International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 04 | April - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Understanding Data compression & overview of its types.

Siddhesh Yeshwante, Chinmay patil

ABSTRACT
This paper explains data compression, including its definition, applications, and its different techniques. A summary of the benefits and drawbacks of the various data compression techniques is provided. We've also covered the procedures for encoding and decoding different codes.

Keywords: Data Compression, lossy, lossless, redundancy, reliability, algorithms.

INTRODUCTION

Large data sets, which are expanding in size and come from multiple heterogeneous sources, are becoming harder to store, maintain, process, and transfer in real time. Often, data files include extraneous and redundant information that can be removed to reduce the file's size. A range of approaches are referred to as compression techniques in order to achieve this goal. Text, picture, audio, and video data can all be handled with these techniques. Condensed data expedites knowledge discovery while also improving processing times and transfer speeds.

WHAT IS DATA COMPRESSION ?

Data compression is the process of reducing the amount of data required to represent information, making it more efficient for storage and transmission. This technique involves encoding information using fewer bits than the original representation, which can significantly save storage space and enhance data transfer speeds.

Challenges associated with lager size of data include:
1. Difficulties in Storage.
2. Complexity in Management.
3. Data Overload.
4. Challenges faced during Integration.
5. Longer Processing Times.
6. Data Security and Privacy Concerns.
7. Real-Time Analytics Challenges.

Reasons for the use of Data Compression:
1. Storage Savings due to smaller file sizes.
2. Faster Data Transmission.
3. Enhanced Data Management.
4. Improved Performance.
5. Cost Efficiency.
6. Improved Scalability.
7. Reduced Data Transfer Costs.

Literature Review

The significance of data compression.
Data compression has the ability to significantly reduce the storage requirements of a file. For example, a file weighing 20 megabytes (MB) requires 10 MB of space when compressed with a 2:1 ratio. Administrators spend less money and time on storage as a result of compression. Data compression improves the performance of backup storage and has been observed lately in the data reduction of primary storage. Since data is growing at an exponential rate, compression will be a crucial data reduction technique. Most file types can be compressed, however when selecting which files to

compress, it's crucial to adhere to best practices. Compressing files that, in general, may already be compressed won't really make a difference.

Working Principle for data compression.

A program that use an algorithm or formula to calculate how to reduce the size of the data performs the data compression. The algorithm can insert a reference or pointer to a string of 0s and 1s that the program has already seen, or it can represent a string of bits, or 0s and 1s, with a smaller string of 0s and 1s by converting them between them using a dictionary. Text compression has the ability to eliminate all unnecessary characters, replace frequently occurring bit strings with smaller bit strings, and insert a single repeat character to represent a string of repeated characters. A text file's initial size can be lowered by 50% or a substantially larger percentage thanks to data compression. Compression of the data content or the full transmission unit, including header data, is possible when transmitting data. The data is transferred or received over the internet in compressed formats such as ZIP, GZIP, or others. It can be sent or received individually or in combination with other data as part of an archive file.

Figure 1: Working of data compression in general

In figure 1, it explains the process of data compression in general of how the uncompressed data will be processed by compression method then the data that has been compressed will produce a smaller file size than the size of the file before it was compressed.

Data Compression Techniques.

There are two major families of compression techniques in terms of the possibility of reconstructing the original source. They are called Lossless and lossy compression.

Data compression types:

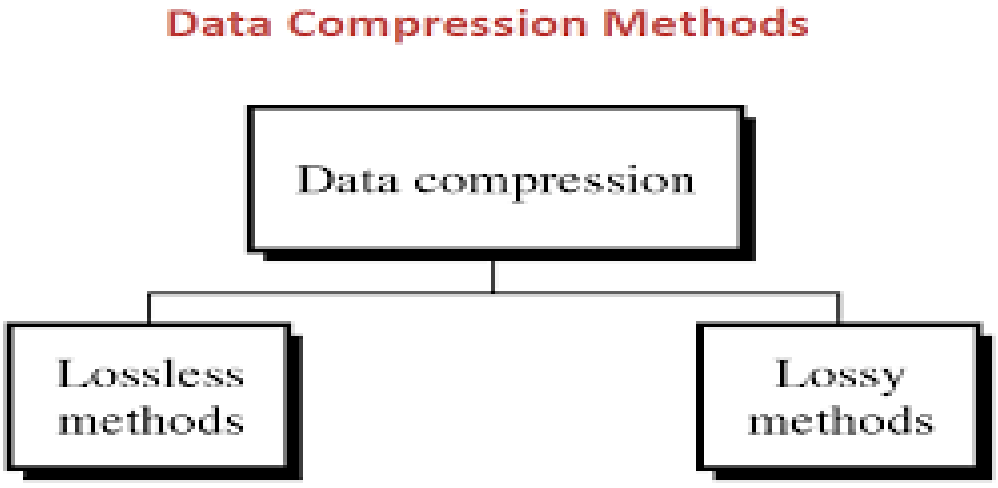


Figure 2 : Types of techniques for Data compression.

In figure 2, it shows the different techniques of data compression. There are two major families of compression techniques in terms of the possibility of reconstructing the original source. They are called Lossless and lossy compression.

Lossless Data Compression
By using data compression algorithms, lossless data compression makes it possible to recreate the exact original data from the compressed data. The idea of lossless data compression is in opposition to lossy data compression, which prevents precise reconstruction of the original data from the compressed data. Applications for lossless data compression are numerous. It is typically utilized by the Unix utility gzip as well as the widely used ZIP file format. Additionally, this method is frequently a part of lossy data compression systems. When it's crucial that the original and decompressed data match or when it's impossible to determine if a particular divergence is unimportant, lossless compression is employed. Source code and executable programs are two examples.
A select number picture file formats—most notably PNG—only employ lossless compression; other formats, like TIFF and MNG, can employ lossy or lossless techniques. The GIF uses a lossless compression technique, but because most implementations of the format cannot display images in full color, they quantize the image to 256 colors or less before encoding it as a GIF (sometimes using dithering). Although the process of color quantization is lossy, no more loss is produced when the color image is rebuilt and then re-quantized. When a file is uncompressed, lossless compression allows it to be restored to its original state without losing any data at all.
The usual method for compressing executables, text, and spreadsheet files—where removing words or numbers would alter the data—is called lossless compression. The kinds of data that lossless compression techniques are intended to compress can be used to classify them. Targets for compression algorithms mostly attempt to compress text, executables, pictures, and sound. A statistical model is created for the input data by one algorithm, and the other uses this model to map the input data to bit strings so that "probable" data will result in shorter output than "improbable" data. These two types of algorithms are used by the majority of lossless compression programs.

Lossless compression algorithms

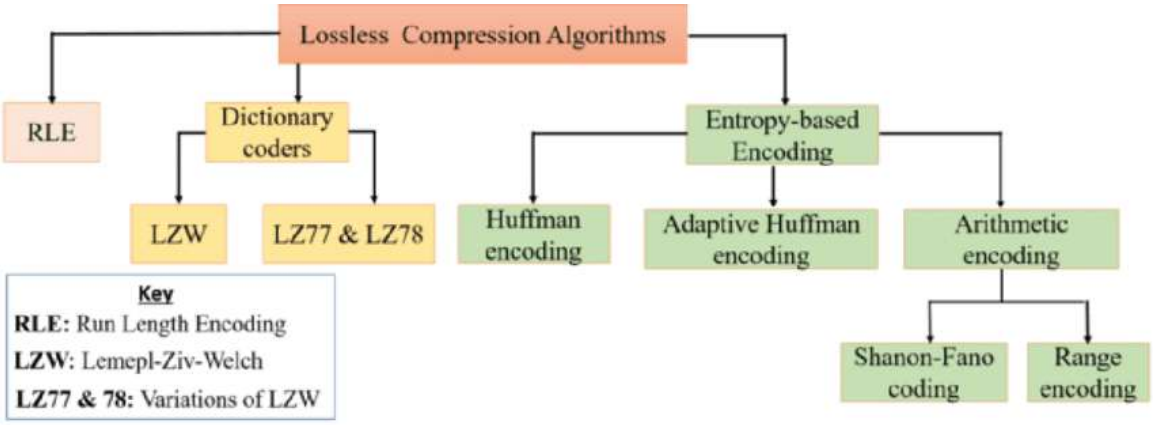


Figure 3: Lossless compression algorithms

In figure 3, the diagrammatic representation of the different compression algorithms for the lossless type of data compression.

Lossless compression algorithms
1. Run-length encoding (also known as RLE)
2. Dictionary coders:
    a. LZW
    b. LZ77 & LZ78

3. The entropy encoding:
    a. The Huffman coding
    b. Adaptive Huffman coding
    c. arithmetic coding (more advanced)

Run-length encoding (RLE)
Run-length encoding is a simple and effective lossless data compression technique that encodes sequences of repeated data values as a single value and a count. This method is particularly useful for compressing data that contains many consecutive repeated characters or pixels, making it ideal for applications like simple images and animations.

How Run-Length Encoding Works
The RLE algorithm operates by identifying runs of consecutive identical elements in the data. The basic steps involved in the encoding process are:
Traverse the Input Data: The algorithm scans through the data to identify sequences of repeated characters.
Count Consecutive Occurrences: For each character, it counts how many times it appears consecutively (the "run length").
Store the Character and Count: The character and its corresponding count are stored as a pair.

Example of RLE
For example, consider the string "aaabcccc". The run-length encoding would transform this into "a3b1c4", where:'a' appears 3 times,
'b' appears once,
'c' appears 4 times.
This method effectively reduces the amount of data required to represent the original string, especially when there are long runs of repeated characters.

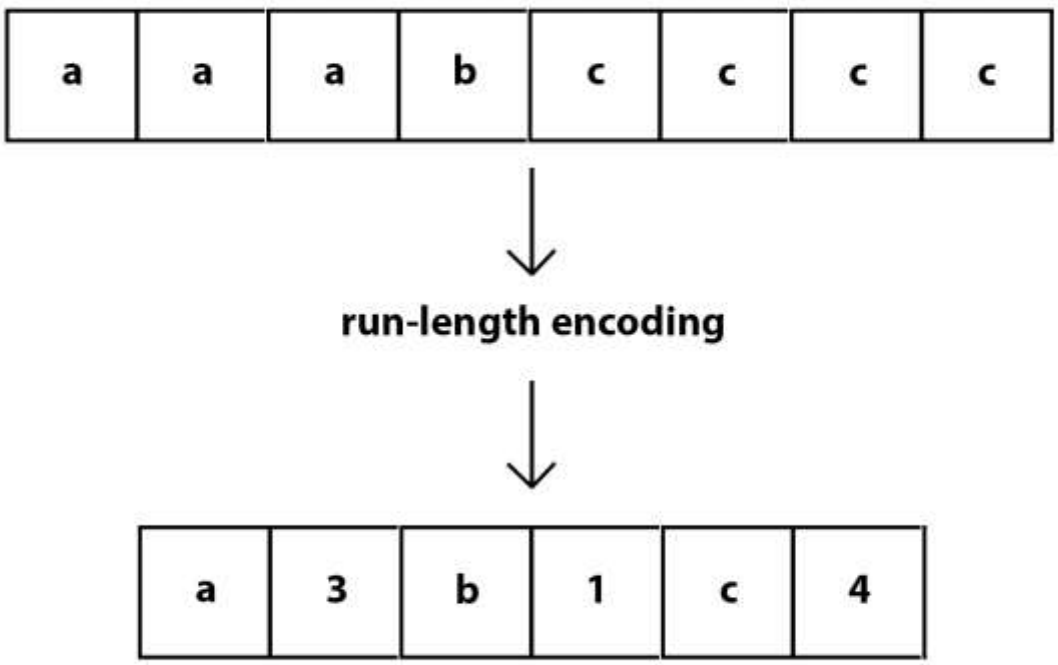


Figure 4: Example of Run-length Encoding

In figure 4, The character and its corresponding count are stored as a pair (e.g., for the string "AAAABBB", the output would be "4A3B").

Applications of Run-Length Encoding
RLE is commonly used in various applications, including:
Image Compression: Particularly effective for black and white images or images with large areas of uniform color, such as icons and simple graphics.
Data Transmission: Used in protocols where data redundancy can be minimized to speed up transmission.
File Formats: Found in formats like TIFF and certain bitmap images, where it helps reduce file size without losing any information.

Advantages and Limitations
Advantages
Simplicity: The algorithm is straightforward and easy to implement.
Lossless Compression: RLE allows for perfect reconstruction of the original data, making it suitable for critical applications where data integrity is essential.
Limitations
Inefficiency with Non-Repetitive Data: RLE performs poorly on data with few or no runs of repeated values. In such cases, it can actually increase the size of the data.
Limited Compression: The effectiveness of RLE diminishes as the randomness of the data increases, making it less suitable for complex images or data sets with high variability.

In summary, run-length encoding is a valuable compression technique for specific types of data, particularly where redundancy is prevalent. Its simplicity and lossless nature make it a useful tool in various fields, despite its limitations in handling more complex data structures.

Dictionarycoders:
A dictionary coder, also known as a substitution coder, is a class of lossless data compression algorithms that encodes data by replacing sequences of characters with shorter representations based on a predefined set of strings stored in a data structure called a dictionary. This method is effective for reducing the size of data by eliminating redundancy.

a.LZW
LZW (Lempel-Ziv-Welch): This is a widely used dynamic dictionary coder that initializes its dictionary with all possible symbols and adds new entries as it encounters new sequences during encoding.

How LZW Works
Encoding Process
a. Initialization: The algorithm begins with a dictionary initialized with single-character entries corresponding to the ASCII character set (0-255). Each character is assigned a unique code.
b. Reading Input: As the algorithm processes the input data, it builds longer sequences of characters. It maintains a current sequence P and reads the next character C.
c. Dictionary Lookup: The algorithm checks if the combination P + C exists in the dictionary:
If it exists, it updates P to P + C and continues.
If it does not exist, it outputs the code for P, adds P + C to the dictionary with a new code, and resets P to C
d. Output: This process continues until the entire input is processed, resulting in a series of output codes that represent the original data in a compressed format.

DecodingProcess
a. Rebuilding the Dictionary: The decoder starts with the same initial dictionary of single-character entries. It reads the output codes and reconstructs the original data by translating the codes back into their corresponding strings.

b. Dynamic Updates: As new sequences are encountered during decoding, the dictionary is updated similarly to the encoding process, allowing the decoder to handle the compressed data effectively.

Advantages of LZW
Lossless Compression: LZW ensures that no data is lost during compression, allowing for exact reconstruction of the original data.
Efficiency: The algorithm is particularly effective for files with a lot of repetitive data, achieving compression ratios of 60-70% for certain text files.
No Prior Knowledge Required: LZW does not require any prior knowledge of the data being compressed, making it versatile for various data types.
Single Pass Compression: The algorithm processes the input data in a single pass, making it efficient in terms of speed.

Limitations of LZW
Inefficiency with Non-Repetitive Data: LZW may not perform well on data with little redundancy, potentially resulting in larger compressed files.
Patent Issues: Some versions of the algorithm have been subject to patent restrictions, leading to licensing fees for commercial use.
Variable Compression Ratios: The effectiveness of LZW can vary significantly depending on the nature of the input data, with better results seen in data with long, repeated sequences.

Applications
LZW is widely used in several formats and applications, including:
GIF Images: The algorithm is integral to the GIF format, allowing for efficient storage of image data.
TIFF and PDF Files: LZW is also used in TIFF and occasionally in PDF files for lossless compression.
Unix Compression Utilities: The algorithm is part of the Unix file compression utility, providing a straightforward method for reducing file sizes.

In summary, LZW is a powerful and efficient lossless compression algorithm that leverages a dynamic dictionary to replace repeated sequences with shorter codes, making it suitable for a variety of applications where data integrity is crucial. Its simplicity and effectiveness have made it a foundational technique in the field of data compression.

b. LZ77 & LZ78
LZ77 and LZ78 Compression Algorithms
LZ77 and LZ78 are two foundational lossless data compression algorithms developed by Abraham Lempel and Jacob Ziv in 1977 and 1978 respectively. Both algorithms belong to the Lempel-Ziv family and utilize dictionary-based compression techniques.

LZ77
The key ideas behind LZ77 are:
It maintains a sliding window of previously seen data during compression.
The encoder searches for matches within this window and encodes them using a tuple: (offset, length, next char).
Offset indicates the distance back in the uncompressed data where the matching substring begins.
Length specifies the length of the matching substring.
next char is the character that follows the matching substring in the input.
This allows LZ77 to effectively compress data by referencing previously seen sequences without explicitly building a dictionary.

LZ78
In contrast to LZ77, LZ78 builds a more explicit dictionary from the input data:
It starts with an empty dictionary and processes the input character by character.

When a match is found, the encoder outputs the index of the last matching sequence along with the next character.
If no match is found, it adds the current sequence to the dictionary and outputs the corresponding index.
LZ78 constructs a dictionary that grows dynamically during compression.

Differences between LZ77 and LZ78
LZ77 maintains a sliding window and encodes matches using a tuple, while LZ78 builds an explicit dictionary.
LZ77 can achieve better compression ratios in some cases, as it can reference any previously seen sequence within the window.
LZ78 has a simpler decompression process, as the dictionary can be rebuilt from the encoded data alone.
LZ77 requires the decoder to maintain a sliding window, while LZ78 allows for random access to the compressed data.

ApplicationsLZ77
Both LZ77 and LZ78 form the basis for many popular compression algorithms and formats:
LZ77 is used in the DEFLATE algorithm, which is used in ZIP and PNG file formats.
LZ78 is used in the Unix compress utility and the GIF image format.
Variants of these algorithms, such as LZW (Lempel-Ziv-Welch) and LZMA, are widely used in various compression tools and file formats.

In summary, LZ77 and LZ78 are fundamental lossless compression algorithms that utilize dictionary-based techniques to replace repeated data with shorter codes. While they share similar goals, they differ in their approach to dictionary management and compression efficiency, making them suitable for different applications and scenarios.

3. The entropy encoding:
Entropy encoding is a lossless data compression technique that assigns variable-length codes to input symbols based on their probabilities. The main goal is to reduce the average code length, making it efficient for compressing data streams.

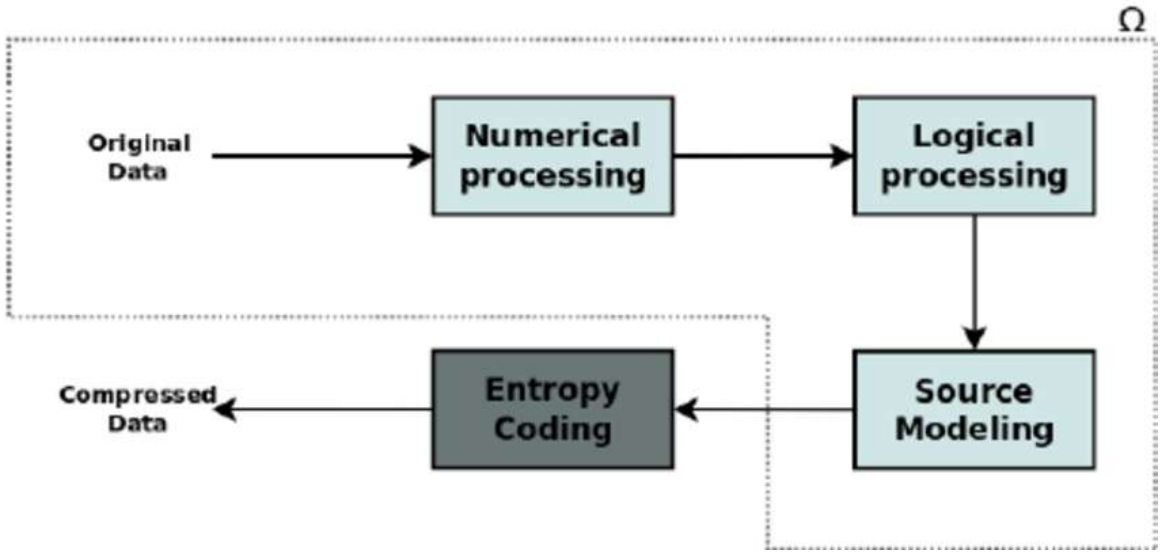


Figure 5: Diagrammatic representation of entropy encoding

In figure 5, The Entropy encoding used in lossless data compression is explained. Where the data goes through various processing units such as numerical processing, logical processing, source modelling and then entropy coding and thus after that we have the compressed data.

a. The Huffman coding
For lossless data compression, this entropy encoding approach is employed. This method describes encoding a source symbol, like a character in a file, using a variable length code table. The variable-length code is derived in a specific

fashion based on the predicted likelihood of occurrence for each potential value of the source symbol. David A. Huffman created it, and he published it in the 1952 paper "A Method for the Construction of Minimum-Redundancy Codes." As a result, Huffman coding is ideal for symbol-by-symbol coding when the input probability distribution is known; nevertheless, its optimality may inadvertently be overstated. LZW and arithmetic coding are frequently more capable of compression.
When input probabilities are not precisely known, the following two approaches can combine an arbitrary number of symbols for more efficient coding, and they can also generally adjust to the actual input statistics.

b. Adaptive Huffman coding
Adaptive Huffman coding is a form of adaptive coding that is based on Huffman coding. It allows for one-phase encoding and adapts to changing data conditions by creating the code as the symbols are being sent without prior knowledge of the source distribution. One-pass encoding has the advantage of allowing for real-time source encoding, but at the cost of increased sensitivity to transmission errors—a single loss can completely destroy the code.

c. arithmetic coding (more advanced)
One technique for lossless data compression is arithmetic coding. This is entropy encoding in a different form. Rather of breaking the input message down into its individual symbols and replacing each one with a code word, as other entropy encoding techniques do, arithmetic coding encodes the entire message into a single number, a fraction n where (0.0 = n < 1.0).

Lossy Data Compression

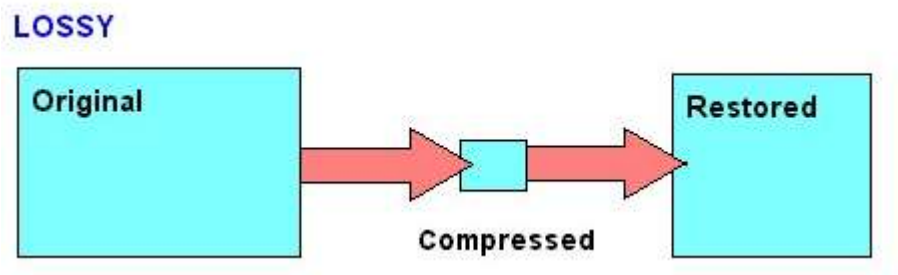


Figure 6 : Lossy compression type for compressing of data.

In figure 6, the given image shows how the lossy data compression would lose data when its restored. Hence, the name lossy compression, The data lost may be crucial.

A compression technique that does not decompress digital data back to 100% of the original. Lossy methods can provide high degrees of compression and result in smaller compressed files, but some number of the original pixels, sound waves or video frames are removed forever. Examples are the widely used JPEG image, MPEG video and MP3 audio formats.

The greater the compression, the smaller the file. However, a high image compression loss can be observed in photos printed very large, and people with excellent hearing can notice a huge difference between MP3 music and high-resolution audio files (see audiophile). Typically, the moving frames of video can tolerate a greater loss of pixels than still images.

Lossy compression is never used for business data and text, which demand a perfect restoration.

Lossy compression on images


40K
2013


200K
2013

Figure 7: Lossy compression on image

In figure 7, The top JPEG image is one fifth the file size of the bottom image, which was compressed the least. Look carefully at the orange section on the top of this cake. Sometimes Hard to Tell.

Lossy compression on text

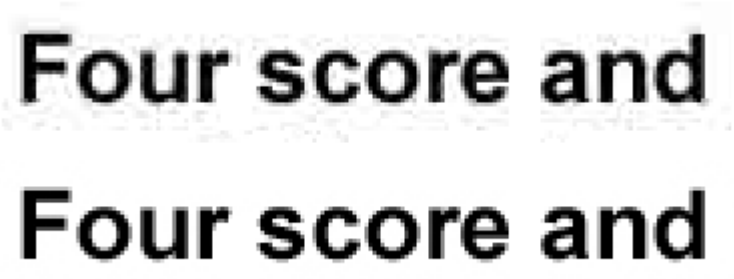


Figure 8: Lossy compression on text

In figure 8, The top JPEG image was compressed the most, and the text is not as sharp as the least compressed. The bottom blow-up shows the distortion clearly. Easier to See with Text.

Comparison with Lossless Compression
Lossless compression reduces file size without any loss of data, allowing the original to be perfectly reconstructed. It is typically used for text files, executable files, and images where quality is critical. However, the compression ratios are much lower compared to lossy compression.

In summary, lossy compression is an essential technique for efficiently storing and transmitting multimedia data by trading off some quality for significantly reduced file sizes. While it introduces irreversible changes, the human perception is often unaffected. The choice between lossy and lossless depends on the specific application and requirements.

Products and technologies that employ data compression
Many technologies, such as databases, operating systems, storage systems, and software programs utilized by corporations and enterprise groups, all include compression. Data compression is also frequently found in consumer electronics including laptops, desktop computers, and mobile phones. Here, a lot of devices and systems carry out compression transparently, while some allow users to switch compression on or off. It can be done more than once on the same file or set of data, however depending on the data compression algorithms used, extra compression is minimal or non-existent and the file size may even increase slightly.
A well-known Windows application called WinZip compresses files when it bundles them into an archive. RAR and ZIP are two archive file formats that allow compression.

METHODOLOGY

This research paper combines qualitative and quantitative analysis to learn what people think about reliability and significance of data compression and its techniques with the increasing volumes of data. We can analyse and draw a conclusion from people's responses. In order to gather data regarding people's awareness, we first polled those who used online form creators and data collection services.

PUBLIC SERVEY

The survey is used to gather the data. Both the outcome and the process by which it was arrived at will be examined. In this instance, 80 people were asked their opinions about questions pertaining to the subject of Understanding Data

compression & overview of its types. Conducting a survey is essential to obtaining reliable data that can be analysed and used to determine the survey's outcome.

QUESTIONNAIRES

Have you heard about data compression before?
(Yes/No)

Do you think data compression is important?
(Yes/No)

What types of data compression techniques are you familiar with?
a. None
b. Lossy compression
c. Lossless compression
d. Both

In your opinion, does increasing data volume affect the reliability of data compression?
a. Yes
b. No
c. Unsure

Which data compression algorithms do you believe maintain reliability as data volume increases? (Select all that apply)
a. Huffman Coding
b. Lempel-Ziv-Welch (LZW)
c. Arithmetic Coding
d. JPEG (for images)

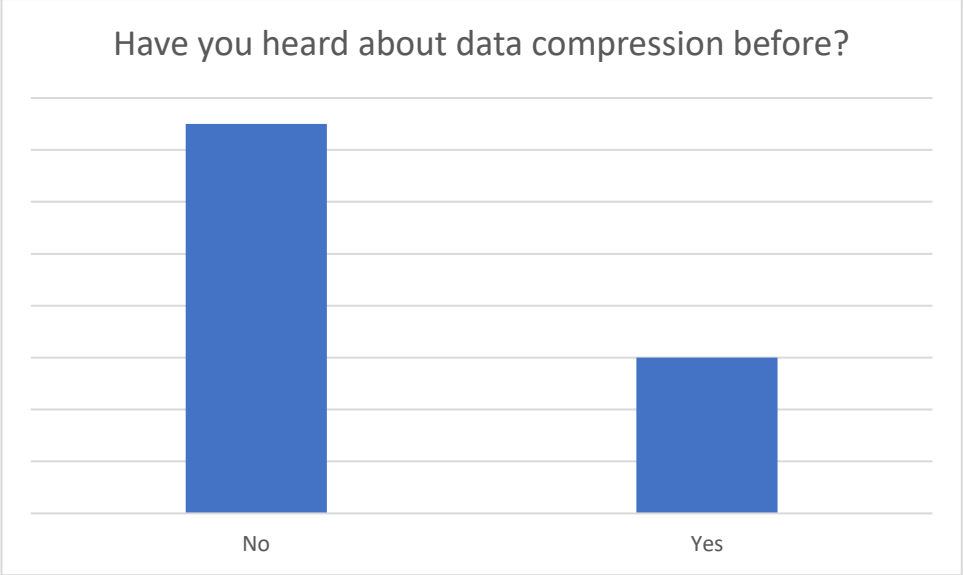
Have you observed a difference in reliability between lossless and lossy compression methods when dealing with large datasets?
a. Yes
b. No
c. Unsure

How important is it for organizations to consider data volume when choosing a compression method?
a. Very important
b. Somewhat important
c. Not important

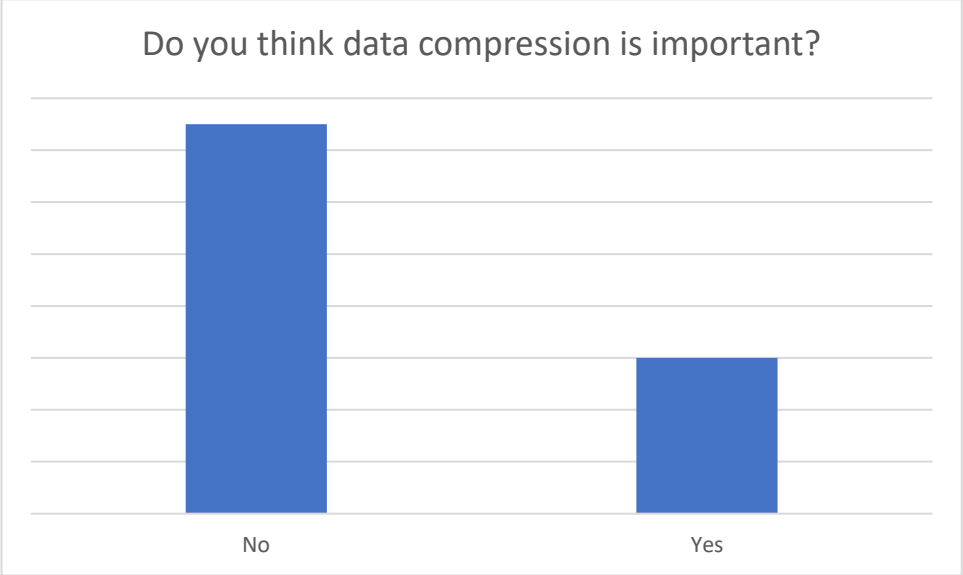
Do you believe that machine learning or AI could play a role in enhancing the reliability of data compression as data volumes grow?
a. Yes
b. No
c. Unsure

RESULTS

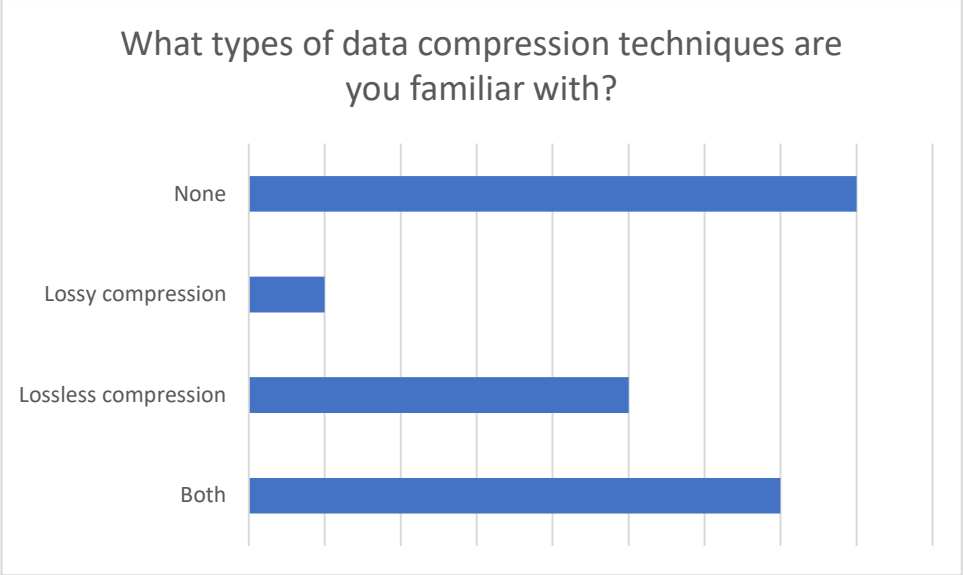
Have you heard about data compression before?
(Yes/No)



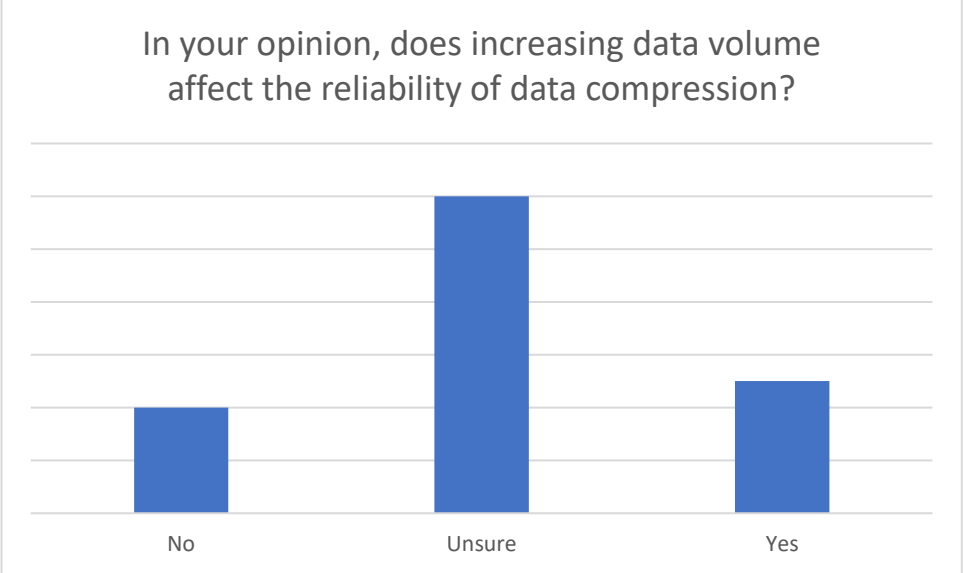
Do you think data compression is important?
(Yes/No)



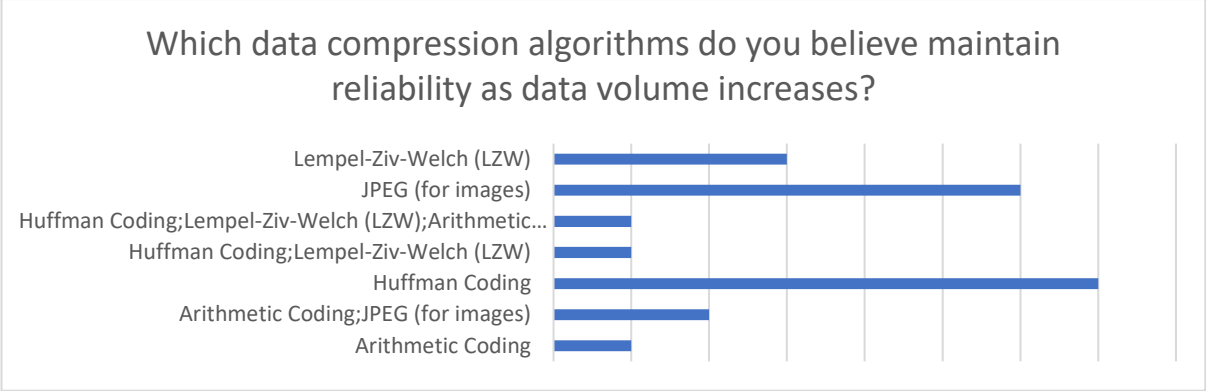
What types of data compression techniques are you familiar with?
a. None
b. Lossy compression
c. Lossless compression
d. Both



In your opinion, does increasing data volume affect the reliability of data compression?
a. Yes
b. No
c. Unsure

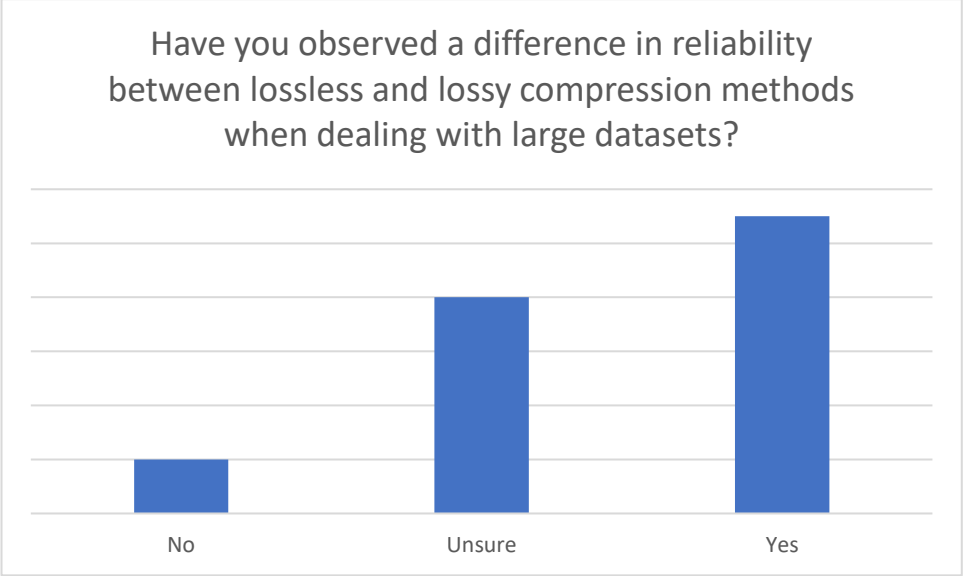


Which data compression algorithms do you believe maintain reliability as data volume increases? (Select all that apply)
a. Huffman Coding
b. Lempel-Ziv-Welch (LZW)
c. Arithmetic Coding
d. JPEG (for images)



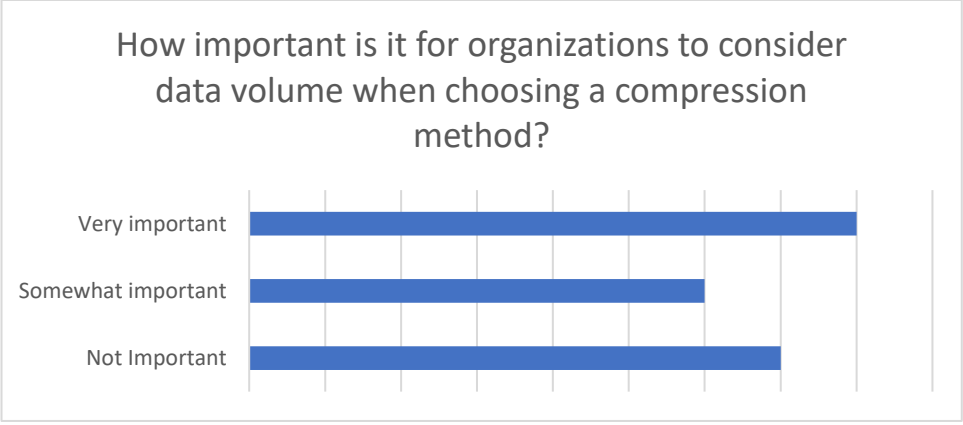
Have you observed a difference in reliability between lossless and lossy compression methods when dealing with large datasets?

a. Yes
b. No
c. Unsure

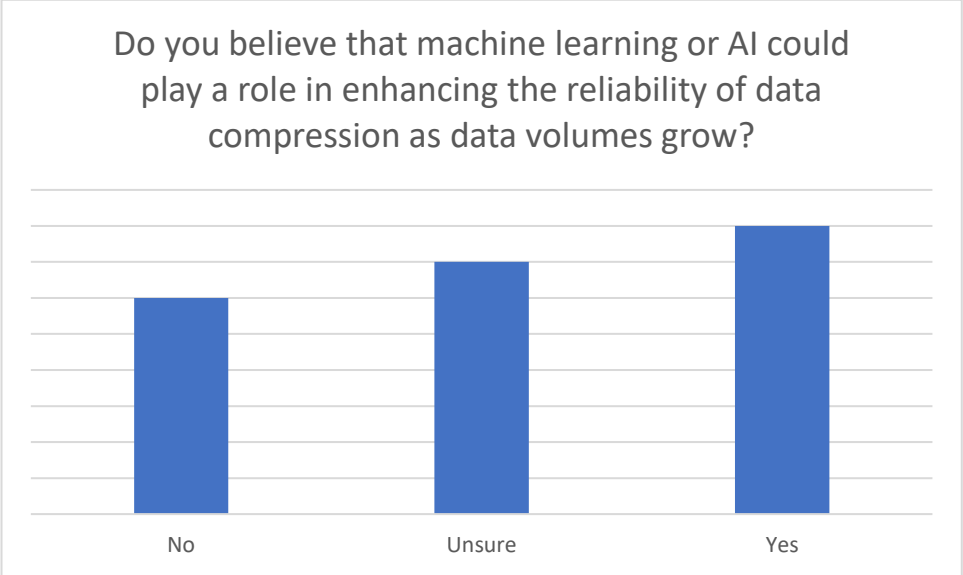


How important is it for organizations to consider data volume when choosing a compression method?

a. Very important
b. Somewhat important
c. Not important



Do you believe that machine learning or AI could play a role in enhancing the reliability of data compression as data volumes grow?
a. Yes
b. No
c. Unsure



HYPOTHESES TESTING

Hypothesis testing is a sort of statistical reasoning that includes analysing data from a sample to derive inferences about a parameter of population or probability distribution. First, a hypothesis is created regarding the parameter or distribution. This is known as the null hypothesis, abbreviated as H0. Following that, we establish an alternative hypothesis (Ha), which contrasts with the null hypothesis.[2] Hypothesis testing utilizes sample data to assess whether the null hypothesis (H0) should be rejected. If H0 is rejected, the statistical conclusion is that the alternative hypothesis (Ha) is supported. For this paper, Null hypothesis (H0): Educational games do not significantly improve students' cognitive skills and engagement levels compared to traditional learning methods. Alternative hypothesis (Ha): Educational games significantly improve students' cognitive skills and engagement levels compared to traditional learning methods.

Test (Statistics)

There are several tests available to decide whether to reject the null hypothesis, including: 1. Chi-squared test 2. T-student test (Ttest) 3. Fisher's Z test. This paper will focus on the Chi-squared Test, which is a statistical method used for categorical data to assess whether the observed data significantly differ from expected values. [3]

Level of significance (also known as alpha or α): A significance level of 0.05 indicates a 5% chance of finding a difference when none exists. Lower significance levels ask for powerful evidence to reject the null hypothesis. [4]

Level of confidence: The confidence level reflects the probability that a statistical parameter (such as the mean) derived from a sample is also representative of the entire population.

Level of significance = 0.05 i.e., confidence = 95%

The likelihood of accepting the null hypothesis in a chi-squared test is influenced by the selected significance level and whether the calculated chi-squared value meets or exceeds that significance level.

Step 1: Identify the null and alternative hypotheses:

Null hypothesis (H0): Educational games do not significantly improve students' cognitive skills and engagement levels compared to traditional learning methods. Alternative hypothesis (Ha): Educational games significantly improve students' cognitive skills and engagement levels compared to traditional learning methods.

We are considering 50 random observations from a public survey that collected 101 responses. The following table provides the data necessary to calculate the Chi-Square test for the selected observations.

table 6.1: observed values

Observed	Yes	No	Total
Student	18	1	19
Parent	8	6	14
Teacher	8	0	8
Others	8	1	9
Total	42	8	50

Step 2: Determine the expected values

E = (Row Total X Column Total) / Grand Total

For example, Student (Yes) = (42*19) / 50 = 15.96

Similarly

table 6.2: excepted values

Excepted	Yes	No
Student	15.96	3.04
Parent	11.76	2.24
Teacher	6.72	1.28
Others	7.56	1.44

Step 3: To calculate the Chi-Square Statistic

table 6.3: statistic values

(Observed – Expected)2 / Expected	Yes	No
Student	0.26075188	1.368947368
Parent	1.202176871	6.311428571
Teacher	0.243809524	1.28
Others	0.025608466	0.134444444

$X2 = \sum (\text{observed value} - \text{expected value})^2 / \text{expected value}$

X2 = 10.82717

Step 4: To calculate Degree of Freedom(df) df = (rows − 1) × (columns − 1) df = (4 − 1) × (2 − 1) **df = 3 * 1 = 3**

Step 5: To calculate critical chi square value

Using a chi-square distribution table at a **significance level of 0.05 with 3 degrees of freedom**, the critical value is approximately **7.8157**.

Given that the **calculated chi-square statistic (10.82717) is higher than the critical value (7.8157)**, we will **reject the null hypothesis.**

This means that we can say that educational games significantly improve students' cognitive skills and engagement levels compared to traditional learning methods.

FINDINGS

Awareness and Support: A significant majority is unaware of Understanding Data compression & overview of its types.

Perceived Benefits: Key benefits identified include improved Understanding Data compression.

Challenges: Concerns exist regarding reliability and necessity.

Future Outlook: Most respondents believe data compression will become essential in reflecting a positive perspective on technology's role despite challenges.

REFERENCES

Huffman, D. A. 1952. A Method for the Construction of Minimum-Redundancy Codes. Proc. IRE 40, 9 (Sept.), 1098-1101.

Ziv, J., and Lempel, A. 1977. A Universal Algorithm for Sequential Data Compression. IEEE Trans. Inform. Theory 23, 3 (May),337-343.

Ziv, J., and Lempel, A. 1978. Compression of Individual Sequences via Variable-Rate Coding. IEEE Trans. Inform. Theory 24, 5 (Sept.), 530-536.
https://en.wikipedia.org/wiki/Data_compression

https://www.geeksforgeeks.org/introduction-to-data-compression/

https://www.youtube.com/watch?v=gmHoX_EqWF4