First Libyan international Conference on Engineering Sciences & Applications (FLICESA_LA)
13 – 15 March 2023, Tripoli – Libya

Data Compression Algorithms and their Applications

Basma Emhamed Dihoum
dept. Computer Science
University of Jafara Tripoli, Libya
basmadihom@gmail.com

***Abstract*—As the need for information storage and data transfer increases, data compression has become increasingly crucial. The purpose of data compression is to reduce the size of data, which is particularly helpful when transferring large files over networks or storing them on limited-capacity devices. With the rise of the internet and mobile devices with limited resources, data compression has become even more essential in conserving storage and bandwidth and shortening download times. There are various techniques to achieve data compression, and in this survey, I will extensively discuss important compression algorithms, evaluate their performance, and highlight their key features. Additionally, I will explore the benefits of combining the RLE and Huffman algorithms to enhance the compression process, and demonstrate how their combined use produces better results than using the RLE algorithm alone.**

Keywords—Data Compression, compression techniques, lossless compression, lossy compression, Compression Ratio, Huffman, Shannon Fano, RLE, LZW, RH.

I. INTRODUCTION

The growing demand for information storage and data transfer has made efficient compression techniques a critical requirement for optimal performance. The exponentialincrease in multimedia files, social media content, and sensor data has presented significant challenges for storage and transfer on networks and limited-capacity devices. This paper provides a comprehensive review of data compressiontechniques and their importance for information storage and transfer. We evaluate the effectiveness of various compression methods, including RLE and Huffman algorithms, based on their compression ratio, computational complexity, and other performance metrics. Our findings indicate that combiningRLE and Huffman algorithms produces better compression outcomes than using the RLE algorithm alone. Furthermore, we explore the benefits of other compression techniques, such as RLE, LZW, Huffman and Shannon, to demonstrate their significance in specific applications.

The implications of this study extend to various applications, including multimedia storage, wireless communications, and data centers, where efficient compression is crucial for optimal performance and resource utilization. Efficient compression significantly reduces storage requirements and transfer time, enabling faster processing, retrieval, and analysisof data. Our research contributes to the existing literature on data compression techniques and performance evaluation, providing valuable insights for both researchers and practitioners. The results of this study can guide the development and optimization of compression algorithms and their applications in diverse fields. Overall, this paper underscores the importance of efficient compression techniques in the growing field of digital data management. [1], [2].

AIMS AND OBJECTIVES OF THE SUBJECT

1) The objectives of the topic are to identify importantissues in data compression, describe a variety of data compression techniques, and explain data compression algorithms
2) The main objective of the research is to improve the work of the RLE algorithm in terms of data compression without losing data.
3) This research aims to compare data compression algorithms and which one is better at compressing text data Maintaining the Integrity of the Specifications

II. RELATED WORK

Data compression algorithms are widely in various applications to reduce the size of digital data and improve efficiency. Here are some related works on data compression algorithms and their applications.

One such study by Al-laham Mohammed and Ibrahiem M. M. El Emary [3] compared the Huffman algorithm and LZW, concluding that Huffman was better suited for compressing text data. Another study by Neha Sharma and Usha Batrab [4] introduced two algorithms, Huffman and RLE, for compressing lossy image data, with Huffman performing better than RLE. In a study by Luluk Anjar Fitriya, Tito Waluyo Purboyo, and Anggunmeka Luhur Prasasti [2], the Huffman algorithm was found to be the best in terms of data compression rate compared to other lossless data compression algorithms. Additionally, Ruchi Gupta, Mukesh Kumar, and Rohit Bathla [5] provided an overview of various data compression methods, evaluating them for their compression ratio, efficiency, and susceptibility to errors. These studies provide insights intothe performance of different compression algorithms and help identify the most suitable algorithm for specific applications

III. DATA COMPRESSION

Data compression is simply a means for efficient digital representation of a source of data such as text, image and the sound. The goal of data compression is to represent a source

in digital form with as few bits as possible while meeting the minimum requirement of reconstruction. This goal is achieved by removing any redundancy presented in the source.
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. [1], [6]

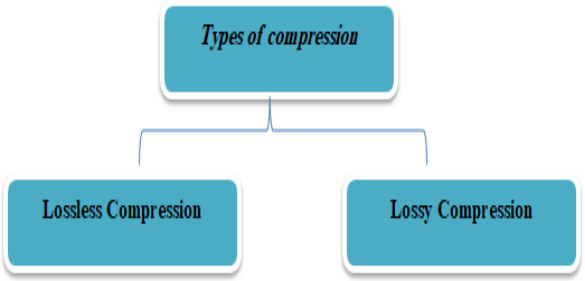

Fig. 1. shows the types of data compression

A. *Lossless compression*

Lossless compression A compression approach is lossless only if it is possible to exactly reconstruct the original data from the compressed version. There is no loss of any information during the compression1 process. [7] Lossless compression techniques are mostly applied to symbolic data such as character text, numeric data, computer source code and executable graphics and icons. Lossless compression techniques are also used when the original data of a source areso important that we cannot afford to lose any details. For example, medical images, text and images preserved for legal reasons; some computer executable files, etc.

B. *Lossy compression*

A compression method is lossy compression only if it is not possible to reconstruct the original exactly from the compressed version. There are some insignificant details that may get lost during the process of compression. Approximate reconstruction may be very good in terms of the compression-ratio but usually it often requires a trade-off between the visual quality and the computation complexity (i.e. speed) [8].

C. *Terms associated with data compression*

Compressor or Encoder: It is the program that compresses the raw data in the input stream and creates an output stream with compressed (low redundancy data) [9]
Decompressor or Decoder: It is the program that converts the compressed data into the original data. [8] [9]
Compression Ratio: It is defined as the ratio between the compressed file and the original file [6].
Compression Ratio=compressed file size / original file size.
Compression Factor: It is defined as the ratio between the original file and the compressed file and is the inverse of the Compression Ratio [5].

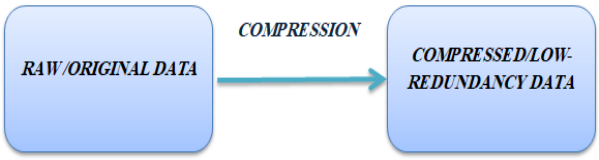

Fig. 2. Compressor or Encoder

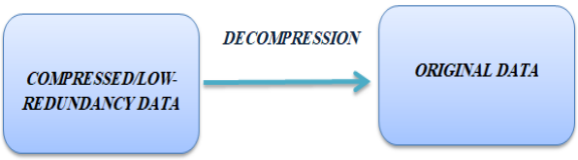

Fig. 3. Decompressor or Decoder

D. *Lossless Compression Algorithms*

In this section, we will give a short review and explanation for each one of the lossless compression methods that can be used on any text files. Compression algorithms have along history, the roots of compression algorithms goes back toearlier twentieth century be used on any text files. Compression algorithms have a long history, the roots of compression algorithms goes back to earlier twentieth century , It is these algorithms: [10] [11]

- Run Length encoding (RLE)
- Huffman Tree
- Lempel – Ziv – Welch (LZW)
- Shannone-fan
- RH (RLE, Huffman)

1) Predictive techniques: ***Run Length Encoding (RLE):*** Run Length Encoding (RLE): is a simple and popular data compression algorithm. It is based on the idea of replacing a long sequence of the same symbol by a shorter sequence. RLE requires only a small amount of hardware and software resources. Run-length encoding is a data compression algorithm that is supported by most bitmap file formats, such as TIFF, BMP. [2] [7]
compression mechanism RLE to compress the text:
(abeecbedcbaeddcbeeabea) : file size is 8 *22=176 bits

- In this algorithm, the number of times the letter is repeated, then the letter itself.
- It must be taken into account that the number of repetitions does not 255, and when it exceeds, the number is divided.
- So the pressure for the previous text is as follows :
- 1A1b2e1c1b1e1d1c1b1a1e2d1c1b2e1a1b1e1a
- Therefore, the file size after compression is 38*8 = 304 bits.
- For Example, the text aaaaaaaaaaattrrrreeeeeeee is 26*8= 208 bits.
- • And when pressed, it will be its size 12a2t4r8e 8 byte any 8*8 =16 bits. [11]

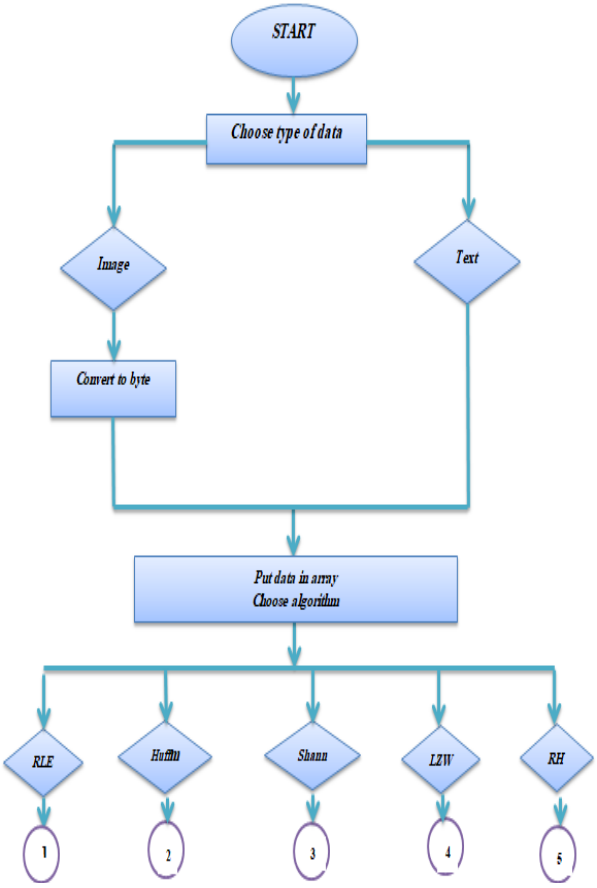

Fig. 4. flowchart algorithm compression

2) Probability techniques: In this section I will show three techniques that uses the probability of each symbol to compress the data. The first one, Huffman coding, was developed by David Huffman in 1951. The idea to the second one, arithmetic coding, came from Claude E. Shannon in 1948 and was further developed by Peter Elias and Norman Abramson in 1963, and algorithm LZW. [10]

3) Huffman coding: Huffman coding is used for lossless data compression. It uses variable length code for encoding a source symbol (such as a character in a file) which is derived based on the estimated probability of occurrence for eachpossible value of the source symbol. In this compression technique, a table is created incorporating the no of occurrences of an individual symbol. This table is known as frequency table and is arranged in a certain order.

Then a tree is generated from that table, in this tree high frequency symbols are assigned codes which have fewer bits, and less frequent symbols are assigned codes with many bits. In this way the code table is [12], [13].

Compression mechanism Huffman:

- to compress the text (abeecbedcbaeddcbeeabea) The file size is 8*22=617 bits.
- Find the number of repetitions of each letter a=4,b=5,c=3,d=3,e=7
- Make sheets with the number of letters, i.e. Make 5 sheets (a contract) and put on each sheet the repetition of the letter it represent

Fig. 5. Compression Huffman

- Select every two cards that have the least frequency and make them daughters of a new node containing their sum. [6] [8]

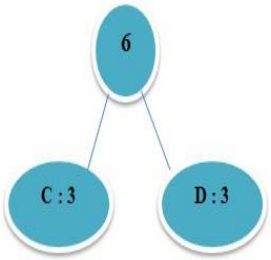

Fig. 6. Compression Huffman

- Apply this process from the bottom up until you get one node, which is the root. [1]

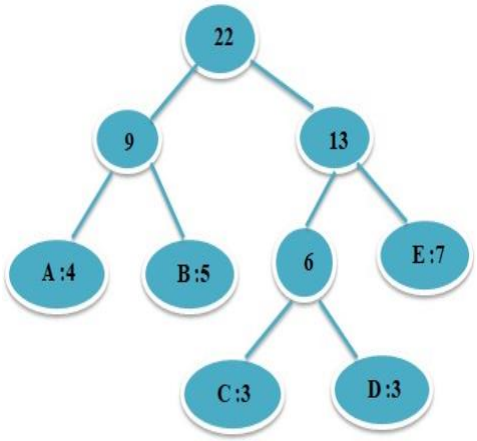

Fig. 7. Compression Huffman

- Number each knot according to its position, if it is to the right of another knot, take the number 1, and if it is to it is to the left of another knot, take the number 0. [4] [7] with these six steps, a Huffman tree has been created, and the file compression and created, and the file compression and decompression processes can be applied. [4]

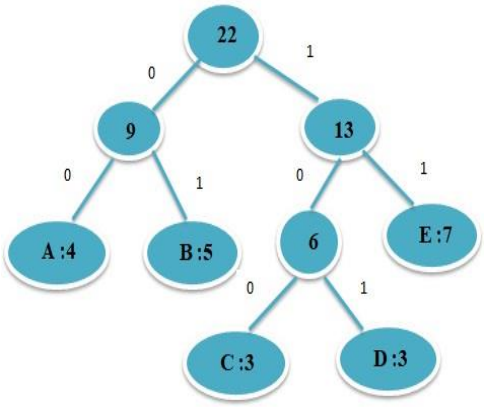

Fig. 8. Compression Huffman

- Then each character in the text is swapped into the new bit as follows :
  Abeecbedcbaeddcbeeabea Becomes (00011101111010100010011101101100111100011101)
  The file size is : 4*2+5*2+3*3+3*3+3*3+7*2= 8+10+9+9+14=50 bit , which is much less than the original file size of 617 bits . [1] [8]

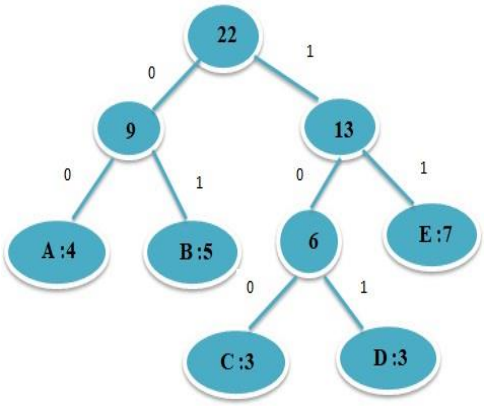

Fig. 9. Compression Huffman

E. Algorithmic Shannon :

This technique is named after Claude Shannon and Robert Fano and is a variable length code for encoding a source symbol. It is a lossless data compression scheme. According to Shannon's source coding theorem, the optimal code length for a symbol is –log b P, where b is the number of symbols used to make output codes and P is the probability of the input symbol. Similar to the Huffman coding, initially a frequency table is generated and then a particular procedure is followed to produce the code table from frequency. [12]

F. LZW Algorithmic

This technique is named after Abraham Lempel, Jacob Zev and Terry Welch. It is dictionary coder or substitution coder, which means a dynamic dictionary is created depending upon the presence of substring chosen from the original file. Then the substring is matched with the Dictionary, if the string is found then a reference of the dictionary is mentioned in the encoded file, if the string is not found then a new dictionary entry is made with a new reference. In all algorithms the en- coded file contains the code table/Dictionary and the encoded text; the encoder matches the codes with the directory (code table/ dictionary) and retrieves the original text iteratively [2].

- LZW Compression Mechanism: To compress babaabaaa text, the file size is 8*9 =72 bits.

BABAABAAA P=A
C=A

ENCODER OUTPUT		STRING TABLE	
output code	representing	codeword	string
66	B	256	BA
65	A	257	AB
256	BA	258	BAA
257	AB	259	ABA
65	A	260	AA

Fig. 10. Compression LZW

- The size of the compressed file is 5*12=60 bits. [4] [10]

G. RH algorithm with RLE optimization algorithm

This algorithm compresses the data twice using the RLE algorithm first and then the Huffman algorithm. [6] [10]

H. RH Algorithm Compression Mechanism

- Data compression using the RLE algorithm.
- Pre-compressed data with Huffman algorithm. [11] [10]

IV. RESULTS

- Applying lossless text compression applying by Visual Studio C# 2015 .NET.

Fig. 11. Implementation of compression algorithms

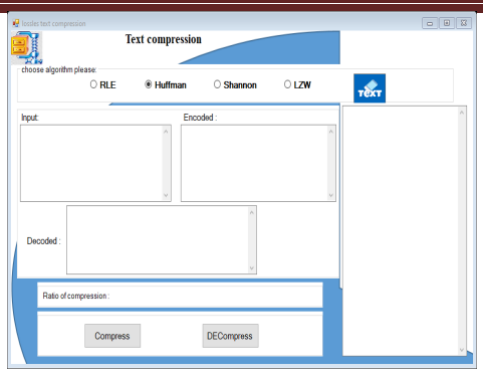

Fig. 12. text compression and decompression

- Comparative of text compression algorithms

The following figure shows the results of the compression ratio for each of the algorithms RLE, Huffman, shannon, LZW,RH. The results showed that the RH algorithm gave the highest compression ratio.

N	N bits	RLE		Huffman		Shannon		LZW		RH	
1	144	272	-89%	63	56%	73	49%	216	-50%	100	31%
2	944	1859	-47%	460	51%	521	45%	1032	-9%	698	26%
3	1680	3312	-97%	866	48%	965	43%	1728	-3%	1290	23%
4	3656	7168	-96%	1923	47%	2094	43%	3408	7%	2848	22%
5	4176	8160	-95%	2191	48%	2392	43%	3792	9%	3245	22%
6	6008	11760	-96%	3138	48%	3443	43%	5028	16%	4658	22%
7	6208	12144	-96%	3241	48%	3556	43%	5172	17%	4811	23%
8	7400	14368	-94%	4234	48%	5027	43%	5916	20%	5731	23%
9	9632	18688	-94%	5027	48%	5509	43%	7128	26%	7451	23%
10	10088	19696	-95%	5433	46%	5659	44%	7692	24%	7818	23%
11	11848	23040	-94%	6210	48%	6823	42%	8472	28%	9194	22%
12	12136	23840	-96%	6601	46%	7210	41%	8796	28%	9640	21%
13	13200	25936	-96%	7202	45%	7943	40%	9420	29%	10506	20%
14	15552	30608	-97%	8510	45%	9391	40%	10740	31%	12408	20%
15	17880	37328	-97%	9895	45%	10968	39%	12288	31%	14344	20%

Fig. 13. Comparative of text compression algorithms

- Comparative of text compression algorithms

The following figure shows that Huffman algorithm gave better data compression results than Shannon algorithm. The

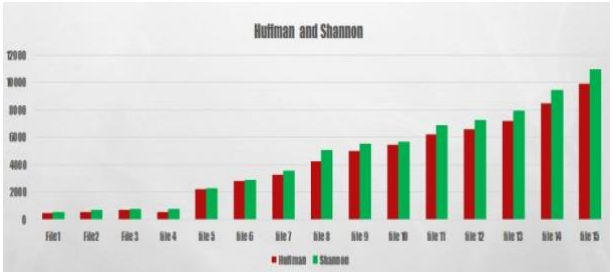

Fig. 14. Comparative Huffman and Shannon

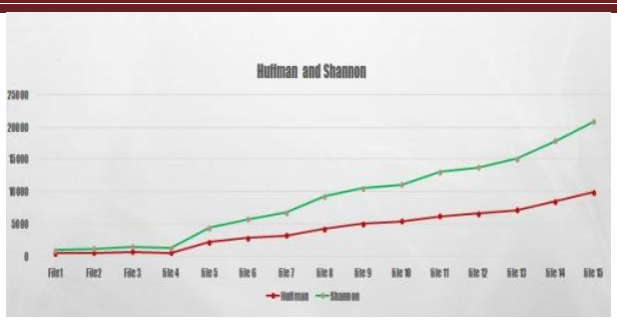

Fig. 15. Comparative Huffman and Shannon

following figure shows the best algorithm for compressing text data is Huffman.

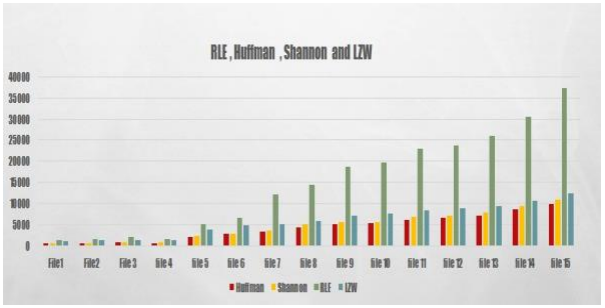

Fig. 16. Comparative Huffman, RLE, Shannon and LZW

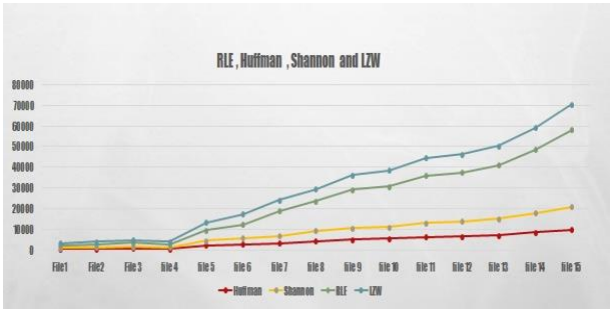

Fig. 17. Comparative Huffman, RLE, Shannon and LZW

The following figure shows that the Huffman algorithm gave the best compression results.

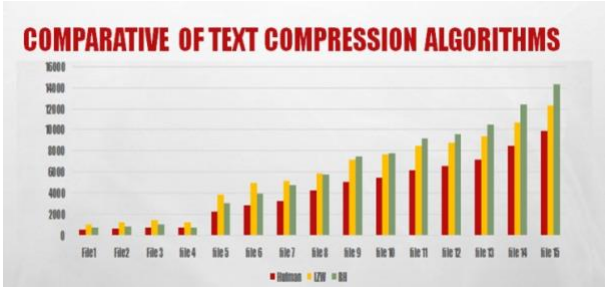

Fig. 18. Comparative Huffman, LZW and RH

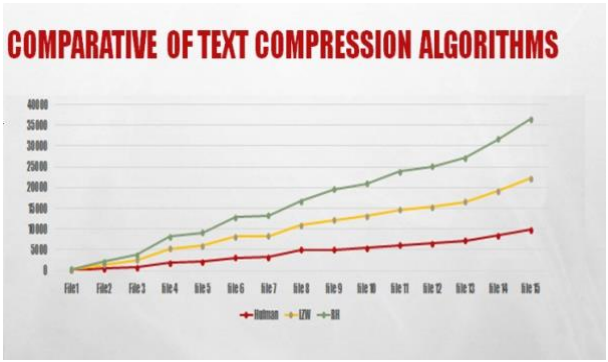

Fig. 19. Comparative Huffman, LZW and RH

A special case of comparing lossless data compression algorithms (Aaaaaaaaaaaaabbbbbbbbbbbbbbbbb-baaaaaaaaaaabbbbbbbbbbbbbbbbbaaaaaaaaaaa)

The following figure shows that in the case of hig repetition, the RH algorithm gave the highest compression ratio for text data.

- Comparative of text compression algorithms

N	N bits	RLE		Huffman		Shannon		LZW		RH	
1	560	80	86%	70	88%	106	81%	216	61%	35	94%

Fig. 20. Comparative of text compression algorithms

Through the results shown, the RH algorithm had the most data compression Ratio. Comparative Huffman, LZW, RLE, Shnnon and RH

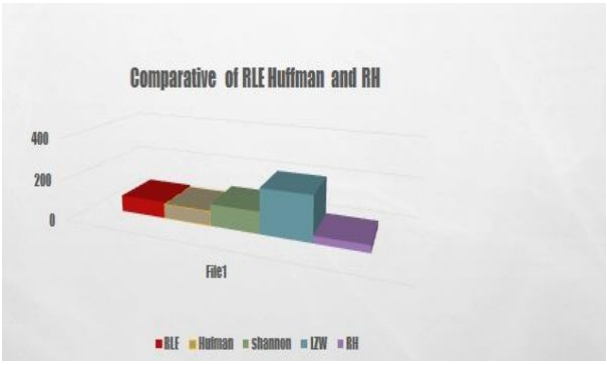

Fig. 21. Comparative Huffman, RLE, Shannon and LZW

V. Conclusion

Data compression technologies have become increasingly important due to the rise in data storage and information transmission. Without compression, many applications would be prohibitively expensive to use, despite advances in bandwidth and storage capabilities. This research survey covers four types of compression, including lossless and lossy compression, as well as the optimization of the RLE algorithm. The survey also explains basic concepts, algorithms, and methods used in data compression and their various applications. Additionally, the survey evaluates the performance of different compression algorithms in compressing text data, with the results showing that Huffman's algorithm achieved the highest compression ratio. Furthermore, the survey found that the improved RH algorithm outperformed the RLE algorithm.

References

[1] Stecuła, Beniamin, Kinga Stecuła, and Adrian Kapczyński. "Compression of Text in Selected Languages—Efficiency, Volume, and Time Comparison." Sensors 22.17 (2022): 6393.

[2] Fitriya, L. Anjar, Tito Waluyo Purboyo, and Anggunmeka Luhur Prasasti. "A review of data compression techniques." International Journal of Applied Engineering Research 12.19 (2017): 8956-8963.

[3] Ayyoub, Belal, and Jamil Al-azzeh. "A comparative analysis of Huffman and LZW methods of color image compression-decompression."

[4] Sharma, Neha, and Usha Batrab. "Evaluation of lossless algorithms for data compression." Top. Intell. Comput. Ind. Des 2 (2020): 40-4.

[5] Gupta, Ruchi, Mukesh Kumar, and Rohit Bathla. "Data compression-lossless and lossy techniques." International Journal of Application or Innovation in Engineering and Management 5.7 (2016): 120-125.

[6] Hosseini, Mohammad. "A survey of data compression algorithms and their applications." Network Systems Laboratory, School of Computing Science, Simon Fraser University, BC, Canada (2012).

[7] Sidhu, Amandeep Singh, and Er Meenakshi Garg. "An Advanced Text Encryption Compression System Based on ASCII Values Arithmetic Encoding to Improve Data Security." International Journal of Computer Science and Mobile Computing 3.10 (2014).

[8] Karmakar, Jayashree, et al. "Sparse representation based compressive video encryption using hyper-chaos and DNA coding." Digital Signal Processing 117 (2021): 103143.

[9] Mubi, Adamu Garba, and P. B. Zirra. "Performance Evaluation of Forward Difference Scheme on Huffman Algorithm to Compress and Decompress Data." International Journal of Computer Science and Information Security 12.7 (2014): 31.

[10] Gupta, Anshul, and Sumit Nigam. "A Review on Different Types of Lossless Data Compression Techniques." International Journal of Scientific Research in Computer Science, Engineering and Information Technology 7.1 (2021): 50-56.

[11] Vijayvargiya, Gaurav, Sanjay Silakari, and Rajeev Pandey. "A sur- vey: various techniques of image compression." arXiv preprint arXiv:1311.6877 (2013).

[12] Singh, Akhand Pratap, Anjali Potnis, and Abhineet Kumar. "A Review on Latest Techniques of Image Compression." International Research Journal of Engineering and Technology (IRJET) 3.7 (2016): 2395-0056.

[13]