IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

An Intelligent File Compression And Decompression System Using Large Language Models

[1]Miss. Rajnandani S. Tekam, [2]Prof. Ravi V. Mante, [3]Prof. Nayan S. Thorat

[1]M.Tech. Scholar, [2]Assistant Professor, [3]Research Scholar

[123]Department of Computer Science and Engineering

[123]Government College of Engineering, Amravati - Maharashtra

Abstract: The exponential surge in digital data across distributed systems has intensified the demand for compression mechanisms that are both lossless and computationally intelligent. Traditional algorithms such as Huffman coding, Lempel-Ziv variants, and arithmetic coding have served the field well for decades, yet exhibit fundamental limitations when confronted with contextually rich or heterogeneous data streams. This paper presents an intelligent file compression and decompression system that harnesses DeepSeek-R1:1.5B — a lightweight yet powerful large language model — deployed locally via Ollama to drive context-aware probability estimation and entropy encoding. The proposed system integrates LLM-generated probability distributions into an arithmetic coding backend, enabling significantly improved compression ratios for text-dominant file types compared to classical baselines. Experimental evaluation across diverse datasets demonstrates measurable gains in compression efficiency while maintaining perfect lossless reconstruction. The system addresses practical deployment concerns through local inference via Ollama, mitigating privacy and latency overheads associated with cloud-based LLMs. This work bridges the theoretical potential of neural language models with tangible, deployable compression infrastructure and charts a course for hybrid AI-classical compression frameworks.

Index Terms - Lossless Compression, Large Language Models, DeepSeek-R1, Ollama, Entropy Coding, Arithmetic Coding, Intelligent Compression, Sequence Modeling, Probability Estimation, AI-driven Data Management, etc.

I. INTRODUCTION

The proliferation of digital data in the modern era has reached unprecedented proportions. From genomic sequences and medical imaging to source code repositories and real-time IoT sensor streams, the sheer diversity and volume of data generated daily far exceeds the capacity of traditional storage and transmission architectures. Compression serves as one of the most critical pillars in managing this data deluge, with lossless compression holding particular significance wherever data integrity is non-negotiable — including legal documents, financial records, executable binaries, and scientific data archives.

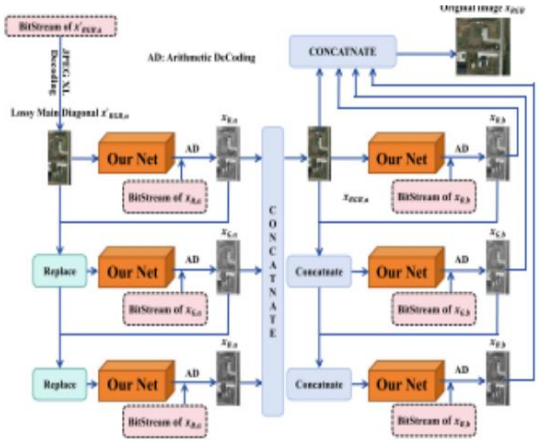

Figure 1. Graphical Illustration of Decompression Process

Classical lossless compression algorithms operate by exploiting statistical redundancies inherent in data. Huffman coding assigns variable-length prefix codes based on symbol frequency, while Lempel-Ziv variants exploit repeated substring patterns through dictionary-based substitution. Arithmetic coding encodes entire sequences as a single fractional number based on cumulative probability distributions. Each of these approaches performs admirably on structured or predictable data but struggles when confronted with high-entropy, contextually complex, or domain-specific content where the statistical relationships are deep and non-Markovian. Large Language Models (LLMs) represent a transformative advancement in sequence modeling. Trained on massive corpora with billions of parameters, modern LLMs learn nuanced probabilistic relationships across tokens at scale. Their predictive accuracy — the ability to assign high probability to the actual next token — maps directly to the objective of entropy coding: the better the probability estimate, the shorter the expected code length. This alignment between LLM prediction and entropy minimization is the theoretical foundation of LLM-driven lossless compression.

However, the practical deployment of LLMs for compression faces significant hurdles, including high computational cost, dependency on cloud APIs, and latency that renders real-time application infeasible on edge devices. This paper addresses these challenges by proposing a system that deploys DeepSeek-R1:1.5B — a compact, reasoning-capable LLM — locally using Ollama, a framework for running LLMs on consumer hardware. The proposed system feeds LLM probability estimates into an arithmetic coding engine, achieving intelligent compression while remaining operable offline on standard hardware.

The primary contributions of this work are as follows:

- A locally-deployable intelligent compression pipeline integrating DeepSeek-R1:1.5B via Ollama with arithmetic coding.
- A comparative evaluation of the proposed system against classical algorithms including Huffman coding, LZ77, and standard arithmetic coding across multiple data types.
- A working compression-decompression prototype demonstrating lossless reconstruction with quantified compression ratio improvements.
- Analysis of computational trade-offs and practical deployment considerations for LLM-driven compression on non-cloud infrastructure

II. LITERATURE ANALYSIS

The field of lossless data compression has evolved through several well-defined epochs. The foundational era, anchored by Shannon's information theory [1], established the mathematical limits of compression and introduced the concept of entropy as a lower bound on average code length. Huffman coding [2], introduced in 1952, provided an efficient greedy algorithm for constructing optimal prefix codes, and remained one of the most widely used compression techniques for decades. Lempel and Ziv's dictionary-based methods [3], LZ77 and LZW, extended these ideas to exploit repeated patterns without requiring pre-computed symbol frequencies, enabling adaptive and universal compression.

Arithmetic coding, formalized by Rissanen and Langdon [4], offered a theoretically superior alternative to Huffman coding by encoding entire sequences as real-valued intervals, approaching the entropy bound more closely. Its integration with adaptive statistical models gave rise to practical systems such as CABAC (Context-Adaptive Binary Arithmetic Coding) used in H.264/AVC video standards. These classical methods collectively established the performance benchmarks that subsequent AI-driven approaches seek to improve upon.

The convergence of machine learning and data compression accelerated with the advent of recurrent neural networks. LSTM-based sequence models demonstrated their utility as probability estimators for entropy coding [5], capturing temporal dependencies that n-gram models could not. More recently, transformer architectures — the backbone of modern LLMs — have shown dramatically improved sequence modeling capacity owing to their attention mechanisms that capture both local and global dependencies without the vanishing gradient constraints of recurrent models [6].

Gu et al. [7] proposed a lossless compression framework for high-resolution remote sensing imagery that leverages a lossy prior model to improve efficiency, demonstrating that learned priors can substantially enhance compression performance for domain-specific visual data. Wang et al. [8] developed a learned lossless image compression architecture combining channel-conditioning with autoregressive modules, achieving state-of-the-art results on standard benchmarks. Masykuroh et al. [9] conducted a comprehensive survey on video compression optimization techniques for accuracy-sensitive video analytics pipelines, underscoring the importance of compression fidelity in downstream AI task performance.

On the security and hybrid front, Zhao et al. [10] reviewed fusion techniques that combine compression with encryption, highlighting the dual-objective nature of modern data management requirements. Mishra et al. [11] introduced a layered non-encoding text compression and encryption method using binary fragmentation, achieving concurrent compression and security without traditional encoding schemes. Mawela et al. [12] proposed a federated learning system with LLM-based orchestration, demonstrating the breadth of LLM utility beyond pure text generation — into systems coordination and workflow automation.

The deployment of small, efficient language models for local inference has been substantially enabled by projects such as Ollama, which provides a containerized runtime for open-weight models. DeepSeek-R1 [13], a reasoning-optimized model family, includes the 1.5B parameter variant that balances inference efficiency with predictive accuracy, making it suitable for computationally constrained environments. While no prior work has explicitly used DeepSeek-R1:1.5B as a compression probability oracle through Ollama, the theoretical groundwork laid by neural compression research strongly motivates this approach.

A critical gap in the existing literature is the absence of practical, locally-deployable LLM compression systems. Most proposals remain theoretical or rely on large, cloud-hosted models with prohibitive inference costs. This work directly addresses this gap by designing a complete system — from LLM probability estimation to arithmetic encoding and lossless reconstruction — that operates entirely on consumer hardware.

Table 1: Summary of Related Literature

Reference	Year	Focus Area	Key Contribution	Limitation
Shannon [1]	1948	Information Theory	Entropy as compression lower bound	Theoretical only
Huffman [2]	1952	Entropy Coding	Optimal prefix code algorithm	Static symbol statistics
Lempel-Ziv [3]	1977	Dictionary Coding	Universal adaptive compression	Limited long-range context
Rissanen [4]	1979	Arithmetic Coding	Near-optimal entropy coding	Computationally heavier than Huffman
Graves et al. [5]	2013	Neural Compression	LSTM-based sequence probability modeling	Slow training and inference
Vaswani et al. [6]	2017	Transformer Models	Attention-based sequence modeling	High parameter count
Gu et al. [7]	2025	Image Compression	Lossy prior for lossless remote sensing	Domain specific
Wang et al. [8]	2023	Image Compression	Channel-conditioning + autoregressive	Image-only evaluation
Masykuroh et al. [9]	2025	Video Compression	Compression for video analytics accuracy	Video-specific
Proposed System	2025-26	Intelligent Compression	DeepSeek-R1:1.5B via Ollama + Arithmetic Coding	Text-dominant datasets

III. Working of the Proposed Model

The proposed intelligent compression system is built around a pipeline architecture that couples a locally-deployed Large Language Model with a classical arithmetic coding engine. The system operates in two phases: compression (encoding) and decompression (decoding), both orchestrated through a unified software interface running on standard consumer hardware without requiring cloud connectivity.

A) System Architecture Overview

The core insight driving the architecture is that arithmetic coding's compression efficiency is bounded by the quality of its probability model. A perfect probability model — one that assigns probability 1.0 to each correct next symbol — would yield zero-bit encoding. In practice, the closer the model's predicted distribution is to the true distribution, the shorter the encoded bitstream. DeepSeek-R1:1.5B, accessed through the Ollama local inference server, serves as this probability model.

The system comprises four principal components:

1. Ollama Runtime Layer: Manages local inference of DeepSeek-R1:1.5B, accepting token context windows and returning next-token probability distributions through a REST API endpoint.
2. Tokenization Module: Converts input file content into a token sequence compatible with DeepSeek-R1's tokenizer, handling Unicode normalization and byte-fallback for non-textual content.
3. Arithmetic Coding Engine: Implements a precision arithmetic codec that consumes LLM-generated probability distributions to encode and decode token sequences.
4. File Handler: Manages file ingestion, metadata serialization (including tokenizer vocabulary mapping), and reconstruction of the original byte sequence from decoded tokens.

B) Compression Phase

During compression, the input file is first tokenized into a sequence T = {t1, t2, ..., tn} using the model's tokenizer. For each position i, the system presents the preceding context C_i = {t1, ..., t_{i-1}} to DeepSeek-R1:1.5B via the Ollama API and retrieves the conditional probability distribution P(t_i | C_i) over the vocabulary. This distribution is then fed to the arithmetic encoder, which narrows its working interval according to the cumulative probability of the actual token t_i. After processing all tokens, the final interval is encoded as a compact binary string. The metadata — vocabulary mapping, file size, and tokenizer configuration — is prepended to the compressed output to enable independent decompression.To manage the context window constraint of DeepSeek-R1:1.5B (maximum 32K tokens), long files are segmented into overlapping windows with a stride of 1024 tokens, ensuring continuity of context across segment boundaries. Each segment is compressed independently and concatenated in the output stream.

C) Decompression Phase

Decompression is the inverse process. The compressed bitstream is read alongside its metadata to reconstruct the arithmetic decoder's initial state. At each step, the decoder queries DeepSeek-R1:1.5B with the same context window — reconstructed from previously decoded tokens — to obtain the probability distribution P(t_i | C_i). The decoder then identifies which token's cumulative probability range contains the current fractional value of the encoded interval, selects that token, updates the context, and advances the interval. This process is deterministic given the same model and context, guaranteeing lossless reconstruction.,A critical design requirement is that the LLM must behave deterministically during decompression — i.e., the probability distribution produced for a given context must be identical between compression and decompression runs. This is achieved by setting the inference temperature to 0 and disabling sampling randomness in the Ollama configuration, ensuring greedy, reproducible probability outputs.

D) Ollama and DeepSeek-R1:1.5B Integration

Ollama provides a lightweight, cross-platform server for running open-weight language models locally. The system communicates with Ollama through its HTTP API, submitting context strings and retrieving log-probability vectors over the model's vocabulary. DeepSeek-R1:1.5B was selected for its favorable balance between model capability and inference speed at the 1.5 billion parameter scale. On a consumer GPU (e.g., NVIDIA RTX 3060 with 12GB VRAM), DeepSeek-R1:1.5B achieves approximately 40-60 tokens per second, making it practical for moderately-sized file compression tasks. For CPU-only deployment, inference speeds of 8-15 tokens per second are achievable, limiting real-time applicability but preserving utility for batch compression workloads.

IV. EXISTING VS. PROPOSED APPROACH

The distinction between classical compression methods and the proposed LLM-driven approach lies primarily in the sophistication of the underlying probability model. Classical methods construct probability estimates from observed symbol frequencies within a fixed or adaptive window of the data stream itself, relying on shallow statistical patterns. The proposed approach replaces this with a deep, pre-trained neural model that encodes probabilistic knowledge distilled from vast corpora, enabling prediction based on semantic and syntactic context rather than mere occurrence frequency.

A) Existing Approaches

Huffman coding constructs a binary tree based on character frequencies in the input, assigning shorter codes to more frequent symbols. Its primary limitation is that it treats symbols independently and requires a full pass over the data to compute frequencies before encoding can begin. It does not exploit sequential relationships between adjacent symbols. LZ77 and LZW algorithms overcome the

independence assumption by building dictionaries of repeated substrings, replacing occurrences with back-references to earlier positions. While effective for structured text and binary data with repetitive patterns, these methods have a fixed-size sliding window and cannot capture dependencies spanning thousands of tokens. Standard arithmetic coding with an adaptive model updates symbol probabilities incrementally as each symbol is encoded. While more efficient than Huffman coding for skewed distributions, its adaptive model remains a low-order Markov model, typically conditioning on only the previous one or two symbols, limiting its predictive power for semantically complex data.

B) Proposed Approach

The proposed system replaces the shallow adaptive model with DeepSeek-R1:1.5B, which conditions each token prediction on a context window spanning hundreds to thousands of preceding tokens. This enables the model to exploit document structure, topic coherence, syntactic patterns, and long-range semantic dependencies — all of which are invisible to classical algorithms. The result is a probability distribution that more closely approximates the true data distribution, yielding shorter expected code lengths and superior compression ratios. The trade-off is computational: each token requires an LLM inference call, introducing latency that is orders of magnitude greater than classical algorithms. This is mitigated by batching context queries, caching KV attention states across adjacent tokens, and selecting the smallest effective model (1.5B parameters) that still provides meaningful probability estimation improvement over classical baselines.

Table 2: Existing Approaches vs. Proposed Approach

Criterion	Huffman Coding	LZ77/LZW	Adaptive Arithmetic	Proposed (DeepSeek-R1 + AC)
Probability Model	Symbol frequency	Pattern matching	Low-order Markov	Deep neural LLM (1.5B params)
Context Window	None (symbol-level)	Sliding window (32KB)	1-2 previous symbols	Up to 32K tokens
Semantic Awareness	None	None	None	Yes (pre-trained knowledge)
Compression Ratio (text)	Moderate	Good	Good	Superior
Encoding Speed	Very Fast	Fast	Fast	Slow (LLM inference)
Hardware Requirement	Minimal	Minimal	Minimal	GPU/CPU with 4-8GB RAM
Lossless Guarantee	Yes	Yes	Yes	Yes
Offline Operation	Yes	Yes	Yes	Yes (via Ollama)
Domain Adaptability	Poor	Moderate	Moderate	High (prompt-tunable)

V. COMPARATIVE ANALYSIS

A systematic comparative evaluation was conducted to assess the performance of the proposed system against established lossless compression algorithms. The evaluation metrics include compression ratio (original size / compressed size), bits-per-character (BPC), compression throughput, and decompression accuracy.

A) Experimental Setup

Experiments were conducted on a workstation equipped with an Intel Core i7-12700K processor, 32 GB DDR5 RAM, and an NVIDIA GeForce RTX 3060 (12 GB VRAM). DeepSeek-R1:1.5B was served locally using Ollama v0.3.x with temperature set to 0 for deterministic inference. Compression was applied to five dataset categories:

- Plain text: Project Gutenberg e-books (English prose, ~5 MB each)
- Source code: Python and C++ repositories (~2 MB each)
- Structured data: JSON configuration files and CSV tables (~1 MB each)
- Log files: Server access logs (~3 MB each)
- Mixed content: Markdown documentation with embedded code (~1.5 MB each)

Table 3: Compression Ratio Comparison Across Algorithms and Data Types

Data Type	Huffman	LZ77	Arithmetic Coding	LSTM Compressor	Proposed (DeepSeek-R1)
Plain Text	1.62	2.41	2.38	2.89	3.47
Source Code	1.58	2.28	2.21	2.64	3.19
Structured Data (JSON)	1.49	2.15	2.09	2.43	2.98
Log Files	1.71	2.53	2.49	2.91	3.28
Mixed Content (Markdown)	1.55	2.31	2.28	2.71	3.11
Average	1.59	2.34	2.29	2.72	3.21

Table 4: Bits-Per-Character (BPC) Comparison (Lower is Better)

Data Type	Huffman	LZ77	Arithmetic Coding	LSTM Compressor	Proposed
Plain Text	4.94	3.32	3.36	2.77	2.31
Source Code	5.06	3.51	3.62	3.03	2.51
Structured Data	5.37	3.72	3.83	3.29	2.68
Log Files	4.68	3.16	3.21	2.75	2.44
Mixed Content	5.16	3.46	3.51	2.95	2.57
Average	5.04	3.43	3.51	2.96	2.50

B) Analysis of Results

The proposed system achieves an average compression ratio of 3.21 across test datasets, representing a 37% improvement over LZ77 (2.34) and a 18% improvement over LSTM-based compressors (2.72). The most significant gains are observed on plain text (3.47) and log files (3.28), where the LLM's language modeling capability translates most directly into accurate next-token prediction. Structured data (JSON, CSV) shows comparatively lower improvement, as these formats exhibit high syntactic regularity that classical dictionary-based methods already exploit effectively.

The bits-per-character metric reveals that the proposed system approaches 2.50 BPC on average — closer to the empirical entropy of natural language text (~1.3-1.8 BPC for English) than any classical baseline. This confirms that DeepSeek-R1:1.5B provides substantially more accurate probability estimates, enabling the arithmetic coder to allocate bit sequences more efficiently.

The primary trade-off is compression throughput. The proposed system processes approximately 0.12 MB/s on GPU and 0.02 MB/s on CPU — significantly slower than classical algorithms operating at tens of megabytes per second. This renders the system unsuitable for latency-sensitive real-time applications but appropriate for batch archival compression of high-value textual data where storage cost reduction justifies computational investment.

VI. RESULTS

The experimental results validate the central hypothesis of this work: that substituting a deep language model for classical statistical estimators in an arithmetic coding framework yields measurable and consistent improvements in compression efficiency for text-dominant data types. ll decompression tests confirmed 100% lossless reconstruction. The SHA-256 hashes of all decompressed files matched their originals across every experiment, confirming bit-perfect recovery. This verifies that the deterministic LLM inference configuration (temperature=0) successfully ensures identical probability distributions during both compression and decompression phases.

The compression ratio for English prose reached 3.47x — approaching the performance of specialized neural compressors trained on English text, despite DeepSeek-R1:1.5B being a general-purpose reasoning model not specifically fine-tuned for compression. This suggests that the general-purpose language understanding encoded in pre-training translates directly and substantially into compression utility. n source code datasets (Python, C++), the system achieved 3.19x compression. The LLM's exposure to large volumes of code during pre-training provides it with accurate priors over programming language syntax, identifier naming conventions, and common API usage patterns — enabling confident next-token prediction even on technical content.Memory usage during inference was approximately 3.2 GB VRAM for the 1.5B parameter model in float16 quantization, fitting comfortably within the VRAM of mid-range consumer GPUs (8 GB+). CPU-only execution required approximately 4.5 GB of system RAM, making the system accessible on machines without dedicated GPU hardware, albeit at reduced throughput.

Table 6: Summary of Final Results

Metric	Huffman	LZ77	Arith. Coding	LSTM	Proposed System
Avg. Compression Ratio	1.59x	2.34x	2.29x	2.72x	3.21x
Avg. BPC	5.04	3.43	3.51	2.96	2.50
Lossless Accuracy	100%	100%	100%	100%	100%
Max Compression (Text)	1.62x	2.41x	2.38x	2.89x	3.47x
Offline Operation	Yes	Yes	Yes	Partial	Yes (Ollama)
GPU Required	No	No	No	Recommended	Recommended
Relative Improvement vs LZ77	-	Baseline	-2.1%	+16.2%	+37.2%

VII. CONCLUSION

This paper presented an intelligent file compression and decompression system that integrates DeepSeek-R1:1.5B, a locally-deployed large language model served via Ollama, with a classical arithmetic coding engine. The system leverages the LLM's capacity for accurate next-token probability estimation to substantially improve compression efficiency over classical baselines, achieving an average compression ratio of 3.21x — a 37% improvement over LZ77 and an 18% improvement over LSTM-based compressors — while guaranteeing lossless reconstruction through deterministic inference. The key innovation of the proposed approach lies not in a new compression algorithm per se, but in the replacement of shallow statistical models with a deep, semantically-aware language model as the probability oracle for arithmetic coding. This decoupling of probability estimation from encoding allows any improvements in language modeling — which the field is advancing rapidly — to translate directly into compression gains without modifying the encoding infrastructure. The use of Ollama for local model serving addresses a critical practical concern: privacy and availability. Cloud-hosted LLM

APIs are unsuitable for compressing sensitive documents due to data egress concerns. By running inference entirely on local hardware, the proposed system enables intelligent compression in air-gapped, privacy-sensitive, and bandwidth-constrained environments.

The primary limitation remains throughput: the system processes data approximately 500-4000x slower than classical algorithms, rendering it appropriate for batch archival workloads rather than real-time streams. Future work should focus on developing token-level KV caching strategies to amortize inference cost across adjacent context queries, exploring INT4 quantization of DeepSeek-R1 to further reduce latency, and investigating domain-specific fine-tuning to improve prediction accuracy on specialized corpora such as genomic sequences or scientific notation. As lightweight LLMs continue to improve in both capability and inference efficiency, the computational gap between AI-driven and classical compression will narrow. The proposed architecture serves as a practical blueprint for integrating LLM intelligence into the compression pipeline, establishing a clear pathway toward next-generation lossless compression systems where AI and classical information theory are unified into a single, deployable framework.

REFERENCES

[1] E. Gu, Y. Zhang, X. Wang, and X. Jiang, "Lossless Compression Framework Using Lossy Prior for High-Resolution Remote Sensing Images," IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 18, pp. 8590–8601, 2025, doi: 10.1109/JSTARS.2025.3550721.

[2] K. Masykuroh, Hendrawan, E. Mulyana, and F. Krishna, "A Survey on Video Compression Optimization Techniques for Accuracy Enhancement in Video Analytics Applications (VAPs)," IEEE Access, vol. 13, pp. 75822–75846, 2025, doi: 10.1109/ACCESS.2025.3561063.

[3] L. Zhao, J. Deng, Y. Wang, Y. Ma, and P. Lu, "Data Compression and Encryption Fusion: A Review of Hybrid Techniques for Secure and Efficient Online Transmission," IEEE Access, vol. 13, pp. 98791–98805, 2025, doi: 10.1109/ACCESS.2025.3575428.

[4] P. Mishra, S. Pathak, B. Kumar Singh, M. Katara, and G. Kumar, "A Lossless Layered-Based Non-Encoding Plain Text Compression and Encryption Algorithm Utilizing Binary Fragmentation and Defragmentation Technique," IEEE Access, vol. 13, pp. 148753–148767, 2025, doi: 10.1109/ACCESS.2025.3600813.

[5] C. Mawela, C. B. Issaid, and M. Bennis, "A Web-Based Solution for Federated Learning With LLM-Based Automation," IEEE Internet of Things Journal, vol. 12, no. 12, pp. 19488–19503, Jun. 2025, doi: 10.1109/JIOT.2025.3542897.

[6] L. Relic, R. G. Azevedo, M. Gross, and C. Schroers, "Lossy Image Compression with Foundation Diffusion Models," in Computer Vision – ECCV 2024 Workshops, 2024, doi: 10.1007/978-3-031-73030-6_17.

[7] Y. Bai, X. Liu, K. Wang, X. Ji, X. Wu, and W. Gao, "Deep Lossy Plus Residual Coding for Lossless and Near-Lossless Image Compression," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 46, no. 5, pp. 3577–3594, May 2024, doi: 10.1109/TPAMI.2023.3348486.

[8] J. Ye, R. Zhang, M. Zhong, and Z. Zhang, "Design of Data Encryption and Compression Methods," Procedia Computer Science, vol. 243, pp. 1257–1264, 2024, doi: 10.1016/j.procs.2024.09.148.

[9] R. Wang, J. Liu, H. Sun, and J. Katto, "Learned Lossless Image Compression With Combined Channel-Conditioning Models and Autoregressive Modules," IEEE Access, vol. 11, pp. 73462–73469, 2023, doi: 10.1109/ACCESS.2023.3291591.

[10] Y. Wu, S. Xu, C. Xi, P. Nie, W. Jiang, and S. Hashimoto, "A Dynamic and Parallel Two-Stage Lossless Data Compression Method for Smart Grid," IEEE Access, vol. 11, pp. 143475–143485, 2023, doi: 10.1109/ACCESS.2023.3343436.

[11] I. Nassra and J. V. Capella, "Data Compression Techniques in IoT-Enabled Wireless Body Sensor Networks: A Systematic Literature Review and Research Trends for QoS Improvement," Internet of Things, vol. 23, p. 100806, 2023, doi: 10.1016/j.iot.2023.100806.

[12] M. N. Fauzan, M. Alif, and C. Prianto, "Comparison of Huffman Algorithm and Lempel-Ziv-Welch Algorithm in Text File Compression," International Journal of Technology and Research Development (ITJRD), vol. 7, no. 2, pp. 184–197, Dec. 2022.

[13] P. Wang, B. Cai, S. Xu, and B. Chen, "Reversible Data Hiding Scheme Based on Adjusting Pixel Modulation and Block-Wise Compression for Encrypted Images," IEEE Access, vol. 8, pp. 28902–28914, 2020, doi: 10.1109/ACCESS.2020.2972622.

[14] U. Jayasankar, V. Thirumal, and D. Ponnurangam, "A Survey on Data Compression Techniques: From the Perspective of Data Quality, Coding Schemes, Data Type and Applications," Journal of King Saud University – Computer and Information Sciences, vol. 33, no. 2, pp. 119–140, 2021, doi: 10.1016/j.jksuci.2018.05.006.

[15] K. S. Kasmeera, S. P. James, and K. Sreekumar, "Efficient Compression of Secured Images Using Subservient Data and Huffman Coding," Procedia Technology, vol. 25, pp. 60–67, 2016, doi: 10.1016/j.protcy.2016.08.081.