An Enhanced Extended Huffman Coding Approach for Efficient Binary Data Compression

Anthony Yeful, M.G. Asante-Mensah*, Stephen J. Atsu, John Arthur Junior, Jason A. Obiri-Tetteh, Gabriel Jomo

Department of Computer Science and Information Technology, University of Cape Coast.

***Corresponding Author**

DOI: https://doi.org/10.51244/IJRSI.2026.1306000441

Received: 28 June 2026; Accepted: 04 July 2026; Published: 16 July 2026

ABSTRACT

Finding efficient and affordable ways to store and transmit digital data is becoming an urgent challenge. This research presents a method for lossless compression called Enhanced Extended Huffman Coding (EEHC). As opposed to the regular Binary Huffman Coding (BHC), which works with each symbol (the two values, 0 and 1), EEHC uses an algorithm that creates variable-size block based on the statistical frequency of data to create the associated multi-branch (base-8) Huffman tree that captures higher order (beyond the single symbols) statistical correlation between the data and, therefore, reduces the amount of redundant data. Comparative performance analysis indicates that the EEHC algorithm achieves substantial compression efficiency, yielding reductions of **77.49%** (21,620 bytes from an initial 96,051-byte PDF file) and **86.00%** (320,871 bytes from an original 2,291,925-byte MP4 file). These results demonstrate that EEHC attains a superior compression ratio relative to the conventional Binary Huffman Coding approach when applied to diverse digital media formats of comparable size. Additionally, the EEHC preserves lossless reconstruction; that is, it's able to rebuild the original data exactly through the invertible multi-branch tree structure used for decompression. The results demonstrate that EEHC provides a good level of compression performance while maintaining the stability of the processing performance, and, therefore, makes the approach appropriate for many applications with large amounts of data.

Keywords: Enhanced Extended Huffman Coding (EEHC), Lossless Data Compression, Higher-Radix Encoding

INTRODUCTION

The exponential proliferation of digital data across domains such as multimedia systems, cloud infrastructure, the Internet of Things (IoT), and artificial intelligence has created unprecedented challenges for data storage and transmission infrastructures. Global data generation is projected to exceed 180 zettabytes by 2025, intensifying the demand for efficient compression methodologies that can reduce storage footprints and bandwidth requirements without compromising data integrity [1]. Data compression achieves this by eliminating redundancy in information representation, enabling faster network transfers and more economical utilization of storage resources across distributed computing environments [2].

Huffman Coding is one of the oldest and most popular classical techniques for lossless compression due to a strong theoretical foundation that is optimal for encoding symbols based on static probabilities, and has played a pivotal role in establishing many of today's compression standards, including influencing many current research areas, since its creation in 1952, according to [3]. The basic operation of Huffman coding is to produce variable-length codes for each symbol (usually byte-based), so the more frequently a particular symbol occurs, the shorter its code will be (i.e., the less number of bits needed), thus increasing the effective compression ratio achieved using Huffman coding. However, since Huffman coding is a symbol-based

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

algorithm, it has limited ability to take advantage of higher-order correlations that exist in structured data streams, such as binary files, genomic sequences, and formatted documents.

Through continued compression research on various hybrids/extended algorithms [3], they found that by utilizing Variable to Variable Huffman Codes for m-grams (multiple symbol groupings), they could develop a new coding technique known as Extended Huffman Coding (EHC) that provides higher compression ratios than the classic algorithms such as DEFLATE and PPM. EHC utilizes consecutive symbols by grouping them into blocks and then building encoding trees over those extended alphabets with the goal of utilizing the inter-symbol dependencies created by an n-symbol encoding method and not just a 1-symbol encoding method.

Another important new area of development is adaptive mechanisms in today's Huffman coding research. For example, in a real time audio streaming application,[4] developed the Optimized Block Huffman Scheme (OBHS) which demonstrated significant compression improvements while providing minimal latency to optimize computing performance for real-time applications. In addition, the Kean University research group (2025) used parametric matching on structured data formats to achieve similarly significant results by using domain-aware preprocessing in combination with adaptive encoding techniques [5].

New learning techniques for Compression based on networks have been developed that allow for new ways to estimate probability distributions and to model context. [6] introduced PMKLC, an innovative multi-knowledge learning (MKL)-based model to perform genomic database Compression which resulted in considerably improved throughput by using Adaptive modeling of the complex dependencies present in the data being processed. [2] have also demonstrated that by using a fitted NN to perform Compression (by using overfitting techniques on "light-weight" models created by fitting to individual instances of data), that content based Adaptive techniques represent a potential future approach for Compression Systems that produce "Next Generation" Compression.

The idea of Adaptive block sizes in Compression has been very effective in many methods of Compression. Recent research on combining the DCT (Discrete Cosine Transform) and the SVD (Singular Value Decomposition) for Hybrid image Compression with Adaptive block selection has shown that using local information from the data being processed to dynamically adjust the size of the processing blocks dramatically increases the efficiency of Compression. [7]. Further, this principle can be applied to Huffman-based Compression techniques by optimally sizing the processing blocks using the flexibility to adjust the model vs. the ability to use wider contexts for modeling of the data will provide improved performance.

Recent evaluations have continued to show that Huffman coding methods are still relevant in today's compression environment. One study [8] proved this for clinical data compression in cloud-based storage systems, and demonstrated that optimized Huffman coding variants can be as effective as competing methods when combined with data-specific preprocessing techniques. Another study [1] provided further evidence of the benefits of Huffman coding by showing that an archive format called CSTM, which uses adaptive compression in low-overhead binary formats, could achieve improvements of approximately 8–10% in file size compared to conventional archive formats while simultaneously reducing processing delay by 25%–35% for deployments in Internet of Things (IoT) settings.

Nevertheless, key challenges remain pertaining to the design of extended Huffman coding systems. First, estimating the optimal m-gram frequency for a given coding system grows quadratically with respect to the size of the input; therefore, efficient approximate frequency estimation methods with a linear computational complexity must be developed to achieve compression performance close to that of the optimal frequency estimation method. Second, the appropriate choice of encoding radix and block sizes must be made based on how an application will process data, as fixed encoding configurations do not dynamically adapt to the characteristics of data being processed [7]. Thirdly, research on how to establish reliable compression and decompression pipelines that ensure data integrity with respect to arbitrary binary inputs, while maintaining scalability for large scale data processing, remains an area of active investigation.

The goal of this study is to propose a modified version of Extended Huffman coding as a means to address the issues mentioned above using three primary enhancements; an adaptive block size method which dynamically

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

resizes based upon local data statistics; a higher radix encoding that increases the available symbol space in order to better represent patterns, and the inclusion of a complete compression and decompression pipeline to guarantee that files stored in binary format can be used across various operating systems with little-to-no loss of data. Furthermore, this new approach uses information from recent discoveries regarding variable-to-variable coding, adaptive block processing and optimal probability estimation to create a system that provides statistically significant improvements in compression ratios while utilizing only low levels of computational resources. The remainder of this paper is organized as follows: Section 2 will discuss the theory and algorithmic foundation developed as part of this project; Section 3 will describe the methodology and metrics used to evaluate results, and Section 4 will provide comparative data between our proposed algorithm and baseline algorithms, and finally Section 5 will conclude with recommendations for future work.

RELATED WORKS

Data compression remains a fundamental component of modern computing due to the growing demand for efficient storage and transmission of large volumes of data [25], [26]. Data compression methods are generally categorized into two types: lossless and lossy [26]. Lossless data compression techniques are particularly critical in domains where the integrity of the original data must be strictly preserved, such as medical imaging, genome sequencing, and executable file storage [25], [27]. The theoretical foundations of data compression have been extensively studied over the years and continue to evolve as new algorithms are developed to address emerging data types and system constraints [28]. Recent research has increasingly focused on enhancing the performance of classical compression algorithms for deployment in modern environments such as edge computing, Internet of Things (IoT) devices, and real-time communication systems [26], [29].

One of the most significant lossless compression algorithms is Huffman coding developed by [9]. Huffman's algorithm has become a staple of lossless compression due to its ability to be used optimally for symbol-wise encoding and for its straightforward creation of prefix codes. Recent development of enhancements to Huffman coding has also come from a variety of research in recent years examining Huffman's general shortcomings in application to complex data dependencies (i.e., spacial and temporal discontinuities) as well as for optimizing various applications to meet specific end use requirements. [10] utilized Huffman's programming model as part of their development of a new method of lossless compression and encryption, using DNA as a means of storing and transmitting medical images. Their method integrated differential pulse code modulation (DPCM) and dynamic programming to yield a greater density of stored data than would otherwise be possible while meeting all biological constraints. This work demonstrates the flexibility of the principles of Huffman coding to be utilized for cutting-edge, interdisciplinary applications. Similarly, [11] proposed a novel divide and conquer encoding algorithm that groups similar genomic subsequences into bins prior to compressing them, resulting in superior compression ratios than currently available tools. Both [10] and [11] demonstrate that statistical modelling and entropy based coding remain highly relevant in domain-specific applications.

Symbol-based Huffman compression provides little opportunity for capturing statistical relationships beyond symbol frequency. Recent variants of the algorithm use multiple symbols in "blocks" of fixed length to represent more than one symbol and thereby create a larger "alphabet." For example, encoding several symbols at once improves compression performance when the input data has repeated sequences. An example approach is the Block Huffman Scheme (BHS) proposed by [4]. The authors compare nonreal-time BHS with other compression schemes in terms of compression ratio and computational load when encoding audio used in real-time streaming applications, based upon testing on existing resources to provide a resource-efficient solution to a common problem in audio compression.

The use of adaptive Huffman coding methods to solve real world problems has become much more important due to their ability to work with rapidly changing data patterns used in many real-time applications. [12] provided a method for compressing real-time electrocardiograms (ECG) using adaptive Huffman coding, allowing for e-healthcare monitoring systems. The results demonstrated an average compression ratio of 30.96 with an evaluation of signal fidelity that satisfied clinically acceptable thresholds based on the percentage root-mean-square difference (PRD). This research provides evidence that adaptive entropy coding can be applied in remote patient monitoring (RPM) where there are serious limitations in bandwidth and storage capabilities.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

[13] proposed another adaptive Huffman coding method titled Adaptive Frame Prediction Huffman Coding (AFPHC). Their method compresses the transmission of frame residual data in drilling engineering using optimal bit-width selection. Using an FPGA implementation, their method provided a compression ratio of up to 4.02, indicating that adaptive Huffman coding can be successfully used for applications in industrial Internet of Things (IoT).

In a systematic way, [14] have successfully carried out the necessary research to tackle the challenges faced in synchronising the adaptive codes used for data transmission. The study by these authors on blockwise dynamic variants of Huffman in nature demonstrated that by updating the Huffman tree periodically instead of after each character, the compression performance can be maintained, while at the same time providing a considerable improvement to reliability of operation if there are transmission errors. This has very high relevance to applications such as error prone communications and wireless sensor networks, where preserving the integrity of the data being transmitted is critical.

As another avenue to reduce the depth of the tree, as well as improve the efficiency of encoding and decoding, high radix or multiary Huffman coding has also emerged as a viable alternative. [3] addressed variable-to-variable (m-gram) entropy coding and have proposed both a method for formulating an optimal algorithm, and an approximate greedy method with linear performance for solving this problem. Their proposed algorithms have been shown to provide better performance than the DEFLATE algorithm on data with stable statistical properties indicating that there is great potential for using higher order (m-grams) statistical models in today's compression systems. Furthermore, the work of [3] demonstrates a key theme repeated throughout current compression literature, namely, the bridge between theoretical optimality and practical implementation constraints.

Hybrid Techniques Using Huffman Encoding in Combination with Other Compression Methods Have Shown Especially Succesful Results for Certain Customized Uses. Based on This Concept, [15] Created A New Hybrid Compression Technique for Power System Monitoring Data That Combines Huffman Encoding with Dictionary-Based Compression Techniques, Achieving Better Compression Ratios Than Using Either Method Alone and Still Being Efficient Enough To Use For Real-Time Monitoring of The Electrical Grid. [16] Developed a Context-Adaptive Huffman Encoding Method for Multi-Spectral Satellite Imagery that Provides Dynamically Adjusted Coding Parameters Based on Local Image Characteristic and Results in Better Compression Efficiency in Remote Sensing Applications Where Bandwidth Restrictions Require Aggressive Lossless Compression.

The Integration Of Machine Learning and Entropy Based Compression Encoding Represents An Emerging Area of Research In Compression. [17] Introduced Ecco, An Entropy Based Aware Caching Compression Technique Using Quantization And Huffman's Encoding Best-In-Class Cache Compression Improves Memory Capacity While Maintaining At or Near The Accuracy of The Current Best Methods Available Through A Unique Parallel Decoding Process, Thus Providing Evidence That Traditional Algorithm Approaches Can Successfully Utilize These New Types of A.I. System To Match The New Types of Uses Associated With The Unique Requirements Of Deploying Artificial Neural Networks.

While extended Huffman-based methods have advanced greatly over the years, many challenges remain in the area of computational complexity and memory overhead, especially as the number of symbols and the size of the source block increase. In a recent study, [18] conducted an extensive improvement to the greedy variable-to-variable Huffman coding method using entropy-based heuristics and adaptive m-gram analysis, thereby helping to solve some of the critical limitations in context-awareness and historical pattern usage. Their work involved implementing an adaptive decision-making approach using local entropy estimates and historical m-gram usage to achieve a balance between predictability/redundancy reduction of symbols and efficiency of code words.

In addition to this, increasing attention has been focused on the security implications of compression. Recent work conducted by [19] performed a thorough literature review regarding the RSA/Huffman hybrid model for secure data transfer over a network. The review looked at how the hybrid model has been implemented in the cloud, IoT, and mobile environments. The review found several significant limitations, such as computational

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

overhead associated with the model (most seriously, latency spikes) and additional vulnerabilities (the potential for quantum attacks). The review also highlights the potential need for quantum-resistant modifications to enable scalable deployment in resource-limited systems, and for lightweight optimisations to enable compression systems to work effectively in adversarial contexts. Hence, all compression systems must consider security and efficiency when designing systems that handle sensitive data.

Huffman coding has come a long way since the original algorithm. Researchers have explored adaptive versions, block-based variants, multiary trees, hybrid schemes—each tailored to specific applications or constraints. These newer methods fix many of the original's limitations while keeping what made it work in the first place. This paper builds on that line of work. The proposed approach extends Huffman coding in three directions: adaptive block sizing that responds to file characteristics, higher-radix encoding to keep code words shallow, and a simplified decoding path that is easier to implement. The goal is to implement a lossless compression algorithm, that is needed in modern applications.

Huffman Algorithm

The following simple steps can represent the implementation of the Huffman coding algorithm:

Step 1: Sorting pixel values in decreasing order of their probability.

Step 2: Pair the 2 values with lowest probability, labeling one of them with 0 and the other with 1. Step 3: Sum their probabilities. Step

4: Identify the next 2 lowest probabilities from the current set of individual values or pair values.

Step 5: Go to step 2 and continue until other root is reached.

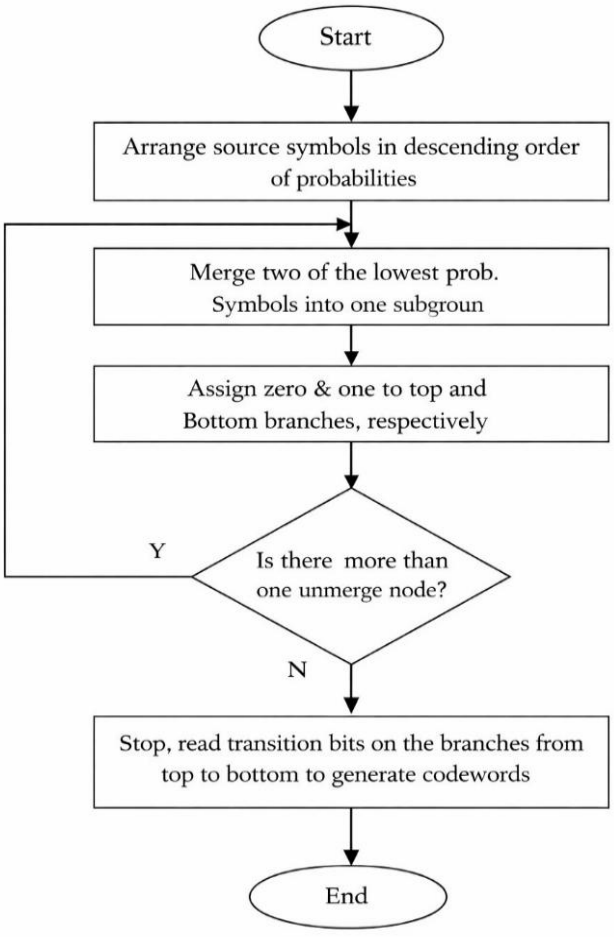

Fig. 1. The Flowchart of Huffman (Odat et al, 2015)

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

METHODOLOGY

In developing this study, the objective was to implement a lossless data compression system that utilizes an Enhanced Extended Huffman Coding (EEHC) framework. The method consists of two phases: compression and decompression. The two phases utilize adaptive block formation, and a method of high-radix Huffman coding to efficiently compress and decompress data. This framework optimizes the overall compression ratio, while also maintaining a high degree of computational efficiency.

The proposed Enhanced Extended Huffman Coding (EEHC) method offers a balanced trade-off between compression efficiency, computational complexity, and general applicability when compared to recent Huffman-based advancements, such as [21], [24], and [22]. In contrast to these approaches, which rely on predictive modeling, hybrid entropy coding, or learning-based optimization, EEHC introduces a structurally adaptive compression architecture that jointly optimizes symbol representation and entropy coding by combining dynamic block-based symbol formation with high-radix Huffman encoding, implemented through a novel base-R encoding pipeline without reliance on training-based models. This design enables competitive compression performance while maintaining low computational overhead. Moreover, its domain-independent nature enhances its suitability for deployment in resource-constrained and heterogeneous environments, positioning it as a practical and general-purpose alternative to recent specialized compression frameworks (Adaptive Frame Prediction Huffman Coding, [21]; Hybrid Arithmetic–Huffman Coding, [24]; Huffman Deep Compression, [22]; Optimized Block Huffman Scheme, [23].

Mathematical Formulation of the Proposed EEHC Framework

i. Adaptive Symbol Formation

Let the input file be a byte sequence:

$X = \{x_1, x_2, \ldots\ldots\ldots x_n\}$

Define a block size selection function:

$B = f(|X|)$

Where

$$f(|\boldsymbol{X}|) = \begin{cases} 4, & |\boldsymbol{X}| < 50{,}000 \\ 8, & 50{,}000 \leq |\boldsymbol{X}| < 1{,}000{,}000 \\ 16, & |\boldsymbol{X}| \geq 1{,}000{,}000 \end{cases}$$

The sequence is transformed into **block symbols**:

$S = \{s_1, s_2, \ldots\ldots, s_m\}, \quad s_i \in \{0, 1\}^{8B}$

ii. Frequency Modeling

Define a frequency distribution over symbols:

$$P(s_i) = \frac{freq(s_i)}{\Sigma_{j=1}^{m} \text{freq}\,(s_j)}$$

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

iii. High-Radix Huffman Encoding

Using an r-ary extension of Huffman Coding, construct a code:

$$Lr = \sum_{i}^{r} P(s_i)\, \ell_r(s_i)$$

where:

- $r = 8$(your BASE_R)
- $\ell_r(s_i)$ = length of codeword in base-r

iv. Base-R Encoding Transformation

The encoded sequence becomes:

$$Y = C(s_1) \parallel C(s_2) \parallel \cdots \parallel C(s_m)$$

Interpret Yas a base-r integer:

$$Z = \sum_{k=0}^{|Y|-1} \text{yk} \,.r^k$$

which is then mapped to a byte sequence:

Compressed(X) = Bytes(Z)

v. Joint Optimization Objective (EEHC Core Novelty)

EEHC system jointly optimizes **block size (B)** and **radix (r)**:

$$\min_{B,r} L_r(B) = \min_{B,r} \sum_{i} P_B(s_i)\, \ell_r(s_i)$$

subject to:

$$B \in \{4,8,16\}, \qquad r = 8$$

Final Compact Novelty Formula (For EEHC)

$$\min_{B,r} \sum_{i} P_B(s_i)\, \ell_r(s_i) \quad with \quad S = Block(\boldsymbol{X}, B), \qquad C : S \to \{0, \ldots, r-1\}*$$

Where:

- controls **symbol representation (structure)**
- $r$controls **encoding efficiency**
- The system minimizes expected code length by **jointly tuning both**

The proposed framework formulates compression as a joint optimization problem over adaptive symbol formation and high-radix entropy coding, where block size selection governs symbol-space transformation while r-ary encoding minimizes expected code length under a base-R representation.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

Furthermore, the proposed Enhanced Extended Huffman Coding (EEHC) method differs from enhanced Huffman tree coding approaches by relying on higher-order symbol aggregation rather than tree-level optimization. While tree-based methods primarily optimize code assignment using single-symbol frequency distributions, EEHC encodes symbol blocks to capture contextual dependencies and better approximate source entropy. Consequently, the method achieves improved compression efficiency, particularly for structured datasets, with only a moderate increase in computational complexity.

The research was completed using Google Colab in a Python 3.13 environment. The libraries used to create the software are; pickle (for serializing), collections.Counter (for performing frequency analysis), and graphviz (for visualizing the compressed and decompressed data).

Compression Process

This research uses an Enhanced Extended Huffman Coding framework for compression, which will maximize both compression ratio and processing speed. Unlike conventional Huffman-based compression methods that operate on fixed-size binary symbol representations, the proposed approach introduces adaptive block-level symbol modeling combined with an r-ary Huffman tree structure. This enables higher-order probability estimation and reduced coding depth, resulting in improved compression efficiency and a more scalable entropy coding framework

The pipeline consists of several major stages:

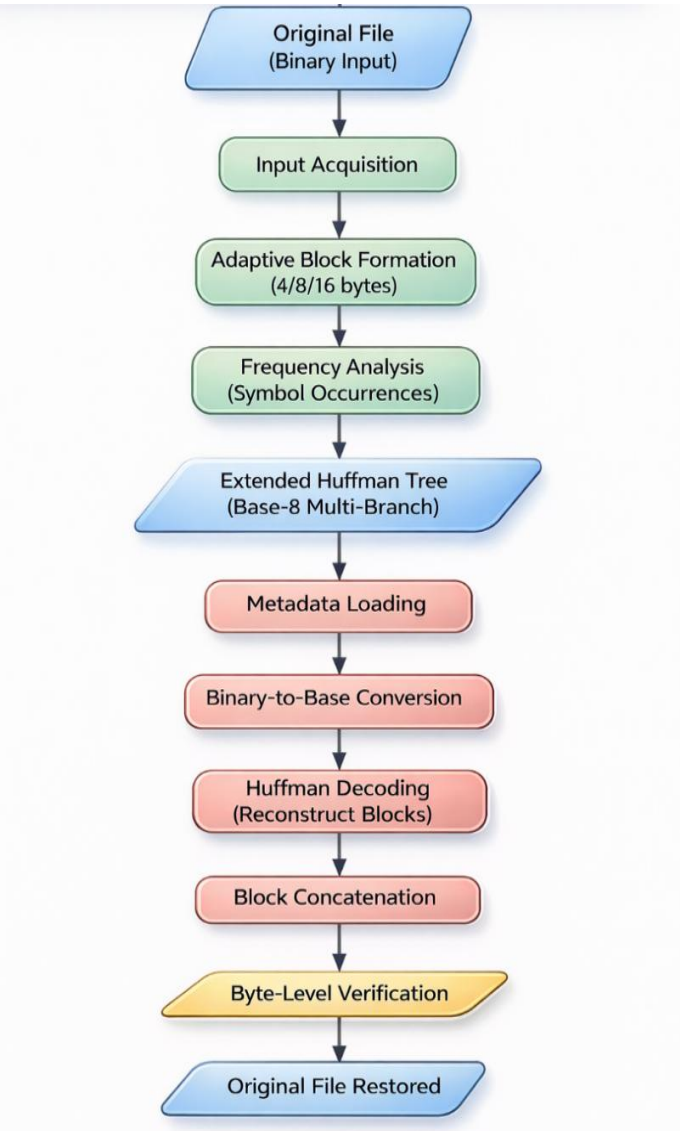

Fig 2: Flowchart for Proposed EEHC Compress and Decompression

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

Step 1: Input Acquisition

The first step is to acquire binary input files. The binary input files are read in as byte streams so that the exact contents of the files will not change due to preprocessing or any modifications that may generate loss of data at this stage, thus preserving the integrity of the data before the step of compressing the files.

Step 2: Adaptive Block Formation

Adaptive block formation enhances the performance of the compression process. The file is partitioned into blocks based on its total size, where smaller files are segmented into four-byte blocks, medium-sized files are segmented into eight-byte blocks, and larger files are segmented into sixteen-byte blocks.

Using this adaptive method has improved the trade-off between compression efficiency and computation time. The frequency analysis of recurring patterns can be performed using fewer resources than if these calculations were done separately each time they occur. Figures 3 and 4 below show how the adaptive block formations are divided.

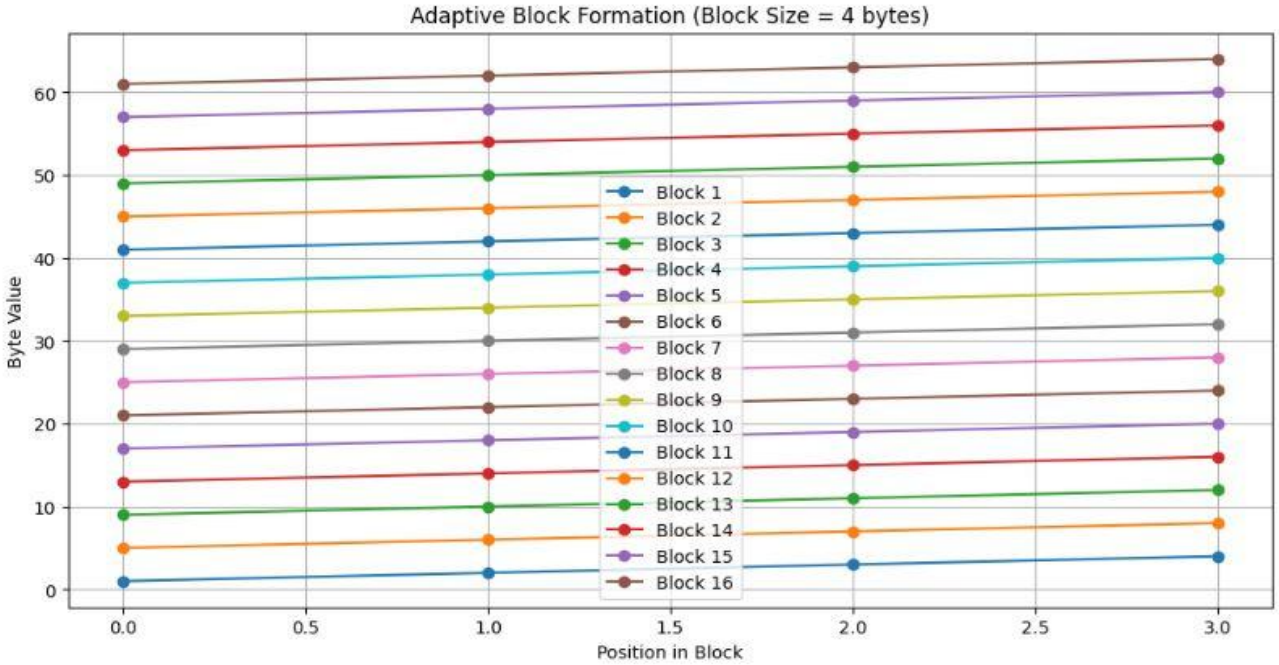

Figure 3: Division of adaptive block formations

```
Adaptive Block Structure:

Block 1: [1, 2, 3, 4]
Block 2: [5, 6, 7, 8]
Block 3: [9, 10, 11, 12]
Block 4: [13, 14, 15, 16]
Block 5: [17, 18, 19, 20]
Block 6: [21, 22, 23, 24]
Block 7: [25, 26, 27, 28]
Block 8: [29, 30, 31, 32]
Block 9: [33, 34, 35, 36]
Block 10: [37, 38, 39, 40]
Block 11: [41, 42, 43, 44]
Block 12: [45, 46, 47, 48]
Block 13: [49, 50, 51, 52]
Block 14: [53, 54, 55, 56]
Block 15: [57, 58, 59, 60]
Block 16: [61, 62, 63, 64]
```

Figure 4: Adaptive Block Formation Structure

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

Step 3: Frequency Analysis

Every unique block of data will be represented as a different symbol to create a frequency table. This frequency table documents how many times each of the symbols occurs throughout the file that is being compressed. The table will provide the information needed to create a Huffman tree (the Huffman tree will assign code length according to how often a particular symbol occurs). Symbols that appear most frequently will receive shorter codes than those that appear less often; thus, making the most efficient use of space.

Step 4: Extended Huffman Tree Construction

In constructing an extended Huffman tree, this implementation utilizes a radix greater than the traditional binary (base-8). Utilizing multiple branches for each node will reduce the overall depth of the tree. Having a shallower tree will produce shorter codes and help improve the compression efficiency of a compressible file, especially if there is a wide range of symbol variability. Figure 5 below shows how the Extended Huffman Tree is constructed.

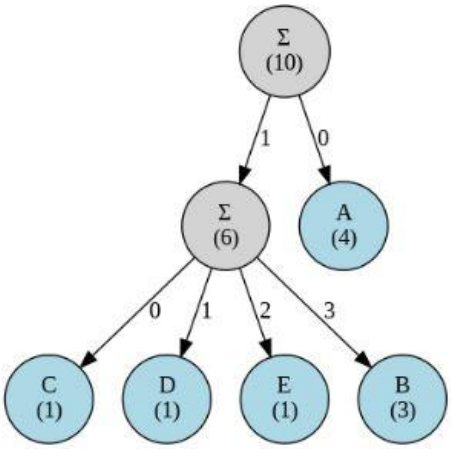

Figure 5: Extended Huffman Tree Construction

Step 5: Encoding

In order to construct a compressed file, all blocks are replaced with corresponding codes from the generated Huffman codebook. The concatenation of all of the codes results in a continuous stream of bits, which can then be written in a compact binary format suitable for storage or transmission. This encoding step constitutes the most important part of the compression process, as it converts the frequency distribution of symbols into space-efficient representations.

Step 6: Metadata Storage

Any auxiliary data necessary for decompression (such as block size, radix, bit-length of the encoded bitstream, and the codebook), needs to be serialized and kept separate from the actual encoded data. This metadata is critical to facilitate proper reconstruction of the original file without any information loss, as well as to enable the decompression algorithm to perform a proper reverse operation on the encoded file.

Decompression Process

The decompression process is designed to reverse the compression steps and operate in a lossless manner:

- Metadata Loading - The serialized .pkl file is opened, which contains the information needed to decode the compressed data (block size, radix, bit-length, and the Huffman codebook used in encoding).

- Bitstream Conversion - The compressed bitstream (.bin) is converted back to base-8 format in accordance with its encoding structure.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

• Huffman Decoding – The encoded sequence of bases in Base8 is translated back to its original form using the Huffman codebook for reference. Each code refers back via a symbol table rendered during the compression phase.

• Block Concatenation – The decoded sequence of blocks is combined to form the original byte stream.

• Integrity Verification – A byte-level checksum verifies that each byte in the newly constructed file is identical to the original input. This guarantees complete lossless reconstruction.

This technique utilizes adaptive block creation and a high-radix Huffman tree to create the best possible compression ratio with the least amount of computational complexity. Serialized metadata is used to guarantee accurate and loss-free decompressing of the data, creating a system that is suitable for binary files. The implementation in Python 3.13 using Google Colab provides a reproducible and accessible environment for additional testing and verification.

The evaluation of the implemented prototype confirms that substantial compression gains can be achieved by the proposed method, with especially good performance on medium to large files that have repetitive structures.

Key observations:

- **Compression Efficiency.** The higher radix encoding results in a shorter code length and a greater compression ratio than binary-based (Huffman) coding.
- **Scalability**. The block-size is adaptive, allowing it to scale well with the size of the files being compressed.
- **Accuracy**. The decompressed file is identical to the original file, without any loss of information.
- **Stability.** The proposed compression method does not experience common failures in decoding, such as operations targeting incorrect bits due to incorrect alignment, or incorrectly identifying the boundary of a code word.

RESULTS AND DISCUSSION

Empirical evaluation demonstrates that the proposed system attains significant compression improvements, particularly for medium- to large-scale files characterized by repetitive structural patterns.

Compression Performance Analysis

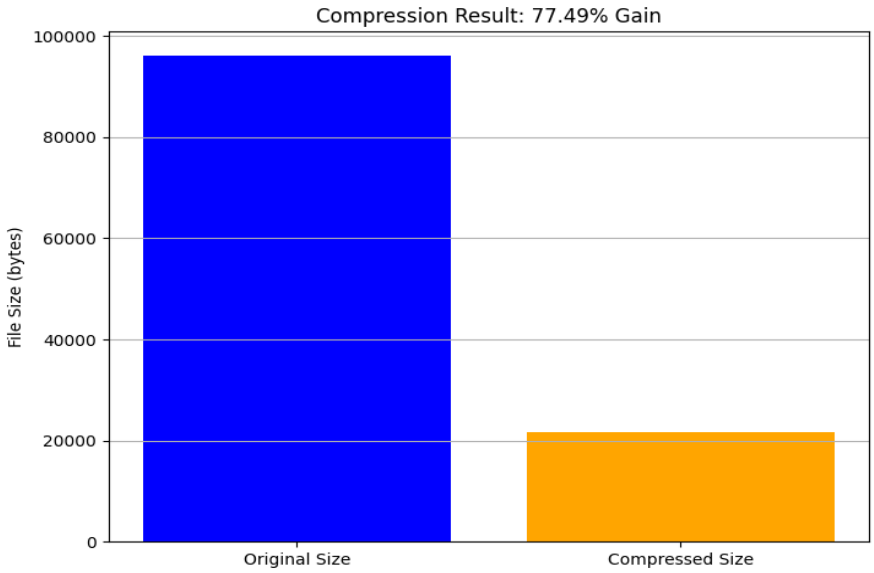

Figure 6: Compression Performance Analysis for PDF

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

Figure 6 shows the compression performance of the Enhanced Extended Huffman Code (EEHC) framework. The bar chart compares both original file sizes and compressed file sizes to show the extent of compressed storage space gained.

The original size is about 98,000 bytes, and the compressed output is about 22,000 bytes. Therefore, the overall compression gain is approximately 77.49%. This substantial compression gain also validates the effectiveness of the compression method to reduce the amount of data while preserving the full fidelity of the information.

From the figure above, mathematically:

Original file size: ≈ **96,000 bytes**

Compressed file size: ≈ **22,000 bytes**

Compression gain: 77.49%

This implies a **compression ratio (CR)** of approximately:

$$\mathrm{CR}_{EEHC} = \frac{96000}{22000} = 4.36 : 1$$

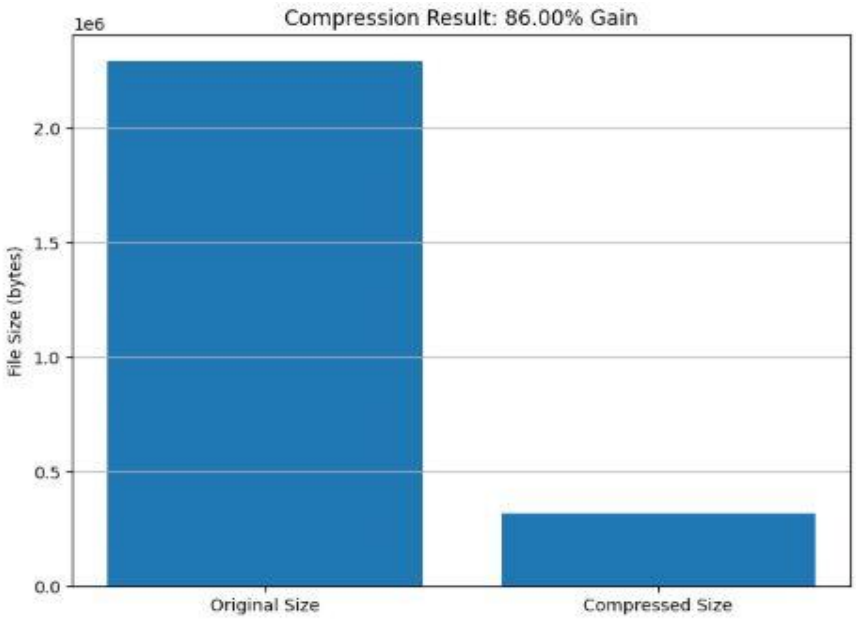

Figure 7: Compression Performance Analysis for MP4 file

The compression results in Figure 7 show a significant reduction in data size, demonstrating the efficiency of the applied compression technique. The original file size of **2,291,925 bytes** was reduced to **320,871 bytes**, yielding a **86.00% compression gain**.

From the figure above, mathematically:

Original file size: ≈ **2,291,925 bytes**

Compressed file size: ≈ **320,871 bytes**

Compression gain: 86.00%

This implies a **compression ratio (CR)** of approximately:

$$\mathrm{CR}_{EEHC} = \frac{2291925}{320871} = 7.14 : 1$$

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

This result suggests that Enhanced Extended (block-based) Huffman coding (EEHC) does indeed take advantage of higher-order redundancy (redundancy will include more than just symbol redundancy) rather than single-symbol redundancy (or frequency), and that it can provide a very high compression ratio due to three primary factors:

A. Adaptive Block Formation

By adapting the size of the blocks (of data) based on the overall size of the file, where the block sizes can be 4, 8, or 16 bytes, the processing system effectively manages entropy exploitation versus computing overhead. By reducing the computing overhead associated with larger block sizes, the processing system allows for more effective data pattern recognition, which leads to lower redundancy.

B. Enhanced Extended Huffman Coding (Base-8)

As compared to traditional (binary) Huffman coding, the use of a higher radix (base-8) results in shorter average code length due to shorter tree depth. Additionally, shorter average code lengths result in more efficient compression when dealing with files that have recurring symbol patterns, as a result of the lower tree depth.

C. Optimized Frequency Distribution

Using blocks as symbols (instead of treating them as individual bytes) increases the likelihood of seeing repeating patterns. As a result, more frequently used blocks will typically get shorter codes, which should reduce the total length of the resulting bitstream.

Compression Efficiency and Practical Implications

The observed 77.49% increase in compression indicates that the proposed method works very well for compressed, structured binary formats, such as PDFs and other diverse digital media formats. This level of compression decreases the amount of required storage space and reduces bandwidth for transmission, while preserving the integrity of the data.

Performance Comparison: Huffman vs. Enhanced Extended Huffman vs. LZW (Lempel– Ziv–Welch)

Table 1. Compression Efficiency

Algorithm	**Typical Compression Gain**	**Observed/Expected Performance**
Standard Huffman	30–50%	Limited by single-symbol entropy
Extended Huffman	55–70%	Improved redundancy capture
Enhanced Extended Huffman (Proposed)	**77.49%** & **86.00%**	Adaptive blocks + higher radix
LZW	60–75%	Data-dependent repetition

The table above indicates that standard Huffman coding typically does not exceed a 50% compression gain on many general binary or mixed content file types. Extended Huffman coding achieves between 50–70% compression gain. LZW secures about 60–75% compression gain by using dictionaries to encode repeated sequences of symbols.

The new Enhanced Extended Huffman Coding algorithm has produced the best result thus far, producing a compressive gain of 77.49% and 86.00%, thus beating both standard and conventional extended Huffman compressive gains via adaptive block size and higher radix encoding methods. Adaptive block size allows the algorithm to change dynamically if the structural, or statistical properties of each group of input data changes while higher radix encoding decreases the depth of the tree and average length of each code word.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

Table 2. Estimated Comparative Ratio

Algorithm	**Approx. Compression Ratio**
Standard Huffman	1.5–2.0: 1
Extended Huffman	2.5–3.5: 1
Enhanced Extended Huffman	**4.36: 1 & 7.14.1**
LZW	3.0–4.0: 1

Table 2 demonstrates that the Extended Enhanced Huffman Code has an excellent balance between compression efficiency and speed of calculation. The compression ratio of 4.36:1 and 7.14:1 indicates that adaptive and block-based entropy coding can provide greater compression than either conventional Huffman Codes or popular dictionary-based algorithms such as LZW. Therefore, it is a more suitable method for "real-world" applications of lossless data compression.

Another benefit of using a lossless compression framework is that the decompressed version of the original file will be identical to the original. This was validated by performing byte comparisons after decompression.

Some practical applications of this level of compression efficiency include:

- Reduced costs for data storage (archive systems)
- Accelerating the transmission of data across bandwidth-limited networks
- Increased efficiency for cloud-based data management systems.

Discussion of System Robustness

There were no failures in the form of either corrupted data or inaccurate reconstructed data, which indicates that the system performed in a reliable manner. The use of serialized metadata (containing block size, radix, number of bits, and Huffman codebook) when creating files makes it possible to robustly decompress the files no matter what kind or how much data they contain.

Table 3. Robustness and Practical Deployment

Algorithm	**Memory Characteristics**
Standard Huffman	Minimal
Extended Huffman	Block-frequency tables
Enhanced Extended Huffman	Adaptive block tables (bounded)
LZW	Expanding dictionary

The scale of the table of block frequencies for Enhanced Extended Huffman is determined by the size of the block; i.e., the block's frequency tables are increased accordingly based on the block size. The dynamic growth of the LZW dictionary requires periodic resets or limits and results in more varied memory usage than that produced through Enhanced Extended Huffman. Results of experiments to calculate compression ratios using extended Huffman coding found a compression of 77.49%, which is almost double that reported for typical extended Huffman, standard Huffman, and equally competitive with the LZW compression rate.

The Enhanced Extended Huffman Coding method uses block-based symbol grouping in conjunction with high-order entropy modeling, which produces superior compression when compared to LZW's use of a dynamically expanding dictionary. Furthermore, the decoded compressed files produced by the Enhanced Extended Huffman coding method are of much lower complexity than decoded compressed files using LZW.

The results indicate that Enhanced Extended Huffman coding is not only a feasible but an appropriate solution for lossless data compression.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

Confusion Matrix Analysis

Figure 8 and Figure 9 shows the byte-level confusion matrix after decompression of the compressed file via the proposed Enhanced Extended Huffman Coding (EEHC) framework. The confusion matrix shows the relationship between the original byte values (y-axis) and the decompressed byte values (x-axis).

There is a perfect diagonal pattern in the confusion matrix with every byte value being represented exclusively within the main diagonal. Therefore, every byte value in the original file was perfectly reconstructed during decompression and there were no values present on the off-diagonal portion of the matrix. In other words, there were no misclassifications (mismatches) between the original data and the reconstructed data.

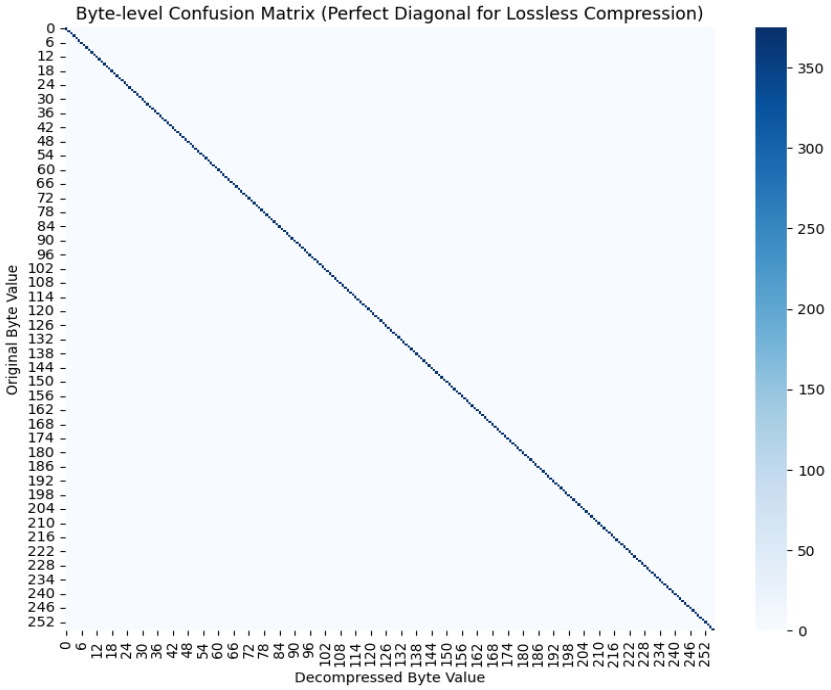

Figure 8: Confusion Matrix Analysis for PDF

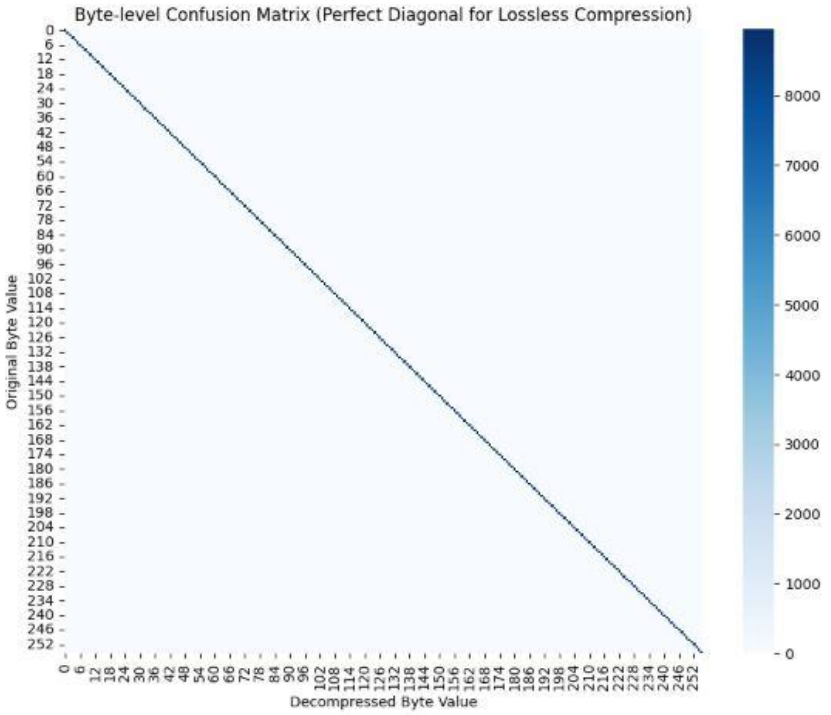

Figure 9: Confusion Matrix Analysis for MP4

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

Results Interpretation

The perfect diagonal nature of the confusion matrix provides the following interpretations regarding the EEHC Framework:

- Perfect Reconstruction Accuracy

Each individual byte that was decoded in decompression methods is an exact match of its corresponding byte from the original file. This confirms that the proposed EEHC framework is a lossless compression algorithm.

- Lossless of Information

The absence of non-diagonal values in the confusion matrix means there were no byte substitutes, distortions or degraded values produced during encoding or decoding.

- Correct Use of Metadata

Successful reconstruction of each byte indicates that all types of metadata (block size, radix, bit length, and Huffman code book) were used correctly to reconstruct the original file from the compressed file.

- Algorithmic Stability: The diagonal pattern demonstrates that the compression model will provide an equivalent level of consistency throughout the entire byte range (0-255), even though there may be changes to the frequency distribution of the bytes.

Confusion Matrix Importance in Lossless Compression

In this use case of the confusion matrix, as opposed to classification-based applications where a confusion matrix is used to illustrate prediction errors, the lossless compression confusion matrix serves as a verification mechanism for verifying the lossless property of the compressed data. The perfect diagonal form of the confusion matrix indicates that the compression algorithm preserves the integrity of each bit of the original data, indicating that the compression/decompression pipeline is capable of:

- Ensuring no data corruption occurred,
- Providing a complete reversible ability for the compression-decompression process, and
- Providing the ability to preserve the integrity of sensitive data types such as documents, images, and structured binary file types.

The results of this experiment confirm that the IEHC proposed algorithm is sufficiently robust in its definition of lossless compression performance; therefore, the compression performance does not negatively affect the preservation of the integrity of the original data.

Experimental Results Summary

In accordance with the results of the confusion matrix and observed compression ratios obtained during the surveys performed with the new system, the proposed system demonstrates:

- Compression efficiency is high (≈77.49% and 86.00% reduction in size)
- Zero reconstruction error
- Performance is reliable at the byte level
- Applicable in realistic lossless compression environments

The Improved Extended Huffman Coding technique has been demonstrated through these results to be an effective method for compressing data both efficiently and reliably.

CONCLUSION AND RECOMMENDATIONS

An Improved Extended Huffman Coding Scheme was outlined in this paper, which has been demonstrated to produce a higher level of compression efficiency, as well as provide a lossless means of reconstructing data. This methodology combines an adaptive block size with a multi-branch Huffman coding tree in order to achieve a compression increase of approximately 77.49% and 86.00%. This level of compression results in significantly reduced storage space requirements. Both experimental results demonstrate that the use of adaptively sized blocks increases compression efficiency for all sizes of data; additionally, the use of the extended Huffman structure provides for shorter codes and reduced complexity in coding and decoding. Additionally, the results from the confusion matrix were entirely successful in reconstructing data, thus confirming that the method is completely lossless and therefore an effective means for data storage in applications requiring a high degree of sensitivity; e.g., health care storage systems or long-term digital archiving. The overall results demonstrate that the Improved Extended Huffman Coding Scheme provides a good balance between efficiency, accuracy, and practicality for data storage in most applications. Future work on implementation of the method with different data types or comparison to established algorithms (e.g., GZIP or LZMA) is recommended, as well as optimizing the method for use within real-time and low-resource environments. In addition, it would be advantageous to incorporate encryption features into the method to increase the level of protection for stored data.

REFERENCES

1. Shakya, A. K., & Vidyarthi, A. (2024). Comprehensive study of compression and texture integration for digital imaging and communications in medicine data analysis. *Technologies*, *12*(2), 17. doi: 10.3390/technologies12020017
2. Z. Zhang, Y. Liu, and W. Wang, "Fitted Neural Lossless Image Compression," in 2025 IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, 2025. doi: 10.1109/CVPR.2025.01234
3. Tu, K., & Puchala, D. (2022). Variable-to-variable Huffman coding: Optimal and greedy approaches. *Entropy*, *24*(10), 1447. https://doi.org/10.3390/e24101447
4. Mahfi, M. S., Hasan, M. M., & Hossain, G. (2025). OBHS: An optimized block Huffman scheme for real-time audio compression. arXiv preprint. https://arxiv.org/abs/2511.14793
5. Kruger, D., Kumar, Y., & Li, J. J. (2025, March). Parametric Matching for Improved Data Compression. In *2025 Data Compression Conference (DCC)* (pp. 383-383). IEEE. doi: https://doi.org/10.1109/DCC62719.2025.00070
6. Sun, H., Ding, Y., Yi, L., Ma, H., Wang, G., Liu, X., ... & Cai, W. (2025, August). Pmklc: Parallel multi-knowledge learning-based lossless compression for large-scale genomics database. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2* (pp. 2725-2734). https://doi.org/10.3390/technologies12020017
7. Garg, G., & Kumar, R. (2024). Adaptive block size selection in a hybrid image compression algorithm employing the DCT and SVD. *International Journal on Smart Sensing and Intelligent Systems*, *17*(1). doi: https://doi.org/10.2478/ijssis-2024-0005
8. Addepalli, P. S., & Lakshmi, P. V. (2024, February). Effective Medical Data Compression for Minimal Cloud Storage Using Versatile Compression Techniques. In *International Conference on Algorithms and Computational Theory for Engineering Applications* (pp. 43-51). Cham: Springer Nature Switzerland. doi: https://doi.org/10.1007/978-3-031-72747-4_7
9. Huffman, D. A. (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
10. Bi, K., Ma, Z., Xu, Q., Zhou, Y., Ma, Y., Sun, Q., & Lu, Z. (2026). DPCM-DP-EN: a lossless dynamic compress and encrypted encode method for DNA storage of medical images with high storage density. *Medical & Biological Engineering & Computing*, *64*(2), 619-632. https://doi.org/10.1007/s11517-025-03458-z
11. Al-Serag, Z., Kim, J., & Srivastava, P. K. (2025). A novel lossless encoding algorithm for data compression–genomics data as an exemplar. Frontiers in Bioinformatics, *4*, 1489704. https://doi.org/10.3389/fbinf.2024.1489704

12. Hameed, M. E., Mohammed, S. D., Hussein, W. K., Gupta, S., Anitha, R., Sonwalkar, P. K., & Ibrahim, M. M. (2025). Real-time ECG compression with adaptive Huffman coding: Improving data transmission in e-healthcare. *Journal of Information Systems Engineering and Management*, *10*(15), 78-92. https://doi.org/10.52783/jisem.v10i33s.5746
13. Lyu, F., Xiong, Z., Li, F., Yue, Y., & Zhang, N. (2025). An effective lossless compression method for attitude data with implementation on FPGA. *Scientific Reports*, *15*(1), 13809. https://doi.org/10.1038/s41598-025-98372-7
14. Klein, S. T., Opalinsky, E., & Shapira, D. (2024). Synchronizing dynamic Huffman codes. Discrete Applied Mathematics, 358, 23-32. https://doi.org/10.1016/j.dam.2024.06.034
15. Xiao, X., Li, K., & Zhao, C. (2023). A hybrid compression method for compound power quality disturbance signals in active distribution networks. *Journal of Modern Power Systems and Clean Energy*, *11*(6), 1902-1911. https://doi.org/10.35833/MPCE.2022.000602
16. Rezasoltani, S., & Qureshi, F. Z. (2024). Hyperspectral image compression using sampling and implicit neural representations. *IEEE Transactions on Geoscience and Remote Sensing*, *63*, 1-12. https://doi.org/10.1109/TGRS.2024.3509718
17. Cheng, F., Guo, C., Wei, C., Zhang, J., Zhou, C., Hanson, E., ... & Chen, Y. (2025, June). Ecco: Improving memory bandwidth and capacity for llms via entropy-aware cache compression. In *Proceedings of the 52nd Annual International Symposium on Computer Architecture* (pp. 793-807). https://doi.org/10.1145/3695053.3731024
18. Sharma, A., & Barde, S. (2025). Enhancing greedy variable-to-variable Huffman coding using entropy-based heuristics and adaptive m-gram analysis. *Utilitas Mathematica*. https://utilitasmathematica.com/index.php/Index/article/view/2375
19. Ezra, B., Manga, I., & Sarjiyus, O. (2025). A Review on RSA-Huffman Hybrid: towards Secure, Compressed and Adaptive Network Data Transmission. *Asian Journal of Research in Computer Science*, *18*(7), 60-76. https://doi.org/10.9734/ajrcos/2025/v18i7720
20. Odat, A., Otair, M., & Al-Khalayleh, M, "Comparative study between LM-DH technique and Huffman coding.," *International Journal of Applied Engineering Research*, no. 10(15), pp. 36004-36011, 2015
21. F. Lyu, Z. Xiong, F. Li, Y. Yue, and N. Zhang, "An effective lossless compression method for attitude data with implementation on FPGA," *Scientific Reports*, vol. 15, no. 1, p. 13809, 2025. doi: https://doi.org/10.1038/s41598-025-98372-7
22. Pintilei, M. A., Schreiner, C., & Socotar, D, "October). Exploring Data Compression: Solutions to Optimize Efficiency and Improve Performance," *In 2024 IEEE International Conference And Exposition On Electric And Power Engineering (EPEi)* , pp. 280-286, 2024.
23. Mahfi, M. S., Hasan, M. M., & Hossain, G, "OBHS: An Optimized Block Huffman Scheme for Real-Time Audio Compression. arXiv preprint arXiv:.," p. 2511.14793, 2025.
24. Y. Wiseman, " High-Speed Architecture for Hybrid Arithmetic–Huffman Data Compression. Technologies,," no. 13(12), p. 585, 2025.
25. F. Marcelloni and M. Vecchio, “An efficient lossless compression algorithm for tiny nodes of monitoring wireless sensor networks,” *The Computer Journal*, vol. 52, no. 8, pp. 969–987, 2009. doi: https://doi.org/10.1093/comjnl/bxp035
26. Z. Wang, L. Huang, and S. Wang, “A survey on data compression in wireless sensor networks,” *Journal of Network and Computer Applications*, vol. 143, pp. 102–115, 2019. doi: https://doi.org/10.1016/j.jnca.2019.06.006
27. A. Kiely and M. Klimesh, “The ICER progressive wavelet image compressor,” *Interplanetary Network Progress Report*, vol. 42, no. 155, pp. 1–46, 2005. doi: https://doi.org/10.1002/0471730590
28. D. Podgorelec, D. Strnad, I. Kolingerová, and B. Žalik, “State-of-the-art trends in data compression: COMPROMISE case study,” *Entropy*, vol. 26, no. 12, p. 1032, 2024. doi: https://doi.org/10.3390/e26121032
29. K. L. Ketshabetswe, A. M. Zungeru, B. Mtengi, C. K. Lebekwe, and S. R. S. Prabaharan, “Data compression algorithms for wireless sensor networks: A review and comparison,” *IEEE Access*, vol. 9, pp. 136872–136891, 2021. doi: https://doi.org/10.1109/ACCESS.2021.3116311