Journal of
Computer & Electrical and Electronics Engineering Sciences
Original Article
DOI: 10.51271/JCEEES-0031

Application and performance evaluation of the dynamic bit-level encoding algorithm (DEA) in text compression: a cross-domain perspective

Erdal Erdal, Atilla Ergüzen, Alperen Önal*
Department of Computer Engineering, Faculty of Engineering and Natural Sciences, Kırıkkale University, Kırıkkale, Turkiye

Cite this article as: Erdal, E., Ergüzen, A., & Önal, A. (2026). Application and performance evaluation of the dynamic bit-level encoding algorithm (DEA) in text compression: a cross-domain perspective. *J Comp Electr Electron Eng Sci, 4*(1), 1-8.

Received: 15.08.2025 ◆ **Accepted**: 04.09.2025 ◆ **Published**: 25.04.2026

ABSTRACT

The exponential growth of digital data has heightened the need for efficient, lossless compression techniques to improve storage and transmission performance. This study investigates the applicability of the dynamic bit-level encoding algorithm (DEA), originally designed for image compression, to text compression. DEA employs a dynamic, frequency-based bit-level coding scheme that models inter-character relationships and constructs two-level Huffman trees for high- and low-frequency character groups. Unlike conventional algorithms such as Huffman, LZ77, and LZ78, DEA adapts its coding structure to local data patterns, aiming to enhance compression efficiency across heterogeneous text datasets. Experiments were conducted on nine datasets-including plain text, SQL queries, and HTML documents-using compression ratio (CR) and space saving (SS) as evaluation metrics. DEA achieved an average CR of 2.72 and SS of 62%, outperforming Huffman (1.87 CR, 43% SS), LZ77 (1.67 CR, 30% SS), and LZ78 (1.73 CR, 14% SS) across all datasets. These results demonstrate DEA's robustness and adaptability to diverse text structures. The findings suggest DEA as a practical alternative for applications requiring high-efficiency, lossless compression, such as archival systems, log management, messaging protocols, and real-time data transmission. Future work will focus on memory optimization, parallelization for large-scale applications, and integration into hybrid compression frameworks.
Keywords: Text compression, lossless compression, dynamic encoding algorithm, cross-domain compression

INTRODUCTION

In the contemporary era, the generation of digital data is undergoing an exponential growth, giving rise to substantial challenges in the domains of data storage and transmission (Dhulavvagol et al., 2024). The explosion of data has led to a strain on physical storage resources and has also exerted pressure on system resources, including network bandwidth, processing time, and energy consumption (Gastón et al., 2013). In this context, data compression techniques enhance storage efficiency by encoding digital data to occupy less space, thereby improving data transmission performance.

Data compression can be categorized into two primary classifications: lossy and lossless (Beemkumar et al., 2024). Lossless compression methods are preferred for sensitive data, such as text, where structural integrity is paramount (Gupta et al., 2017). Text compression is a process that represents character sequences with shorter bit sequences (KeskiN et al., 2023). It is widely used in many areas, such as natural language processing, data archiving, messaging protocols, and log analysis. In such applications, the utilization of algorithms that offer both high CRs and low computational costs is of critical importance.

Conventional algorithms, including Huffman encoding, LZ77, and LZ78, have been demonstrated to be effective by capitalizing on the statistical and structural characteristics inherent in text data (Huffman, 1997). These encoding algorithms form the foundation of contemporary compression algorithms. However, the optimization of these methods for specific data types can impose various limitations in cross-domain applications. Consequently, the evaluation of algorithms developed for disparate domains on text data is imperative for two primary reasons. Firstly, it facilitates the identification of the potential inherent in novel methodologies. Secondly, it enables the assessment of the generalizability of these algorithms.

This study evaluates the performance of the dynamic bit-level encoding algorithm (DEA) (Erdal & Önal, 2025), initially developed for image compression, in text compression. We test the algorithm in comparison with traditional text compression methods, such as Huffman, LZ77, and LZ78. The goal is to determine whether DEA is effective with text data, under what conditions it provides advantages, and what its limitations are.

Corresponding Author: Alperen Önal, 238802017@kku.edu.tr

The paper begins with an overview of existing compression techniques, followed by an introduction to the DEA algorithm's basic structure. Then, the experimental setup and metrics used are explained, and the results are evaluated in detail to discuss DEA's potential in text compression. The final section summarizes the findings and suggests future research directions.

BACKGROUND AND RELATED WORK

Data compression techniques use coding strategies to reduce file size by taking advantage of statistical properties and repetitive structures (Gopinath & Ravisankar, 2020). Lossless compression algorithms are designed to ensure that the original data can be fully recovered. Huffman encoding is one of the most widely used methods for this purpose (Huffman, 1952). It assigns short and long bit sequences based on the frequency of characters; more frequently occurring characters are represented by shorter codes. This approach provides a highly effective CR, especially in texts with uneven character distributions. However, because it operates on a symbol-based system, Huffman coding is limited in its ability to consider local context or larger patterns.

The LZ77 and LZ78 algorithms belong to the Lempel-Ziv family (Welch, 1984). They replace repetitive structures in data with pointer structures by performing a broader context analysis. The LZ77 algorithm identifies repetitive sequences by referencing previous data with a sliding window approach (Ziv & Lempel, 1977). LZ78, on the other hand, builds a dictionary-based structure that represents previously encountered sequences through indexes (Ziv & Lempel, 1978). These techniques produce effective results, especially in long texts or data sets with frequent repetition of certain patterns. However, parameters such as dictionary size and update strategy directly affect the performance of these algorithms. These classical algorithms all offer different advantages and limitations depending on the data type and pattern density. This necessitates developing new algorithms and conducting comparative performance analyses.

Huffman Encoding Algorithm

The Huffman coding algorithm, developed by David A. Huffman in 1952, is a statistical lossless data compression method (Huffman, 1952). It works by representing more frequently occurring symbols with shorter bit sequences and less frequently occurring symbols with longer ones. This minimizes the total bit length of the data, thereby achieving compression. Huffman coding first calculates the frequencies of characters and then creates a binary tree structure based on these frequencies.

The Huffman algorithm uses a prefix-free code structure, meaning no code can be the beginning of another code. This feature facilitates decoding compressed data and ensures that it can be decoded only once. The Huffman tree created during the encoding process is referenced during both encoding and decoding. Typically stored alongside the data, this tree enhances the algorithm's universality but may slightly reduce the CR.

The algorithm's greatest strengths are its low complexity and its ability to deliver highly effective results based on the statistical distribution of data. Huffman coding is particularly effective in texts with uneven symbol frequencies. However, its tendency to treat each character independently and its inability to evaluate contextual patterns within the data can result in poorer performance in more complex data structures.

For this reason, Huffman coding is often used with other techniques or supported by more advanced, artificial intelligence-based, or pattern recognition methods. Nevertheless, its simple structure, high speed, and deterministic analysis feature make it an attractive option, especially for systems with limited processing power.

LZ77 Algorithm

The LZ77 algorithm, developed in 1977 by Abraham Lempel and Jacob Ziv, is a lossless compression method (Ziv & Lempel, 1977). It is based on identifying repeating data patterns and representing these repetitions with references, or pointers. LZ77 uses a sliding window approach to identify repeating sequences within previous data blocks, expressing these repetitions with a triplet structure that specifies the position and length of the data.

The algorithm does not consider both past and future data; it only uses a specific portion of the past window as a reference. The longest matching sequences within the window are identified and replaced with "distance, length, character" tokens. This structure provides a highly effective CR, especially when sequential repetitions are present in the data.

The LZ77 algorithm's greatest advantage is its ability to capture long and frequent repetitions in data effectively and perform compression at the sequence level rather than the character level. However, this method may require large buffers to increase the CR, which increases processing time proportionally to the window size. Additionally, its performance may be limited in short texts or data with low repetition rates.

Today, various LZ77 variants have been developed, forming the basis for common compression formats such as ZIP and GZIP. Thus, LZ77 remains one of the most important reference algorithms in text compression, both theoretically and practically.

LZ78 Algorithm

Developed in 1978 by Abraham Lempel and Jacob Ziv, the LZ78 algorithm is an extension of the LZ77 algorithm (Ziv & Lempel, 1978). It uses a dynamically generated dictionary structure to compress data. Each new sequence is evaluated based on its presence in the dictionary. If the sequence is not present, it is added as a new entry. Compressed data consists of an index pointing to the previous sequence in the dictionary combined with the new character.

Unlike LZ77, which references past data, LZ78 creates a dictionary that grows with each new sequence of symbols. This enables the algorithm to capture direct repetitions and more complex patterns within the data. As the algorithm learns each new sequence, the CR increases, making it particularly effective for large data sets.

The biggest advantage of LZ78 is its ability to adapt to data over time thanks to its constantly updated dictionary. However, controlling the size of the dictionary is also necessary. Efficient memory management is critical for the algorithm to work properly. Additionally, properly synchronizing the dictionary may require storing additional structures for data analysis.

The LZ78 algorithm formed the basis for many modern compression methods, particularly laying the groundwork for more advanced algorithms, such as LZW (Lempel-Ziv-Welch). Thus, LZ78 is considered one of the most fundamental algorithms, retaining its importance in both theoretical studies and practical data compression systems.

Application of DEA in Image Compression

In the field of image compression, lossless compression requires that data be represented in smaller sizes while preserving its integrity. Traditional algorithms often offer limited performance in this context. Increasing data diversity and resolution levels has heightened the need for more flexible, data-centric encoding methods. The DEA algorithm was developed to address this need. It offers a dynamic bit-level encoding approach that can easily adapt to different image types, particularly thanks to its adaptive structure (Erdal & Önal, 2025). DEA performs frequency-based encoding and works with an innovative strategy that includes contextual analysis based on inter-character relationships.

DEA's fundamental principle is evaluating characters likely to follow each other in sequence. This results in a dynamic coding dictionary, unlike the traditional Huffman encoding algorithm, which uses a fixed dictionary. The DEA analyzes these relationships to represent frequently recurring patterns with shorter bit sequences, ensuring that rarely encountered structures are encoded optimally. This contributes to more efficient CRs, particularly in high-resolution images with a wide range of colors. DEA's adaptive structure increases the CR and enables the algorithm to automatically configure itself according to different image contents.

The algorithm creates two groups within the application: a "palette" (group 1) for frequently used characters and a secondary group (group 2) for less common ones. The algorithm dynamically determines these two groups by analyzing the frequency with which each character occurs. Separate Huffman trees are created for each group and used during the encoding process according to which group each character belongs to. This structure enables DEA to encode frequently used characters with short bit sequences while preventing the unnecessary length of the representation of rarely used characters. Thus, DEA provides a bit-level optimized structure. This structure allows for effective compression of the entire image and its sub-segments. The flowchart of the DEA algorithms presented below in **Figure 1**.

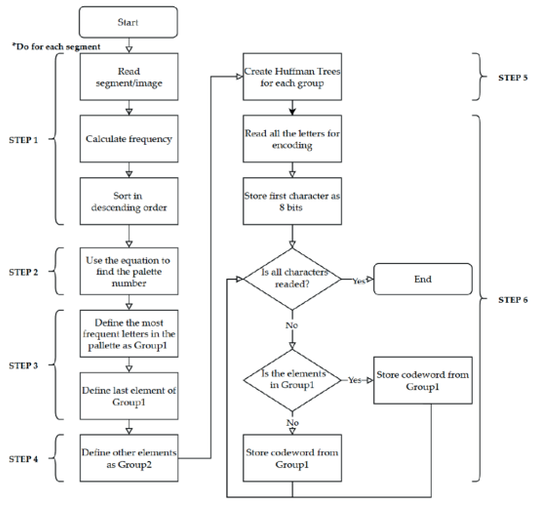

Figure 1. DEA flowchart (Erdal & Önal, 2025)

DEA: Dynamic bit-level encoding algorithm

The applicability of this algorithm in image compression has been enhanced by a segmentation-based preprocessing step. DEA processed images divided into segments based on color and texture similarities separately, thus enabling encoding on homogeneous data sets. This approach maximizes DEA's pattern recognition capabilities and increases compression efficiency. Additionally, the data structures created after segmentation (header and data blocks) ensure that the compressed data can be reconstructed without loss.

We tested the DEA algorithm in comparison with current lossless and lossy algorithms, such as JPEG2000, JPEG-LS, and BPG (Erdal & Önal, 2025). The results showed that DEA offers significant advantages, particularly for data sets with high color density and pattern repetition rates. Furthermore, the S+DEA version, which operates alongside segmentation, has been found to deliver superior performance across all datasets except for certain medical images. These results demonstrate that DEA is an innovative approach offering a comprehensive compression framework when combined with segmentation and data structures, not merely an encoding algorithm.

Gap in the Literature and the Need for Cross-Domain Evaluation

A review of the extant literature on data compression reveals that numerous algorithms have been developed for specific data types. Performance evaluations are typically conducted only for the designated domain. Algorithms that operate on different data types, such as image compression and text compression, are often addressed separately. As a result, the cross-domain validity of these algorithms is largely overlooked. However, the increasing demands of modern data processing have led to a growing need for adaptive and general-purpose compression techniques capable of operating with multiple data types concurrently.

In this context, the most dynamic and adaptive compression algorithms in the extant literature are either limited to fixed coding structures or dependent on specific pattern types. Classical algorithms such as Huffman, LZ77, and LZ78 have been demonstrated to be most effective in text or data sets with specific characteristics. However, the efficacy of these algorithms may be constrained in the context of alternative data types. This situation suggests that confining the evaluation of testing algorithms to their specific domains impedes the assessment of their potential application areas.

A notable lacuna in the extant literature pertains to the evaluation of algorithms developed for disparate domains in the context of their applicability to diverse data types. For instance, while the DEA algorithm has demonstrated notable efficacy in the domain of image compression, its effectiveness on character-based and structurally patterned data, such as text, remains to be systematically investigated. However, the dynamic structure of the DEA, its frequency-based coding approach that is sensitive to character sequences, and its ability to process on a segment basis render it a potential candidate for text data.

The primary motivation behind this study is to extend the boundaries of existing algorithms, assess their cross-domain validity, and identify novel application domains. In particular, the adaptability of algorithms with unique approaches, such as the data envelopment analysis (DEA), to different content types, such as text, should be analyzed without being limited

to a single data type. Such cross-evaluations are critical for both testing the generalizability of the algorithm and increasing the diversity of applications in the literature.

In summary, the present study undertakes two primary objectives. Firstly, it evaluates the performance of DEA on text data. Secondly, it proposes a model in terms of cross-domain validity. Thus, the study aims to fill an important gap in the literature. Consequently, the efficacy of DEA, which has been demonstrated to exhibit high efficiency in the context of image compression on various data types, including text, will be systematically investigated.

METHODS

Ethics

Ethical approval was not required for this study as it does not involve human participants, animal subjects, or identifiable personal data. All procedures were carried out in accordance with the ethical rules and principles.

Dynamic Bit-Level Encoding Algorithm (DEA)

Algorithm overview: The DEA is an innovative encoding method that is adaptable to different data types, lossless, and provides high efficiency. In contrast to the fixed-structure classical methods, the algorithm autonomously adjusts its configuration in a manner that is sensitive to local patterns and character sequences within the data. Initially developed for the purpose of image compression, DEA has the potential to offer a viable solution for character-based data types, such as text data, due to its context-sensitive structure.

The fundamental principle of DEA is to analyze the characters that are likely to follow each character and incorporate these relationships into the model based on frequency. In this process, character pairs that occur with high frequency are represented by shorter bit sequences, while character pairs that occur with low frequency are represented by longer sequences. This structure, in contrast to the classical Huffman algorithm, offers data-specific adaptation rather than a fixed symbol-frequency distribution. Consequently, the compression process becomes both more precise and more effective.

The algorithm delineates two distinct groups for each character: the first group (palette) consists of high-frequency characters, and the second group consists of low-frequency characters. Huffman trees are created separately for each group, and the encoding is performed based on which group the character belongs to. This structure ensures that frequently used characters are encoded efficiently while preventing waste in the encoding of low-frequency characters. Consequently, DEA attains bit-level optimized compression.

A salient feature of DEA is its reliance on a mathematical model that dynamically determines the palette size and character groups. This model determines the optimal bit distribution for each data segment, thereby ensuring the algorithm's compatibility with diverse data structures. The flexibility to work on a segment-based or entire data level makes DEA a strong candidate for both image and text data.

In summary, DEA represents a contemporary coding strategy that transcends conventional fixed coding methodologies, demonstrating a capacity for adaptability to the inherent dynamics of data. This approach facilitates the attainment of substantial compression rates, particularly in datasets characterized by a high prevalence of repetitive patterns. In this regard, the present study explores an innovative algorithm for its applicability to different content types, such as images and text.

Original design for image compression: The DEA was initially developed for the purpose of lossless image compression. It was designed with an innovative structure that aims to overcome the limitations of traditional coding algorithms. This is particularly evident in scenarios involving images with intricate color distributions or dense textures. The efficacy of conventional compression methods is diminished, resulting in substantial performance degradation with respect to both storage and transmission. The DEA addresses this issue by analyzing the transition probabilities between characters and assigning dynamic bit lengths, offering a statistically adaptive coding approach.

The algorithm is predicated on the division of an image into smaller segments that exhibit analogous color and texture characteristics, with these segments then being encoded in a distinct manner. This approach facilitates the compression of each segment in the most optimal manner, based on its unique statistical characteristics. In the design of DEA, each pixel value is converted to an ASCII character, and the frequency relationships between characters are analyzed. Subsequently, two distinct Huffman trees are generated based on these relationships. This configuration facilitates the generation of abbreviated codes for frequently occurring characters and streamlined, extended codes for infrequent characters. Consequently, DEA provides a lossless solution that maintains both a high CR and data integrity.

Byte-level implementation details: The byte-level applicability of the DEA algorithm has been meticulously engineered to enhance the CR and optimize processing time. In practice, the RGB components (red, green, blue) of each pixel are processed separately. As these values are in the range of 0-255, they are directly represented by 8 bits (1 byte). These values are then converted to ASCII characters to initiate the algorithm's character frequency analysis process. Consequently, each color component is evaluated as an independent input sequence for character-based statistical analysis.

The algorithm calculates transition frequencies on these byte-level character sequences to determine the frequency of subsequent characters for each character. According to the resulting transition matrices, two distinct character sets (group 1 and group 2) are delineated for each character. Group 1 comprises characters that are frequently reiterated, while group 2 encompasses characters that are less frequently encountered. Huffman trees are created for each group, enabling shorter bit sequences to be assigned to frequently used characters, while optimized but longer bit sequences are used for rare characters. The flowchart presented in **Figure 2** provides a detailed visual representation of the steps summarized.

This structure at the byte level enables the algorithm to adapt dynamically to variations in image data, even when processing on a segment basis and encountering different contents in each segment. Furthermore, during the encoding process, the initial character is recorded directly with an 8-bit ASCII code, while subsequent characters are encoded at the byte level using the relevant Huffman trees. Characters that

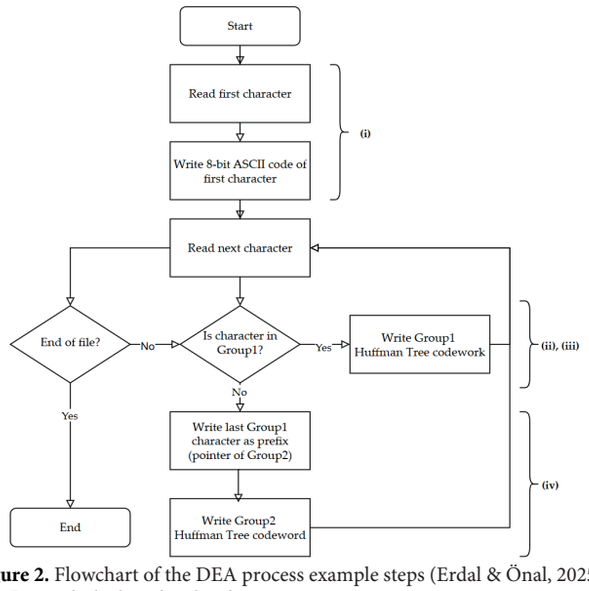

Figure 2. Flowchart of the DEA process example steps (Erdal & Önal, 2025)
DEA: Dynamic bit-level encoding algorithm

do not belong to group 1 are directed to group 2 through a specialized "flag code" and encoded via the Huffman tree. The byte-level modular structure of the system renders both the decoding process and the reconstruction of compressed data highly efficient and lossless.

The DEA algorithm's modular, dynamic, and data-type-independent structure renders it suitable for utilization as an encoding component within the core of any desired compression algorithm. Its capacity to execute two-level Huffman encoding, predicated on inter-character transition frequencies, facilitates DEA's expeditious adaptation to both fixed-structure and statistically variable data sets. Its byte-level operation enables direct application to text, images, biomedical signals, and other data types, facilitating seamless integration into compression frameworks. This feature enables DEA to function not only as a compression algorithm but also as a flexible encoding infrastructure that can be integrated into various applications.

EXPERIMENTAL SETUP

Datasets Used

To evaluate the proposed algorithm with greater accuracy and comprehensiveness, nine distinct data sets were utilized, exhibiting variation in terms of content and size. These datasets encompass a variety of data types, including structured, semi-structured, and unstructured formats, comprising diverse data structures such as plain text, SQL queries, and HTML documents. The nomenclature and dimensions (in bytes) of the datasets utilized in the experiments are enumerated in Table 1.

Table 1. Dataset details

Data set number	Data set name	Size (Byte)	Character type
1	Alice	148.481	UTF-8
2	Genome	5.244.846	UTF-8
3	Pizza Chill English 50	52.428.799	ISO-8891-1
4	Pizza Chill English 100	104.851.942	ISO-8899-1
5	enwik9	100.000.000	ISO-8859-1
6	SQL Query	22.210.031	ISO-8859-9
7	SQL Query 2	9.651.518	ISO-8859-9
8	SQL Query 3	11.647.877	ISO-8859-9
9	HTML	12.993	UTF-8

The type and characteristics of the data contained in the data sets directly affect compression performance. Table 2 presents a cross-section of each data set.

Table 2. Data sample

Data set number	Data sample
1	Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to
2	>NC_007946.1 *Escherichia coli* UTI89, complete sequenceAGCTTTTCATTCTGACTGCAACGGGCAATATGTC TCTGTGTGGAT
3	[Redactor's note: This document uses the ISO 8859-1 Latin1 character set (Windows). The book is composed
4	[Redactor's note: This document uses the ISO 8859-1 Latin1 character set (Windows). The book is composed
5	<mediawiki xmlns="http://www.mediawiki.org/xml/export-0.3/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6	SQL Query: -- Tablo yapısı: `products` CREATE TABLE `products` (`id` int(11) NOT NULL, `name` varchar(255)
7	SQL Query2: CREATE TABLE `products` (`id` int(11) NOT NULL, `name` varchar(255) NOT NULL, `description` text
8	SQL Query3: CREATE TABLE `products` (`id` int(11) NOT NULL AUTO_INCREMENT, `name` varchar(255) NOT NULL, `description`
9	HTML: <!DOCTYPE html> <html lang="tr"> <head> <meta charset="UTF-8"> <meta name="viewport" content

Evaluation Metrics

In order to evaluate the performance of the algorithms applied to the data more clearly and to perform comparative analyses, some criteria that have been widely accepted in the literature have been used. In this context, the CR and SS metrics have been preferred, especially for the quantitative expression of compression performance. The utilization of both metrics facilitates the objective interpretation of the results obtained and enables comparative evaluation between disparate algorithms.

The CR is a metric of paramount importance in the evaluation of the efficacy of a compression algorithm. This ratio is a quantitative metric that quantifies the extent to which the compressed data has been reduced in size compared to the original (uncompressed) data. In general, an elevated CR is indicative of the algorithm's capacity to achieve enhanced data savings and optimize operational efficiency. This is particularly salient in applications that process voluminous data sets, where the CR exerts a pivotal function in reducing storage expenditures and accelerating data transmission times.

The CR is calculated using the following mathematical formula:

$$\text{Compression ratio} = \frac{\text{Original data size}}{\text{Compressed data size}}$$

According to the formula, if a 1000 KB file is compressed to 250 KB, the CR is 4.0. This indicates that the data has been reduced to a quarter of its original size. However, it should be noted that the CR does not always directly indicate the superiority of the algorithm. It is imperative to consider additional factors, such as compression time, algorithm complexity, and data type, to ensure a comprehensive evaluation.

SS is a critical metric that quantifies the amount of space saved on data that has undergone compression as a percentage of the original size. This metric, which is closely related to the CR, provides a more intuitive indication of the extent to which the data has been reduced after compression. This approach is frequently favored, particularly in systems with constrained storage capacity, as it provides a direct illustration of the extent to which the algorithm successfully reduces data storage requirements. The decision to express the results in percentage format was made with the intention of facilitating a more straightforward comparison of the performance of different algorithms.

SS is calculated using the following formula:

$$\text{Space saving} = \left(\frac{\text{Original data size} - \text{Compressed data size}}{\text{Original data size}}\right) x100$$

This formula demonstrates the amount of data that has been compressed as a percentage. To illustrate, when a 1000 kilobyte (KB) file is compressed to 300 KB, it is understood that a 70% SS has been achieved. This metric not only completes the CR but also renders compression performance more intelligible by clearly demonstrating to the user the extent to which data has been eliminated.

This study focused exclusively on compression efficiency metrics (CR and SS) to assess the cross-domain applicability of DEA. Computational complexity and runtime performance were not within the current scope but represent important areas for future research.

Implementation Environment and Tools

All development processes, simulations, and performance evaluations of the algorithm proposed in this study were carried out on a laptop computer with an Intel Core i7-4720HQ processor, 16 GB RAM, and 256 GB SSD hardware specifications. The efficacy of the algorithm was demonstrated through its ability to function seamlessly without inducing bottlenecks in processor power and memory usage during tests conducted on substantial datasets.

The NetBeans IDE 22 development environment was utilized during the software development process, and the algorithms were written in the Java programming language. Specifically, Java Development Kit (JDK) 17.0.12 and Java SE Runtime Environment 17.0.12+8-LTS-286 versions were preferred. The Java language facilitated the development of the algorithm in a modular, portable, and scalable manner, thanks to its advantages, including platform independence, automatic memory management, and extensive library support. The modules pertaining to image segmentation, DEA, and the calculation of evaluation metrics were executed in an integrated manner within this environment.

Compression Performance Comparison

The original sizes of the nine distinct data sets utilized in the study, along with the sizes obtained after implementing four distinct coding algorithms-Huffman, DEA, LZ77, and LZ78-to these data, are presented in **Table 3** for comparison.

Four distinct compression algorithms were implemented on nine distinct data sets utilized in the study, and CR values were derived for each. The results obtained are presented in **Table 4** and provide meaningful findings for comparing the overall performance of the algorithms. After the calculation of the CR values separately for each dataset, the average CR values for the algorithms were also determined. In this context, the DEA algorithm exhibited the optimal compression performance, with an average CR value of 2.72. Conversely, the Huffman algorithm attained an average CR value of 1.87, the LZ77 algorithm 1.67, and the LZ78 algorithm 1.73.

Table 3. Data set data sizes

Data set number	Original (Byte)	Huffman encoding (Byte)	DEA (Byte)	LZ77 Byte	LZ78 Byte
1	152.089	87.687	65.916	135.824	174.546
2	5.244.846	1.487.301	1.459.943	3.587.444	4.097.436
3	52.428.800	29.908.512	23.779.013	50.583.488	56.939.454
4	104.857.600	60.171.023	47.856.491	101.142.020	113.803.264
5	1.000.000.000	644.617.698	488.849.555	840.867.648	1.071.890.988
6	22.440.068	14.747.840	6.543.813	6.413.192	4.056.114
7	10.020.370	6.136.903	3.072.309	3.979.624	3.707.214
8	12.047.167	7.648.161	4.346.789	6.662.588	7.028.568
9	13.974	7.787	5.151	10.076	20.016

DEA: Dynamic bit-level encoding algorithm

Table 4. Compression ratio and average CR value

Data set number	Huffman encoding	DEA	LZ77	LZ78
1	1.73	2.31	1.12	0.87
2	3.53	3.59	1.46	1.28
3	1.75	2.20	1.04	0.92
4	1.74	2.19	1.04	0.92
5	1.55	2.05	1.19	0.93
6	1.52	3.43	3.50	5.53
7	1.63	3.26	2.52	2.70
8	1.58	2.77	1.81	1.71
9	1.79	2.71	1.39	0.70
Average	1.87	2.72	1.67	1.73

CR: Compression ratio

The findings indicate that DEA provides superior CRs in comparison to conventional Huffman and dictionary-based algorithms. This enhancement is attributed to the dynamic nature of DEA's structure, which incorporates the probabilities associated with transitions between characters. This is particularly evident when considering the diversity of data sets, as DEA's statistically sensitive structure renders it more efficient for different content types. In this context, the DEA algorithm has proven to be a significant alternative in terms of compression efficiency, both in theoretical models and in practical applications.

SS values calculated to evaluate the effectiveness of compression algorithms show the percentage of SSs achieved by the applied methods compared to the original data. Pursuant to the experiments conducted, an examination of the mean SS values obtained from nine distinct data sets revealed that the DEA algorithm exhibited the optimal efficiency, yielding a 62% reduction in costs. After this, the Huffman algorithm yielded an average savings value of 43%, the LZ78 algorithm yielded 14%, and the LZ77 algorithm yielded 30% as shown in **Table 5**.

Table 5. Space saving values and average SD values

Data set number	Huffman encoding	DEA	LZ77	LZ78
1	0.42	0.57	0.11	-0.15
2	0.72	0.72	0.32	0.22
3	0.43	0.55	0.04	-0.09
4	0.43	0.54	0.04	-0.09
5	0.36	0.51	0.16	-0.07
6	0.34	0.71	0.71	0.82
7	0.39	0.69	0.60	0.63
8	0.37	0.64	0.45	0.42
9	0.44	0.63	0.28	-0.43
Average	0.43	0.62	0.30	0.14

SD: Standard deviation, DEA: Dynamic bit-level encoding algorithm

The findings indicate that DEA can more efficiently eliminate semantic redundancy within data by dynamically analyzing repetitive patterns and sequential character transitions. Specifically, the 19% increase in efficiency over the classical Huffman algorithm underscores the notion that DEA is not merely a theoretical strength but a practical alternative for data compression purposes. The observation that the LZ77 and LZ78 algorithms yield lower SS values suggests that the dictionary-based structures of these methods are constrained in certain data types and are not universally capable of achieving optimal compression. Overall, the DEA's capacity to yield substantial SSs across diverse data types renders it a versatile and effective compression method.

Statistical Analysis

In this study, the proposed DEA algorithm was analyzed in comparison with the classic Huffman, LZ77, and LZ78 algorithms to objectively evaluate compression performance. Each algorithm was implemented on nine distinct data sets, and the outcomes were assessed using fundamental performance metrics such as CR and SS. The numerical results obtained are presented in tabular form, followed by a general comparison of performance based on arithmetic means. Statistical analyses indicate that DEA provides a significant advantage over other algorithms in terms of both CR and SS values.

Specifically, the DEA algorithm demonstrated superior performance in comparison to all competing algorithms, exhibiting an average CR of 2.72 and an SS value of 62%. The values were calculated as 1.87 CR and 43% SS for the classic Huffman algorithm, 1.67 CR and 30% SS for LZ77, and 1.73 CR and only 14% SS for LZ78. The consistent observation of these differences across all datasets demonstrates that DEA delivers stable performance not only in selected cases but generally across different data structures. The findings indicate that DEA's success in modeling inter-character sequential relationships directly contributes to compression efficiency and offers a statistically significant advantage. This finding suggests that DEA can be regarded as an effective coding infrastructure for both academic and industrial applications.

DEA offers several notable advantages. First, it is an advanced encoding technique capable of efficiently representing repetitive patterns within data, leading to improved compression performance. A key strength of DEA is its data-type independence, which allows it to operate effectively across diverse datasets without requiring significant structural adjustments. Additionally, when combined with appropriate preprocessing steps-such as normalization or tokenization-DEA can deliver even better results by reducing data variability before encoding. These characteristics make DEA a highly versatile component that can serve as the core of a compression pipeline, ensuring both flexibility and adaptability in various application scenarios. However, its implementation complexity and potential overhead in dictionary management should be considered when integrating DEA into large-scale or resource-constrained systems.

Limitations

The proposed DEA is distinguished by its flexible structure and dynamic encoding approach, which can be applied to different data types. A notable strength of this approach is its ability to generate a data-specific two-level Huffman tree through a systematic analysis of the frequencies of successive characters for each data element. This configuration facilitates high CRs by efficiently diminishing data density in data sets characterized by elevated repetition rates. Furthermore, its capacity to function at the byte level facilitates seamless adaptation to image, text, and other numerical data types. When integrated with segmentation, the encoding of each segment according to its own statistical structure represents another advantage that enhances the algorithm's adaptability and compression performance.

However, it is important to note that DEA does possess certain limitations. Initially, the implementation of discrete frequency analyses and Huffman trees for each character has the potential to prolong processing duration and augment the algorithm's computational intricacy. This phenomenon is particularly evident in datasets characterized by low repetition or high diversity. Consequently, the compression gain provided by the algorithm may be constrained. Furthermore, the group separation and flag encoding mechanism necessitates precise parameter adjustments, which may require additional adaptations to ensure optimal performance in different applications. Consequently, strategic choices based on data type and distribution are imperative when utilizing DEA.

CONCLUSION

In this study, the performance of the DEA, which was originally developed for image compression, was systematically examined in the field of text compression. A series of comparative experiments were conducted on nine distinct datasets, employing the classical Huffman, LZ77, and LZ78 algorithms. The effectiveness of these algorithms was gauged by two key metrics: CR and SS. The findings indicated that DEA exhibited statistically significant superiority over all competing algorithms, with an average CR of 2.72 and an SS value of 62%. These findings suggest that DEA can provide stable and high efficiency not only on specific data sets but also on different data types. The findings indicate that DEA is an algorithm with considerable promise for practical applications. In particular, DEA can provide significant advantages in areas that require low latency, high data volumes, and lossless data transmission. The high CR it provides in areas such as text-based archiving systems, log management, messaging services, and real-time data transmission can contribute to

both reducing storage costs and optimizing bandwidth usage. Furthermore, the algorithm's dynamic structure facilitates rapid adaptation to diverse data types, rendering it applicable in heterogeneous data processing environments. While DEA currently demonstrates robust performance, there are numerous opportunities for further research to enhance the efficacy of the algorithm. First, the algorithm's memory management strategies can be optimized to ensure efficient operation even on systems with low hardware capacity. Furthermore, the development of parallel processing versions will be a significant advancement, particularly for the real-time processing of large-scale data sets. A comprehensive analysis of prevailing compression algorithms will be conducted to identify opportunities for the integration of DEA coding components within the algorithmic framework. This will result in the formulation of a hybrid compression approach that is specific to DEA. The outcomes obtained from this approach will be evaluated in comparison with existing methods. An integration study will demonstrate how DEA can be positioned within the existing algorithm ecosystem and expand its potential areas of application. The Future Work section will move beyond generic statements and include specific, actionable directions such as extending DEA for large-scale text datasets, exploring its adaptability to big data environments, and examining its integration into distributed or cloud-based systems. Furthermore, an integration study will demonstrate how DEA can be positioned within the existing algorithm ecosystem and expand its potential areas of application. The presentation quality of tables and figures will be enhanced by ensuring consistent captions and seamless references within the text, and the reference list will be revised to correct formatting and special character issues in line with the journal's style guide. Finally, the development of an open-source DEA library will contribute to the broader adoption of the algorithm in both academic and industrial domains. This initiative will allow researchers to test DEA in diverse scenarios and enable continuous improvement of the algorithm through feedback from a wide user base.

ETHICAL DECLARATIONS

Ethics Committee Approval

Ethical approval was not required for this study as it does not involve human participants, animal subjects, or identifiable personal data.

Peer Review Process

This manuscript was subject to external peer review.

Conflict of Interest

The authors declare no conflicts of interest related to this study.

Financial Disclosure

This work is supported by the Kırıkkale University Department of Scientific Research Projects (2025/034).

Author Contributions

Concept: EE, AE, AÖ; Design: EE, AE, AÖ; Control: EE, AE, AÖ; Resources: EE, AE, AÖ; Materials: EE, AE, AÖ; Data Collection and/or Processing: EE, AE, AÖ; Analysis and/or Interpretation: EE, AE, AÖ; Literature Review: EE, AE, AÖ; Writing the Article: EE, AE, AÖ; Critical Review: EE, AE, AÖ.

Acknowledgments

The author acknowledges the assistance of the DeepL tool, which was used for improving the clarity and fluency of the English in this manuscript.

REFERENCES

Beemkumar, N., Riyat, S., & Kumar, D. (2024). An Investigation of Lossless and Lossy Data Compression & Source Coding. 2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT), 1-6. https://doi.org/10.1109/ICCCNT61001.2024.10725458

Dhulavvagol, P. M., Gadagkar, A., Kj, A., Hegade, G., Poonia, R., & Totad, S. G. (2024). Lossless text compression using recurrent neural networks. *Procedia Computer Science, 235*, 3340-3349. https://doi.org/10.1016/j.procs.2024.04.315

Erdal, E., & Önal, A. (2025). Enhanced framework for lossless image compression using image segmentation and a novel dynamic bit-level encoding algorithm. *Applied Sciences, 15*(6), 2964. https://doi.org/10.3390/app15062964

Gastón, B., Pujol, J., & Villanueva, M. (2013). A realistic distributed storage system that minimizes data storage and repair bandwidth (Versiyon 1). *arXiv.* https://doi.org/10.48550/ARXIV.1301.1549

Gopinath, A., & Ravisankar, M. (2020). Comparison of lossless data compression techniques. *2020 International Conference on Inventive Computation Technologies (ICICT),* 628-633. https://doi.org/10.1109/ICICT48043.2020.9112516

Gupta, A., Bansal, A., & Khanduja, V. (2017). Modern lossless compression techniques: review, comparison and analysis. *2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT),* 1-8. https://doi.org/10.1109/ICECCT.2017.8117850

Hoffman, R. (1997). Compression Algorithms for Symbolic Data. In; R. Hoffman, Data Compression in Digital Systems (ss. 53-74). Springer US. https://doi.org/10.1007/978-1-4615-6031-9_4

Huffman, D. (1952). A method for the construction of minimum-redundancy codes. *Proceedings of the IRE, 40*(9), 1098-1101. https://doi.org/10.1109/JRPROC.1952.273898

KeskiN, S., Sevli, O., & Okatan, E. (2023). Single and binary performance comparison of data compression algorithms for text files. *Bitlis Eren Üniversitesi Fen Bilimleri Dergisi, 12*(3), 783-796. https://doi.org/10.17798/bitlisfen.1301546

Welch. (1984). A technique for high-performance data compression. *Computer, 17*(6), 8-19. https://doi.org/10.1109/MC.1984.1659158

Ziv, J., & Lempel, A. (1977). A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory, 23*(3), 337-343. https://doi.org/10.1109/TIT.1977.1055714

Ziv, J., & Lempel, A. (1978). Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory, 24*(5), 530-536. https://doi.org/10.1109/TIT.1978.1055934