A Review Of Data Compression Algorithms

Christan Wilbert
UG, School of Computer Science and Engineering,
Vellore Institute of Technology,
Chennai, India
christan.wilbert2021@vitstudent.ac.in

Alen Paul Alapatt
UG, School of Computer Science and Engineering,
Vellore Institute of Technology,
Chennai, India
alenpaul.alapatt2021@vitstudent.ac.in

Sri Sai Ravi Charan Siddabathuni
UG, School of Computer Science and Engineering,
Vellore Institute of Technology,
Chennai, India
sri.sairavicharan2021@vitsudent.ac.in

Dr Suguna M
Assistant Professor, School of Computer Science and Engineering,
Vellore Institute of Technology,
Chennai, India
suguna.m@vit.ac.in

Dr Om Kumar C.U
Assistant Professor, School of Computer Science and Engineering,
Vellore Institute of Technology,
Chennai, India
omkumar.cu@vit.ac.in

***Abstract*— This Research paper is a review on various compression algorithms that are used for the purpose of data compression. A variety of algorithms are used for data compression and the choice of algorithms often depends on the type of data that is being compressed. This paper will be concentrating on data compression techniques that can be used in high volume data storage, data transmission and other applications where the rate of compression, rate of retrieval of data, compression factor, loss of quality and processability of compressed data are significant factors in determining the effectiveness of the algorithm.**

Keywords*— *Compression, storage, Transformation, Arithmetic encoding, quantization, algorithm, lossy and lossless

I. INTRODUCTION

Modern computing systems depend on data compression algorithms to effectively store, transmit, and manipulate huge amounts of data. Data can be reduced in size without significantly losing information by using compression methods to remove unnecessary information. Data compression's major objective is to lower the size of data while retaining accuracy and high quality. Many types of data, such as text, photos, audio, and video, can be compressed using a variety of techniques.

In this paper, we will examine the many data compression algorithms available and the methods these algorithms employ to obtain the best possible compression. The essential ideas underlying data compression, such as entropy coding, prediction, dictionary-based compression, and transform coding, will be made clear. Also, we will look at the trade-offs between various compression algorithms, such as compression ratio, speed, and whether to use lossless or lossy compression.

The practical uses of compression algorithms in a variety of domains, such as computer networks, data storage, multimedia systems, and scientific data analysis, will also be a focus of this study. We'll go through how crucial it is to choose the best compression algorithm depending on the particular requirements of the application and the type of data being compressed.

Ultimately, the goal of this work is to give a comprehensive view of compression algorithms, their uses, and trade-offs. The paper will be of interest to researchers and practitioners working in the fields of data compression, data storage, multimedia systems, computer networks, and scientific data analysis.

II. RESEARCH PAPER REVIEW

A. Image Compression Algorithms

Image compression algorithms are widely used in various applications. Mostly a huge amount of data of image are required to be stored for long time in various fields for instance in the field of medicine, data regarding symptoms and tests of lots of people needs to be stored for longer time so that it can be processed to get useful information that can help in more accurate and faster detection of diseases. This requires an efficient image compression technique that has a high compression factor and better processability of data in the compressed state.

[1] The variation of accuracy of various CNN models was compared with, when MRI scan data fed is compressed using the compression technique ‘Lossy JPEG 2000’ and when it is not compressed. The lossy compression technique reduces the accuracy of the models by less than 2 % (ResNet50), about 6%(VGG16) with a good compression ratio.

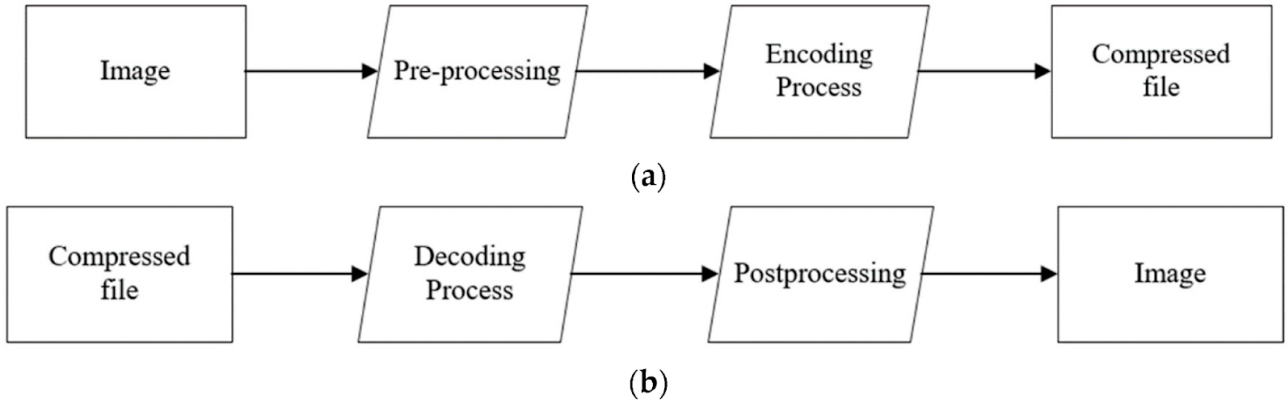
This enables more efficient working of the computation model as it is able to process more information in the compressed form and the cost of

storage of data is also reduced considerably as the compression ratio is high.

In the case mentioned above we have used a lossy compression algorithm. This kind of algorithm is a trade-off between high compression ratio and the quality of the decompressed image.[4] We have lossless compression algorithms but we can only expect a modest compression rate (much lower than the compression rate of lossy compression). The data retrieved from the compressed image is identical to the image compressed with all details preserved. A few lossy image compressions are Discrete Cosine Transform (DCT), Discrete Wavelet Transform (DWT) and Block Truncation Coding (BTC).

The techniques used in lossless image compression are Huffman Coding and Run Length Encoding (RLE).

Image compression and decompression has to be done using minimal computation resources and time with optimum quality. Convolutional Neural Network (CNN) helps in attaining all these goals. CNN is a network architecture used for deep learning algorithms and is particularly used in image processing as it recognises patterns within images which helps in more efficient compression.



B. Video Compression Algorithms

[2] Calculations show that uncompressed video produces a large amount of data, which increases the bandwidth requirements. Fortunately, digital video contains a great amount of redundancy, which facilitates compression. The higher the compression ratio the lower the size and the lower the quality. There are many standards for video compression and MPEG standards are the most widely used ones. MPEG stands for Moving Pictures Coding Export Group. It describes a whole family of standards- MPEG-1, MPEG-2, MPEG-4.

The MPEG compression algorithms compress data in 5 steps

The reduction of resolution comes first, this is followed by a motion compensation in order to reduce temporal redundancy. The next steps are the Discrete Cosine Transformation (DCT) and a quantization, this reduces the spatial redundancy. The last step is entropy coding using Run Length Encoding and Huffman coding algorithm.

Other commonly used video compression algorithms include:

- AVC/H.264: It uses inter-frame prediction, intra-frame prediction, variable block sizes, and multiple reference frames to compress the video.
- HEVC/H.264: It improves upon H.264 by using larger block sizes, improved motion vector prediction, and better intra-frame prediction.

There are more other modified versions of these models.

C. Audio Compression Algorithm

Audio compression allows us to store and transmit audio files more efficiently, while also improving compatibility and adjusting dynamic range for better listening experience. Its common uses include music production, broadcasting, digital audio recordings and live sound reinforcement.

The categorization of digital audio compression techniques can be done based on Predictive and Perceptual encoding methods.

Predictive encoding involves encoding the variance between samples rather than encoding every individual sample value. It is normally used for compression of speech. Several standards are GSM (13 kbps), G.729 (8 kbps), and G.723.3 (6.4 or 5.3

kbps). Perceptual encoding is done to compress CD-quality audio which needs a minimum of 1.411 Mbps that cannot be sent over the internet without compression. MP3 (MPEG audio layer 3) uses this technique.

[9] IEEE 1857.2 is also known as the HEVC (High Efficiency Video Coding standard) or H.265 which is a lossless audio compression algorithm.

A new entropy algorithm has been introduced and combined with LPC (Linear Predictive Coding) to create this standard. It was designed to provide higher compression efficiency than its predecessor, H.264/ AVC (Advanced Video Coding). It achieves this by using advanced coding techniques such as larger block sizes, more efficient prediction algorithms, and better motion compensation.

In both the encoding and decoding process, the audio samples are first processed by the predictor where the correlations in the audio samples are reduced which is a loss of information. After which the signals are flattened by the pre-processor. Then the Entropy Coding compensates the lossy part by encoding the error residue of the Predictor block. In this way, the combination of LPC and Entropy Coding performs lossless compression.

In comparison to other lossless codecs like MPEG-4 ALS (MPEG-4 Audio Lossless Coding) and FLAC (Free Audio Lossless Codec), the tool can provide higher compression efficiency, flexible scalability, robustness, and low complexity.

The limitations include high computational complexity - difficult to implement the standard on low -power devices and platforms, limited compatibility - difficult to integrate with existing video systems and workflow and higher bit rate - encoding process involves the generation of multiple layers of video data that can increase storage and bandwidth requirements.

Compression Algorithms

D. Arithmetic Encoding

Arithmetic coding is a common algorithm used in both lossless and lossy data compression algorithms (mostly lossless). It operates by converting a message into a single large number in a specific range, typically between 0 and 1, representing the possibility of a message. This number is then encoded into binary form and transmitted. It is an entropy encoding technique, in which frequently seen symbols are encoded with fewer bits rarely seen as symbols.

[3] Arithmetic coding compresses the message by representing it as a fraction in the range [0,1]. Encoding process involves dividing the range into sub-intervals corresponding to the probabilities of the symbols in the message. The symbols are then mapped to sub-intervals and are assigned a probability range. Then, a fraction within the range is chosen to represent the message. The fraction encoded to the binary form is then transmitted.

As the message becomes longer, the intervals needed to represent it becomes smaller, hence the number of symbols needed to specify the intervals grows. Therefore, this algorithm is particularly useful for relatively small and skewed alphabets. It is also very useful for compressing messages that have a non-uniform distribution of symbols, as larger fractions of the encoding space can be assigned to frequently occurring symbols.

The decoding process involves reversing the process by calculating the intervals where the encoded fractions fall. The symbols are mapped with the sub-intervals and the message is reconstructed

E. Run-Length Encoding

This is a lossless data compression algorithm that works by replacing sequences of repeated values in a message with a count of repetitions and the value being repeated. This is particularly useful for compressing data that contains long runs of repeated values, like in text documents and images.

[5] The algorithm scans the message and identifies sequences of repeated values. The algorithm then replaces the sequence with two values: the number of continued occurrences of the values and the value itself. For example, a sequence of seven consecutive ones would be replaced by (7,1).

The coded message is usually smaller than the real message if there are long sequences of repeated values. If there are no repetitions, the coded message might be longer than the original, as the number of occurrences is also added as a prefix for each value.

The compression is reversed during decoding as the encoded message is scanned by the decoder which reads the count and value of each pair and outputs the specified number of the values.

RLE is simple, fast and easy to implement, making it suitable for real-time compression and decompression of data.

F. Lempel-Ziv-Welch (Lzw)

It is a lossless data compression algorithm that works by replacing repeated sequences of symbols with shorter codes. LZW compression works best when applied on monochrome images and text files that contain repetitive text/pattern.

The algorithm constructs a dictionary of frequently occurring sequences of symbols. It initially contains all possible single symbols of the message. The algorithm reads the input message and tries to find the longest sequence that is already in the dictionary. Upon discovery, the algorithm replaces it with a code that represents the sequence. The new sequence and the code is added to the dictionary. [7] If the dictionary overflows, it is reinitialized and a bit is added to each one of the code words. The codes are typically binary values that increase in length as the length grows.

During encoding, the message is scanned and repeated sequences of symbols are replaced with their corresponding codes from the dictionary.

During decoding, the compression is reversed by reading the dictionary to reconstruct the original message. The decoder reads the codes from the encoded message, looks up the corresponding sequence in the dictionary, and outputs the original.

LZW achieves high compression ratios with minimal loss of quality. Significant processing power is required to build and maintain the dictionary, and the size of the dictionary becomes large for long messages, which may affect compression efficiency.

G. Transform Coding

Transform coding is a lossy data compression technique that transforms a message from its original form to a different representation that is more compact. The transformation process may involve several mathematical operations, such as Fourier transform, Discrete Cosine transform (DCT), or wavelet transform. These algorithms work by applying a mathematical transformation to the input message that generates a set of transformed coefficients.

[11] DCT is popular due to the fact that it performs a good data compaction, as it focuses the information content in a relatively few transform coefficients. It forms periodic, symmetric sequences from a finite length sequence such that the original sequence can be recovered uniquely. This contains the real part of DFT (discrete fourier transform).

Discrete Wavelength transform (DWT) can be defined as a ‘small wave’ that has its energy centered in time. It provides a means to analyze transient, non-stationary or time varying phenomena. The wavelet transform is executed by a discrete set of the wavelet scales and translation obeying a number of defined rules. Recently, JPEG committee released a new image coding standard, JPEG-2000, which is based upon DWT.

Quantization decreases the number of bits required to store coefficient values by decreasing its precision (eg. rounding). Its aim is to decrease most of the less significant high frequency coefficients to zero. The Quantized coefficients are then encoded and transmitted.

The decoding process involves reversing the compression ratios while maintaining a reasonable level of quality. Transform coding is commonly used in audio and video compression algorithms like MPEG and JPEG, where transformation is typically DCT.

The choice of transformation and quantization tables can critically affect the quality of the reconstructed message. Different quantization tables help control the balance between compression ratio and the level of distortion developed in the reconstructed message.

Optimal quantization tables can be determined by analyzing the statistical properties of the input data and the characteristics of the human visual and auditory system.

H. Burrows-Wheeler Transform

This block encoding compression algorithm reorganizes the data input into a more compressible form in order to function. This is done by taking all the cyclic permutations of a string and sorting it. Then we select the last element in each permutation in the order in which each permutation of the string was sorted. Selection of the last letter or symbol ensures that the inversion process can yield the original string and also ensure maximum repetition of character. Increased repetition of characters ensures higher compressibility. After that, a variable-length encoding procedure, such as Huffman coding, is used to compress the altered data.[8] Recently research has been on BWS and combinations of various algorithms have been tried and tested and are found to be

generating the most optimum BWT (generation of BWT with minimum number of iterations or runs) for a large string. The improvement of compression ratio when Hamming code is applied on BWT is significantly high keeping in mind the lossless nature of the compression. The improvised BWT [8] also reduces the computational time and resources for generation of BWT. Thus, making the overall compression rate better.

I. Block Encoding Compression

A form of data compression technology known as block encoding compresses each block of data separately as it works with fixed-size data blocks. In contrast, other kinds of compression algorithms, such dictionary compression, work with data sequences of varying lengths.

J. Huffman Encoding

Huffman encoding is a lossless compression technique that can give 20% to 90% compression factor. The most frequent symbols in data are assigned a binary code of least length such that there is no common prefix in the binary codes generated enabling lossless recovery of data on decompression. This technique is widely used in combination with various other compression algorithms. [10] Huffman encoding along with weight prediction is used to attain reversible data hiding. Research has been on various techniques to implement this. One of the ways in which this task is achieved is by segregating the image into three different parts that are not overlapping, white, black and mixed. Using Huffman white and black blocks are compressed and decompressed in a loss-less way and weight prediction done with the help of picture correlations help in predicting the mixed block.

K. Algorithms In Cloud

In cloud computing and cloud storage services large quantities of data must be stored, transferred, and processed in the cloud, and compression algorithms are frequently used to shrink the size of this data to improve its storage, transfer, and processing efficiency.[6]. Overall space efficiency of cloud storage can be increased up to 90% by using efficient compression techniques. This is useful for efficient storage of sensor data when you have a large number of IOT sensors. Some of the most popular compression methods in cloud computing are listed below: Gzip: In cloud computing, Gzip is a popular compression technique. It is a lossless compression method that can compress data by up to 60%. Gzip is frequently used to compress web material, including JavaScript, CSS, and HTML files. Brotli: Launched in 2015, Brotli is a more recent compression method. It is a lossless compression method that can outperform Gzip in terms of compression ratio. Brotli is often used in compression of HTML, CSS and javascript files.

LZ77 and LZ78 are two well-known lossless encoding techniques that are frequently applied in cloud computing. Both of these algorithms are dictionary-based, which means they compress data by using a pre-defined lexicon of previously observed patterns. The compression of images and videos frequently uses these methods.

Deflate: Another popular lossless compression method for cloud computing is deflate. It combines LZ77 and Huffman coding and is frequently used to reduce the size of online content like HTML, CSS, and JavaScript files.

L. Lossy Compression Algorithm For Hyperspectral Image Systems

[12] The HyperLCA is a transform-based lossy image compression algorithm. The technique selects the most dissimilar pixels and preserves them which is the most significant difference amongst the other state-of-the-art lossy compression approaches in which the most different pixels are lost. Thus, this technique is preferred for applications involving anomaly detection, target detection, tracking or classification. Popular lossy compressions for Hyperspectral Image compression include JPEG2000, CCSDS123.0, and FastLZ. JPEG2000 is a widely used standard for lossy compression of images and offers high compression ratios with good image quality. CCSDS123.0 is another popular standard that offers both lossy and lossless compression options. FastLZ is a lightweight algorithm that is fast and simple to implement, making it suitable for resource-constrained environments.

The algorithm first divides the image into blocks of pixels, each of which is compressed independently. This process consists of three primary stages: a spectral transform, a preprocessing stage, and an entropy coding stage. During the HyperLCA spectral

transform, orthogonal projection techniques are utilized to sequentially select the most dissimilar pixels from the hyperspectral data set. These selected pixels are then used to project the hyperspectral image, resulting in a compressed and decorrelated version of the data. Following the HyperLCA transform, the output data is adapted for more efficient entropy coding using the HyperLCA preprocessing stage. Finally, the entropy coding stage utilizes a Golomb-Rice coding strategy to encode the extracted vectors.

The evaluation metrics used were SNR (signal to noise ratio) - high implies good average compression performance, Maximum Single Error(MaxSE)- low implies most diff pixels are being preserved in the process, Spectral Angle (SA) - low implies low spectral distortions.

The HyperLCA in comparison to the Karhunen-Loeve Transform (KLT) and Principal Component Analysis (PCA) approach provides better compression ratio at a reasonable computation cost for hyperspectral remote sensing applications. The KLT and PCA approaches are linear transformation techniques where PCA focuses on finding the directions of maximum variance, while KLT finds the optimal basis functions that minimize the reconstruction error. Also the HyperLCA compressor achieves higher compression ratio (less than 2 bpppb) than CCSDS123.0 compressor at a good rate-distortion relation.

M. Medical Imaging Ascii Character Encoding

[13] The ASCII character encoding technique is a lossless ECG data compression technique. ECG data is to be stored for longer duration so the amount of ECG data keeps increasing and also the real time transmission speed of these signals is to be reduced. Hence, compression of biomedical signals is done.

For the compression process, the extracted ECG values are grouped into arrays of 8 values after which sign bit generation is done in order to preserve the negative values. Then the string of sign bit is converted into decimal format further converted into its ASCII equivalent which is stored in another array. The negative numbers and multiplied with -1 so as to get the ASCII equivalent which is possible only for positive numbers. the fractional number are multiplied by 1000 after which if it exceeds 255 then floor function is performed. Then grouping - Forward, reverse (or) No grouping is performed where two or three integers are concatenated. The additional information of the array is represented by the ascii characters - s(sign bit), k(location of forward), reverse (z) and rounded quotient (f). The improved version of the technique eliminates the necessity of 'r', 's', 'q' and 'u' that are necessary for reconstruction of the compressed signal.

The ASCII-encoded ECG is a human readable format which is easier for professionals in the medical field to interpret as they may not be familiar with binary-encoded format. The technique includes a parity bit, which can be used to detect errors in the data transmission process thus helping in accuracy and reliability of ECG data.

It was designed to provide a better compression ratio over its predecessor which required 12 ASCII characters to store an array of 8 ECG voltage values but the improved version requires just 8 ASCII characters for the same, improving the compression ratio by 65% without affecting the quality of the signal. The technique achieves a compression ratio (CR) of 11.25 and percent root mean squared difference (PRD) of 0.0206 on average for all 12 ECG signals.

N. Bandelet - Set Partitioning In Hierarchical Trees

[14] SPIHT is an image based lossy compression algorithm that is typically suited for compressing digital images with a high degree of spatial redundancy. It achieves high compression ratio by exploiting the redundancies in the image and removing them during compression with minimal distortion.

The image is first divided into blocks of pixels, and then each block is recursively subdivided into smaller blocks until each block contains only a single pixel forming a tree structure called a wavelet tree. It then applies a set partitioning algorithm to the wavelet coefficients to identify which coefficients can be discarded without significantly degrading the quality of the reconstructed image. The algorithm achieves high compression ratios by exploiting the statistical properties of the wavelet coefficients and using an adaptive bit allocation scheme to assign more bits to the coefficients that contribute more to the overall image quality.

The limitations of this technique include complexity - its complexity increases with the size of the image which is not suitable for high-resolution images; sensitivity - images with a lot of detail or sharp edges may not be compressed well; limited application -

preferred for compressing grayscale and color images and is not suitable for video or audio data types.

The performance of Bandelet-SPIHT is superior to that of the wavelet-SPIHT (classical) in terms of peak signal to noise ratio (PSNR), visual information fidelity (VIF) and mean structural similarity (MSSIM) at low bit rates. The computational complexity of bandelet SPIHT is same as that of H264/AVC but significantly lower than H.265/HEVC where the HEVC has 25% higher computation complexity.

O. Fractal Compression

[15] This technique is a lossy compression technique for compressing digital images based on the concept of self-similarity (fractals). It has the ability to achieve high high compression ratios and is preferred for complex or irregular images.

The image is divided into a set of non-overlapping blocks of equal size. Then for each block, a similar block in the image is to be found such that it can be scaled, rotated, and translated to match the original block. This is done by iteratively applying a set of affine transformations to the block and measuring the difference between the transformed block and the original block. The self-similarities are encoded using a set of parameters that describe the affine transformations required to transform the matching block into the original block. The parameters are stored in a codebook, which is used to represent the image. Then the codebook is compressed using a lossless compression algorithm, such as Huffman coding or arithmetic coding. To decompress the image, the codebook is first decompressed and used to reconstruct the self-similarities. The self-similarities are then used to reconstruct the original image.

The technique involves a large number of computations to find the best matches for each block of the image. Since it relies on self-similarities it is sensitive to any noise or artifacts present in the image which can lead to poor compression if any. Sometimes the fractal codebook could be quite large.

Fractal coding based on M-estimators (RMFIC) had the encoding time reduced by 87.5% compared with the full search Huber Image Fractal Compression (HIFC) and its retrieved image quality is better than other robust algorithms such as HFIC and LAD-FIC.

III. CONCLUSION

In conclusion, it should be noted that compression algorithms are crucial for lowering the amount of storage space and bandwidth needed to transmit digital data. A broad variety of compression algorithms are available, each with its strengths and limitations. Other techniques, like arithmetic coding and transform coding, are more sophisticated and can achieve higher compression ratios on a wider variety of data types, but some algorithms, like run-length encoding, are quick and easy to build and perform well with specific types of data.

Although compression methods can minimize the amount of storage space and transmission bandwidth needed, they frequently involve a trade-off between the quality of the reconstructed data and the level of compression that can be achieved. Because of this, it's essential to choose the right compression algorithm based on the application's particular needs and the desired trade-off between compression ratio and quality. The development of algorithms with even higher compression ratios while maintaining high quality, as well as the investigation of novel lossless or lossy compression techniques, may be the main areas of compression algorithm research in the future. Overall, to meet the growing demand for effective and dependable data storage and transmission, continued development and improvement of compression algorithms are essential.

REFERENCES

[1] Ghamry, F. M., Emara, H. M., Hagag, A., El-Shafai, W., El-Banby, G. M., Dessouky, M. I., ... & El-Samie, F. E. A. (2023). Efficient algorithms for compression and classification of brain tumor images. Journal of Optics, 1-13.

[2] Mitrovic, D. (2012). Video compression. University of Edinburgh.

[3] Witten, I. H., Neal, R. M., & Cleary, J. G. (1987). Arithmetic coding for data compression. Communications of the ACM, 30(6), 520-540.

[4] Sivakumar, R. D., & Ruba Soundar, K. A General Survey on Lossy Compression Algorithms for Online Learning Images in Cloud Environments.

[5] Khairi, N. A., & Jambek, A. B. (2021). Run-Length Encoding (RLE) Data Compression Algorithm Performance Analysis on Climate Datasets for Internet of Things (IoT) Application. International Journal of Nanoelectronics & Materials, 14.

[6] Hossain, K., & Roy, S. (2018, December). A data compression and storage optimization framework for iot sensor data in cloud storage. In 2018 21st International Conference of Computer and Information Technology (ICCIT) (pp. 1-6). IEEE.

[7] Dheemanth, H. N. (2014). LZW data compression. American Journal of Engineering Research, 3(2), 22-26.

[8] Cenzato, D., Guerrini, V., Lipták, Z., & Rosone, G. (2022). Computing the optimal BWT of very large string collections. arXiv preprint arXiv:2212.01156.
[9] Auristin, F. N., & Mali, S. (2016). New Ieee Standard For Advanced Audio Coding In Lossless Audio Compression: A Literature Review. IJECS, 5(4).
[10] Zhang, L., Li, F., & Qin, C. (2022). Efficient reversible data hiding in encrypted binary image with Huffman encoding and weight prediction. Multimedia Tools and Applications, 81(20), 29347-29365.
[11] Al-Azawi, R. J., & Drweesh, Z. T. (2019). Compression of Audio Using Transform Coding. J. Commun., 14(4), 301-306.
[12] Guerra, R., Barrios, Y., Díaz, M., Santos, L., López, S., & Sarmiento, R. (2018). A new algorithm for the on-board compression of hyperspectral images. Remote Sensing, 10(3), 428.
[13] Gurve, D., Saini, B. S., & Saini, I. (2016, March). An improved lossless ECG data compression using ASCII character encoding. In 2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET) (pp. 758-764). IEEE.
[14] Ferroukhi, M., Ouahabi, A., Attari, M., Habchi, Y., & Taleb-Ahmed, A. (2019). Medical video coding based on 2nd-generation wavelets: Performance evaluation. Electronics, 8(1), 88.
[15] Huang, P., Li, D., & Zhao, H. (2022). An Improved Robust Fractal Image Compression Based on M-Estimator. Applied Sciences, 12(15), 7533.
[16] Erdal, E., & Ergüzen, A. (2019). An efficient encoding algorithm using local path on huffman encoding algorithm for compression. Applied Sciences, 9(4), 782.