

Analysing Trends and Applications in Data Compression Techniques

Sabri KOCER

Necmettin Erbakan University

Bashir Sheikh Abdullahi JAMA

Necmettin Erbakan University

Yasin ER

Necmettin Erbakan University

Beyza DUNDAR

KTO Karatay University

Introduction

Information storage and transmission are frequently significant issues for businesses, states, and different associations. These organizations can deal with additional information by packing it, which likewise decreases the space and cost expected to store it. It could be valuable to appreciate how compress capabilities and the advantages it might propose to you and your association on the off chance that you save or send information as an aspect of your responsibilities. Data compression (DC) makes documents occupy less space on a hard drive and gets some margin time to transport or download. Cost decreases may be huge because of this decrease in existence. For example, organizations and medical care offices that keep a ton of information could lessen their capacity costs by utilizing compression to store more records on a given amount of space. Furthermore, since compressed documents transfer more quickly over the web or internet, these businesses and organizations don't need to spend as much money on pricey bandwidth upgrades. Compression empowers a few different organizations to offer the most ideal help at the most advantageous time. Consider how much sound and video information is dealt with by telecoms organizations. They can serve much more customers because of compression, which doesn't altogether bring down the sound or video quality.

In this chapter, we present a brief definition of the terminologies used in data compression, examine their significance, outline different compression techniques, explain the metrics used to compression performance and give an overview of how to apply compression to various types of data.

A Brief History of Data Compression

The history of DC dates back to the 18th century, when Morse code was invented, which was used for broadcast use in 1838. it dates back to the 18. century, is an exemplary illustration of the DC that depends on using more limited codewords for English alphabetic letters like “e” and “t” that are more predominant. With the appearance of data hypothesis

in the last part of the 1940s, current DC research started off (Wolfram, 2003). Claude Shannon and Robert Fano created a methodical methodology for allocating codewords based on block probability In 1949. David Huffman later discovered an ideal approach for doing this in 1951. Early implementations were frequently carried out in hardware, and certain codeword selections were made as trade-offs between compression and error correction(Blelloch, 2013). A methodical methodology for allocating codewords based on block probabilities was developed in 1949 by Claude Shannon and Robert Fano. Then, in 1951, David Huffman discovered the best technique for doing this. Early implementations were primarily hardware-based, with particular codeword selections made as trade-offs between compression and error-correction(Thanki & Kothari, 2019b). With the rise of the Internet in the 1970s, the transfer speed of the File increased significantly. The first studies on universal lossless compression algorithms were on Lempel-Ziv-Welch (LZW) in the mid-1980s. (Badshah et al., 2016). With LZW, the first widely used data compression algorithm introduced in computers, a text file could be reduced to about half. (Thanki & Kothari, 2019b). The Morse code, created by Samuel Morse in the middle of the 19th century, is an early instance of data compression. Dots and dashes are used to encode leers conveyed by telegraph. Morse observed that some letters appeared more frequently than others (Salomon, 2008). He assigned shorter sequences to leers that happen more frequently in order to decrease the average amount of time needed to transmit a message like e (·) and a(· -), and longer successions to sneers that happen less oftentimes, like q (- - · -) and j (· - - -). This thought of involving more limited codes for all the more every now and again happening characters is utilized in Huffman coding, which we will portray in the chapter.

How Data Compression Works

He gave leers that occur more frequently shorter sequences in order to decrease the average amount of time needed to transmit a message (Sayood, 2017). Every DC method has its own set of guidelines. As an illustration, when text compression is started, the computer will give each of the text gaps a single byte. It will then pack the byte into a string that instructs the decoder where to place everything back (Jovanovic, 2022).

Image compression operates same way as text. Depending on the algorithm, you can get a file that is significantly smaller but with noticeably worse image quality, or something that is about the same size and virtually similar to the original. By either eliminating extraneous data or collecting identical or related bytes and assigning them a new value, compression enables the computer to reassemble the original data (Jamil et al., 2022; Otair et al., 2022).

The same information is repeated throughout most computer file types, making them fairly redundant. Programs for compressing files merely eliminate redundant information. A file-compression program presents a piece of information only once and then references

it anytime it appears in the original program, as opposed to listing the same information repeatedly. Consider John F. Kennedy's 1961 inaugural address, where he gave the following well-known quote: **"Ask not what your country can do for you -- ask what you can do for your country."** The quote consists of 17 words, encompassed 61 letters, 16 spaces, 1 dash, and 1 period make up the quotation. The entire file size is 79 units if each letter, space, and punctuation mark eats up one unit of memory. We must look for duplications in order to reduce the file size.

Word	Number of occurrences
Ask	2
What	2
Your	2
Country	2
Can	2
Do	2
For	2
You	2

About half of the phrase is unnecessary if the distinction between capital and lower-case letters is ignored. We have practically all the information we require for the complete quote in nine words. We just point to the words in the first half of the statement and add the necessary punctuation and spaces to complete the second half.

Classification of Data Compression Techniques

The data compression (DC) approach has recently been created for a variety of data kinds, making it challenging to choose the right strategy for a given application. The quality of the data, different coding methods, different types of data, and applications can all be used to classify data compression approaches (Jayasankar et al., 2021). The following subsections provide more information on various DC techniques.

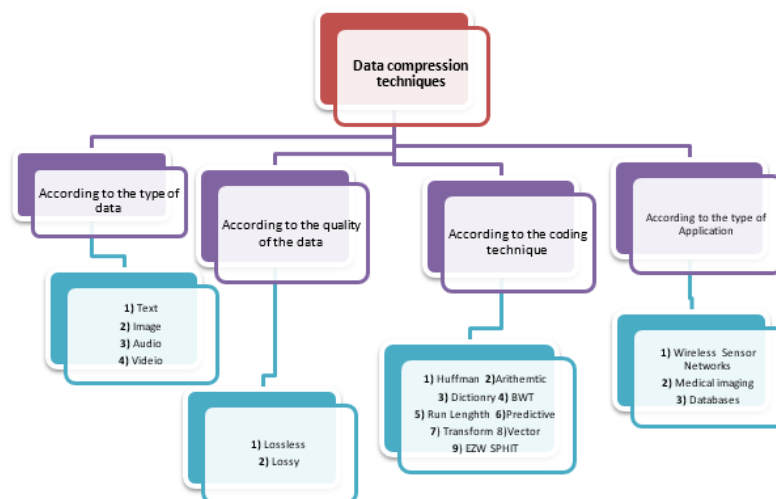


Figure 1. Classification of the DC methods

Classification of DC by data type

Techniques for DC are frequently employed for text, image, video, and audio data (Thanki & Kothari, 2019a). This section covers information on various DC methods and their features for each type of data.

Text Data

Lossless compression methods are frequently employed for text data. In order to compress textual data, Abel and Teahan created a number of preprocessing methods in 2005, including word substitution, phrase replacement, transliteration (from capital to tiny), coding of line endings, and alphabet recording (Navvabi et al., 2022). The drawback of these techniques is that data compression requires longer processing times. Plato presented a BWT-based compression method for repeating text files in about 2008 (Díaz-Domínguez & Navarro, 2022). The BWT approach investigated binary text data's Boolean reductions to produce compressed text data. Here, compressed text data was encoded using BWT coding (Platoš et al., 2008). Numerous real-world applications, including Facebook and texting, use this concept. Reversible transformation-based pre-processing method for converting text data to different forms was created by Roberts and Nadarajan (Sayood, 2018b, 2018a).

Image Data

The idea of picture compression was developed as a result of efforts to optimize website and media data over the network. By using picture compression, the image size can be decreased without significantly affecting the image's quality. You'll be able to do this to cut down on transfer and storage costs as well as time (Nuha et al., 2022).

The process of compressing image files can be done in a variety of ways. The JPEG format and the GIF format are the two most popular compressed graphic image formats for use on the Internet. While the GIF approach is frequently used for line art and other images with relatively simple geometric features, the JPEG methodology is more frequently utilized for photography. Wavelets and fractals are two other methods of image reduction (Javed et al., 2018). As of the time of this writing, these techniques have not become widely accepted for use on the Internet. Nevertheless, both techniques show potential as they provide higher compression ratios for particular types of images than the JPEG or GIF techniques (Larobina & Murino, 2014). The PNG format is another emerging technique that might eventually take the place of the GIF format. There is a limit to how much a text file or software may be compressed without introducing mistakes. Lossless compression is the term for this. Errors are introduced after this. Lossless compression is essential for text and program files because a single error can significantly alter a text file's meaning or prevent a program from running. A slight quality reduction during image compression is typically undetectable. There is no "critical point" beyond which compression is impractical but up to which it functions perfectly (Javed et al., 2018). The compression factor can be higher when there is some

loss tolerance than it can be when there is none. Graphical pictures can therefore be compressed more than text or program files.

You can gain from image compression by having your photos load more quickly and having your webpages load more quickly (Brahimi et al., 2022). Additionally, image compression might provide you with extra advantages like:

- **Reduction of the File Size:** One of the main advantages of image compression is this. You can reduce the image until it is the right size, regardless of the file size you are working with. This suggests that, until you alter the image, it occupies less space on the webpage and stays the same size physically.
- **Create Websites with less Bandwidth:** Webmasters can build websites with numerous images without using a lot of storage or bandwidth if they choose reduced image sizes.
- **Faster Loading on Mobiles:** People access websites on a variety of devices, including tablets, iPads, and much more, in addition to mobile phones. Your website will load more quickly on all devices if the picture files on it are small.
- **Maintain Image Quality:** If the quality of your photographs is an issue for you, you can conduct lossless image compression without suffering any image quality loss.
- **Easy Storage:** You can save more photographs on your hard disk since lower image sizes use less storage space than larger image sizes.

Audio Data

In order to drastically reduce file sizes and make it easier to share music and recordings over the network, audio data compression is a crucial and essential technique. Simply put, the current audio streaming environment would not exist without audio compression. Recently, authors created the linear prediction coding-based audio compression standard IEEE 1857.2 (LPC) (RANDHAWA, 2022). This standard offers higher-quality, lossless compression for audio signals with fast processing. In this method, pre-processing and an entropy coding block are done after the audio stream has already been encoded using LPC. When applied to voice and audio signals, this approach is effective. This method allows compressed original audio signals to be recreated. The development of a scalable variable bit rate encoding-based audio compression method (Hoang Thi Hang, 2022) It can be used for any variation in bandwidth.

Video Data

Real-time video compression is a challenging and significant subject that has sparked a lot of study. This corpus of knowledge has been heavily included into the H.261/H263 and MPEG motion video specifications (Ma et al., 2020). However, certain crucial issues have not yet been addressed. This chapter describes a potential replacement for these standards that has superior compression properties and requires far less processing

power to fully implement. (Duan et al., 2020).

Moving digital video pictures have been incorporated into applications since around 1989. The enormous bandwidth needed for the encoding of video data is what makes it challenging to integrate moving digital video. A quarter-screen image (320 x 240 pixels), for instance, needs to be stored and sent at a rate of 6.9 million bytes per second in order to play at full speed on an RGB video screen at 30 frames per second (FPS). This data rate is just unaffordable, so finding ways to compress digital video so that it may be played back in real-time is a must for the mainstream adoption of digital motion video applications (Chowdhery & Chiang, 2018). There are numerous digital video compression techniques that have been created and used. These algorithms' compression ratios (CR) change depending on the subjectively accepted degree of error, how the term "compression" is used, and who is making the claim. The digital video often has a lot of information for both transmission and storage (Javed et al., 2018). Compression is therefore required for the storage and transmission of any video stream. Researchers created a number of methods to effectively compress video streams. The Moving Pictures Experts Body (MPEG) is an ISO working group that creates numerous video stream compression standards (Duan et al., 2020).

Classification of DC by data quality

Generally, the employment of a data compression technique in a particular application has an impact on the quality of the data. The quality of compressed data is not given much consideration when compression techniques are utilized in applications that pertain to communication. However, the data loss is intolerable in applications involving text data compression. Small changes in image pixels are also undesirable when compressing satellite images or medical images. As a result, the type of data or application involved has a significant impact on the quality of the data in the data compression technique. Data compression techniques can be classified into lossless compression and lossy compression based on the quality of the compressed data that is required (Thanki & Kothari, 2019a).

Reconstructed data from compressed data in lossless compression resembles the original data. Data cannot be increased by the noise. Thus, it is referred as lossless. Most applications where information loss is unacceptable, such as those involving medical images, text, satellite imagery, etc., use lossless data compression. Reconstructed data in lossy compression is not an exact replica of the original data. This kind of approach involves a certain amount of data or information loss. In comparison to lossless compression techniques, lossy compression approaches can obtain a larger compression ratio (Blelloch, 2013; Rao & Yip, 2018).

Classification of DC by coding approach

This section categorizes data compression methods based on well-known coding

methods like dictionary-based predictive coding (PC), transform coding (TC), fractal compression (FC), wavelet transform (WT), and vector quantization (VQ). Also these methods include arithmetic, run-length encoding (RLE), Burrows-Wheeler Transform (BWT), Huffman coding (HC), and run-length encoding (RLE).

Huffman Coding

The fundamental concept of file compression is Huffman coding, a data compression algorithm. The method functions by building a binary tree with nodes. A node can either be an internal node or a leaf node. All nodes are initially leaf nodes, which include the character's weight (frequency of presence), as well as the character itself. Character weight and linkages to two child nodes are both found in internal nodes. Following the left child is denoted by bit 0 and following the right child by bit 1, respectively. n leaf nodes and $n-1$ interior nodes make to a finished tree. For the most ideal code lengths, it is advised that Huffman Tree throw away any unneeded characters in the text (Sayood, 2018a). The Huffman Tree can be built using a priority sequence as a basic illustration of the HC method, with the node with the lowest frequency having the highest priority. Below are all of the steps:

1. For every character, make a leaf node and add them to the priority queue.
2. Although there are multiple nodes in the queue:
 - Remove the two nodes of the highest priority (the lowest frequency) from the queue.
 - Add these two nodes as children of a new internal node with a frequency equal to the sum of both nodes' frequencies.
 - Add the new node to the priority queue.
3. The tree is finished when just the root node remains.

Arithmetic Coding

Important coding for the creation of compressed data utilising variable length codes was created by Langdon in 1984 (Kumar et al., 2021). Arithmetic coding is the name of this coding method. This method is more popular in practical applications and is superior to human coding. Lossless data compression employs a type of entropy encoding known as arithmetic coding (AC). As with the ASCII code, a string of characters is often represented using a fixed number of bits per character. An algorithm for arithmetic coding converts a whole file's worth of symbols into a single decimal integer (Guo et al., 2021). Each iteration processes a single input symbol. The code word for the complete string of symbols is chosen using the interval that is generated at the end of this division operation. Two categories of coding exist, including binary arithmetic coding (Klein & Shapira, 2021) and adaptive arithmetic coding (Dudhagara & Patel, 2017). When dealing with sources with limited alphabets, including binary sources, and alphabets with significantly skewed probability, the AC is very helpful. When the

modelling and coding components of lossless compression need to be kept apart for a variety of reasons, it is also a very helpful method.

Dictionary Coding

In order to produce greater compression, dictionary compression algorithms use a variety of strategies that take into account the data's structure (Cinque et al., 2009). The idea is to avoid keeping redundant strings for words and phrases that repeatedly appear in the text stream. The most frequently used words or phrases are kept in a dictionary, which is used by the coder to generate tokens for output. Ideally, the words or phrases are repeated throughout the page and the tokens are significantly shorter than the actual words or phrases (Thanki & Kothari, 2019b).

When reading an input string, an encoder looks for recurrent words and outputs those words' dictionary indices. A new word is produced in its uncompressed form and added as a new entry to the dictionary. The primary operations are efficient encoding, dictionary upkeep, and string comparison. Both compressors and decompressors keep a lexicon on their own. Generally speaking, dictionary-based algorithms are quicker than entropy-based ones (Cinque et al., 2009; Navvabi et al., 2022). Instead of treating the input as a stream of bits, they treat it as a series of characters. The compression method receives a stream of symbols as input, and the output is a combination of tokens and words in their original form. The coding system can be categorized as working in variable-to-fixed fashion when it outputs tokens since, in its simplest version, each string to be encoded is of a varied length but the code words, or the dictionary's indexes, are of the same length. Due to the fact that the dictionary is changed during the compression and decompression processes, dictionary-based techniques are adaptable (Blelloch, 2013). The dictionary's contents change depending on the order in which the compressed text is entered. Dictionary techniques rely on finding recurring patterns rather than any statistical model. As a result, neither the statistical model's quality nor the source's entropy have any bearing on the compression effect. As a result, it can frequently obtain a better compression ratio than techniques that use statistical models (Cinque et al., 2009).

Run Length Encoding (RLE)

A straightforward method of lossless data compression called run-length encoding (RLE) applies to sequences where the same value appears repeatedly. In order to store just one value and its count, the sequence is encoded (Birajdar et al., 2019). For instance, if the sample string is "mmmmaaadexxxxxx", then the process should return "m4a3d1e1x6". To solve this issue, adhere to the methods listed below::

- Choose the first letter of the source string.
- Add the chosen character to the string's final destination.
- Count how many times the chosen character appears after that and add that number to the final string.

- If the string's end isn't reached, choose the next character and repeat steps 2, 3, and 4.

Predictive and Transform

Predictive coding and transform coding are the bases for another classification of data compression techniques. Predictive coding uses input data that is already accessible to anticipate compressed data, and it performs encoding depending on the discrepancy between the predicate data and the actual input data (Chowdhery & Chiang, 2018). This method is the simplest, clearest, and most useful for compressing texture images. When using transform coding, the input data is first converted into its transformed coefficients, and then these coefficients are encoded. Here, different transformations like the discrete Fourier transform (DFT), discrete cosine transform (DCT), and discrete wavelet transform (DWT) are widely used (Rao & Yip, 2018). The transform coding method, which is referred to as the better compression method, offers superior compression to data utilising a small number of coefficients.

Vector-based Compression

In vector quantization, blocks or vectors are formed from the source output. As an illustration, we can think of $[X]$ successive speech samples as the elements of a $[x\text{-dimensional}]$ vector. Alternately, we can treat each pixel value in a block of L pixels from an image as a component of a vector of size or dimension $[X]$. The vector quantizer receives this vector of source outputs as input. We have a collection of $[X\text{-dimensional}]$ vectors at the vector quantizer's encoder and decoder known as the vector quantizer's codebook (Blelloch, 2013; Faria et al., 2012). The "code-vectors" in this codebook are chosen to be typical of the vectors we produce from the source output. A binary index is assigned to each code-vector. The input vector is compared to each code-vector at the encoder to determine which code-vector is most similar to the input vector (Blelloch, 2013). The quantized values of the source output make up the codevector's elements. We broadcast or store the binary index of the code-vector in order to provide the decoder with information about which code-vector was determined to be the most similar to the input vector. Given its binary index, the decoder can obtain the code-vector because it uses the exact codebook.

EZW SPHIT

The Embedded Zero-tree Wavelet is known as EZW. On many test photos, it provides good results for compression. J. Shapiro provided the introduction. It uses adaptive arithmetic coding and is a lossless compression technology. It is a useful algorithm for compressing images (Dudhagara & Patel, 2017). For picture coding, it generates a fully embedded bit stream. This method uses DWT, critical information is missing, entropy coding, and lossless compression using arithmetic coding. Most coefficients produced by a subband transform will be 0 or very close to zero when the bit rate is low. Since low frequency information is included in nearly all standard test photos, it is close to

zero. High frequency information is sent using high quality schemes. This is crucial to how people judge the quality of images (Dudhagara & Patel, 2017). Set Partitioning in a Hierarchical Tree is referred to as SPIHT. This EZW technique has been changed. It is an image compression coder based on wavelets. It transforms the image into a wavelet. The best image quality and high PSNR values were produced by this method, which is what makes it so essential. It provides a fully progressive bit stream and can be used for lossless image compression. This algorithm's key benefit is that it is entirely progressive. Wavelet transform coefficients are coded and sent using this technique. Therefore, it is beneficial to gradually regenerate the original copy of the image. This technique for compressing images is quite effective. It produces a compressed bit-stream and has fewer complexes, which makes it easier to decode at various transmission rates (Dudhagara & Patel, 2017).

Classification of DC by type of Application

Depending on the application, the DC techniques can be used in a variety of apps. It is challenging to categorise each method according to where it can be used. In this part, various application areas of DC methods are briefly explored along with their features for diverse purposes. A data compression strategy can be used to limit the quantity of information being transferred in a network, resulting in a reduction in power consumption, which is a major power consumer in wireless sensor networks (WSN). This implies that a higher data compression ratio results in a higher percentage of power savings (Shin & Kim, 2016). Image compression significantly reduces the amount of storage space and transmission bandwidth needed, which not only supports the broad use of medical imaging but also serves as one of the most crucial foundational technologies for telemedicine applications. This is crucial in endoscopy as well, which is the process of gathering and saving a lot of data for more accurate diagnosis. Different DC approaches are utilised for different kinds of data, including databases, signals, data from remote sensing, and DNA sequences. As a result, the DC has a wide range of applications, which demonstrates its vital role in current technological trends (Thanki & Kothari, 2019b).

Evaluation metrics for data compression

There are several ways to assess a compression algorithm (AA Azeez, 2016; Baker et al., 2014). We may evaluate the algorithm's relative complexity, the memory needed to implement it, the speed at which it runs on a specific machine, the level of compression, and how closely the reconstruction resembles the original (Faria et al., 2012). The final two criteria will be our main focus in this chapter. Let's examine each one individually. Looking at the ratio of the number of bits needed to represent the data before compression to the number of bits needed to represent the data after compression is a pretty natural way to gauge how well a compression algorithm compresses a given batch of data. The

compression ratio is the name of this ratio. Let's say that 65,536 bytes are needed to store a 256x256 pixel square image. 16,384 bytes are needed to store the compressed version of the image. The compression ratio, in our estimation, is 4:1. The reduction in the quantity of data required as a percentage of the size of the original data is another way to express the compression ratio. The compression ratio determined using this method in this specific case would be 75%.

The average number of bits used to represent a single sample is another approach to report compression performance. Typically, this is referred to as the rate. For instance, in the above-mentioned compressed image, the compressed representation's average bit rate per pixel is 2. We could therefore state that the rate is 2 bits per pixel.

The reconstruction in a lossy compression is different from the original data. As a result, we need a method for measuring the difference in order to establish how effective a compression algorithm is. The distortion is the term used to describe the variation between the original and the reconstruction. Lossy compression methods are typically employed for data that started off as analog signals, such as speech and video (AA Azeez, 2016). Human beings are the ultimate arbiters of quality in speech and video compression. The quality of the reconstructed waveforms is assessed using a variety of approximations of distortion because it is challenging to mathematically represent human responses.

Conclusion

Data compression is the technology term utilized in current computing technology that spotlights on the decrease in the quantity of pieces expected to represent data. Compression can increase file transfer speed, preserve storage space, and lower the cost of storage hardware and network bandwidth. In this chapter, the basic concepts and importance of advanced technological DC methods used in many real-life applications are discussed. The role and importance of data compression technology are described in detail in the first chapters, and in the second, the categories and classification of DC methods are presented. The main methods of the DC are described in detail in this section. In the section, the evaluation criteria of DC, which are more important for evaluating the performance of DC techniques, are presented. In the last part of the chapter, general and special purpose applications such as WSN, medical imaging, DNA data, signal data, databases and remote detection using DC methods are discussed.

References

- AA Azeez, A. L. (2016). Empirical and statistical evaluation of the effectiveness of four lossless data compression algorithms. *Igerian Journal of Technological*, 13(2). <https://www.ajol.info/index.php/njtd/article/view/152843>.
- Badshah, G., Liew, S. C., Zain, J. M., & Ali, M. (2016). Watermark Compression in Medical Image Watermarking Using Lempel-Ziv-Welch (LZW) Lossless Compression Technique. *Journal of Digital Imaging*, 29(2), 216–225. <https://doi.org/10.1007/S10278-015-9822-4>.

- Baker, A. H., Levy, M. N., Xu, H., Nychka, D., Dennis, J. M., & Mickelson, S. A. (2014). A methodology for evaluating the impact of data compression on climate simulation data. *HPDC 2014 - Proceedings of the 23rd International Symposium on High-Performance Parallel and Distributed Computing*, 203–214. <https://doi.org/10.1145/2600212.2600217>.
- Birajdar, A., Agarwal, H., Bolia, M., & Gupte, V. (2019). Image Compression using Run Length Encoding and its Optimisation. *2019 Global Conference for Advancement in Technology, GCAT 2019*. <https://doi.org/10.1109/GCAT47503.2019.8978464>.
- Blelloch, G. (2013). Introduction to data compression. *Orion.Towson.Edu*. <http://orion.towson.edu/~mzimand/ainternetalg/Blelloch-Compression.pdf>
- Brahimi, T., Khelifi, F., Laouir, F., & Kacha, A. (2022). A new, enhanced EZW image codec with subband classification. *Multimedia Systems*, 28(1). <https://doi.org/10.1007/S00530-021-00781-X>.
- Chowdhery, A., & Chiang, M. (2018). Model Predictive Compression for Drone Video Analytics. *2018 IEEE International Conference on Sensing, Communication and Networking, SECON Workshops 2018*, 1–5. <https://doi.org/10.1109/SECONW.2018.8396351>.
- Cinque, L., De Agostino, S., & Lombardi, L. (2009). Practical parallel algorithms for dictionary data compression. *Proceedings - 2009 Data Compression Conference, DCC 2009*, 441. <https://doi.org/10.1109/DCC.2009.84>.
- Díaz-Domínguez, D., & Navarro, G. (2022). *Efficient Construction of the BWT for Repetitive Text Using String Compression*. <http://arxiv.org/abs/2204.05969>.
- Duan, L., Liu, J., Yang, W., Huang, T., & Gao, W. (2020). Video Coding for Machines: A Paradigm of Collaborative Compression and Intelligent Analytics. *IEEE Transactions on Image Processing*, 29, 8680–8695. <https://doi.org/10.1109/TIP.2020.3016485>.
- Dudhagara, C. R., & Patel, M. M. (2017). *A Comparative Study and Analysis of EZW and SPIHT methods for Wavelet based Image Compression*. <https://doi.org/10.13005/ojst/10.03.17>.
- Faria, L. N., Fonseca, L. M. G., & Costa, M. H. M. (2012). Performance evaluation of data compression systems applied to satellite imagery. *Journal of Electrical and Computer Engineering*. <https://doi.org/10.1155/2012/471857>.
- Guo, Z., Fu, J., Feng, R., & Chen, Z. (2021). Accelerate neural image compression with channel-adaptive arithmetic coding. *Proceedings - IEEE International Symposium on Circuits and Systems, 2021-May*. <https://doi.org/10.1109/ISCAS51556.2021.9401277>.
- Hoang Thi Hang, N. V. V. (2022). *APPLY QMF FILTER TAPE IN DIGITAL FILTER BAND TO COMPRESS DIGITAL AUDIO SIGNALS*. *UTEHY Journal of Science and Technology*. <http://tapchi.utehy.edu.vn/index.php/jst/article/view/495>.
- Jamil, S., Jalil Piran, M., & Rahman, M. (2022). *Learning-Driven Lossy Image Compression; A Comprehensive Survey*. <http://arxiv.org/abs/2201.09240>.
- Javed, M., Nagabhushan, P., & Chaudhuri, B. B. (2018). A review on document image analysis techniques directly in the compressed domain. *Artificial Intelligence Review*, 50(4), 539–568. <https://doi.org/10.1007/S10462-017-9551-9>.

- Jayasankar, U., Thirumal, V., & Ponnuram, D. (2021). 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*, 33(2), 119–140. <https://doi.org/10.1016/J.JKSUCI.2018.05.006>.
- Jovanovic, B. (2022). *Data Compression Explained*. Dataport. <https://dataprot.net/guides/data-compression/>.
- Klein, S. T., & Shapira, D. (2021). Integrated encryption in dynamic arithmetic compression. *Information and Computation*, 279, 104617. <https://doi.org/10.1016/J.IC.2020.104617>.
- Kumar, G., Processing, R. K.-G. and S., & 2021, undefined. (2021). Analysis of Arithmetic and Huffman Compression Techniques by Using DWT-DCT. *Mecs-Press.Org*, 4, 63–70. <https://doi.org/10.5815/ijigsp.2021.04.05>.
- Larobina, M., & Murino, L. (2014). Medical image file formats. *Journal of Digital Imaging*, 27(2), 200–206. <https://doi.org/10.1007/S10278-013-9657-9>.
- Ma, S., Zhang, X., Jia, C., Zhao, Z., Wang, S., & Wang, S. (2020). Image and Video Compression with Neural Networks: A Review. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(6), 1683–1698. <https://doi.org/10.1109/TCSVT.2019.2910119>.
- Navvabi, S., Najafi Oshnari, M., & Novikov, B. (2022). Lossless Genome Data Compression Using V-Gram. *Springer Proceedings in Physics*, 268, 141–149. https://doi.org/10.1007/978-3-030-81119-8_15.
- Nuha, H., Mohandes, M., Liu, B., & Al-Shaikhi, A. (2022). Seismic Data Compression: A Survey. *Advances in Science, Technology and Innovation*, 253–255. https://doi.org/10.1007/978-3-030-73026-0_58.
- Otaïr, M., Abualigah, L., & Qawaqzeh, M. K. (2022). Improved near-lossless technique using the Huffman coding for enhancing the quality of image compression. *Multimedia Tools and Applications*. <https://doi.org/10.1007/S11042-022-12846-8>.
- Platoš, J., Snášel, V., & El-Qawasmeh, E. (2008). Compression of small text files. *Advanced Engineering Informatics*, 22(3), 410–417. <https://doi.org/10.1016/J.AEI.2008.05.001>.
- RANDHAWA, S. S. (2022). *EEE 1857.2 LOSSLESS AUDIO COMPRESSION: A LITERATURE REVIEW*. Researchgate. https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&as_ylo=2018&q=IEEE+1857.2++linear+prediction+coding+&btnG=.
- Rao, K., & Yip, P. (2018). *The transform and data compression handbook*. <https://www.taylorfrancis.com/books/mono/10.1201/9781315220529/transform-data-compression-handbook-kamisetty-ramam-rao-patrick-yip>.
- Salomon, D. (2008). *A Concise Introduction to Data Compression*. <https://doi.org/10.1007/978-1-84800-072-8>.
- Sayood, K. (2017). *Introduction to data compression*. <https://books.google.com/books?hl=en&lr=&id=3DFHDgAAQBAJ&oi=fnd&pg=PP1&dq=how+data+compression+works&ots=gGMmmxOh74&sig=dv3DcxUHkYiElz-kpjpBxLj69Gk>.
- Sayood, K. (2018a). Huffman Coding. *Introduction to Data Compression*, 41–88. <https://>

doi.org/10.1016/B978-0-12-809474-7.00003-3.

Sayood, K. (2018b). *Text Compression - an overview* |. ScienceDirect. <https://www.sciencedirect.com/topics/computer-science/text-compression>.

Shin, D., & Kim, C. (2016). Data Compression Method for Reducing Sensor Data Loss and Error in Wireless Sensor Networks. *Journal of Korea Multimedia Society*, 19(2), 360–374. <https://doi.org/10.9717/KMMS.2016.19.2.360>.

Thanki, R. M., & Kothari, A. (2019a). Classification in Data Compression. *Hybrid and Advanced Compression Techniques for Medical Images*, 17–29. https://doi.org/10.1007/978-3-030-12575-2_2.

Thanki, R. M., & Kothari, A. (2019b). Hybrid and advanced compression techniques for medical images. *Hybrid and Advanced Compression Techniques for Medical Images*, 1–95. <https://doi.org/10.1007/978-3-030-12575-2/COVER>

Wolfram, S. (2003). *Historical Notes from Stephen Wolfram's A New Kind of Science*.

About the Authors

Sabri KOCER, He is a professor and senior researcher in the Department of Computer Engineering at Konya Necmettin Erbakan University. He completed his doctorate at the Department of Electronic Computer Education at Gazi University. He has held academic positions at different Universities in Turkey and has made great contributions to the on-going technological academic development in the region. His main research interests are biomedical engineering, artificial intelligence, optimization, and machine learning and microcontroller applications. In addition, he has many academic works such as books, articles, essays, posters published in many national and international journals.

E-mail: skocer@erbakan.edu.tr, **ORCID:** 0000-0002-4849-747X

Bashir Sheikh Abdullahi JAMA, He is a computer engineer and junior researcher in the field of computer science and engineering. He is interested in conducting research that improves human life and helps to adapt to the modern technological world. His research interests include artificial intelligence, machine learning, computer vision and digital image processing. Currently, he is a PhD student in the Department of Computer Engineering at Konya Necmettin Erbakan University.

E-mail: bashiir204@gmail.com, **ORCID:** 0000-0002-2417-9223

Yasin ER, he works as a computer engineer at the Turkish State Meteorological Service. He has a master's degree from Hoca Ahmet Yesevi University. His studies are meteorological communication, enterprise-specific software development, and Windows/Linux-based system management. He is a member of various expert groups and working groups at the World Meteorological Organization, currently, He is studying for a doctorate at the Institute of Pure and Applied Science of Necmettin Erbakan University.

E-mail: yasineryasiner@gmail.com, **ORCID:** 0000-0001-6729-5278

Beyza DUNDAR, She is a senior student in Computer Engineering at KTO Karatay University. System has studied data science, artificial intelligence, image processing and natural language processing. She works on smart cities on the Capsule Technology Platform.

E-mail: dundarbeyza777@gmail.com, **ORCID:** 0000-0002-2843-1043

Similarity Index

The similarity index obtained from the plagiarism software for this book chapter is 16%.

To Cite This Chapter:

Kocer, S. , Jama, B.S.A ., Er Y. , Dundar, B. (2022). Analysing Trends and Applications in Data Compression Techniques, In S. Kocer, O. Dundar (Eds .), *Current Studies in Basic Sciences Engineering and Technology* (pp. 114–128). ISRES Publishing.