

Exploring Text Data Compression: A Comparative Study of Adaptive Huffman and LZW Approaches

Doaa J. Kadhim^{1*}, Mahmood F. Mosleh² and Faeza A. Abed²

¹Electrical Engineering Technical College, Middle Technical University, Baghdad, Iraq

²Institute of Technology, Middle Technical University, Baghdad, Iraq

Abstract. Data compression is a critical procedure in computer science that aims to minimize the size of data files while maintaining their vital information. It is extensively utilized in Numerous applications, including communication, data storage, and multimedia transmission. In this work, we investigated the results of compressing four different text files with Lempel-Ziv-Welch compression techniques and Adaptive Huffman coding. The experiment used four text files: Arabic and English paragraphs and repeated Arabic and English characters. We measured Bit-rate, Compression Time, and Decompression Time to evaluate the algorithms' performance. With a compression time of around 22 μ sec/char, the results demonstrated that the Adaptive Huffman algorithm was quicker at compressing Arabic and English text files. On the other hand, the decompression time for the LZW technique was 23 μ sec/char, which was quicker. The Adaptive Huffman algorithm outperforms the LZW with a Bit rate of about 1.25 bits per character for Arabic text. The English-formatted encoded text's Bit rate in Adaptive Huffman was 4.495 bit/char, lower than LZW's Bit rates of 3.363 and 6.824 bit/char for the Arabic and English texts, respectively. When it came to texts containing Arabic and English characters, the LZW algorithm outperformed the Adaptive Huffman algorithm in terms of decompression time and Bit-rate. The decompression time for a text with Arabic letters was 6 μ sec/char, and the Bit-rate was 0.717 bits/char. These values were lower compared to the text with English letters, which had a decompression time of 16 μ sec/char and a Bit-rate of 1.694 bit/char. For compression time Adaptive Huffman outperform LZW and achieve 15 μ sec/char, and 47 μ sec/char for both Arabic and English letters files respectively.

1 Introduction

As a result of the development of communication systems, which caused the emergence of a large amount of data [1]. A vital procedure in computer science called data compression tries to shrink the size of data files while maintaining their essential information [2]. Data compression aims to render files smaller in order to ensure they use less storage space and bandwidth (BW) when transmitting through data communication channels, it was necessary to find a way to reduce the volume of data, thus the urgent need for data compression [3]. Data compression lowers the number of errors occurring during data transmission by decreasing the amount of data required to be transmitted shared through channels which may be prone to errors [4]. This process enables efficient data management and promotes faster and more effective communication between devices and systems [5, 6]. Many compression algorithms compress different files, either with lossy or lossless, and they differ in their advantages [7, 8]. Lossy compression looks for "redundant" elements and removes them from existence permanently, so it is not suitable to be used in text compression. This strategy is helpful situations when it is possible to erase data from specific ranges that human brain cannot recognize, such as video, audio, and images [9]. While lossless data compression techniques similarly shrink files' sizes while maintaining their information without any losing or changing thus, when data is uncompressed, there is no data loss and the integrity of the information is preserved [10-12]. In simple terms, the reconstructed data is a perfect replica of the original data. It applied in text, satellite imagery, medical imaging, and other fields [13]. When compressing a text file, it is necessary to use a hassle-free technique that will not result in data loss. This method restores the text file to its initial condition [14]. Text compression algorithms convert a string of characters into a new string that retains the original information but is smaller [15]. Among the most famous algorithms that compress texts without loss are Huffman, Lempel-Ziv Welch (LZW), Run Length Encoding (RLE), and Shannon- Fano Technique [16]. Sharma *et al.* 2017 [17] analysed of multiple lossless compression methods examined the effectiveness and performance of each one by using time and space complexity. The Huffman algorithm is the best for text file compression and takes an optimal Execution Time (ET), based on the results and analysis. For image file formats, the LZW compression technique works well, whereas the Shannon-Fano technique works well for hybrid text and audio file formats. Muhammad *et al.* 2018 [18] presented a parallel processing technique based on open multiprocessing to reduce ET for compression techniques. Their proposed method offers a highly efficient and

* Corresponding author: doaa1jk@gmail.com

reliable method for compression and decompression processes. The findings showed that Arithmetic Coding outperformed Lempel-Zev-77 and K-RLE, achieving a Compression Ratio (CR) of 46% as opposed to 44% and 37%, respectively. Gopinath *et al.* 2020 [19] discussed the analysis of different lossless compression techniques by evaluating the measurement parameters. The results showed that the CR for LZW technique, which equals to 4.33 and 76.9% with respect to the Saving Ratio (SR), has a better CR than other compression strategies for identical text. However, the Delta technique performs better than LZW, RLE, and Huffman for calculating ET. While Ignatoski *et al.* 2020 [20] focused on text compression in different languages. The results showed that the SR of the Arithmetic technique by 72.54%, 72%, 72.87%, 72.42%, 73.43%, 69.94%, 69.62%, and 71.47% while the LZW technique achieved 75.79%, 74.51%, 77.33%, 75.14%, 76.84%, 74.63%, 73.25%, and 76.17% for the English, Croatian, French, Finnish, Italian, Hungarian, Czech, and German languages, respectively. A summary of the lossless Tamil compression method for Tamil text files and the extraction of the testing findings were delivered by Vijayalakshmi *et al.* in 2021 [21]. The proposed Tamil technique evaluated against several data compression methods, such as statistical-based coding, dictionary-based coding, and compression utility applications [22]. The amount of storage space and time needed to compress and decompress a file are two ways the effectiveness of compression algorithms is determined. The average SR for the Huffman, LZW, and ZIP approaches are 40.23%, 52.5%, and 68.27%, respectively. The final results have been compared to the average SR of the Tamil approach, which offers the best outcome with improvements over Huffman, LZW, and ZIP of 31.89%, 19.58%, and 3.81%, respectively. With respect to compression and decompression times, the Tamil approach performs best at 0.22 seconds and 0.07 seconds after Huffman coding, LZW, and ZIP respectively.

In this work, we compared the Adaptive Huffman (AH) with the LZW and examined the outcomes based on compression time (CT), decompression time (DT), and Bit-rate (Br). We demonstrated the behaviour and efficiency of each of them in the text compression process.

2 Adopted Compression Methods

In this study, it has been used two compression methods namely AH, and LZW to compress different texts.

2.1 Adaptive Huffman Coding Technique (AH)

David Huffman developed this method in 1950 [23], and it is a lossless compression that shortens the original file's size based on symbol repetition. Each symbol is assigned a variable length under the use of variable-length encoding. The shorter codes assigned to symbols that appear more frequently, while symbols that appear less frequently assigned longer codes [24]. There are two types of Huffman compression techniques: static and adaptive. The static Huffman algorithm efficiently compresses and decompresses text by creating a shared binary tree based on character frequencies. AH creates two trees by developing the tree and then calculating frequencies [25].

2.2 Lempel-Zev Welch Technique (LZW)

Abraham Lempel, Jacob Ziv, and Terry Welch developed the LZW compression algorithm in 1977 [26]. It is an effective and popular lossless compression technique, particularly for files containing repeated data. It's easy to use and doesn't require any prior understanding of statistics, data types, or file structures. LZW employs a dynamic dictionary to substitute character strings with codes [27]. The LZW compression technique gives a single code for recurrent patterns [28]. The technique starts by listing every ASCII symbol in a table and giving each symbol a code between 0 and 255. If there isn't already an entry in the table when reading one from the input file, it is added [29]. In contrast, decompression is achieved by taking each code from the compression file and translating it through the table to find the original symbol [30]. As the number of repetitive sequences increases, the efficiency of LZW also increases.

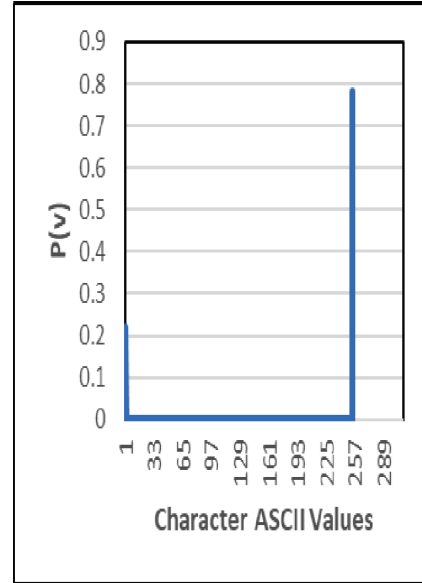
3 Methodology

3.1 Text compression

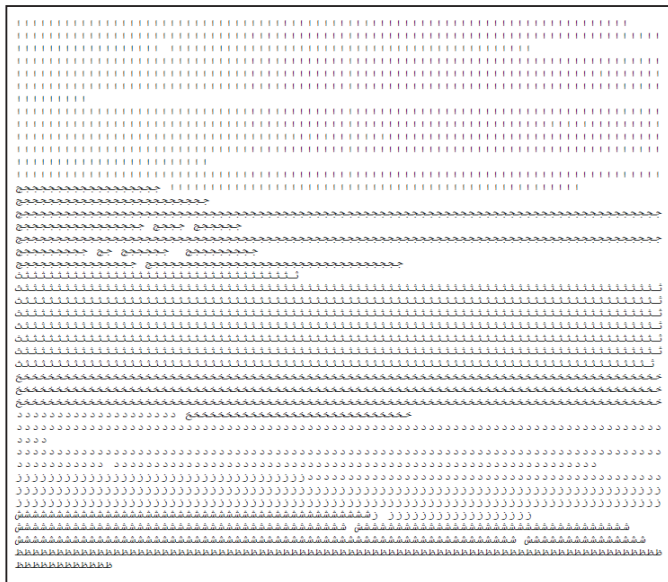
In this study, the performance of AH and LZW compression algorithms using different text files is compared and analysed. It is an Arabic text with a size of 3484 characters, an English text with a size of 3102 characters, a text made up of different Arabic letters with a size of 3146 characters, and another text made up of English letters with a size of 2210 characters. Figure (1) shows the texts used in this study with their histograms of the ASCII values for the symbols of each text. A rate (RT) of (10%, 20%, ..., 100%) taken from each text file and these RT of text data compressed for each of those percentages.



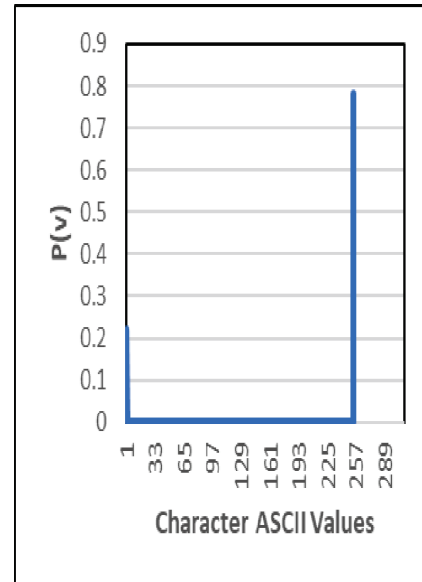
(e)



(f)



(g)



(h)

Fig.1. The data in this study (a) Arabic Text contain 3484 characters (b) Histogram for ASCII, (c) English Text contain 3102 characters, (d) Histogram for ASCII of English Text, (e) Text with Arabic character contain 3146 characters, (f) Histogram for ASCII of Arabic Text characters, (g) Text with English character contain 3146 characters, and (h) Histogram for ASCII of English Text characters.

The relationship between the ASCII values involved in the compression process for each input text and the ASCII values for each output text after the decompression process. Where can be noted that a linear relationship between input data and output data, that mean the retrieval process was complete and correct and there were no defects. While for using LZW in a sequential manner demonstrated the relation between input and output data also noted a linear relationship when using four texts with RT=100%.

4 Results and Discussions

In this study, the AH and LZW compression technique applied to compress four texts. The CT, DT, and BR were computed.

4.1 Arabic text

In order to evaluate the performance of each AH and LZW algorithms in term of Arabic text. Figure (2) shows the relationship of CT, DT, and Br of each RT. The Arabic text displayed in Figure (2) contains a total of 3484 characters. Changes is noticed in the factors for both LZW and Huffman. When RT is 20%, it means that the number of characters reaches 700 characters, the CT intersects and started from number of characters. But, when the number of characters' increases to 3484 characters, the CT for Huffman and LZW were 22 μ sec/char, and 49 μ sec/char, respectively. DT for Huffman is more than the CT and reaches 72 μ sec/char. While, in case of LZW the opposite happens i.e., as the time required to decompression is less than the time of compression and arrives at 23 μ sec/char for the entire text, which is less than in Huffman. The Br increases when the text size (increased RT), so the whole text reaches 1.25 bit/char and 3.363 bit/char for both Adaptive Huffman and LZW, respectively.

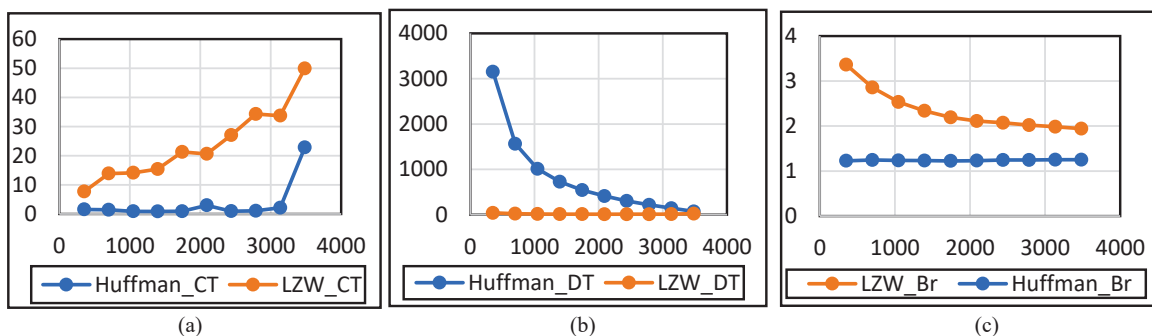


Fig.2 The relationship between AH, and LZW in term of (a) CT, (b) DT, and (C) BR by using Arabic text.

From Figure 4, it is clear that AH is better than the LZW in terms of the CT and Br. But in terms of DT, the LZW is better as it requires less time.

4.2 English text

In Figure (3) English text used consists of 3102 characters. When comparing the performance of both AH and LZW algorithms, the CT converges when the number of characters is less than 1000. Also, the total CT reaches 58 μ sec/char in the case of AH, while in LZW it is 110 μ sec/char. The DT in LZW is 23 μ sec/char less than in AH 370 μ sec/char. As for the Br, it is high and reaches 6.824 bit/char in the case of LZW and 4.495 bit/char in the AH case. Which indicates a low CR for the text used.

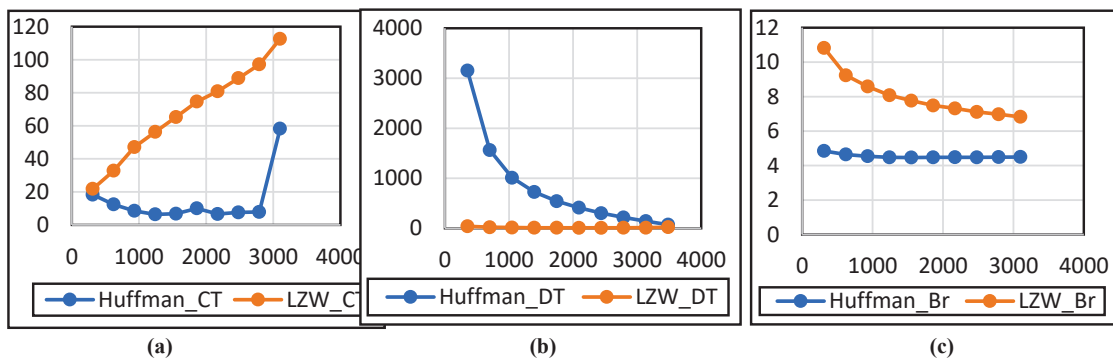


Fig.3 The relationship between AH, and LZW in term of (a) CT, (b) DT, and (C) BR using English text.

4.3 Arabic Letters

Figure (4) shows Arabic letters which consists of 3146 characters, this give that the time required for compression is short, and the AH level is 15 $\mu\text{sec}/\text{char}$, less than that noted when using LZW method 30 $\mu\text{sec}/\text{char}$. In this case, the time required to decompression the entire text is 41 $\mu\text{sec}/\text{char}$ for AH and 6 $\mu\text{sec}/\text{char}$ for LZW. According to Br, it intersects for both algorithms at RT of 20%, which is about 626 characters. When the number of characters increases, the BR decreases, and it is less than comparing with previous state and reaches 1.080 bit/char and 0.717 bit/char for both AH, and LZW, respectively.

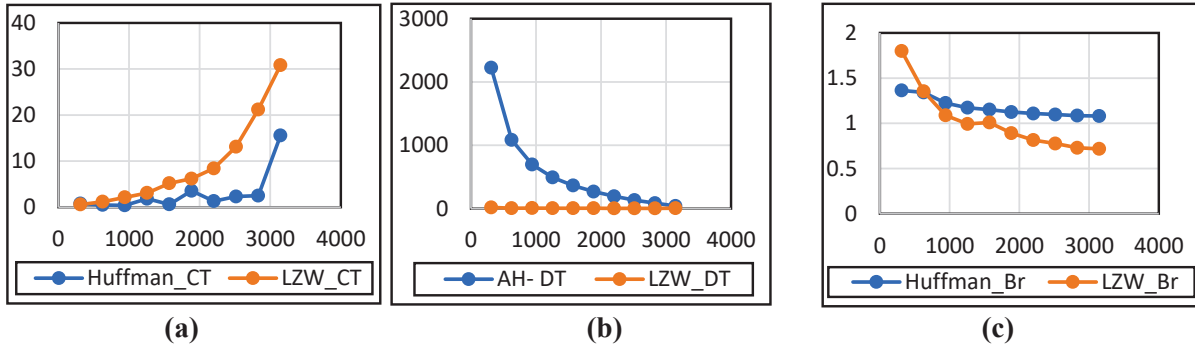


Fig.4. The relationship between AH, and LZW in term of (a) CT, (b) DT, and (c) BR by using Arabic letters.

4.4 English Letters

The CT is decreasing in Figure (5) for English characters consisting of 2210. In the case of AH, it is 47 $\mu\text{sec}/\text{char}$, while for LZW reaches 74 $\mu\text{sec}/\text{char}$. In this case, the time required to decompress the entire text is 120 $\mu\text{sec}/\text{char}$ for AH and 16 $\mu\text{sec}/\text{char}$ for LZW. The Br intersects at RT= 30% for both algorithms when the number of characters is less than 1000 characters. The Br is lower than that in the texts and arrives at 3.145 bit/char for AH and 1.694 bit/char for LZW.

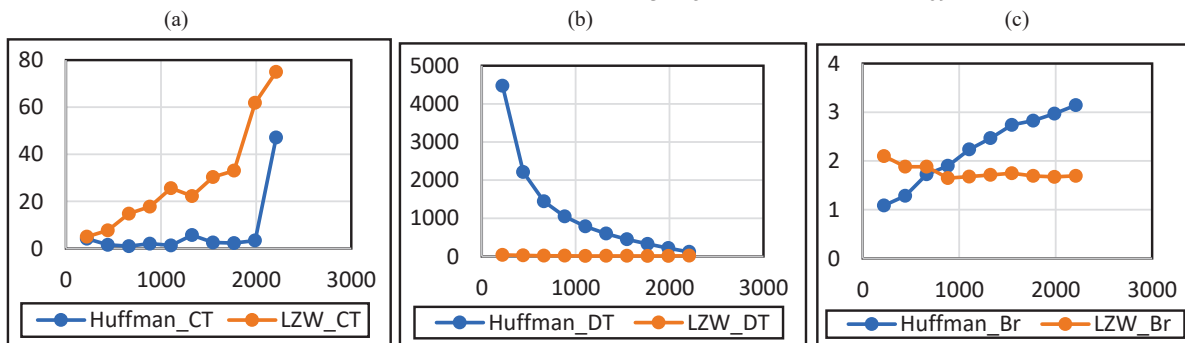


Fig.5. The relationship between AH, and LZW in term of (a) CT, (b) DT, and (c) BR using English letters.

5 Conclusions

Arabic text has low CT in AH and reached 22 $\mu\text{sec}/\text{char}$. Also, Br is small comparing with LZW which equal to 1.251 bit/char. While the DT is low for LZW algorithm reach 23 $\mu\text{sec}/\text{char}$. For English text, the CT in AH it less than for LZW and reach 58 $\mu\text{sec}/\text{char}$. Also, Br is small comparing with LZW which equal to 4.495 bit/char. While DT is low for LZW algorithm reach 23 $\mu\text{sec}/\text{char}$. For Arabic letters, the CT it is low in AH and reach 15 $\mu\text{sec}/\text{char}$. While Br and DT for LZW is less than for AH equal to 0.717 bit/char, and 6 $\mu\text{sec}/\text{char}$, respectively.

For English letters, the CT it is low in AH and reach 47 $\mu\text{sec}/\text{char}$. While Br for LZW is less than for AH equal to 1.694 bit/char. The DT is low for LZW algorithm reached 16 $\mu\text{sec}/\text{char}$.

The results show that the AH appear better performance than LZW in term of CT and Br. So, we conclude the CT for AH in both Arabic and English text it is less than for LZW. While the opposite happens in DT. The LZW required high Br than AH. For both Arabic and English letters the CT for AH is less than for LZW. While both DT and Br there are low for LZW compare with AH.

References

1. R. T. Anto and R. Ramachandran, "A Compression System for Unicode Files Using an Enhanced Lzw Method," *Pertanika Journal of Science & Technology*, vol. 28, no. 4, 2020.
2. D. F. Djusdek, H. Studiawan, and T. Ahmad, "Adaptive image compression using adaptive Huffman and LZW," in *2016 International Conference on Information & Communication Technology and Systems (ICTS)*, 2016: IEEE, pp. 101-106.
3. Ö. ALTAŞ and K. TÛTÛNCÛ, "Implementation and Comparison of Text Compression Algorithms in Image Steganography."
4. S. A. Abdulzahra, A. K. M. Al-Qurabat, and A. K. Idrees, "Data reduction based on compression technique for big data in IoT," in *2020 international conference on emerging smart computing and informatics (ESCI)*, 2020: IEEE, pp. 103-108.
5. M. S. Modabbes, "Study on impact of changing the nature of data on the overall file compression ratio," *Association of Arab Universities Journal of Engineering Sciences*, vol. 29, no. 1, pp. 56-63, 2022.
6. B. Stecuła, K. Stecuła, and A. Kapczyński, "Compression of Text in Selected Languages—Efficiency, Volume, and Time Comparison," *Sensors*, vol. 22, no. 17, p. 6393, 2022.
7. G. Drost and N. G. Bourbakis, "A hybrid system for real-time lossless image compression," *Microprocessors and Microsystems*, vol. 25, no. 1, pp. 19-31, 2001.
8. A. Habib and M. S. Rahman, "Balancing decoding speed and memory usage for Huffman codes using quaternary tree," in *Applied Informatics*, 2017, vol. 4, no. 1: SpringerOpen, pp. 1-15.
9. G. Navarro, "A self-index on block trees," in *International Symposium on String Processing and Information Retrieval*, 2017: Springer, pp. 278-289.
10. U. Jayasankar, V. Thirumal, and D. Ponnurangam, "A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications," *Journal of King Saud University-Computer and Information Sciences*, vol. 33, no. 2, pp. 119-140, 2021.
11. D. A. Lelewer and D. S. Hirschberg, "Data compression," *ACM Computing Surveys (CSUR)*, vol. 19, no. 3, pp. 261-296, 1987.
12. D. R.-J. G.-J. Rydning, J. Reinsel, and J. Gantz, "The digitization of the world from edge to core," Framingham: International Data Corporation, vol. 16, pp. 1-28, 2018.
13. D. Salomon, *A concise introduction to data compression*. Springer Science & Business Media, 2007.
14. A. P. Sridhar and P. Lakshmi, "An efficient lossless medical data compression using lzw compression for optimal cloud data storage," *Annals of the Romanian Society for Cell Biology*, vol. 25, no. 6, pp. 17144-17160, 2021.
15. A. Quddus and M. M. Fahmy, "A new compression technique for binary text images," in *Proceedings Second IEEE Symposium on Computer and Communications*, 1997: IEEE, pp. 194-198.
16. Z.-N. Li, M. S. Drew, J. Liu, Z.-N. Li, M. S. Drew, and J. Liu, "MPEG Audio Compression," *Fundamentals of Multimedia*, pp. 505-531, 2021.
17. S. Porwal, Y. Chaudhary, J. Joshi, and M. Jain, "Data compression methodologies for lossless data and comparison between algorithms," *International Journal of Engineering Science and Innovative Technology (IJESIT) Volume*, vol. 2, pp. 142-147, 2013.
18. K. Sharma and K. Gupta, "Lossless data compression techniques and their performance," in *2017 International Conference on Computing, Communication and Automation (ICCCA)*, 2017: IEEE, pp. 256-261.
19. F. S. Mahammad and V. M. Viswanatham, "Performance analysis of data compression algorithms for heterogeneous architecture through parallel approach," *The Journal of Supercomputing*, vol. 76, no. 4, pp. 2275-2288, 2020.
20. A. Gopinath and M. Ravisankar, "Comparison of lossless data compression techniques," in *2020 International Conference on Inventive Computation Technologies (ICICT)*, 2020: IEEE, pp. 628-633.
21. M. Ignatoski, J. Lerga, L. Stanković, and M. Daković, "Comparison of entropy and dictionary based text compression in English, German, French, Italian, Czech, Hungarian, Finnish, and Croatian," *Mathematics*, vol. 8, no. 7, p. 1059, 2020.
22. B. Vijayalakshmi and N. Sasirekha, "Comparative Analysis of Lossless Text Compression Methods with Novel Tamil Compression Technique," *International Journal of Research in Engineering and Science (IJRES)*, vol. 9, no. 7, pp. 38-44, 2021.
23. D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098-1101, 1952.
24. R. Ranjan, "Canonical Huffman coding based image compression using wavelet," *Wireless Personal Communications*, vol. 117, no. 3, pp. 2193-2206, 2021.

25. M. A. Rahman and M. Hamada, "Burrows–wheeler transform based lossless text compression using keys and Huffman coding," *Symmetry*, vol. 12, no. 10, p. 1654, 2020.
26. J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on information theory*, vol. 23, no. 3, pp. 337-343, 1977.
27. Q. Jiancheng, L. Yiqin, and Z. Yu, "Block-Split Array Coding Algorithm for Long-Stream Data Compression," *Journal of Sensors*, vol. 2020, pp. 1-22, 2020.
28. B. Vijayalakshmi and N. Sasirekha, "Lossless Text Compression Technique Based on Static Dictionary for Unicode Tamil Document," *Int. J. Pure Appl. Math*, vol. 118, pp. 85-91, 2018.
29. N. H. Resham, H. K. Abbas, H. J. Mohamad, and A. H. Al-Saleh, "Noise Reduction, Enhancement and Classification for Sonar Images," *Iraqi Journal of Science*, vol. 62, no. 11, pp. 4439-4452, 12/23 2021, doi: 10.24996/ijs.2021.62.11(SI).25.
30. H. K. Abbas, A. H. Al-Saleh, H. J. Mohamad, and A. A. Al-Zuky, "New algorithms to Enhanced Fused Images from Auto-Focus Images," *Baghdad Science Journal*, vol. 18, no. 1, p. 0124, 03/10 2021, doi: 10.21123/bsj.2021.18.1.0124.