*International Journal of Engineering Science Invention (IJESI)*
*ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726*
*www.ijesi.org || PP. 176-180*

# Comparative Analysis of Lossless Data Compression Techniques

Dr. Nagamani N P[1], Sushruth Nagaraj[2], Sagar J[3], Shashank S P[4], Tushar Kulkarni[5]

[1]*(Department of Information Science and Engineering, JSS Academy of Technical Education, Bengaluru, India)*
[2,3,4,5]*(Department of Information Science and Engineering, JSS Academy of Technical Education, Bengaluru, India)*

***Abstract:*** *With the tremendous growth of data in digital world, the data being generated is increasing drastically. This enormous amount of data tends to be very expensive in terms of data storage and transmission. Data compression involves encoding the information in a file to reduce the redundancy and represent the data in a condensed way. Data compression results in reduced storage space, faster transmission, reduced access time, effective utilization of communication bandwidth and achieves higher streaming speed. Data compression plays a very important role for both text and multimedia content. Many algorithms and techniques have been proposed in the past for performing the compression on different types of data formats to generate different file formats after compression. As numerous data compression techniques have been developed, a need arises to review and compare the techniques and approximately select the suitable algorithm for a particular application. This paper provides an overview on the various existing lossless data compression techniques, and comparative analysis has been carried out to explore and identify the data compression techniques in terms of their characteristics, limitations and applications. The algorithms such as Huffman Coding, Shannon-Fano and Lempel-Ziv-Welch are considered. The parameters like compression ratio, compression factor, space saving, percentage of compression are used to measure the efficiency of algorithms.*

***Keywords -*** *Data Compression, Huffman coding, Lempel-Ziv-Welch, Lossless, Shannon-Fano*

## I.Introduction

Digitization is at its peak in today's world. According to the survey, over 2.5 quintillion bytes of data are generatedevery day [2]. Such rapid development contributes to the exponential growth of data, requiring more storage, which implies an increase in the cost for storage. Data Compression is one of the vital approaches which addressesthis issue. Data compression results in reduced storage space, faster transmission, reduced access time, effective utilization of communication bandwidth and achieves higher streaming speed. Data compression plays a very important role for both text and multimedia content. Different types of files like video, text, audio can be compressed with data compression and the original file can be recovered without any data loss. Such a compressed file can be conveniently shared over the internet with the help of the compression. Since compression reduces the size of the file, it can be uploaded or downloaded much faster.

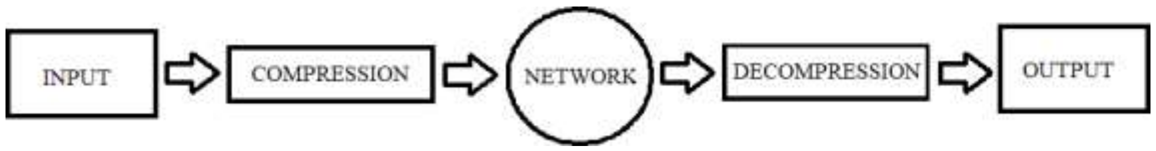

Fig. 1 Compression and Decompression Process

There are two types of Data Compression techniques, viz., Lossless and Lossy techniques as shown in Fig. 2

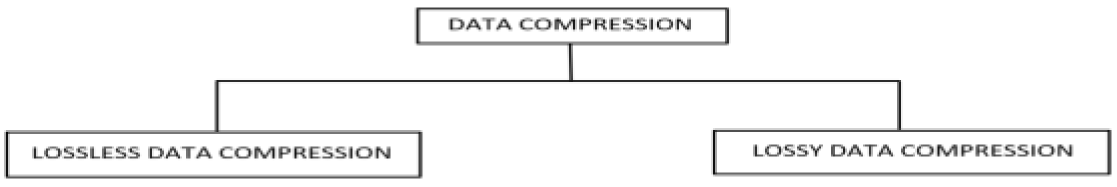

Fig. 2 Classification of Data Compression Techniques

Lossless compression technique compresses the file without any damage or loss of data or reduction in quality. A method of data compression in which some data is lost is known as lossy compression. In this technique, reconstructed data does not match perfectly with the original data. Original bits are retrieved after decompression. Based on the application, data compression technique influences the quality of data [3]. For applications like messaging or browsing the internet, the quality of the reconstructed data is not highly considered. Whereas in text compression, even a single character modification is not tolerable since the entire meaning may change. In applications like text, law forensics, military imagery, medical imaging, satellite imaging, etc. where loss of information is undesirable, lossless compression techniques are employed. In some applications, lossy compression techniques are preferred where approximation of the original data is acceptable. It leads to higher compression for lossy compression techniques when compared to lossless compression techniques. The process of lossy compression searches for 'redundant' pixels and discards this information permanently. Lossy data compression is used for MP3, JPEG, and MPEG, etc.

Lossless techniques can be categorized as Entropy and Dictionary based compression [1], as shown in Fig. 3.

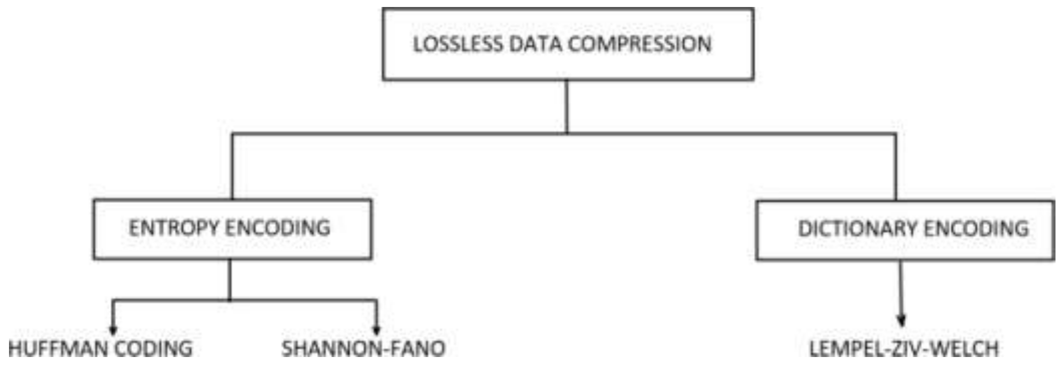

Fig. 3 Classification of Lossless Data Compression Techniques

Entropy compression is a type of lossless data compression technique. The frequency of most repeating data is first found out, and the repeating sequences are encoded[5]. Frequently occurring patterns are represented withfew bits, and more bits are used to represent rarely occurring patterns to compress digital content. Entropy encoding is the backbone of Huffman coding, and Shannon-Fano coding. Huffman coding is a type of entropy coding that encodes symbols by using several bits that are inversely proportional to the symbols' likelihood. The length of the data is decreased by replacing the small variable length code word in the place of the most frequently occurring word. The data is compressed in the Shannon-Fano algorithm by encoding the text data with the help of the probabilities of the occurrence of the symbols.

In dictionary-based algorithms, the strings of random length symbols are encoded as single tokens [3]. The encoder searches for the exact pair of strings in the dictionary that match. If this match is found, it replaces it withthe dictionary pair. Dictionary based algorithms are useful in places where the original data has more repeated patterns. If the input sequence has a pattern, then it is coded with an index to the dictionary. Otherwise, it is codedwith a less efficient approach. Patterns can be categorized as frequently and infrequently occurring patterns. Efficiency of this method will be more if shorter code words are assigned for frequently occurring patterns. Dictionaries are of two types: static and dynamic. With the availability of prior knowledge of the source output, astatic dictionary will be used, else a dynamic dictionary is used. Lempel–Ziv (LZ) is the commonly used dictionarybased method in lossless file compression, and it is widely used due to its adaptability to many file formats. The two versions of LZ viz. LZ77 and LZ78 were developed by Lempel and Ziv in 1977 and 1978. An enhanced version of these two which is named as Lempel–Ziv-Welch (LZW) was developed by Terry Welch in 1984. UNIXcompress, GIF images, PNG images and other file formats use LZW coding whereas LZ77 is used in Gzip and ZIP.

## II. Lossless Data Compression Algorithms

### 2.1 Huffman Coding

David Huffman discovered the Huffman coding algorithm in 1952[4]. Huffman coding is a type of optimal prefix code and it is popularly used in lossless data compression[6]. The method employed in Huffman encoding is a binary code tree generation. Due to this, each symbol's probability of occurrence results in the length of its code. Basically it assigns variable length codes to input characters based on the frequency of occurrence. The output consists of a variable length code table for coding a source symbol. This technique is uniquely decodable and it consists of two steps viz., constructing Huffman tree from the input sequence and tree traversal to assign codes to characters. Due to its simpler implementation and faster compression, Huffman coding is still popular. The various versions of Huffman coding includes minimum variance Huffman code, non-binary Huffman code, canonical Huffman code, adaptive Huffman code, length-limited Huffman code, etc. Compression methods like Deflate, JPEG, MP3, etc. use Huffman code. Huffman coding is the popular coding technique used for effectively compressing the data in almost all file formats.
Algorithm[7]:

1. Construct a frequency table that is sorted in the descending order.
2. Build a binary tree by carrying out the iterations until the completion of a complete binary tree:
(a). Merge the last two items that have the minimum frequencies in the frequency table toform a new combined item with a sum frequency of the two.
(b). Insert the combined item and update the frequency table.
3. Derive the Huffman tree by starting at the root, and tracing down to every other leaf. Mark '0' for the left branchand '1' for the right branch.
4. Generate the Huffman code by collecting all the 0s and 1s for each path from the root to a leaf and assigning a0-1 codeword for each of the symbols.

### 2.2. Shannon-Fano

Claude Shannon and Robert Fano implemented this technique for data compression in 1949. This process involves a binary tree that is generated showing the probabilities of each symbol that occurs. Ordering of the symbols is done as per their occurrences. This means that the symbols at the top of the tree appear most and the symbols at the bottom are least likely to occur. This algorithm also uses the probability of each symbol's occurrence to construct a code in which each codeword can be of different lengths. The codes for symbols with low probabilities are assigned more bits, and the codewords of various lengths can be uniquely decoded. Shannon-Fano is similar to Huffman coding but the only difference is that it uses the top-down approach whereas Huffman algorithm uses the bottom-up approach on the text. When the probabilities are very much closer to 2 power inverses, this technique is more successful. The drawback of this method is that it does not provide any guarantee that an optimal code will be generated.

Algorithm[7]:
1. Develop a frequency or probability table.
2. Sort the table according to frequency where the most frequent one is at the top.
3. Divide the table into two halves with similar frequency counts.
4. Assign the upper half of the list a value of '0' and the lower half a value of '1'.
5. Recursively apply the step of division and assignment to the two halves, subdividing groups and adding bits tothe codewords until each symbol has become a corresponding leaf on the tree.

### 2.3 Lempel-Ziv-Welch

Lempel-Ziv-Welch is a dictionary based compression technique. A set of possible words of a language makes up a dictionary, and it is stored in a structure that is similar to a table that uses the indexes of the entries to represent larger and repeating dictionary words. The Lempel-Zev-Welch algorithm or simply LZW is one of the algorithms in which a dictionary is used to store and index the previously seen string patterns. In the compression process, the index values are used instead of repeating the string patterns. There is no need to transfer the dictionary with the encoded message for decompressing since it is generated dynamically in the process of compression. Since the same dictionary is generated dynamically in the decompression process, this algorithm is known as an adaptive compression algorithm. This algorithm reads a sequence of symbols and performs the grouping of the symbols into string and then converts it into code, so that the code takes less space than the actual string. The main idea behind this technique is that, as the number of long repetitive sequences increases, the efficiency of the algorithm also increases. This technique is used in PDF, GIF and TIFF format. UNIX file compression utility also uses this technique.

Algorithm[8]:

1. Initialize the dictionary so that it holds all the strings of length one.
2. Find the longest string W in the dictionary that matches with the current input.
3. Emit the dictionary index for W to the output and remove W from the input.
4. Add W followed by the next symbol in the input to the dictionary.
5. Go to step 2

## III. Performance Parameters

Compression Ratio**:** It is the ratio of the uncompressed file size to the compressed file size[1]. It shows the level of compression achieved.

$$\text{Compression Ratio} = \frac{Number\ o*\ bytes\ in\ t0e\ uncompressed\ *ile}{Number\ o*\ bytes\ in\ t0e\ compressed\ *ile}$$

Compression Factor**:** The compression factor is the inverse of compression ratio and, it gives the ratio between compressed and the uncompressed file size.

$$\text{Compression Factor} = \frac{Number\ o*\ bytes\ in\ t0e\ compressed\ *ile}{Number\ o*\ bytes\ in\ t0e\ uncompressed\ *ile}$$

Space Saving: Space-saving is known as the size reduction compared to the uncompressed size[1]. It is the reduction in file size relative to the uncompressed size.

$$\text{Space Savings} = 1 - \frac{Number\ of\ bytes\ in\ the\ compressed\ file}{Number\ of\ bytes\ in\ the\ uncompressed\ file}$$

Percentage of Compression: It gives the percentage of reduction in the memory size from the original filesize[3].

$$\text{Percentage of Compression} = \frac{Bytes\ be*ore\ compression - Bytes\ a*ter\ compression}{Bytes\ be*ore\ compression} * 100$$

## IV. Results And Discussions

This section discusses the results and shows the comparative analysis of various algorithms in terms of performance parameters. The performance of the various data compression algorithms can be evaluated in terms of compression ratio, compression factor, space savings and percentage of compression. Table 1. and Fig. 4 depicts the file size before and after compression. Table 2. and Fig. 5 illustrates the performance parameters of various compression techniques. The Shannon-Fano technique shows better efficiency than the other lossless data compression techniques when the file size is less than 50 kb.

Table. 1 Comparison of Various Lossless Data Compression Techniques Based on File Size

| Compression Technique | Uncompressed File Size (kb) | Compressed File Size (kb) |
|---|---|---|
| Huffman | 46.3 | 25.6 |
| Shannon-Fano | 46.3 | 23.8 |
| Lempel-Ziv-Welch | 46.3 | 24.7 |

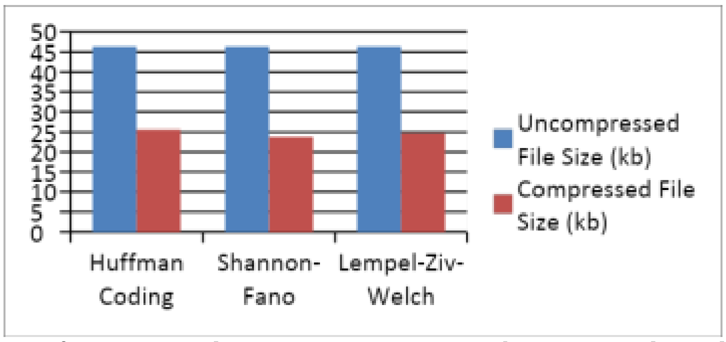

Fig. 4 Comparison of Various Lossless Data Compression Techniques Based on File Size

Table. 2 Performance parameter Comparison of Various Lossless Data Compression Techniques

| Compression Technique | Huffman | Shannon-Fano | Lempel-Ziv-Welch |
|---|---|---|---|
| Compression Ratio | 1.80859375 | 1.945378151 | 1.874493927 |
| Compression Factor | 0.552915767 | 0.514038877 | 0.533477322 |
| Space Saving | 0.447084233 | 0.485961123 | 0.466522678 |
| Percentage of Compression | 44.70842333 | 48.59611231 | 46.65226782 |

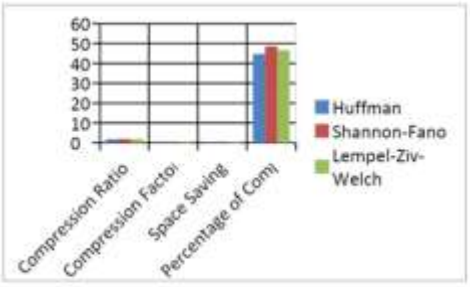

Fig. 5 Performance parameter Comparison of Various Lossless Data Compression Techniques

## V. Conclusion

Data compression techniques play a crucial role in handling large amounts of data generated in the digital world. Various data compression techniques have been proposed in the past to compress various forms of data like audio, images, text, video and so on. This paper discusses different techniques for compressing data and also gives an overview of the various algorithms for lossless data compression. Different methods of data compression, mainly entropy-based and dictionary-based methods are discussed. The Shannon-Fano technique shows better efficiencythan the other lossless data compression techniques when the file size is less than 50 kb. Data compression techniques are reviewed and the performance comparison is carried out and also their suitability to various applications is discussed.

## References

[1]. Gupta, A., & Nigam, S. (2021). A Review on Different Types of Lossless Data Compression Techniques, International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 7(1),50-56

[2]. Barman, R., Deshpande, S., Kulkarni, N., Agarwal, S., & Badade, S. (2021). A review on lossless data compression techniques. Int J Sci Res Eng Trends, 7(1), 2395-566X.

[3]. Saʻūd, J. A. M. (1994). Journal of King Saud University: Computer and Information Sciences. King Saud University.

[4]. Vijayalakshmi, B., & Sasirekha, N. (2021). Comparative Analysis of Lossless Text Compression Methods with Novel Tamil Compression Technique. vol, 9, 38-44.

[5]. Pooja Raundale. (2019), Comparative Study of Data Compression Techniques, International Journal of Computer Applications, 178(28), International Journal of Computer Applications

[6]. Fitriya, L. A., Purboyo, T. W., & Prasasti, A. L. (2017). A review of data compression techniques. International Journal of Applied Engineering Research, 12(19), 8956-8963.

[7]. Data Compression, University of London. (2004), United Kingdom.

[8]. Ziv, J.; Lempel, A. (1978). "Compression of individual sequences via variable-rate coding" (PDF). IEEE Transactions on Information Theory. 24 (5): 530. CiteSeerX 10.1.1.14.2892. doi:10.1109/TIT.1978.1055934.