed: May 23, 2023. [Online]. Available: https://link.springer.com/article/10.1007/s10489-021-03118-3

[24] Papia Ray, S. Surender Reddy, and Tuhina Banerjee, "Various dimension reduction techniques for high dimensional data analysis: a review," Artif Intell Rev, vol. 54, no. 5, pp. 3473–3515, Jun. 2021, Accessed: May 23, 2023. [Online]. Available: https://link.springer.com/article/10.1007/s10462-020-09928-0

[25] G. M. Dias, B. Bellalta, and S. Oechsner, "The impact of dual prediction schemes on the reduction of the number of transmissions in sensor networks," Comput Commun, vol. 112, pp. 58–72, Nov. 2017, Accessed: Oct. 26, 2022. [Online]. Available: https://dl.acm.org/doi/10.1016/j.comcom.2017.08.002

[26] G. T. Reddy et al., "Analysis of Dimensionality Reduction Techniques on Big Data," IEEE Access, vol. 8, pp. 54776–54788, 2020.

[27] S. A. Abdulzahra, A. K. M. Al-Qurabat, and A. K. Idrees, "Data Reduction Based on Compression Technique for Big Data in IoT," 2020 International Conference on Emerging Smart Computing and Informatics, ESCI 2020, pp. 103–108, Mar. 2020.

[28] M. Agarwal, V. Gupta, A. Goel, and N. Dhiman, "Near Lossless Image Compression Using Discrete Cosine Transformation and Principal Component Analysis," AIP Conf Proc, vol. 2481, no. 1, Nov. 2022, Accessed: May 24, 2023. [Online]. Available: /aip/acp/article/2481/1/020002/2826499/Near-lossless-image-compression-using-discrete

[29] I. F. Ince, F. Bulut, I. Kilic, M. E. Yildirim, and O. F. Ince, "Low dynamic range discrete cosine transform (LDR-DCT) for high-performance JPEG image compression," Visual Computer, vol. 38, no. 5, pp. 1845–1870, May 2022, Accessed: May 24, 2023. [Online]. Available: https://link.springer.com/article/10.1007/s00371-022-02418-0

[30] A. C. Pinto, M. D. Maciel, M. S. Pinho, R. R. Medeiros, F. Motta S, and A. O. Moraes, "Evaluation of lossy compression algorithms using discrete cosine transform for sounding rocket vibration data," Meas Sci Technol, vol. 34, no. 1, p. 015117, Oct. 2022, Accessed: May 24, 2023. [Online]. Available: https://iopscience.iop.org/article/10.1088/1361-6501/ac97fe

[31] S. Sharanyaa, P. N. Renjith, and K. Ramesh, "Classification of parkinson's disease using speech attributes with parametric and nonparametric machine learning techniques," Proceedings of the 3rd International Conference on Intelligent Sustainable Systems, ICISS 2020, pp. 437–442, Dec. 2020.

[32] H. Harb and C. A. Jaoude, "Combining compression and clustering techniques to handle big data collected in sensor networks," 2018 IEEE Middle East and North Africa Communications Conference, MENACOMM 2018, pp. 1–6, Jun. 2018.

[33] Z. Cui, X. Jing, P. Zhao, W. Zhang, and J. Chen, "A New Subspace Clustering Strategy for AI-Based Data Analysis in IoT System," IEEE Internet Things J, vol. 8, no. 16, pp. 12540–12549, Aug. 2021.

[34] S. Singh and R. Devgon, "Analysis of encryption and lossless compression techniques for secure data transmission," 2019 IEEE 4th International Conference on Computer and Communication Systems, ICCCS 2019, pp. 120–124, Feb. 2019.

[35] T. N. Gia, L. Qingqing, J. Pena Queralta, H. Tenhunen, Z. Zou, and T. Westerlund, "Lossless Compression Techniques in Edge Computing for Mission-Critical Applications in the IoT," 2019 12th International Conference on Mobile Computing and Ubiquitous Network, ICMU 2019, Nov. 2019.

[36] A. Nasif, Z. A. Othman, and N. S. Sani, "The Deep Learning Solutions on Lossless Compression Methods for Alleviating Data Load on IoT Nodes in Smart Cities," Sensors 2021, Vol. 21, Page 4223, vol. 21, no. 12, p. 4223, Jun. 2021, Accessed: Jun. 27, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/21/12/4223/htm

[37] R. Jindal, N. Kumar, and S. Patidar, "IoT streamed data handling model using delta encoding," International Journal of Communication Systems, vol. 35, no. 13, p. e5243, Sep. 2022, Accessed: Jun. 27, 2023. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/dac.5243

[38] A. K. M. Al-Qurabat, S. A. Abdulzahra, and A. K. Idrees, "Two-level energy-efficient data reduction strategies based on SAX-LZW and hierarchical clustering for minimizing the huge data conveyed on the internet of things networks," Journal of Supercomputing, vol. 78, no. 16, pp. 17844–17890, 2022.

[39] Q. Qiu, M. Wang, J. Guo, Z. Liu, and Q. Wang, "An adaptive down-sampling method of laser scan data for scan-to-BIM," Autom Constr, vol. 135, p. 104135, Mar. 2022, doi: 10.1016/J.AUTCON.2022.104135.

[40] J. Lötsch, S. Malkusch, and A. Ultsch, "Optimal distribution-preserving downsampling of large biomedical data sets (opdisDownsampling)," PLoS One, vol. 16, no. 8, p. e0255838, Aug. 2021, doi: 10.1371/JOURNAL.PONE.0255838.

[41] B. Ghojogh and M. Crowley, "Principal Sample Analysis for Data Reduction".

[42] J. E. Everett, "Block model data reduction and sensitivity analysis," Transactions of the Institutions of Mining and Metallurgy, Section B: Applied Earth Science, vol. 126, no. 1, pp. 2–10, Jan. 2017, doi: 10.1080/03717453.2016.1195051.

[43] G. N. K. Reddy, M. Sabarimalai Manikandan, and N. V. L. Narasimha Murty, "Lightweight Compressed Sensing (CS) and Partial DCT Based Compression Schemes for Energy-Efficient Wearable PPG Monitoring Devices," InHeNce 2021 - 2021 IEEE International Conference on Health, Instrumentation and Measurement, and Natural Sciences, Jul. 2021, doi: 10.1109/INHENCE52833.2021.9537262.

[44] T. Suel, "Delta Compression Techniques," in Sakr, S., Zomaya, A. (eds) Encyclopedia of Big Data Technologies, Proceedings of the Encyclopedia of Big Data Technologies, 2018. Accessed: Apr. 08, 2024. [Online]. Available: https://doi.org/10.1007/978-3-319-63962-8_63-1

[45] W. Qiao, Z. Fang, M. C. F. Chang, and J. Cong, "An FPGA-based bwt accelerator for bzip2 data compression," in Proceedings - 27th IEEE International Symposium on Field-Programmable Custom Computing Machines, FCCM 2019, 2019. doi: 10.1109/FCCM.2019.00023.

[46] "VitorFernandes2/compression-methods: Master degree thesis in IoT data compression." Accessed: Jul. 20, 2025. [Online]. Available: https://github.com/VitorFernandes2/compression-methods

[47] "Kaggle." Accessed: Nov. 04, 2025. [Online]. Available: https://www.kaggle.com/datasets

[48] "Daily Minimum Temperatures in Melbourne." Accessed: Jan. 10, 2024. [Online]. Available: https://www.kaggle.com/datasets/samfaraday/daily-minimum-temperatures-in-me

[49] "Electric Production." Accessed: Feb. 26, 2024. [Online]. Available: https://www.kaggle.com/datasets/mwafia/electric-production

[50] "beer production | Kaggle." Accessed: Feb. 26, 2024. [Online]. Available: https://www.kaggle.com/code/nainapandey96/beer-production

[51] "Accelerometer Data Set." Accessed: Feb. 26, 2024. [Online]. Available: https://www.kaggle.com/datasets/dhinaharp/accelerometer-data-set

[52] "Smoke Detection Dataset." Accessed: Jan. 10, 2024. [Online]. Available: https://www.kaggle.com/datasets/deepcontractor/smoke-detection-dataset

[53] "Temperature Readings : IOT Devices." Accessed: Jan. 10, 2024. [Online]. Available: https://www.kaggle.com/datasets/atulanandjha/temperature-readings-iot-devices?resource=download

[54] "Musical Instrument Chord Classification (Audio)." Accessed: Jul. 20, 2025. [Online]. Available: https://www.kaggle.com/datasets/deepcontractor/musical-instrument-chord-classification?resource=download

[55] "Body Parts X-Ray Images in PNG." Accessed: Jul. 20, 2025. [Online]. Available: https://www.kaggle.com/datasets/ibombonato/xray-body-images-in-png-unifesp-competion?select=image_png.png

[56] "Weather Image Recognition." Accessed: Jul. 20, 2025. [Online]. Available: https://www.kaggle.com/datasets/jehanbhathena/weather-dataset

Instituto Superior
de Engenharia
Politécnico de Coimbra