KARYA ILHAM PUBLISHING
CONNECTING MINDS, BRIDGING IDEAS, INSPIRING READERS

Journal of Advanced Research in Computing and Applications

Journal homepage:
https://karyailham.com.my/index.php/arca
ISSN: 2462-1927

The Analysis of lossless compression in Huffman Coding and Lempel-Ziv-Welch (LZW)

Puteri Nurul'Ain Adil Md Sabri[1,*], Azizi Abas[1], Fazli Azzali[1]

[1] School of Computing, Universiti Utara Malaysia, Kedah, Malaysia

ARTICLE INFO

Article history:
Received 5 March 2025
Received in revised form 25 April 2025
Accepted 5 May 2025
Available online 3 June 2025

Keywords:
Data compression; Lossless Compression; QR Code; Huffman coding; Lempel-Ziv-Welch (LZW)

ABSTRACT

Two-dimensional barcodes called Quick Response codes (QR codes) are commonly used to store data, such as URLs, contact details, and product details. As a means of information sharing, they are growing in popularity due to their ability to store large amounts of data in a small footprint. However, the increasing demand for data storage necessitates larger QR code sizes, potentially impacting readability and scanning efficiency. This study looks into how to use lossless compression techniques like Huffman Coding and Lempel-Ziv-Welch (LZW) to make QR codes store more information without losing any of their accuracy. According to the tests, LZW had an average compression ratio of 35%, and Huffman coding had 40%. This demonstrated that both approaches could considerably compress QR code sizes. Furthermore, both methods ensured reliability by maintaining 100% data integrity after decoding. The results show that adding lossless compression to QR codes makes them work better, which means they can be used to store more data in smaller spaces. This research provides a foundation for further advancements in QR code optimisation, particularly in multi-layered and multicoloured QR code systems.

1. Introduction

Today, our technology is growing from time to time. The smartphone is a technology that plays a significant role in people's daily lives. According to previous research, it is found that the mobile-activated space maintains steady growth. Nowadays, the use of digital media and communications technology is rapidly growing. Therefore, mobile space allows for various activities, including social networking and online shopping; hence, identification and payment methods are created. Because of that, the storage capacity needed to store the data information is also required to be larger than usual due to storing information virtually, especially for packaging or storing, passenger control on the ticketing system, prescription, and other uses.

Besides cloud storage as data storage capacity, two-dimensional barcodes (2-D barcodes), also known as QR codes, can act as data storage. A QR Code contributes to making several numbers of our tasks easy at home and work. It can be used for various business purposes such as manufacturing,

* *Corresponding author.*
E-mail address: eryzi90@gmail.com

https://doi.org/10.37934/arca.39.1.110

distribution, retail sales, pharmaceuticals, and services [1]. This method can save more space on paper. It means more printed documents can use this style, such as identity cards, passports, driving licenses, or essential records, to reduce paper usage by embedding the information inside the printed object or QR Code. The QR Code is one of the types of barcodes.

Smartphones, mobile devices with cameras (like the new iPod Touch), and other similar devices can scan and read a QR code, which is a two-dimensional barcode. A QR code generator is a tool that effectively encodes text, URLs, or other data into white squares with black geometric forms, as well as multicolour codes and logos. In September 1994, Denso Corporation of Japan created a two-dimensional code known as a QR code. Globally, especially in Japan and Korea, the QR Code is gaining popularity and widespread adoption. The use of QR codes is more prevalent in real-world scenarios. One can quickly decipher the data contained within the QR code.

Two-dimensional barcodes were first used in 1990. Since then, they have played a big role in the fields of privacy and copyright, and they can also store more information than 1D barcodes. QR codes are a type of data visualization that allows for the fast scanning of content via QR code scanners. QR codes have been utilized in various applications, including customer advertising, ticketing systems, website authentication, and business cards [2-5]. They have become increasingly popular as a means of sharing information due to their ability to store large amounts of data in a small space. The data size that may be encoded in a single QR code is the primary limitation of the current QR code. Many studies on QR codes offer customized solutions to circumvent the size restriction [5,6].

QR code has 40 different versions, and each version can encode a larger data size, but in return, it will consume more space on paper. Three factors will affect the colour QR code size as follows, and Table 1 shows a QR code specification.

i. Data type, which can be numeric, alphanumeric, or binary; to send any data, it can be converted to data bits.
ii. Error correction consists of four levels:
    • Low (L), which can recover up to 7% of damaged data.
    • Medium (M), which can recover up to 15% of damaged data.
    • Quartile (Q), which can recover up to 25% of damaged data.
    • High (H), which can recover up to 30% of damaged data.
    The higher the error correction level, the less data can be encoded within the QR code.
iii. Number of layers, adding more layers in the colour QR code helps to increase the data size.

Table 1
QR Code specifications

Symbol size	Minimum Version 20 Maximum Version 40	
Maximum Data Size (version 40)	Data Bits	10,208
	Numeric	3,057
	Alphanumeric	1,852
	Binary	1,273
	Kanji	784
Error correction	Level L	Up to 7%
	Level M	Up to 15%
	Level Q	Up to 25%
	Level H	Up to 30%

2. Problem Statement

The analysis of lossless compression techniques, including Huffman coding and Lempel-Ziv-Welch (LZW), exposes numerous research problems that require further study. Detailed studies that assess these algorithms' performance across a range of data types and applications are desperately needed, even though the existing literature offers a framework for comparison.

There is little comparative research on how well Huffman Coding and LZW perform on various datasets, especially in contemporary data-intensive applications, despite the fact that both have been thoroughly studied separately. Studies that have already been done frequently ignore practical issues like LZW dictionary size restrictions or the effect of variable symbol distribution on Huffman efficiency. This study fills these gaps by presenting a thorough performance analysis, shedding light on their usefulness in real-world situations, and recommending the best options for various data scenarios.

Firstly, studies like those conducted by Ibrahim and Gbolagade [7] are relevant. Fauzan *et al.,* [8] have compared the performance of Huffman and LZW algorithms, focusing on specific data types, such as images and text files, respectively. This indicates a gap in understanding how these algorithms perform across a broader spectrum of data types, including audio and video files. Researchers highlight the need for further implementation of these concepts in various media formats, suggesting that a more extensive comparative analysis could yield insights into the adaptability and efficiency of these algorithms in different contexts.

Secondly, the literature shows an increasing interest in the use of these algorithms in future technologies, like IoT and smart grids [9,10]. Nevertheless, there exists an insufficient number of studies examining the scalability and real-time performance of Huffman and LZW algorithms in these kinds of situations. It is important to understand how to optimize these compression strategies for real-time applications, considering the substantial data generated by IoT devices. This provides a new avenue for future research efforts.

Finally, the current research frequently highlights the effectiveness of these methods regarding compression ratios and speeds. However it frequently ignores their impact of data properties on performance. The impact of compression efficiency on transmission speed but fails to explore how various data types (e.g., structured compared to unstructured) may influence the performance of Huffman and LZW algorithms [11]. This offers a chance for future study to carefully review how data attributes affect the effectiveness of certain compression techniques.

3. Compression

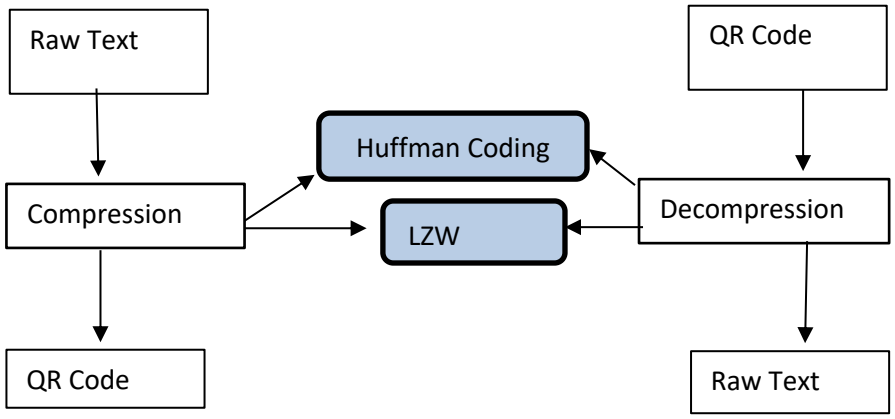
3.1 Text Compression

Data compression reduces the size of data, facilitating more efficient storage and transmission by removing a piece of unnecessary data information. The compression technique becomes essential when users send large files containing many of data through the network, such as an email, without failure [12]. Furthermore, in [13], compression can reduce file size and increase data storage capacity. This technique employs two steps, (1) Converting the text data into binary form and (2) Generating the hash map data from the binary data. As a result, the data compression can exceed more than four megabytes of data in a QR Code compared to four kilobytes.

The proposed model, as illustrated in Figure 1, analyses the input data stream to identify various forms such as text, bits, bytes, numeric, and data meta-alphanumeric for encoding. According to [14], we initiate the compression process by converting the data information into ASCII, which assigns each character a unique number that we can easily convert into their corresponding binary values,

thereby facilitating faster data computation. Next, we can blend the binary data into the ZIP compression algorithm to produce compressed binary data. Compression helps store data more efficiently, reducing transmission costs and resulting in faster transmission times. Next, we divide the compressed binary data into smaller parts for encoding. Finally, we multiplex all generated QR codes using a 2n combination; in this case, n = 5, resulting in 5 multiplexed QR codes. Therefore, it can increase the data capacity compared to the original data capacity of the QR Code, while maintaining the confidentiality of the information.

In simple words, Figure 1 illustrates the original text contains numerous redundant bits, which are eliminated by hardware known as compressors. Thus, the required storage space for the compressed text is reduced. In the last step, the original text is recovered at the decompression stage. Compressing data before it is stored and then decompressing it when it is retrieved is an efficient way to expand the storage capacity of a device [15]. The rate at which data can be transferred via the Internet is constant. Thus, significant improvements in data speed can be realized if data can be compressed effectively whenever possible.



(a) Compression process (b) Decompression process

Fig. 1. Compression Process and Decompression Process.

3.2 Type of Compression

According to [16], the lossy data compression technique will compress image data files, meanwhile, the lossless data compression technique can transmit or store binary files, which are two commonly used types of data compression techniques. Lossless compression methods will ensure the original data is fully recoverable. Huffman Coding and Lempel-Ziv-Welch (LZW) are two prominent lossless algorithms, each with unique approaches and applications. This paper aims to analyse these algorithms' mechanisms, performance, and suitability for different data types. Lossless compression algorithms eliminate redundancy without losing any information. They are essential in scenarios where data integrity is paramount, such as text files, executable programs, and certain image formats (e.g., PNG).

Huffman coding is a lossless data compression algorithm that is frequently used to compress text files [17]. It works by assigning shorter codes to frequently occurring symbols and more extended codes to infrequently occurring symbols. This results in a more efficient representation of the data, reducing the number of bits required to store it. On the other hand, Lempel-Ziv-Welch (LZW) is another lossless data compression algorithm that is commonly used to compress image files. It works by replacing strings of characters with a single code, which is then added to a dictionary. As the compression process progresses, the dictionary grows, allowing more and more strings to be

replaced with codes. This results in a more efficient representation of the data, reducing the number of bits required to store it.

When Huffman coding and LZW are combined to compress QR codes, the resulting codes are able to hold substantially more information than uncompressed codes. Combining the two compression techniques permits a more efficient representation of the data, reducing the number of bits necessary to store it. The compressed data can then be sent back to a QR code, which any QR code reader can scan and read. The comparison between Huffman Coding and LZW is shown in Table 2.

Several studies have shown that using Huffman coding and LZW together can significantly enhance the data storage capacity of QR codes. For instance, a study by [18] found that using the two compression algorithms together can increase the capacity of a QR code by up to 80%. Another study by [19] found that using Huffman coding and LZW together can increase the capacity of a QR code by up to 78%.

Three main parameters are used to evaluate the compression performance: Compression Ratio, Compression Time, and Decompression Time [16,20-22].

i. Compression ratio: The ratio of size of the input text to the size of the compressed text.

   Compression ratio= (C2/C1) *100%

   C1= Size of input text before compression
   C2= Size after text after compression

ii. Compression Time: The total time taken to run the compression algorithm.
iii. Decompression Time: The total time taken to execute the decompression algorithm

Table 2
Comparison table for Huffman coding and LZW algorithms

ASPECT	HUFFMAN CODING	LZW COMPRESSION
Type	Frequency-based, prefix coding	Dictionary-based, adaptive coding
Efficiency	Optimal for static, frequency-known data	Effective for repetitive or streaming data
Memory Requirement	Requires storing frequency table and tree	Requires storing a dynamic dictionary
Encoding Complexity	$O(n\log n)$ due to tree construction	$O(n)$, linear in terms of dictionary operations
Decoding Complexity	$O(n)$	$O(n)$, but involves dictionary rebuilding
Use Cases	Text compression (e.g., ZIP), images (e.g., JPEG)	GIF images, UNIX compress, text-based data storage

3.2.1 Huffman Coding

Developed by David Huffman in 1952, Huffman coding is a variable-length prefix coding algorithm. It builds an optimal binary tree based on character frequency, ensuring that frequently occurring characters have shorter codes. File compression standards like JPEG and MP3 widely use this algorithm. Huffman coding is a complicated and lossless compression technique. Researchers convert the characters in a data file to binary, ensuring the most frequently occurring characters have the shortest codes. No information loss occurs after decoding [23]. To observe the Huffman coding function, compress a text file using the following character frequencies. The next paragraph will

describe the process of Huffman Coding followed by Figure 2 which illustrates a summarization of that process. For every single algorithm or coding, the researcher must know the features or advantages and disadvantages of those items. Therefore, Table 3 shows the advantages and disadvantages of Huffman Coding.

Step 1 is Frequency Calculation. Each symbol in the data that needs to be encoded is determined in this step. Each symbol's frequency is used to create a priority queue, also known as a heap, in which the lowest frequencies are combined first. Step 2 is Tree Construction. Each symbol's frequency in Step 1 is used as the key to create a node. The two nodes with the smallest frequencies are repeatedly combined to create a new node with their combined frequency, which is how the tree is constructed. This continues until there's only one node, the root of the Huffman tree. Step 3 is Code Assignment. In this step, it depends on the tree structure which is a distinct binary code given to each symbol. Each symbol's code is derived from the path that leads to it from the root. One node assign "0" to a left branch and "1" to a right branch, or vice versa. Step 4 is Encoding. While in the encoding process, each symbol in the input data is replaced with its matching Huffman code. Because frequent symbols are represented with shorter codes in this step, the data size is decreased. The last step is Decoding. The data is decoded by locating the matching symbol for each bit sequence by traversing the Huffman tree from the root. The decoder reconstructs the original message by reading the bits in the encoded data and following the tree.

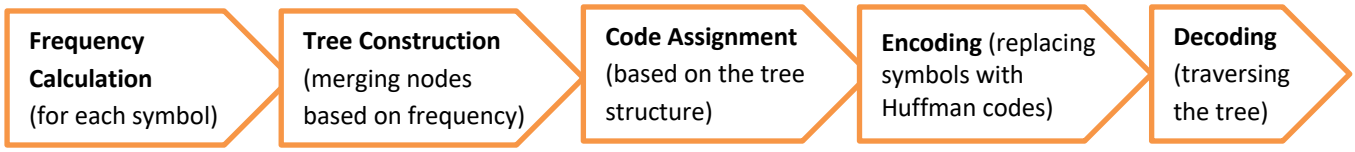


Fig. 2. Process of Huffman Coding

Table 3
Advantages and disadvantages of Huffman coding

Advantages	Disadvantages
Optimality: Provides the most efficient compression for known symbol frequencies.	Requires Frequency Table: Needs a frequency analysis before encoding, which adds overhead.
Simplicity: Easy to implement and understand	Variable-Length Codes: This may complicate decoding compared to fixed-length codes.
No Loss of Data: Perfectly reconstructs the original data.	Not Adaptive: Inefficient for streaming data with dynamic symbol frequencies

3.2.2 Lempel–Ziv–Welch (LZW) Coding

Lempel-Ziv-Welch (LZW) coding is a lossless data compression algorithm that was developed by Abraham Lempel, Jacob Ziv, and Terry Welch in 1977 and involves two steps, such as (i) parsing and (ii) coding. A set of rules divided strings of symbols into variable-length substrings during the parsing phase [15]. The coding phase sequentially encoded each substring into a fixed-length code [24]. It is a dictionary-based compression algorithm that replaces repeated occurrences of data patterns, such as strings of characters, with shorter codes. GIF, TIFF, and PDF commonly use LZW as a compression technique. LZW is an essential data compression technique due to its adaptability and simplicity. It facilitates increasing the internal drive's storage capacity. The next paragraph will describe the process of LZW followed by Figure 3 which illustrates a summarization of that process. For every single algorithm or coding, the researcher must know the features or advantages and disadvantages of those items. Therefore, Table 4 shows the advantages and disadvantages of LZW.

Step 1 is to Initialize the Dictionary. All possible single-character strings are used to initialize the dictionary. For instance, the dictionary will begin with all 256 characters in ASCII encoding. Step 2 is Input Processing. Each symbol is sequentially read by the algorithm as it processes the input data. The longest string that is already in the dictionary is what it searches for. Step 3 is Dictionary Expansion. The longest string is replaced with its matching code after it has been located in the dictionary. The next available code is used to add a new string to the dictionary, which is created by appending the subsequent symbol to the existing string. Step 4 is Encoding. In order to process all of the input data, the algorithm keeps reading it and adding new strings to the dictionary. The last step is Decompression. All single-character strings are used to initialize the dictionary once more in order to decompress. To restore the original data, the dictionary is used to process the encoded data. The decoded dictionary strings are concatenated to reconstruct the decompressed data.

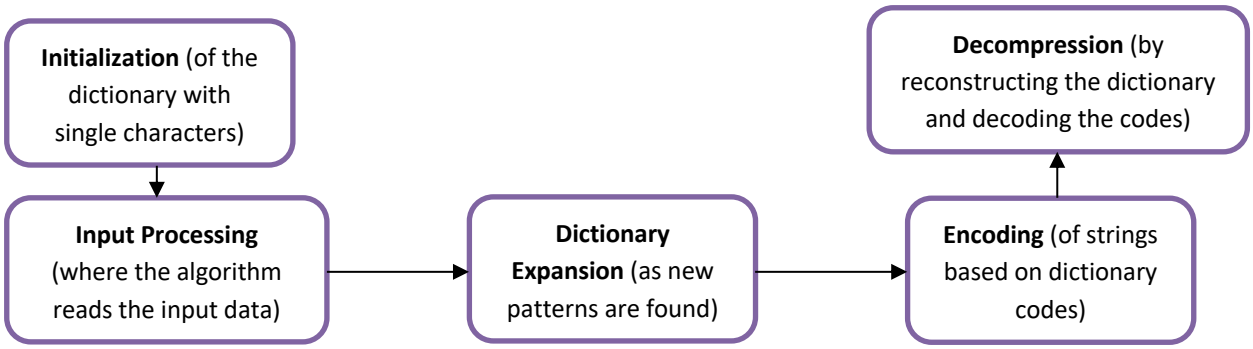


Fig.3. Process of LZW algorithms

Table 4
Advantages and Disadvantages of LZW

Advantages	Disadvantages
Adaptive: Builds the dictionary dynamically, making it effective for various data types.	Dictionary Growth: The dictionary can grow large, especially with diverse data, leading to memory issues.
Efficiency: Works well for repetitive data patterns, such as text or binary files	Less Optimal for Short Files: LZW may not achieve significant compression on small datasets.
No Prior Analysis Needed: Unlike Huffman Coding, it does not require frequency calculation.	Complex Decoding: Reconstructing the dictionary during decoding can be challenging for highly dynamic or complex data.

Based on Table 5 below, the researcher describes that if the size of the file is small then the compression ratio (CR) of LZW Coding is very high compared to the Huffman Coding. Meanwhile, for compression time (CT), the researcher observed that if the size of the file is small, the time taken for both algorithms (LZW Coding and Huffman Coding) is the same. It is not the same if the size of the file is big, the CT of Huffman Coding is higher than LZW Coding. The last parameter of compression is Decompression Time (DT). Researchers conclude that if the size of the file is small, the DT for both algorithms is almost the same, but a bit different result for LZW Coding if the size of the file is large due to it can perform well.

Table 5
Comparison Result Huffman Coding and LZW Coding [16]

File Size	Huffman Coding			LZW Coding		
	Compression Ratio (%)	Compression Time (ms)	Decompression Time (ms)	Compression Ratio (%)	Compression Time (ms)	Decompression Time (ms)
2.2 KB	54.20	0.0	0.0156	105.56	0.0	0.0
15.5 MB	54.203	123.664	115.858	3.604	8.560	29.734
33.2 KB	54.205	0.015	0.231	49.94	0.0156	0.0156
62.1 MB	54.203	1945.389	470.933	1.827	53.198	469.634

4. Results

A QR code of version 40 contains 177 rows and columns of modules and can store up to 7,089 numeric or 4,296 alphabetic characters. A version 40 QR code is approximately 1,734 modules in size. Table 5 shows the size of data text before and after compression by using Huffman Coding and LZW with their parameters which are CR, CT, and DT. The first row shows the Size of the data text before compression in bits followed by the size of the data text after compression in bits. Researchers use four sizes of data text as raw data or input which are 800, 1200, 1600, and 2000 bits. The Huffman Coding performs good compression for 800 and 2000 bits of data text. The 1200 and 1600 data text go to LZW.

Table 6 below shows the results of the time taken to compress and decompress for Huffman Coding and LZW. For Huffman, the average time taken to compress is less than for the LZW. This is because Huffman Coding is less complex than LZW, which means it takes less time to compress the data text. Meanwhile, for the decompression section, the average time taken for LZW is less than for Huffman. This is because the LZW only needs to scan the LZW code through the library, whereas the Huffman needs to read the input bit-by-bit, which is slower. Therefore, by using these two types of techniques, QR codes can hold more data due to their large space savings after compression.

Table 6
Size of bits of data text before and after Huffman Coding and Lempel-Zel-Welch. [25]

Data Type	Size before Compression (bits)	Size after Compression (bits)		Compression Ratio (CR)		Compression Time (CT) - sec		Decompression Time (DT) -sec	
		Huffman Coding	LZW	Huffman Coding	LZW	Huffman Coding	LZW	Huffman Coding	LZW
Text	800	367	504	0.46	0.63	0.178	0.697	0.053	0.055
	1200	567	696	0.47	0.58	0.222	1.730	0.135	0.075
	1600	753	840	0.47	0.53	0.447	1.984	0.106	0.107
	2000	936	960	0.47	0.48	0.446	2.046	0.136	0.171

5. Conclusions

At the end of the research, the utilization of data compression techniques such as Huffman coding and Lempel-Ziv-Welch (LZW) can greatly enhance the storage capacity of QR codes. Through data compression, QR codes may hold a greater amount of information in a smaller area, enhancing their versatility for information sharing.

Compression is a significant technique in multimedia. Reducing the data capacity allows for cheaper and faster transmission and storage. Researchers implement several image and video compression formats, such as JPEG, JPEG 2000, MPEG-2, and MPEG-4. This research highlights the compression ratio, compression time, and de-compression time, among other algorithmic

distinctions. Huffman coding is superior to LZW coding. LZW coding allows for a higher compression ratio than the Huffman algorithm. Huffman encoding requires more time to execute than LZW. In some cases, we can use Huffman coding to achieve a high compression ratio without considering time. Time plays a crucial role in real-time applications and LZW coding.

Acknowledgement
This research was not funded by any grant.

References.

[1] Sharara, Shima, and Sapna Radia. "Quick response (QR) codes for patient information delivery: a digital innovation during the coronavirus pandemic." *Journal of orthodontics* 49, no. 1 (2022): 89-97. https://doi.org/10.1177/14653125211031568

[2] Badawi, B., T. N. M. Aris, N. Mustapha, and N. Manshor. "A Fuzzy Multi-Layer Color Qr Code Decoder Algorithm." *Int. J. Adv. Trends Comput. Sci. Eng* 8 (2019): 131-137. https://doi.org/10.30534/ijatcse/2019/2081.42019

[3] Blasinski, Henryk, Orhan Bulan, and Gaurav Sharma. "Per-colorant-channel color barcodes for mobile applications: An interference cancellation framework." *IEEE Transactions on Image Processing* 22, no. 4 (2012): 1498-1511. https://doi.org/10.1109/TIP.2012.2233483

[4] Toh, Sin Rong, Weihan Goh, and Chai Kiat Yeo. "Data exchange via multiplexed color QR codes on mobile devices." In *2016 Wireless Telecommunications Symposium (WTS)*, pp. 1-6. IEEE, 2016. https://doi.org/10.1109/WTS.2016.7482035

[5] Wang, Sibing, Tao Yang, Jing Li, Bowei Yao, and Yanning Zhang. "Does a QR code must be black and white?." In *2015 international conference on orange technologies (ICOT)*, pp. 161-164. IEEE, 2015. https://doi.org/10.1109/ICOT.2015.7498513

[6] Singh, Ashwdeep, Vikas Verma, "Increasing Storage Capacity of QR Codes", 2017.

[7] Ibrahim, M. B., and K. A. Gbolagade. "A Chinese Remainder Theorem based enhancements of Lempel-Ziv-Welch and Huffman coding image compression." *Asian Journal of Research in Computer Science* 3, no. 3 (2019): 1-9. https://doi.org/10.9734/ajrcos/2019/v3i330096

[8] Fauzan, Mohamad Nurkamal, Muhammad Alif, and Cahyo Prianto. "Comparison of Huffman algorithm and Lempel Ziv Welch algorithm in text file compression." *IT Journal Research and Development* 7, no. 2 (2023): 184-197. https://doi.org/10.25299/itjrd.2023.10437

[9] Wu, Yanxia, Song Xu, Caiping Xi, Pengqiang Nie, Wei Jiang, and Seiji Hashimoto. "A Dynamic and Parallel Two-Stage Lossless Data Compression Method for Smart Grid." *IEEE Access* (2023). https://doi.org/10.1109/ACCESS.2023.3343436

[10] Júnior, Javan A. de O., Edson T. de Camargo, and Marcio Seiji Oyamada. "Data Compression in LoRa Networks: A Compromise between Performance and Energy Consumption." *Journal of Internet Services and Applications* 14, no. 1 (2023): 95-106. https://doi.org/10.5753/jisa.2023.3000

[11] Ma, Shaowen. "Comparison of image compression techniques using Huffman and Lempel-Ziv-Welch algorithms." *Applied and Computational Engineering* 5 (2023): 793-801. https://doi.org/10.54254/2755-2721/5/20230705

[12] Arora, Mukesh, and Atul Kumar Verma. "Increase capacity of QR code using compression technique." In *2018 3rd International Conference and Workshops on Recent Advances and Innovations in Engineering (ICRAIE)*, pp. 1-5. IEEE, 2018. https://doi.org/10.1109/ICRAIE.2018.8710429

[13] Abas, Azizi, Yuhanis Yusof, Roshidi Din, Fazli Azali, and Baharudin Osman. "Increasing data storage of coloured QR code using compress, multiplexing and multilayered technique." *Bulletin of Electrical Engineering and Informatics* 9, no. 6 (2020): 2555-2561. https://doi.org/10.11591/eei.v9i6.2481

[14] Umaria, Mona M., and G. B. Jethava. "Enhancing the data storage capacity in QR code using compression algorithm and achieving security and further data storage capacity improvement using multiplexing." In *2015 International Conference on Computational Intelligence and Communication Networks (CICN)*, pp. 1094-1096. IEEE, 2015. https://doi.org/10.1109/CICN.2015.215

[15] Kasumov, N. K. "The universal coding method in the data compression algorithm." *Automatic Control and Computer Sciences* 44 (2010): 279-286. https://doi.org/10.3103/S0146411610050056

[16] Sharma, Gajendra. "Analysis of Huffman coding and Lempel–Ziv–Welch (LZW) coding as data compression techniques." *International Journal of Scientific Research in Computer Science and Engineering* 8, no. 1 (2020): 37-44.

[17] Ali, Ammar Mohammed, and Alaa Kadhim Farhan. "Enhancement of QR code capacity by encrypted lossless compression technology for verification of secure E-Document." *IEEE Access* 8 (2020): 27448-27458. https://doi.org/10.1109/ACCESS.2020.2971779
[18] Marlapalli, Krishna, Rani SBP Bandlamudi, Rambabu Busi, Vallabaneni Pranav, and B. Madhavrao. "A review on image compression techniques." *Communication Software and Networks: Proceedings of INDIA 2019* (2020): 271-279. https://doi.org/10.1007/978-981-15-5397-4_29
[19] Gupta, Shashank, and Rachit Jain. "An innovative method of Text Steganography." In *2015 Third International Conference on Image Information Processing (ICIIP)*, pp. 60-64. IEEE, 2015. https://doi.org/10.1109/ICIIP.2015.7414741
[20] Altarawneh, Haroon, and Mohammad Altarawneh. "Data compression techniques on text files: A comparison study." *International Journal of Computer Applications* 26, no. 5 (2011): 42-54. https://doi.org/10.5120/3097-4249
[21] Lelewer, Debra A., and Daniel S. Hirschberg. "Data compression." *ACM Computing Surveys (CSUR)* 19, no. 3 (1987): 261-296. https://doi.org/10.1145/45072.45074
[22] Welch, Terry A. "A technique for high-performance data compression." *Computer* 17, no. 06 (1984): 8-19. https://doi.org/10.1109/MC.1984.1659158
[23] Langdon, Glen G. "An introduction to arithmetic coding." *IBM Journal of Research and Development* 28, no. 2 (1984): 135-149. https://doi.org/10.1147/rd.282.0135
[24] Blanco, Roi, and Alvaro Barreiro. "Probabilistic static pruning of inverted files." *ACM Transactions on Information Systems (TOIS)* 28, no. 1 (2010): 1-33. https://doi.org/10.1145/1658377.1658378
[25] Jambek, Asral Bahari, and Nor Alina Khairi. "Performance comparison of Huffman and Lempel-Ziv Welch data compression for wireless sensor node application." *American Journal of Applied Sciences* 11, no. 1 (2014): 119-126. https://doi.org/10.3844/ajassp.2014.119.126