International Journal of Trend in Scientific Research and Development (IJTSRD)
Volume 9 Issue 3, May-Jun 2025 Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

Compression Algorithms for Efficient Big Data Storage

Dr. Gopal Prasad Sharma[1], Prof. Raj Kumar Thakur[2], Prof. Dr. Pawan Kumar Jha[3]

[1]Associate Professor, Purbanchal University School of Science & Technology (PUSAT), Biratnagar, Nepal
[2,3]Professor, Purbanchal University School of Science & Technology (PUSAT), Biratnagar, Nepal

ABSTRACT

In today's digital world, big data is growing at an unprecedented rate, presenting significant challenges in terms of storage and management. Compression algorithms play a pivotal role in addressing these challenges by reducing data size without compromising essential information. This article provides an in-depth exploration of data compression techniques, categorizing them into lossless and lossy compression methods, and evaluates their effectiveness in big data applications. The integration of these algorithms into popular big data frameworks such as Hadoop, Spark, and cloud storage systems is discussed, highlighting their impact on storage efficiency and data accessibility. A comparison of several compression algorithms, including Snappy, Zstandard, Gzip, and Brotli, is also presented to guide the selection of the most suitable algorithm based on compression ratio, speed, and the type of data being processed. The article concludes with a look into future trends and innovations in compression, including AI and machine learning-based techniques, adaptive methods, and the potential for quantum compression. These advancements offer exciting prospects for improving data storage and processing capabilities, while addressing ongoing challenges in big data environments.

KEYWORDS: *Big data, Compression Algorithms, Hadoop, Lossless Compression, Lossy Compression*

How to cite this paper: Dr. Gopal Prasad Sharma | Prof. Raj Kumar Thakur | Prof. Dr. Pawan Kumar Jha "Compression Algorithms for Efficient Big Data Storage" Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470, Volume-9 | Issue-3, June 2025, pp.788-794, URL: www.ijtsrd.com/papers/ijtsrd81127.pdf

IJTSRD81127

I. INTRODUCTION

The digital age's massive data generation has transformed business, government, and relationships. IoT devices, social media, and advanced analytics have increased data production to unprecedented levels. This trend, called "big data," involves massive, complex datasets that are difficult to store, process, and analyse. Big data helps identify patterns, improve decision-making, and drive innovation in medicine, economics, retail, and science [1]. Big data management and storage are difficult despite its many benefits. Big data's size and complexity make storage difficult. Infrastructure costs are high because structured and unstructured data require a lot of hardware. Keeping such data accessible, reliable, and latency-free complicates storage. Businesses are always looking for ways to maximise their limited resources because data is growing faster than most storage solutions can handle. Due to this need, storage solutions must be efficient, scalable, and affordable while protecting data integrity and accessibility.

Compression algorithms make big data storage easier. Compression algorithms allow organisations to store more data in less space while retaining essential information. Real-time applications need this optimisation to lower storage costs and boost processing and data transfer speeds. Lossless compression is the optimal choice for mission-critical data, while losing compression allows certain applications to make acceptable quality and size compromises [2]. Together, these algorithms support current data management methods. Compression algorithms are important and complicated in big data storage. It discusses their principles, rates popular methods, and integrates them into big data frameworks. The article discusses recent and future compression algorithm developments to demonstrate their importance in big data storage and management.

II. FUNDAMENTALS OF DATA COMPRESSION

Compressing data reduces its size without compromising its quality. Data compression reduces

space and bandwidth needed to transmit and store data to maximise resource use. Data-driven industries and the need to efficiently manage massive amounts of data have made compression crucial. Lossless and lossy compression are the main data compression methods [3].

The original data can be perfectly reconstructed from compressed data using lossless compression. This method is often used for high-fidelity data files like PNG images or text files like ZIP. Lossless compression preserves all data, making it essential for applications where small errors can have big effects.

Lossy compression reduces file size while maintaining quality by using some of the original data. Multimedia files like images, audio, and video use this method because even a small data loss doesn't affect user experience. JPEG and MP3 use lossy compression to balance file size and quality [4]. Lossy methods help streaming services save bandwidth. A compression algorithm's effectiveness is measured by compression ratio, speed, and efficiency. Data compression is often expressed as a percentage of its original size. For real-time applications, data compression and decompression times matter. An algorithm is efficient if it compresses data with little processing power. These metrics help decide if a compression method is right for a task.

Big data requires compression to overcome processing and storage limitations. Big data is too large to send or store unprocessed. Data compression saves computing resources, speeds network data transfer, and lowers storage costs. Compression algorithms in Hadoop and Spark help organisations manage large datasets. Therefore, big data management strategies must include compression.

III. OVERVIEW OF BIG DATA STORAGE CHALLENGES

Big data is defined by its 5Vs: volume, velocity, variety, veracity, and value. Volume terabytes to exabytes of data generated daily is called volume. This massive data flood comes from enterprise apps, social media, and IoT devices. Due to data generation and processing velocity, real-time or near-real-time analytics systems are needed. Big data includes structured databases, unstructured text, photos, videos, and sensor data. Truthfulness highlights the challenges of data accuracy and reliability in the face of noise and inconsistencies. Finally, big data analysis and use yield practical insights and benefits [5]. Traditional storage methods struggle with these features. Traditional database and file storage systems cannot handle big data's volume and velocity. They're not flexible or scalable enough to handle all data formats. This causes inefficient storage use, latency, and higher costs.

Because managing large datasets is complicated, data accessibility, security, and integrity are difficult to ensure. Big data companies aim to reduce storage costs and increase data accessibility [6]. Servers, data centres, and other storage infrastructure are expensive upfront and over time. Effective data management can reduce these expenses and improve system performance and data retrieval speeds. Improved accessibility ensures quick data retrieval and processing, enabling timely decision-making and competitive advantage preservation.

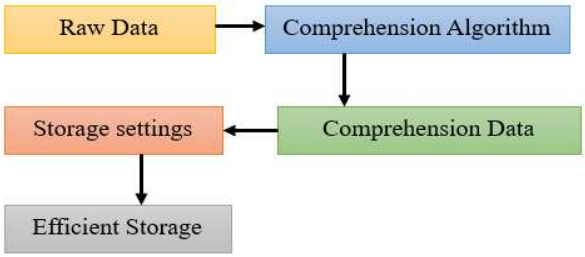

FIGURE 1 Data Flow Before and After Compression (Source: Self-Created)

Effective compression algorithms are needed to address these issues. This algorithm greatly reduces data file sizes, allowing businesses to store more data with less resources. Compression lowers infrastructure costs and hardware upgrades by reducing storage footprint [7]. Compression speeds up data processing and transfers, which is crucial for real-time applications and analytics. Strong compression techniques are needed to make storage strategies scalable, cost-effective, and operationally efficient as big data grows.

IV. TYPES OF COMPRESSION ALGORITHMS FOR BIG DATA

Massive data sets require compression algorithms for data management. These algorithms are mostly lossless and lossy. Both have different approaches, benefits, and drawbacks, making them optimal for specific applications. Hybrid compression methods maximise both approaches.

A. LOSSLESS COMPRESSION ALGORITHMS

The early lossless compression algorithm Huffman Coding stands out. To make it work, frequently appearing symbols have shorter binary codes and rarely appearing symbols have longer codes. Huffman Coding uses a symbol frequency-based binary tree structure to ensure no code is a prefix for efficient decoding [8]. The ZIP and GZIP file formats use this method. Its main benefit is reproducing initial data. Due to low compression ratios, Huffman Coding may not benefit datasets with normal frequency distributions.

Arithmetic coding, another lossless method, converts messages to integers between zero and one. Arithmetic Coding represents the probability of the entire data stream, unlike Huffman Coding, which uses discrete symbols, making it efficient for datasets with skewed symbol distributions [9]. This algorithm is used in text and multimedia codecs. Although it outperforms Huffman Coding in compression ratio, it is computationally heavy and may not work well in real time.

Compression algorithms like LZ77 and LZW (Lempel-Ziv) can replace repetitive patterns with dictionary-based shorter references. LZ77 replaces repeating strings with references to earlier ones, while LZW creates a data stream pattern dictionary. GIF and ZIP use these methods extensively. Their adaptability to different data types is one of their many advantages.

Quick and efficient lossless compression algorithms Snappy and Zstandard are new [10]. Google's Snappy prioritises fast compression and decompression for real-time applications like log management. Facebook's Zstandard balances speed and compression ratio and offers adjustable parameters. In fast-processing big data frameworks like Spark and Hadoop, these algorithms are becoming more popular. Though efficient, they may not compress as well as more complicated algorithms like Arithmetic Coding.

B. LOSSY COMPRESSION ALGORITHMS

Lossy compression relies on transform coding, which includes the DCT. Data can be converted from spatial to frequency domain to remove low-frequency components (important features) and keep high-frequency components (details). JPEG and MPEG use this method to compress images and videos [11]. Its main benefit is size reduction, but it may lose fine details, making it unsuitable for some uses.

Wavelet transforms use transform coding at multiple resolutions to improve data representation. Progressive transmission and scalable storage make this method ideal for audio and image compression. Wavelet-based compression allows JPEG 2000 to compress at high ratios without sacrificing quality. However, these methods may be too computationally intensive for real-time big data processing.

Run-Length Encoding (RLE) is a simple and efficient lossy compression method that replaces repeated values with a single value and count [12]. A string of ten "A"s is "A10." Bitmaps and other data with long repeated values are good RLE candidates. Processing is fast and computational overhead is low due to its simplicity. However, RLE doesn't work for highly variable data because repetitions are rare, preventing compression gains.

C. HYBRID COMPRESSION TECHNIQUES

Hybrid compression algorithms combine lossy and lossless algorithms for optimal results. Multimedia codecs like HEVC and H.264 use transform coding for lossy video frame compression and lossless entropy coding for metadata and headers, like Arithmetic Coding [13]. This combination preserves important data while achieving high compression ratios. These methods excel in big data applications like streaming and video analytics, where efficiency and quality are crucial.

D. ADVANTAGES AND LIMITATIONS OF EACH METHOD

Compression algorithms are optimal for certain tasks due to their pros and cons. Use LZW, Arithmetic Coding, or Huffman Coding for lossless data recovery of text files, databases, or important logs. However, their compression ratios are lower than lossy methods. Although they may cause artefacts or quality loss, lossy algorithms like DCT and wavelet-based methods compress multimedia files well. Hybrid approaches can optimise size and quality, but they are complex and computationally intensive.

Big data compression algorithms depend on data type, use, and storage efficiency vs. processing demands. Lossy algorithms work better for photos and videos than lossless algorithms for structured and critical data. For storage and resource efficiency, big data ecosystems will always need compression algorithms.

V. ROLE OF COMPRESSION IN BIG DATA FRAMEWORKS

Large amounts of data are created and processed in the big data era, making compression essential for storage management and performance optimisation. Cloud storage platforms and big data ecosystems like Hadoop and Spark manage massive amounts of diverse data [14]. These frameworks improve storage costs, data transfer speeds, and processing efficiency by using efficient compression algorithms. Modern big data applications use compression to scale, speed up, and reduce storage.

A. INTEGRATION OF COMPRESSION ALGORITHMS IN BIG DATA ECOSYSTEMS

Big data frameworks are used for datasets too large for conventional systems. Since these systems are scalable, data compression algorithms are integrated into their processing and storage layers to optimise performance. Compressing data before storage allows these systems to fit datasets into distributed storage clusters [15]. Data compression during processing

speeds up analysis and reduces network bandwidth by reducing data read and transferred. Big data frameworks like Hadoop and Spark compress and decompress large datasets before processing them to reduce storage costs. These frameworks allow real-time data analytics with high compression ratios by balancing processing speed and use-case-specific compression methods.

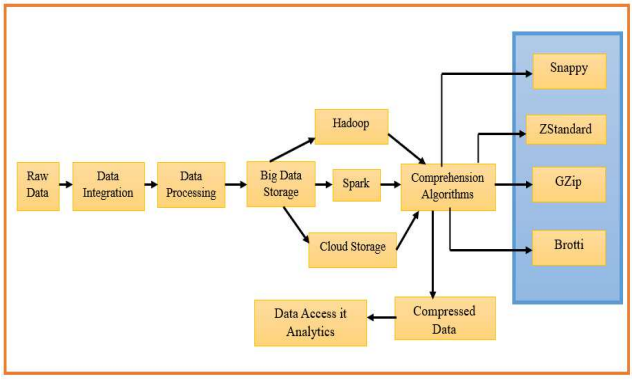

FIGURE 2 Big Data Storage Architecture

B. USE CASES OF COMPRESSION ALGORITHMS IN BIG DATA FRAMEWORKS

1. HADOOP: SNAPPY AND LZO

Hadoop, a popular distributed processing and storage framework, optimises storage and performance with compression. In the Hadoop Distributed File System (HDFS), which distributes data across many nodes, Snappy and LZO are widely used to compress data [16].

Snappy's speed-focused compression ratio attracts Hadoop users. Its efficient compression and decompression mechanisms make it ideal for real-time or near-real-time applications.

Snappy speeds up data processing, which is essential for Hadoop's batch processing model despite its low compression ratios. Many workflows use it, including log management, data ingestion, and streaming analytics. Another lightweight Hadoop algorithm is LZO, which prioritises speed and moderate compression ratios. Applications that need real-time data compression and decompression without delays benefit from this feature. For time-sensitive large-scale data processing, LZO and Snappy are faster than more computationally intensive algorithms despite having lower compression ratios.

2. SPARK: ZSTD AND BROTLI

Apache Spark's in-memory big data processing is quick thanks to efficient compression methods. Spark often uses ZSTD and Brotli compression to boost performance.

ZSTD, an advanced compression method, balances processing speed and compression ratio [17]. ZSTD speeds up read/write speeds and lowers transmission and storage costs in Spark, making it ideal for remote computing and large datasets. Spark and the algorithm's distributed nature enable efficient decompression and I/O bottleneck reduction.

Spark uses Brotli, originally for web compression, for data storage and transmission. Spark tasks that handle log files, JSON, and other textual data may benefit from its ability to compress large volumes of text. Brotli is a suitable alternative to Gzip for Spark workload optimisation, especially in cloud situations, because it has greater compression ratios and comparable speeds.

3. CLOUD STORAGE: GZIP AND PARQUET

Modern big data architectures use Amazon S3, Google Cloud Storage, and Azure Blob Storage. Optimising storage and data transport requires compression. Cloud storage companies use Gzip and Parquet for massive data compression. Gzip is a popular data compression method. It is supported by various cloud storage services and big data frameworks because to its efficiency and simplicity. Gzip is ideal for compressing CSV files, logs, and other unstructured data because to its excellent compression ratios [18]. Cloud systems often compress data with Gzip before storage to reduce storage costs and network bandwidth. Columnar storage file format Parquet is suitable for Hadoop and Spark. Parquet includes snappy compression and lets users choose from Gzip and LZO. Because it compresses columns, Parquet is great for analytical queries because it decreases storage costs and improves query performance. For big data applications that need scalable, fast data processing, cloud-stored Parquet files are perfect for storing enormous datasets for analytics.

C. CASE STUDIES AND EXAMPLES OF COMPRESSION IN REAL-WORLD BIG DATA FRAMEWORKS

1. Netflix: Compression for Streaming Video Analytics

Industry leaders like Netflix employ Spark and Hadoop to process large amounts of user and streaming data. Video storage, transport, and user data analysis depend on compression. Snappy and Zstandard compression help Netflix save money on storage and improve its recommendation systems [19]. Netflix compresses logs and user behaviour data to process billions of user interactions daily with low latency and high throughput.

2. LinkedIn: Real-time Data Processing with Kafka

LinkedIn uses distributed streaming technology Apache Kafka for real-time data analytics. Snappy

and Gzip compress Kafka messages to save space and bandwidth. LinkedIn reduces its data storage demands by compressing streaming data before storage without affecting its ability to quickly process huge volumes of incoming data. LinkedIn can provide real-time user interactions and platform performance metrics [21]. Compression strategies boost massive data framework efficiency. These methods reduce storage needs, speed up data transport, and improve data processing task performance, ensuring huge data system scalability. Data and application requirements should determine lossless or lossy compression.

Modern compression techniques are needed to solve large-scale data system storage and processing problems, which will only expand as big data grows.

VI. COMPARISON OF POPULAR COMPRESSION ALGORITHMS

Different compression algorithms work optimal with different data formats and have different speed, compression ratios, and efficacy. Knowing these differences is crucial when picking a large data algorithm since they directly affect processing speed and storage efficiency. The table below compares popular big data compression methods. These algorithms are Snappy, ZSTD, LZ77/LZW, Gzip, and Brotli.

TABLE 1 COMPARISON OF POPULAR COMPRESSION ALGORITHM

Compression Algorithm	Compression Ratio	Speed (Compression/ Decompression)	Suitability for Different Types of Data	Use Cases
Snappy	Moderate	Very Fast (Compression: Fast, Decompression: Very Fast)	Ideal for structured and semi-structured data such as logs, CSV, or simple datasets	Hadoop, Spark, real-time processing, log data
Zstandard (ZSTD)	High	Moderate (Compression: Fast, Decompression: Very Fast)	Suitable for large, high-volume datasets such as transactional logs and analytical data	Hadoop, Spark, cloud storage, file systems
LZ77/LZW	Moderate to High	Fast to Moderate	Works well with text data, codebooks, and simple data	General-purpose compression, text-based files
Gzip	High	Moderate to Slow	Optimal for compressing text-based data, including logs, CSV, and XML	Cloud storage, file systems, web applications
Brotli	Very High	Moderate	Effective for compressing text and web data, especially HTTP compression	Web applications, cloud storage, static file serving

A. COMPRESSION RATIO

Zstandard (ZSTD) and Gzip have the optimal compression ratios for space savings. Gzip excels at text-heavy material, but ZSTD balances speed and compression. Snappy may not have the optimal compression ratios, but its lightning-fast performance makes it suitable for real-time or high-performance settings where every second counts. LZ77/LZW compresses moderately, making it suitable for smaller datasets or older compression methods.

B. SPEED

Snappy's lightning-fast compression and decompression speeds are ideal for real-time applications that value speed over compression efficiency. Its fast data processing benefits Hadoop and Spark, two massive data systems. Zstandard (ZSTD) is ideal for storage and processing due to its fast compression and decompression rates, especially in high-throughput applications. Gzip is slower than LZ77/LZW but better for compression ratio over speed [22].

C. SUITABILITY FOR DIFFERENT TYPES OF DATA

Snappy is optimal for semi-structured or log-based data if speed trumps compression ratio.

ZSTD is adaptable and works with multimedia files, large databases, and transactional logs [23]. With sophisticated, high-volume data storage activities, LZ77/LZW may not be as efficient as with text-based data or simpler datasets. Gzip can compress logs, CSV, and XML in distributed storage systems with limited bandwidth. Brotli optimises web-based data compression, especially HTTP traffic, and

compresses static things like scripts and images faster.

D. CHOOSING THE RIGHT ALGORITHM

Due to their lightning-fast speeds and optimum compression/decompression combo, Snappy and ZSTD surpass the competition in real-time processing in Spark and Hadoop. Gzip and ZSTD reduce storage and processing time, making them ideal for cloud storage or archive data that needs compression ratio.

Brotli optimises web traffic and HTTP compression for websites that use CDNs or need fast data transfer [24]. Storage efficiency, data type, and processing speed determine which compression method is ideal for a big data application. Gzip and Brotli are good for high-compression applications, whereas Snappy and ZSTD are good for speed and real-time performance. Understanding compression ratio and speed trade-offs helps big data frameworks optimise storage and performance.

VII. FUTURE TRENDS AND INNOVATIONS

AI and machine learning could dramatically improve compression methods in the future. These methods are promising for context-aware and adaptive compression, which optimises processing speed and storage economy by making real-time adaptations based on data attributes. AI-driven compression improves compression efficiency without quality loss by studying data patterns. Quantum computing can use quantum states to compress enormous amounts of data at record rates, which could revolutionise data compression. Thus, quantum compression research is intriguing. AI-driven approaches are computationally complex, large-scale systems must process in real time, and quantum computing's practical uses are unclear. Compression techniques are constantly changing, but they can improve data storage, processing efficiency, and big data application possibilities while overcoming these challenges.

VIII. CONCLUSION

Compression techniques can aid with enormous data storage, as mentioned in this article. We covered data compression basics, including lossless and lossy compression and big data applications. Compression algorithms include Huffman coding, Zstandard, Snappy, and Gzip. Each has strengths in compression ratio, speed, and data type compatibility. We examined how Hadoop and Spark use the algorithms to improve storage optimisation and data accessibility. For optimal compression, storage efficiency, processing speed, and data type, choose the right algorithm. Data storage and processing may change soon due to AI-driven and quantum compression. Though the field's constant evolution presents challenges and opportunities, big data storage's future is bright with better and more adaptable compression approaches.

REFERENCE

[1] M. Pandey, S. Shrivastava, S. Pandey, and S. Shridevi, "An enhanced data compression algorithm," in *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, 2020, pp. 1–4.

[2] J. Latif, P. Mehryar, L. Hou, and Z. Ali, "An efficient data compression algorithm for real-time monitoring applications in healthcare," in *2020 5th International Conference on Computer and Communication Systems (ICCCS)*, 2020, pp. 71–75.

[3] T. A. S. Srinivas, S. Ramasubbareddy, G. Kannayaram, and C. P. Kumar, "Storage optimization using file compression techniques for big data," in *FICTA (2)*, 2020, pp. 409–416.

[4] A. N. Kahdim and M. E. Manaa, "Design an efficient Internet of Things data compression for healthcare applications," *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 3, pp. 1678–1686, 2022.

[5] H. Astsatryan, A. Lalayan, A. Kocharyan, and D. Hagimont, "Performance-efficient recommendation and prediction service for big data frameworks focusing on data compression and in-memory data storage indicators," *Scalable Computing: Practice and Experience*, vol. 22, no. 4, pp. 401–412, 2021.

[6] S. Kalaivani, C. Tharini, K. Saranya, and K. Priyanka, "Design and implementation of hybrid compression algorithm for personal health care big data applications," *Wireless Personal Communications*, vol. 113, no. 1, pp. 599–615, 2020.

[7] K. Meena and J. Sujatha, "Reduced time compression in big data using MapReduce approach and Hadoop," *Journal of Medical Systems*, vol. 43, no. 8, p. 239, 2019.

[8] J. Song, S. Hu, Y. Bao, and G. Yu, "Compress blocks or not: Tradeoffs for energy consumption of a big data processing system," *IEEE Transactions on Sustainable Computing*, vol. 7, no. 1, pp. 112–124, 2020.

[9] Bakir, "New blockchain based special keys security model with path compression algorithm for big data," *IEEE Access*, vol. 10, pp. 94738–94753, 2022.

[10] Yu, S. Lu, T. Wang, X. Zhang, and S. Wan, "Towards higher efficiency in a distributed

memory storage system using data compression," *International Journal of Bio-Inspired Computation*, vol. 20, no. 4, pp. 232–240, 2022.

[11] S. Qi, J. Wang, M. Miao, M. Zhang, and X. Chen, "Tinyenc: Enabling compressed and encrypted big data stores with rich query support," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 1, pp. 176–192, 2021.

[12] Carpentieri, "Data compression in massive data storage systems," in *2024 International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA)*, 2024, pp. 1–6.

[13] Hu, F. Wang, W. Li, J. Li, and H. Guan, "QZFS: QAT accelerated compression in file system for application agnostic and cost efficient data storage," in *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, 2019, pp. 163–176.

[14] U. Narayanan, V. Paul, and S. Joseph, "A novel system architecture for secure authentication and data sharing in cloud enabled big data environment," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 6, pp. 3121–3135, 2022.

[15] H. Yao, Y. Ji, K. Li, S. Liu, J. He, and R. Wang, "HRCM: An efficient hybrid referential compression method for genomic big data," *BioMed Research International*, vol. 2019, no. 1, p. 3108950, 2019.

[16] J. Chen, M. Daverveldt, and Z. Al-Ars, "FPGA acceleration of ZSTD compression algorithm," in *2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2021, pp. 188–191.

[17] S. Vatedka and A. Tchamkerten, "Local decode and update for big data compression," *IEEE Transactions on Information Theory*, vol. 66, no. 9, pp. 5790–5805, 2020.

[18] G. Xiong, "Research on big data compression algorithm based on BIM," in *2021 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS)*, 2021, pp. 97–100.

[19] S. Pal, S. Mondal, G. Das, S. Khatua, and Z. Ghosh, "Big data in biology: The hope and present-day challenges in it," *Gene Reports*, vol. 21, p. 100869, 2020.

[20] K. Sansanwal, G. Shrivastava, R. Anand, and K. Sharma, "Big data analysis and compression for indoor air quality," in *Handbook of IoT and Big Data*, CRC Press, 2019, pp. 1–21.

[21] S. A. Abdulzahra, A. K. M. Al-Qurabat, and A. K. Idrees, "Data reduction based on compression technique for big data in IoT," in *2020 International Conference on Emerging Smart Computing and Informatics (ESCI)*, 2020, pp. 103–108.

[22] Zhang et al., "Compress DB: Enabling efficient compressed data direct processing for various databases," in *Proceedings of the 2022 International Conference on Management of Data*, 2022, pp. 1655–1669.

[23] A. Abdo, T. Salem Karamany, and A. Yakoub, "Enhanced data security and storage efficiency in cloud computing: A survey of data compression and encryption techniques," vol. 6, no. 2, pp. 81–88, 2024.

[24] R. Pratap, K. Revanuru, R. Anirudh, and R. Kulkarni, "Efficient compression algorithm for multimedia data," in *2020 IEEE Sixth International Conference on Multimedia Big Data (BigMM)*, 2020, pp. 245–250.