Journal of Artificial Intelligence for Automation
https://www.sciltp.com/journals/jaia

Article

Check for updates

Compressor-Aware Feature Analysis and Learning-Based Performance Prediction for Scientific Data Compression

Zhenlu Qin [1], Qirui Tian [2], Lei Wu [1,*] and Jinzhen Wang [3,*]

[1] Department of Computer Science & Computer Information Systems, Auburn University at Montgomery, Montgomery, AL 36117, USA

[2] Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102, USA

[3] Department of Computer Science, University of North Carolina at Charlotte, Charlotte, NC 28223, USA

* Correspondence: lwu@aum.edu (L.W.); jwang96@charlotte.edu (J.W.)

How To Cite: Qin, Z.; Tian, Q.; Wu, L.; et al. Compressor-Aware Feature Analysis and Learning-Based Performance Prediction for Scientific Data Compression. *Journal of Artificial Intelligence for Automation* **2026**, *1*(2), 13. https://doi.org/10.53941/jaia.2026.100013

Received: 24 May 2026
Revised: 21 August 2026
Accepted: 27 August 2026
Published: 21 September 2026

Abstract: High-fidelity scientific instruments and simulations are producing data at unprecedented volumes and rates, imposing substantial pressure on storage, transmission, and analysis. In order to alleviate the challenges brought to high-performance computing I/O and storage, compression techniques have been introduced in various scenarios, but understanding compression performance remains challenging due to the complex interactions among data characteristics and compressor-internal behaviors. In this paper, we analyze the internal behaviors of SZ, a representative prediction-based error-bounded lossy compressor, and identify representative compressor-related features for performance prediction. We then develop learning-based models to predict compression ratio and throughput, and further design a simplified prediction model using representative features. We compare the proposed method with existing sample-based and white-box prediction methods. The results show that compressor-internal features are important for performance prediction, but the best prediction method is scenario-dependent.

Keywords: high-performance computing; data compression; learning-based model; performance prediction

1. Introduction

Scientific simulations running on high-performance computing (HPC) systems are generating data at rapidly increasing scales and resolutions. This trend is driven by advances in computing hardware, accelerator technologies, and the scientific software ecosystem. To ensure numerical accuracy, simulations often adopt fine spatial and temporal resolutions, which can introduce substantial redundancy into the generated data. As data volume continues to grow, storing, moving, and analyzing these outputs become increasingly expensive. Data reduction is therefore becoming an important technique for reducing storage, transmission, and post-processing costs. In the past decade, several scientific lossy compressors, such as ZFP [1,2], SZ [3–7], and MGARD [8–10], have been developed for different application scenarios. Compared with lossless compression, which is mainly used when information loss is not allowed, lossy compression can achieve higher compression ratios by allowing controlled information loss. Therefore, lossy compression has been widely used for scientific data analysis and visualization. However, compression performance is affected by multiple factors, including data characteristics, compression algorithms, and user-specified configurations. The trade-off among compression ratio, compression throughput, and data quality is complex and nontrivial, making performance understanding and prediction important for both users and compressor designers.

There are numerous studies that focus on analyzing compression performance. A recent survey summarizes the state of the art in lossy compression technologies [11]. Claggett et al. [12] integrate and compare state-of-the-art

lossless compression algorithms for performance understanding. As for the lossy compression, Tao et al. [13] propose an online lightweight selection algorithm that helps users choose between SZ and ZFP to satisfy compression quality requirements. Liang et al. [14] optimize compression ratio by testing different SZ configurations combined with decimation methods. Another study [15] comprehensively evaluates compressor performance by considering compressor design, data features, performance analysis, and compression-ratio prediction. Wang et al. [16] further develop a white-box analytical model to predict the compression ratios of SZ and ZFP by analyzing their internal mechanisms. More recently, Artificial Intelligence (AI) has also been introduced into scientific data compression. These studies incorporate AI-based methods into compression design to improve compression ratio, error control, or data reconstruction quality [17–23]. For example, Jia et al. [20] propose an online neural learning method that adapts during runtime to improve the effectiveness of scientific lossy compression. Liu et al. [18] propose a deep learning-based scientific lossy compression method utilizing the HAT super-resolution network. These studies show the potential of AI in compression performance prediction and optimization. Such models achieve promising compression ratios due to the superior capability of deep learning models in capturing data distributions. However, existing performance prediction studies are often based on compressor-specific white-box analysis, which may require detailed knowledge of internal mechanisms and complex analytical modeling. Meanwhile, recent learning-based compression studies mainly focus on compression design for specific data or model types. It remains insufficiently understood how compressor-internal features affect compression performance and how such features can be used for reliable prediction of compression performance. This motivates our compressor-aware feature analysis and learning-based performance prediction framework.

In this paper, we analyze scientific compression performance from the perspective of compressor-internal features. Using SZ as a representative example, we first study how feature-related components affect compressed size and throughput. Based on these remarks, we develop learning-based models to predict compression ratio and throughput using compressor-related features. We further study whether a simplified model based on representative features can preserve prediction accuracy while reducing the feature requirement. Finally, we compare the proposed learning-based method with existing methods to understand their advantages, limitations, and suitable scenarios. The contributions of this paper are summarized as follows.

- We conduct a feature-level analysis of SZ compression performance by studying important compressor-internal features. The analysis shows that these features play an important role in compression performance, but their effects must be carefully examined to support accurate performance estimation.
- We develop learning-based models to predict compression ratio and throughput using compressor-related features, and analyze the relationship between representative features and performance.
- We design a simplified prediction model based on representative features and evaluate its effectiveness for compression-ratio prediction. We also compare the proposed learning-based method with existing methods.

The remainder of this paper is organized as follows. Section 2 provides the background and motivation for this study. Section 3 presents the feature-level analysis of SZ compression performance. Section 4 introduces the learning-based prediction models and discusses the impact of compressor-related features on performance prediction. Section 5 concludes the paper.

2. Background and Motivation

2.1. Scientific Data Compression and SZ Description

Scientific data compression is typically done in situ before data are moved to persistent storage or being transmitted on the network, in order to reduce the I/O overhead. Depending on whether there is a loss of accuracy during compression, scientific data compression in general is categorized into lossless (e.g., Zstd [24], Blosc [25], gzip [26, 27]) and lossy compression (e.g., SZ [3–7], ZFP [1] and MGARD [8–10]). Recent studies have also explored compression for other multidimensional data-intensive applications. For example, Klus et al. [28] proposed EWOk for efficient multidimensional compression of indoor positioning datasets. Performance metrics for scientific data compression generally include compression ratio, which is the ratio between original data size and compressed data size, and compression throughput (MB/s), which is the original data size divided by compression time. As for the error-bounded lossy compression, users typically need to specify an error bound before compression in order to control the information loss. During compression, the error bound is strictly observed, and error metrics, such as Root Mean Squared Error (RMSE) and Peak Signal-to-Noise Ratio (PSNR), are used to assess the data quality after compression. On the other hand, the performance of a floating-point compressor is also influenced by various factors, including data characteristics, compression algorithm, and compression configurations. In short, understanding and estimating the compression performance has become increasingly challenging.

In this paper, we use SZ version 2, referred to as SZ as a representative example, as SZ has been widely adopted in scientific data compression due to its high compression ratio and effective error-bounded compression capability. It first tries to represent the input data using predictive (Lorenzo-based and regression-based) methods. Then the prediction error values are mapped to quantization intervals if they fall within the prediction range. The prediction range can be calculated based on the error bound (e.g., relative error bound e) and the number of quantization intervals q. The effectiveness of the prediction H can be measured by the ratio of prediction-hit points N_{H} to the total points N_{T}, indicating the proportion of data points successfully predicted within the error bound. The quantization intervals are then encoded using a Huffman tree, with the size of the tree represented as S_{T}. This encoding takes advantage of the repeated occurrence of quantization intervals: the more frequently an interval occurs, the shorter its encoding size S_{E} will be. Since the distribution of quantization intervals directly affects Huffman encoding efficiency, we characterize this distribution using the most frequently occurring quantization intervals and their occurrence counts. Specifically, the index of the quantization interval with the i^{th} highest occurrence is denoted as S_i, and its occurrence count is denoted as C_i. Additionally, byte-level lossless entropy encoding, such as Zstd or Zlib, can be applied after Huffman encoding to further reduce the storage footprint. However, the decision to use these lossless entropy encoding methods depends on the desired trade-off between compression ratio and throughput. For data points whose prediction errors fall outside the check radius r, SZ treats them as curve-missed points. These points are stored separately through an error-bounded truncation procedure, where the truncated values are further represented with optimized leading-zero encoding to reduce the storage footprint. To facilitate the discussion, the datasets used in this work are listed in Table 1, and the notations and features used throughout the paper are shown in Table 2.

Table 1. Application descriptions. Note that each application may contain multiple datasets; therefore, the number of datasets used in this work is larger than the number of applications listed in this table. For example, NYX includes temperature and other scientific variables, which are treated as separate datasets. All datasets used in this work are FP64.

Name	# Points	Description
Astro	65,536	Velocity in a supernova simulation
Blast2	578,880	Pressure of strong shocks in a gas dynamical simulation
Bump	55,692	Flow density of an axisymmetric bump
Eddy	282,616	Velocity in a 2D solution to Navier Stokes equations
Yf17_p	97,104	Pressure in a computational fluid dynamics calculation
Yf17_t	97,104	Temperature in a computational fluid dynamics calculation
Fish	65,536	Velocity in a CFD calculation of cooling air being injected into a mixing tank
Sedov	78,144	Pressure of strong shocks in a hydrodynamical simulation
NYX	134,217,728	Adaptive mesh hydrodynamics and cosmological simulation
Miranda	37,748,736	Hydrodynamics code for large turbulence simulations
Exaalt	2,869,440	Molecular dynamics simulation
Dpot	20,694	Electric potential deviation in a plasma physics simulation

Table 2. Description of frequently used symbols.

Category	Symbol	Description
General	S	Data size.
	T	Compression time.
	N_{T}	Number of data points.
SZ settings	q	Number of quantization intervals.
	e	Relative error bound.
	l	Compression level.
	a	Lossless backend algorithms.
Data Prediction	D	Data range about the input dataset.
	r	Radius of prediction range.
	H	Efficiency of the data prediction.
Quantization	S_i	The quantization level with the i-th largest number of occurrences.
	C_i	The count of occurrences of S_i.

Table 2. *Cont.*

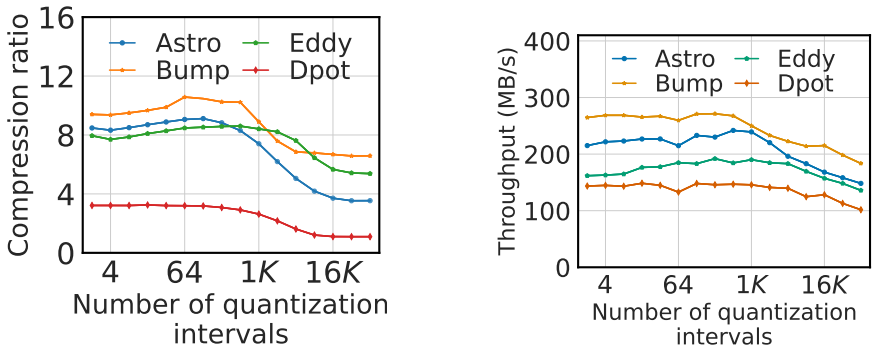
Category	Symbol	Description
Huffman encoding	N_R	Number of Huffman tree leaf nodes.
	S_T	Huffman tree size.
	S_E	Huffman encoding size.
	N_H	Number of hit points.
Binary Representation	S_L	The leading-zero encoding size.
	S_M	The mid-byte size .
	S_R	The residual size.

2.2. AI-Assisted Scientific Data Compression and Performance Prediction

Motivated by the capability of AI models to capture complex data distributions and nonlinear performance behaviors, there have been increasing efforts devoted to improving lossy data compression with AI algorithms. Several studies focus on the use of AI models for compression performance or quality prediction. Qin et al. [29] employ a deep-learning model for compression ratio prediction. They observed that incorporating compressor-related features into the model significantly improved prediction accuracy. Mumenin et al. [30] developed a deep surrogate model to predict the resulting quality of error-bounded lossy compression, enabling compression behavior to be estimated without directly performing full compression. AI models have also been incorporated into the design of scientific lossy compressors. Liu et al . [17] introduced an autoencoder-based lossy compression technique for scientific datasets, which improves compression ratio while maintaining high visualization fidelity. Recent AI-based scientific lossy compression methods have further incorporated online neural learning and super-resolution networks to improve compression effectiveness and reconstruction quality by capturing complex data distributions [18,20]. Related learning-based compression research has also explored quality prediction and learned compression for image and video coding and streaming [31–33]. More recent studies further extend AI-assisted compression to broader data types and learning architectures, including temporal graph autoencoders for error-bounded scientific data compression [22], foundation-model-based lossy compression for spatiotemporal scientific data [23], implicit neural representations for adaptive volumetric data compression [34,35], learned/context-based volumetric medical image compression [36], and topology-preserving error-bounded lossy compression [37]. Beyond scientific datasets, AI-related compression has also been studied for neural network models, including error-bounded lossy compression of deep neural network models [21] and tensor-decomposition-based rank reduction for deep learning model compression [38]. These studies show that AI-assisted compression has been widely explored for scientific data, volumetric data, and neural-network models.

2.3. Motivation

This paper is motivated by the fact that compression performance is highly sensitive to both compression configurations and data characteristics. We first illustrate this observation through SZ by examining the impact of the number of quantization intervals on compression performance. In Figure 1, we vary the number of quantization intervals q in SZ–a critical parameter that affects the number of Huffman tree nodes–and assess its impact on compression performance.



(a) Compression ratio **(b)** Compression throughput

Figure 1. Compression performance of SZ vs. the number of quantization intervals. The relative error bound is 10^{-5}.

It is observed that for the same dataset, different numbers of quantization intervals have different compression performance. The compression ratio and throughput are observed to be positively correlated, i.e., the higher the compression ratio is, the higher the throughput it will be. The reason is that a smaller q can generally lead to a higher occurrence of a quantization interval, thus resulting in higher compression ratios, and a smaller Huffman tree, thus faster encoding during compression.

In Figure 2, we show the compression ratio and throughput across 17 datasets for SZ, where the relative error bound is set to 10^{-3}. It is observed that both compression ratio and throughput vary significantly across datasets. For example, the highest compression ratio achieved for Blast2 is more than 20× higher than the lowest compression ratio for Nyx_temper.

Remark 1. *Compression performance is highly sensitive to both compressor features and datasets. Therefore, performance prediction requires a feature-level understanding of compressor-internal behaviors rather than relying only on coarse performance metrics.*

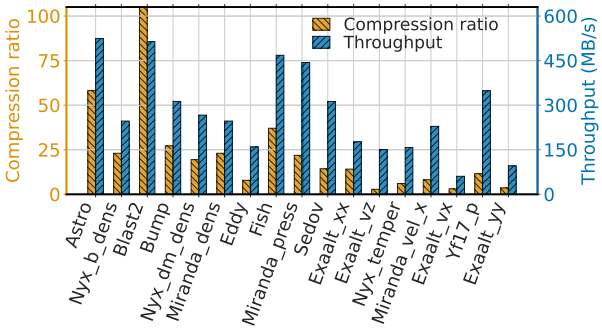


Figure 2. Compression performance of SZ on different datasets. The relative error bound is set to be $e = 10^{-3}$.

3. Feature-Level Understanding of SZ Compression Performance

In this section, we investigate how compressor-internal features affect SZ compression performance. This analysis helps explain the performance variations observed in Section 2.3 and provides the basis for the learning-based prediction models discussed later.

3.1. Impact of Quantization Intervals on Compressed Size

To explain the feature sensitivity observed in Figure 1, we first analyze how the number of quantization intervals affects the compressed file size. A larger number of quantization intervals q can cover more data points during prediction and thus increase the hit efficiency. However, as shown in Figure 1, the best compression performance does not necessarily occur at the highest q. In what follows, we aim to understand the impact of q on the compressed file size S_c. As described in Section 2.1, the output size for curve-hit data points is generally comprised of the Huffman tree size S_T, Huffman encoding size S_E; and the output size for curve-missed data points is composed of the leading-zero encoding size S_L, the mid-byte size S_M, and the residual size S_R. The size of the compressed file S_c can then be expressed as

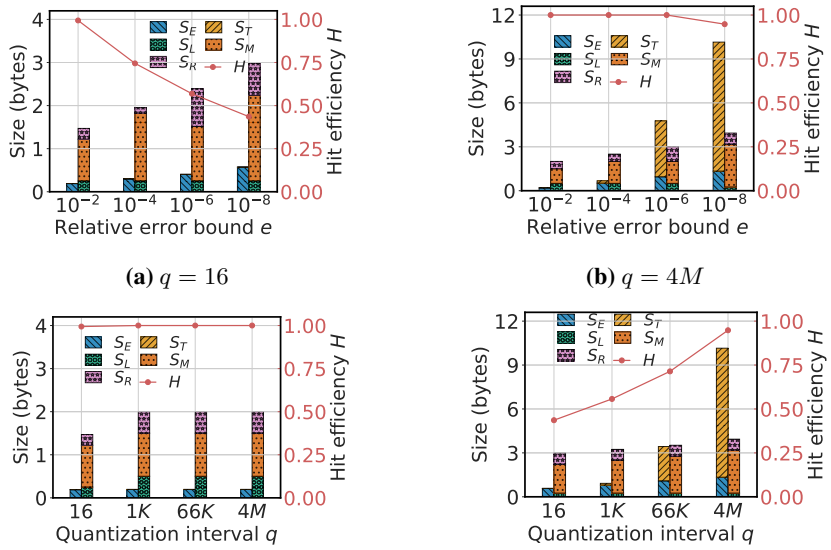
$$S_c = S_T + S_E + S_L + S_M + S_R \tag{1}$$

In Figure 3, we measure the size per point (plotted in bars) for curve-hit and curve-missed data points, together with the hit efficiency H (plotted in line). H is defined as the ratio of the number of hit points N_H to the total number of points N_T. As e becomes tighter, the required size per data point generally increases, as shown in Figure 3b. However, the per-point sizes of curve-hit and curve-missed data points show different trends. At relatively loose error bounds, such as 10^{-2} and 10^{-4}, the size per point for curve-hit data points is significantly lower than that for curve-missed data points, indicating that Huffman encoding provides a more efficient representation. As the error bound becomes tighter, this relation is reversed, and representing curve-hit data points becomes more expensive than representing curve-missed data points. This increase in the curve-hit size is mainly caused by the growth of the Huffman tree size S_T. Such a change is directly related to the increase in the number of Huffman tree leaf nodes N_R [15], which is further affected by the distribution of quantization intervals. When e becomes smaller, more quantization intervals are involved in Huffman encoding, and the average bit length of Huffman codes increases accordingly.

For curve-missed data points, the size increase is mainly contributed by S_M. As q increases, the number of missed points decreases, and the difference between the remaining missed points and the original median value increases. Consequently, the byte-level XOR representation can expose fewer leading zeros for each missed data

point, resulting in a gradual increase in S_{M}. In other words, when q is small, although more points are missed, these missed points can be represented more efficiently by leading-zero encoding, resulting in a smaller S_{M}. As shown in Figure 3, the size per point of the missed part is more stable than that of the curve-hit part. This is due to the fact that truncating a fixed number of bits is sufficient to maintain accuracy for the curve-missed data points. Furthermore, when q is large in Figure 3d, such as $q \geq 66K$, the Huffman encoding size S_{E} and the Huffman tree size S_{T} increase significantly, leading to a substantial increase in the compressed file size. Therefore, a smaller q may reduce the hit efficiency H, but it can also improve the compression ratio by reducing the Huffman-related storage cost for curve-hit data points.

Remark 2. *Under tight relative error bounds, increasing the number of quantization intervals can increase the percentage of curve-hit points. However, an excessively large number of quantization intervals can also increase the compressed file size due to the increased Huffman encoding size.*



(a) $q = 16$ **(b)** $q = 4M$

(c) $e = 10^{-2}$ **(d)** $e = 10^{-8}$

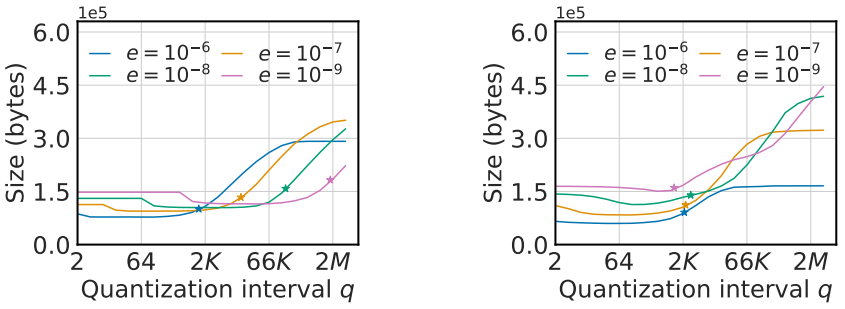
Figure 3. The average output size per data point and the data prediction efficiency across relative error and quantization intervals. We use the dataset Eddy as the example.

3.2. Trade-Off between Huffman Encoding and Truncation

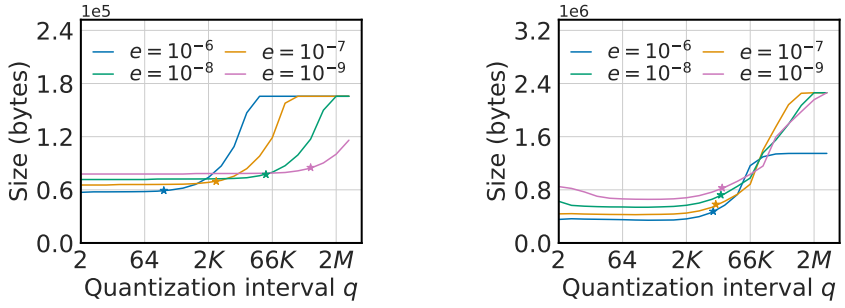
As previously mentioned, simply increasing the number of quantization intervals q to allow more points to be hit does not necessarily reduce the compressed file size, especially when e is tight. To further understand this behavior, we conduct a baseline experiment where only truncation is used for compression without Huffman encoding. This baseline mimics an extreme case in which all data points are treated as prediction-missed points, allowing us to isolate the effectiveness of truncation. Figure 4 illustrates the role of truncation, where asterisks ($*$) represent the compressed file size when only truncation is used. Empirically, as the number of quantization intervals increases, more points are represented by Huffman encoding. However, this does not always lead to a smaller compressed file size. For tight e, the truncation-only baseline often produces a compressed size comparable to or smaller than the regular SZ output, suggesting that truncation alone can be effective under such settings.

This trade-off between Huffman encoding and truncation is influenced by the fixed nature of the compression through truncation once e is determined. However, as q increases, the effectiveness of Huffman encoding diminishes due to the need for more bits to encode lower-frequency symbols, as shown in Figure 3b and Figure 3d. If the size of the compressed file achieved solely through truncation does not exceed the normal SZ compressed file size, truncation for missed data points is deemed to be very effective.

We also analyze the impact on throughput using only truncation and different numbers of quantization intervals, as shown in Figure 5. The overall trend is that throughput first increases and then decreases as the number of quantization intervals increases. The increased cost of tree construction associated with higher quantization intervals is offset by encoding more data points, thereby improving overall throughput. However, the subsequent decreasing trend suggests that the additional overhead of building the tree is not sufficiently offset by the additional use of Huffman coding points.

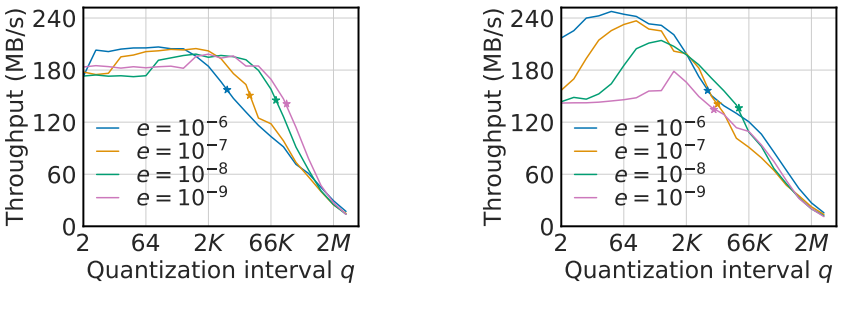


(a) Astro **(b)** Bump

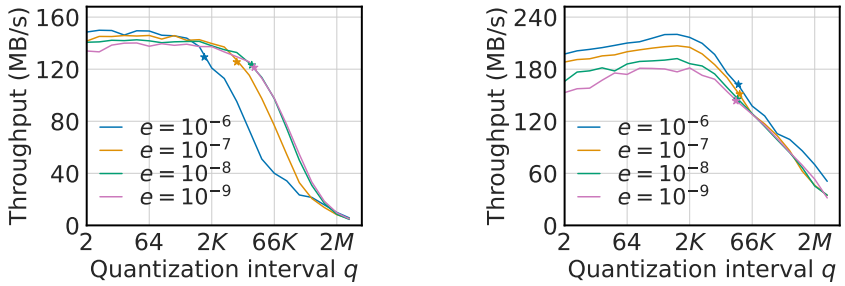


(c) Dpot **(d)** Eddy

Figure 4. The relationship between number of quantization intervals and compressed size under the tighter relative error bounds e. The symbol $*$ indicates the baseline where only truncation is used without quantization and Huffman encoding.



(a) Astro **(b)** Bump



(c) Dpot **(d)** Eddy

Figure 5. The relationship between number of quantization intervals and throughput under the tighter relative error bounds e. The symbol $*$ indicates the baseline where only truncation is used without quantization and Huffman encoding.

4. Learning-Based Compression Performance Prediction

Based on the feature-level remarks in Section 3, we develop a learning-based model to predict compression performance from compressor-related features.

4.1. Model Design and Setup

The idea is to build a deep neural network to capture the complex interaction between compressor-internal features and compression performance. We first assemble the training data by running compression on 20 datasets [15,39]

across various settings. In order to alleviate the overhead of running compression, we randomly sample the original data at the rate of 10%, and the compression is carried out on the sampled data blocks of 64 bytes.

We design a fully connected neural network with three hidden layers and ReLU activation [29]. The model is implemented using TensorFlow/Keras 2.12.0 and trained using the RMSprop optimizer with an initial learning rate of 0.001. The dataset is randomly split into 75% training data and 25% testing data. The network consists of two output channels for predicted compression ratio and compression throughput, respectively. Both output channels use RMSE as the loss function, which is defined as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2}$$

where y_i and $\hat{y}_i$ are the ground truth and model output, respectively. The overall loss function is computed by combining the *RMSEs* of the two output targets. To determine the number of neurons N_{N} in each layer, we use the rule of thumb $N_{\mathrm{N}} = N_{\mathrm{S}}/(\rho \cdot (N_{\mathrm{I}} + N_{\mathrm{O}}))$ [40], where N_{I} and N_{O} are the numbers of input and output neurons, respectively, consistent with the number of extracted features. N_{S} denotes the number of instances in the training data, and ρ is a scaling factor ranging from 2 to 10. In this paper, we set ρ to 2, a commonly used basic setting. The model uses 30 extracted features and two output targets; therefore, $N_{\mathrm{I}} = 30$ and $N_{\mathrm{O}} = 2$.

Besides *RMSE*, we also use the coefficient of determination R^2, which is defined as follows:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{3}$$

where $\bar{y}$ represents the mean value of y_i. to evaluate the model's performance. A larger R^2 value indicates better agreement between the predicted and ground-truth values, with $R^2 = 1$ indicating a perfect fit. Additionally, we use the Pearson correlation coefficient R_{xy}, which is defined as follows:

$$R_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{4}$$

to quantitatively assess the linear relationship between each feature X and the target Y. Furthermore, we use Accumulated Local Effects (ALE) plots [41] to analyze the impact of representative features on the model prediction, which helps us understand how these features influence compression performance.

4.2. Prediction Accuracy of the Full-Feature Model

4.2.1. Prediction Accuracy

We first utilize the deep learning model to predict the compression ratio and compression throughput for SZ. The data are partitioned into training and testing sets, with 75% used for training and the remaining 25% used for testing. Based on the learning-based model described in Section 4.1, we develop regression models for SZ. We use R^2 to evaluate the prediction accuracy, as shown in Table 3. Overall, the training R^2 values reach 0.961, while the testing R^2 values reach 0.926 , indicating that the model can accurately predict both compression ratio and throughput.

Table 3. Regression performance.

Compressor	Performance	R^2 (Train)	R^2 (Test)
SZ	Throughput	0.972	0.941
	Compression ratio	0.961	0.926

4.2.2. Correlation-Based Feature Analysis

To understand the relationships between compressor-related features and performance targets, we use the Pearson correlation coefficient. Figure 6 shows the correlations of selected SZ features with compression ratio and compression throughput. These features are primarily derived from prior studies [16,29]. Overall, compression ratio and throughput show a strong positive correlation in Figure 6. This is because when SZ achieves a high compression ratio, Huffman encoding is usually effective and most data points are represented by a small number of frequently occurring quantization intervals. In this case, the compression throughput is also large, as discussed earlier in Section 3.

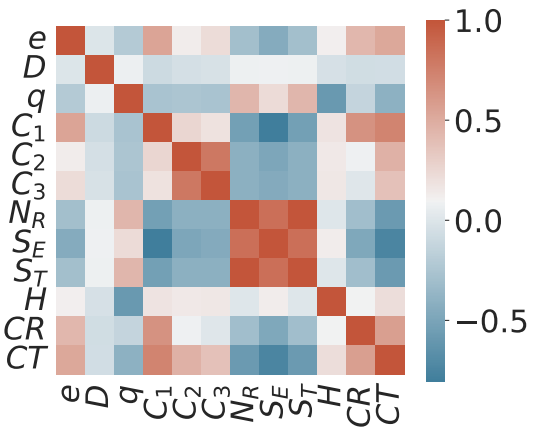


Figure 6. Pearson correlation coefficients for SZ between compression ratio and throughput.

The relative error bound e, features associated with quantization intervals, and features associated with Huffman encoding directly reflect compression performance, showing higher Pearson correlation coefficients in Figure 6. However, some other features, such as the efficiency of data prediction H and the data range D, have limitations in capturing the complete inner mechanisms of SZ. For instance, when the relative error bound e is loose (e.g., $e = 0.01$), the efficiency H can approach 1, yet the distributions of quantization intervals may still differ, leading to performance variations.

The strong correlation between compression ratio and throughput also provides a coarse way to infer the relative magnitude of one performance when the other is available. This correlation helps address whether compression is necessary when some performance values are already known. Our goal here is not to build a separate prediction model, but to examine whether such a cross-performance relationship can provide an order-of-magnitude estimate. By examining the available data, we find that 95% of the compression-ratio-to-throughput ratios fall within the range of 0.235 to 11.6. This range is commonly used by statisticians to describe the spread of data, helping us with performance estimation within an order of magnitude. Table 4 compares the compression ratio estimated from throughput with the actual compression ratio on four datasets. In this case, we use an empirical scaling factor of 10. The error, Err(%), is defined as the absolute relative error between the predicted value (Pred) and the actual value (Real):

$$\text{Err}(\%) = \frac{|\text{Pred} - \text{Real}|}{\text{Real}} \times 100\%.$$

Most predictions achieve an error rate below 65%, while larger errors generally occur under tighter relative error bounds. In this case, using throughput alone tends to overestimate the compression ratio.

Table 4. Compression ratio estimation from throughput via Pearson correlation.

Dataset	Metric	10^{-3}	10^{-4}	10^{-5}	10^{-6}
Astro	Real	28.72	12.22	5.92	3.57
	Pred	19.81	15.21	9.83	6.51
	Err(%)	31.02	24.44	66.14	82.41
Bump	Real	50.54	20.90	9.58	4.58
	Pred	19.60	17.67	13.22	8.95
	Err(%)	61.22	15.42	37.94	95.19
Eddy	Real	29.23	15.66	8.07	3.99
	Pred	26.69	23.83	16.47	9.16
	Err(%)	8.70	52.12	104.22	129.76
Fish	Real	34.77	17.54	8.41	6.42
	Pred	19.74	16.23	11.03	9.27
	Err(%)	43.22	7.47	31.13	44.39

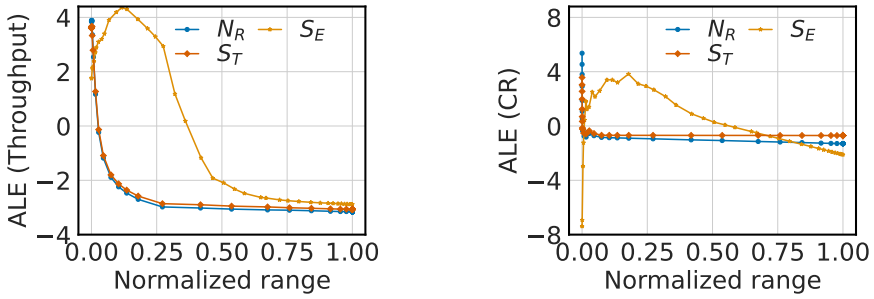
Remark 3. *For SZ, compression ratio and throughput show a strong correlation in the evaluated cases. This correlation can provide a coarse order-of-magnitude estimate across performance, although the empirical scaling factor should be selected with respect to the error bound.*

4.2.3. Interpretable Feature Effects through ALE

In addition to correlation analysis, we further analyze the impact of important features from the perspective of the trained model. ALE plots [41] have been proven to be a powerful tool and are used to show how changes in a feature affect the model prediction. Compared with simple correlation coefficients, ALE plots provide a model-based interpretation of feature effects.

In the context of Huffman encoding-associated features (N_R, S_T, and S_E), Figure 7 shows their effects on compression ratio and throughput. The curves of the number of Huffman tree nodes N_R and the Huffman tree size S_T show similar patterns. This similarity arises from the fact that the size of the Huffman tree is primarily determined by the frequency distribution of symbols in the input data, which influences the number of Huffman nodes. As both N_R and S_T increase, the predicted compression ratio and throughput decrease, indicating that a larger Huffman tree increases both storage and construction costs.

However, Huffman encoding size S_E exhibits a non-monotonic trend. The behavior of S_E is influenced by both the number of hit points and the size required for each hit point. Initially, as the normalized range of S_E increases, the increase is primarily driven by the number of hit points rather than the size per hit point. In such cases, a smaller number of quantization intervals can effectively cover enough points. However, as S_E continues to increase, more quantization intervals and size per hit point are required, leading to a decrease in compression ratio and compression throughput. A basic explanation of size per point and quantization intervals can be found in the previous discussion in Figure 3.



(a) Throughput **(b)** Compression ratio

Figure 7. The relation between the Huffman encoding associated features and normalized ALE performances for SZ.

4.3. Simplified Prediction with Representative Features

4.3.1. Model Design

Based on the preceding analysis, it is evident that not all features contribute equally to compression performance prediction. Therefore, we examine whether a smaller set of representative features can replace the full feature set while preserving prediction accuracy. In this section, we choose compression ratio as the prediction target, since it allows direct comparison with existing compression-ratio prediction methods. To select representative features, we use the Pearson correlation coefficients discussed earlier and keep features with compression ratio whose absolute correlation coefficients are greater than 0.5, which indicates moderate or strong correlation. These chosen features will serve as our representative variables for predicting compression ratio.

Inspired by knowledge distillation [42,43], we train a downstream submodel $\mathcal{S}$ with representative features by leveraging the prediction behavior of an upstream full model $\mathcal{F}$ trained with all SZ features. This design provides a more compact downstream model for compression-ratio prediction while preserving useful information learned by the full model. As shown in Figure 8, the full model $\mathcal{F}$ takes all SZ features as input, whereas the submodel $\mathcal{S}$ takes only the selected representative features. In other words, these features are two different representations of the same instance.

The training process begins by pre-training the full model $\mathcal{F}$ using all SZ features. After that, $\mathcal{F}$ is fixed and used to guide the training of the submodel $\mathcal{S}$. During forward propagation, both models $\mathcal{F}$ and $\mathcal{S}$ make predictions separately, and their outputs are compared to compute the prediction loss $\mathcal{L}_M$. This loss quantifies the discrepancy between the predictions of the full model and the sub-model. In contrast, during backpropagation, only the sub-model $\mathcal{S}$ is updated using the ground truth labels to optimize the prediction loss $\mathcal{L}_G$. The overall loss $\mathcal{L}_O$ is a combination of $\mathcal{L}_M$ and $\mathcal{L}_G$, weighted by an importance parameter θ, and $\mathcal{L}_O$ is defined as follows:

$$\mathcal{L}_O = \theta \cdot \mathcal{L}_M + (1 - \theta) \cdot \mathcal{L}_G \tag{5}$$

where θ is typically set to 0.1, as suggested by Hinton et al. [42], and controls the relative influence of mimicking the full model's predictions versus fitting the ground truth. In this way, the submodel can benefit from the knowledge encoded in the full model while maintaining a compact feature representation.

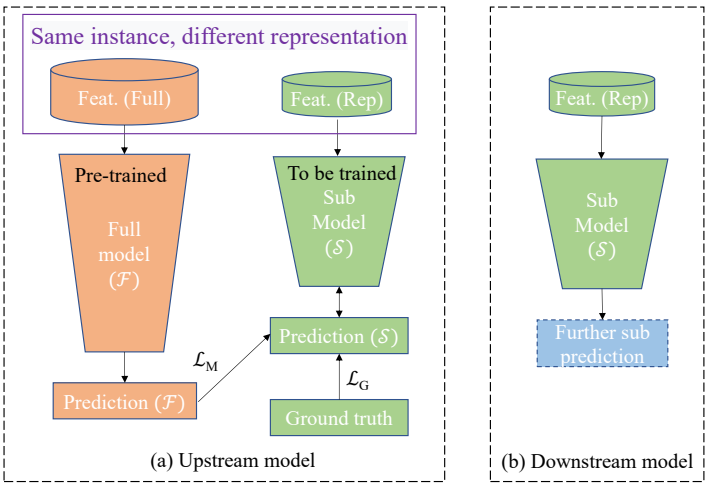


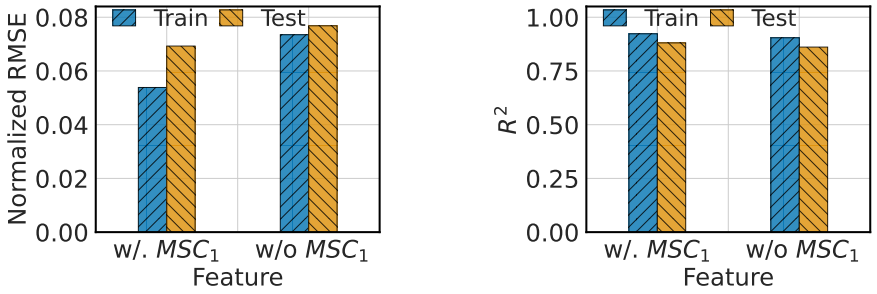
Figure 8. Structure of simplified learning model. (**a**) The upstream model, including the pre-trained full model $\mathcal{F}$ whose inputs are all SZ features, and the sub model $\mathcal{S}$ that contains only representative features. (**b**) The downstream model for which the input contains only representative features. In particular, the full and representative features of the upstream model represent the same instance.

4.3.2. Evaluation of the Simplified Prediction Model

We present the performance of the full model $\mathcal{F}$ and the submodel $\mathcal{S}$ in Table 5. Overall, the full model $\mathcal{F}$ slightly outperforms the submodel $\mathcal{S}$ because it uses the complete set of compression-related features. In contrast, the submodel $\mathcal{S}$ uses only representative features, resulting in a more compact but less complete representation. Nevertheless, the submodel still achieves comparable prediction accuracy. For instance, C_1, which has demonstrated its representative role in compression ratio in Figure 6, also plays a significant role in the distribution of quantization intervals. Therefore, we particularly focus on C_1 in the submodel $\mathcal{S}$ in Figure 9. It is evident that including C_1 reduces the normalized *RMSE* from 0.0768 to 0.0735, corresponding to a 4.30% reduction. This result indicates that C_1 not only serves as a representative feature in the correlation analysis but also improves the prediction accuracy of the learning-based model.

Table 5. Performance of Full $\mathcal{F}$ and sub $\mathcal{S}$ models.

Metrics	Train ($\mathcal{F}$)	Test ($\mathcal{F}$)	Train ($\mathcal{S}$)	Test ($\mathcal{S}$)
Normalized *RMSE*	0.037	0.055	0.051	0.074
R^2	0.965	0.946	0.935	0.916



(a) Normalized *RMSE* **(b)** R^2 score

Figure 9. Downstream model performance with normalized *RMSE* and R^2 score.

After the learning-based model is trained, it can be directly used for compression performance prediction. As shown in Figure 10, the prediction pipeline starts by sampling the scientific data whose compression performance needs to be estimated. To reduce the profiling overhead while preserving representative data characteristics, we

extract compressor-related features from the sampled data following the feature analysis described in Section 3. The extracted representative features are then fed into the trained submodel $\mathcal{S}$ to predict compression performance.

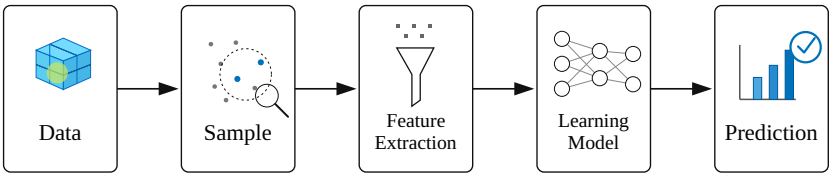


Figure 10. Pipeline of the proposed learning-based compression performance prediction workflow.

4.4. Comparison of Various Models

Based on the previous discussion, we compare the proposed learning-based method with existing compression-ratio prediction methods, and the results are presented in Table 6. Specifically, we differentiate these methods based on their use of sampling and its implementation. The Sample method [15] uses the compression ratio obtained from sampled data as the estimate for the full dataset. This method is simple to implement, but it usually has the largest error because the compression ratio measured from sampled data can be a biased estimate of the full-data compression ratio. The Sample w/Gaussian method [15] improves upon the sample-based estimate by assuming that the full dataset has the same hit efficiency H as the sampled data and that the distribution of quantization intervals follows a Gaussian distribution. Based on these assumptions, it estimates the number of Huffman tree nodes N_{R} for the full dataset and further estimates the Huffman tree size S_{T}, Huffman encoding size S_{E}, and outlier size. The Full w/Cross-bound method [16] uses similar Gaussian assumptions and compression statistics, but its objective is different: it predicts the compression ratio across different error bounds using statistics collected from the full dataset. This requires keeping the mean of the Gaussian distribution of quantization intervals consistent across error bounds while adjusting the number of quantization intervals in the tail to estimate the new Huffman tree nodes N_{R}.

Table 6. Compression ratio estimation using various methods. Bold values indicate the best results, and underlined values indicate the second-best results. $*$ indicates our proposed approach.

Method	Normalized *RMSE*	R^2
Sample	0.116	0.600
Sample w/Gaussian	0.103	0.680
Sample w/Learning $*$	**0.074**	**0.916**
Full w/Cross-bound	<u>0.080</u>	<u>0.878</u>

As for time complexity, the compared prediction methods mainly differ in their compression/profiling overhead. According to the SZ compression pipeline, prediction and quantization need to scan the data points and are mainly related to N_T. Huffman encoding requires collecting symbol-frequency information and constructing the Huffman tree, which introduces a cost related to the number of Huffman tree nodes, such as $O(N_R \log N_R)$. Therefore, the full-data compression cost can be roughly represented as $O(N_T + N_R \log N_R)$. For sampling-based methods, this cost is reduced because SZ is applied to sampled data, resulting in an approximate cost of $sO(N_T + N_R \log N_R)$, where s is the sampling ratio. The Sample w/Gaussian method further introduces distribution-fitting and tail-estimation overhead, which is related to $O(N_T)$. The Full w/Cross-bound method requires offline full-data compression and Gaussian fitting on the full dataset, with an additional fitting cost related to $O(N_T)$. The proposed learning-based method also includes an offline stage for feature extraction and model training, but its additional overhead during online prediction is small because it only performs sampled-data feature extraction and lightweight model inference and can be approximately represented as $sO(N_T + N_R \log N_R)$.

Overall, the proposed Sample w/Learning method achieves the best performance among the sample-based methods. As shown in Table 6, its *RMSE* is 28.16% lower than that of Sample w/Gaussian. This improvement comes from the learning model's ability to capture nonlinear relationships between compressor-related features and compression performance within the range covered by the training data. The comparison between the sample-based data prediction models and the cross-error bound full dataset prediction model Full w/Cross-bound can also be found in Table 6. In general, except for the Sample w/Learning model, which has an *RMSE* 7.5% smaller than the Full w/Cross-bound model, the rest of the sample-based models perform worse than the full data prediction. This is because more accurate compression-related statistics can be obtained from the full dataset.

Although the learning-based model achieves the best overall accuracy, it also has limitations. When the relative error bound e is very loose and the dataset behavior falls outside the range covered by the training data, it may lead to a higher potential for errors. Table 7 shows such an example using the Bump dataset under $e = 0.1$. In this case, the Full w/Cross-bound method achieves the best prediction accuracy, while the Sample w/Learning method does not perform as well as in the general cases. For the other sample-based methods, the errors mainly come from the fact that the sampled data do not accurately reflect the Huffman encoding behavior of the full dataset. In addition, ignoring the lossless backend can further increase the prediction error when the relative error bound is loose.

Table 7. Different methods for Bump compression ratio prediction and error rate under $e = 10^{-1}$. Bold: best prediction; underlined: second-best prediction.

Methods	Original	Sample	Sample w/Gaussian	Sample w/Learning	Full w/Cross-Bound
Compression ratio	604.53	185.39	189.54	<u>305.81</u>	**464.58**
Err(%)	0	69.30	68.65	<u>49.41</u>	**23.15**

Beyond prediction accuracy, Table 8 summarizes the practical requirements and major error sources of different methods. The Sample and Sample w/Gaussian methods require multiple compressions on sampled data for multiple predictions. The former does not require component-level information but has the largest error, while the latter records key compression components during sampling and uses Gaussian assumptions to improve prediction accuracy. However, it can introduce errors when estimating the Huffman tree nodes of the full dataset and ignoring the lossless compression backend. The Full w/Cross-bound method requires one full-data compression and can then support prediction across error bounds on the same dataset, but it also ignores the effect of the lossless backend. Finally, the proposed Sample with learning method requires extracting features consistent with the trained model during prediction.

Table 8. Comparison of different methods. 'C:P' denotes the compression-to-prediction ratio, where $n : n$ means that predicting n cases requires compressing the data n times, and $1 : n$ means that one compression can support n predictions. 'No Lossless' indicates whether the method ignores the lossless backend. The symbol '-' indicates that the item is not considered. The symbols ✓ and ✗ indicate that the corresponding feature is supported and not supported, respectively.

Method	Preparation			Scenario		Source of Error	
	Components	C:P	Source	Cross Error Bound	Cross Datasets	Est. N_R	No Lossless
Sample [15]	-	$n : n$	Sample	✗	✗	-	-
Sample w/Gaussian [15]	N_R, H, q, S_T, S_E	$n : n$	Sample	✗	✗	✓	✓
Sample w/Learning (Section 4.3)	Selected features	$n : n$	Sample	✓	✓	-	-
Full w/Cross-bound [16]	N_R, H, q, S_T, S_E	$1 : n$	Full	✓	✗	✓	✓

Remark 4. *The choice of prediction method is scenario-dependent. A learning-based model does not always outperform a white-box model, and a full-data method is not necessarily better than a sampling-based method when sampled features can effectively capture the compression behavior.*

5. Conclusions

In this paper, we analyze scientific compression performance from the perspective of compressor-internal features. Using SZ as a representative example, we study how feature-related components, such as quantization intervals, Huffman encoding, and truncation, affect compression performance. Based on these remarks, we develop learning-based models to predict compression ratio and throughput using compressor-related features. We then compare the proposed learning-based method with existing sample-based and white-box prediction methods. The experimental results show that compressor-internal features can provide useful information for compression performance prediction. However, the prediction errors indicate that the currently extracted features may still be insufficient to fully capture fine-grained compressor mechanisms, especially the behavior of Huffman encoding and the lossless backend. For sample-based prediction methods, the errors may also mainly come from the fact that the sampled data do not accurately reflect the Huffman encoding behavior of the full dataset. These findings suggest that more fine-grained compressor-internal features are needed for more robust performance prediction. Therefore, future work may further investigate finer-grained compressor-internal features and extend the analysis to more compressors and their corresponding compression pipelines. We will also investigate whether important

model parameters, such as the importance parameter θ, can be optimized using systematic or intelligent search strategies to further improve the robustness of the prediction framework.

Author Contributions

Z.Q.: Conceptualization, Methodology, Software, Investigation, Data curation, Formal analysis, Visualization, Writing—original draft, and Writing—review and editing; Q.T.: Validation and Data curation; L.W.: Writing—review and editing; J.W.: Methodology, Validation, Formal analysis, Supervision. All authors have read and agreed to the published version of the manuscript.

Funding

This research was supported by the AUM Start-up Fund (Grant No. 102001 215224).

Institutional Review Board Statement

Not applicable.

Informed Consent Statement

Not applicable.

Data Availability Statement

The datasets analyzed in this study are publicly available from the corresponding original repositories and sources cited in the paper.

Conflicts of Interest

The authors declare no conflict of interest.

Use of AI and AI-Assisted Technologies

During the preparation of this work, the authors used ChatGPT for language polishing and improving readability. The authors reviewed and revised all AI-assisted content and take full responsibility for the content of the published article.

References

1. Lindstrom, P. Fixed-Rate Compressed Floating-Point Arrays. *IEEE Trans. Vis. Comput. Graph.* **2014**, *20*, 2674–2683. https://doi.org/10.1109/tvcg.2014.2346458.
2. Lu, B.; Li, Y.; Wang, J.; et al. ZFP-X: Efficient Embedded Coding for Accelerating Lossy Floating Point Compression. In Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS), St. Petersburg, FL, USA, 15–19 May 2023; pp. 1041–1050. https://doi.org/10.1109/ipdps54959.2023.00107.
3. Di, S.; Cappello, F. Fast Error-Bounded Lossy HPC Data Compression with SZ. In Proceedings of the 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS), Chicago, IL, USA, 23–27 May 2016; pp. 730–739. https://doi.org/10.1109/ipdps.2016.11.
4. Liang, X.; Di, S.; Tao, D.; et al. Error-Controlled Lossy Compression Optimized for High Compression Ratios of Scientific Datasets. In Proceedings of the IEEE International Conference on Big Data, Seattle, WA, USA, 10–13 December 2018; pp. 438–447. https://doi.org/10.1109/bigdata.2018.8622520.
5. Zhao, K.; Di, S.; Liang, X.; et al. Significantly Improving Lossy Compression for HPC Datasets with Second-Order Prediction and Parameter Optimization. In Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing, Virtual, 23–26 June 2020; pp. 89–100.
6. Zhao, K.; Di, S.; Dmitriev, M.; et al. Optimizing Error-Bounded Lossy Compression for Scientific Data by Dynamic Spline Interpolation. In Proceedings of the 2021 IEEE 37th International Conference on Data Engineering (ICDE), Chania, Greece, 19–22 April 2021; pp. 1643–1654. https://doi.org/10.1109/icde51399.2021.00145.
7. Liang, X.; Zhao, K.; Di, S.; et al. SZ3: A Modular Framework for Composing Prediction-Based Error-Bounded Lossy Compressors. *IEEE Trans. Big Data* **2023**, *9*, 485–498. https://doi.org/10.1109/tbdata.2022.3201176.
8. Ainsworth, M.; Tugluk, O.; Whitney, B.; et al. Multilevel Techniques for Compression and Reduction of Scientific Data—The Multivariate Case. *SIAM J. Sci. Comput.* **2019**, *41*, A1278–A1303. https://doi.org/10.1137/18m1166651.
9. Liang, X.; Whitney, B.; Chen, J.; et al. MGARD+: Optimizing Multilevel Methods for Error-Bounded Scientific Data Reduction. *IEEE Trans. Comput.* **2022**, *71*, 1522–1536.

10. Ainsworth, M.; Tugluk, O.; Whitney, B.; et al. Multilevel Techniques for Compression and Reduction of Scientific Data—Quantitative Control of Accuracy in Derived Quantities. *SIAM J. Sci. Comput.* **2019**, *41*, A2146–A2171. https://doi.org/10.1137/18m1208885.
11. Di, S.; Liu, J.; Zhao, K.; et al. A Survey on Error-Bounded Lossy Compression for Scientific Datasets. *ACM Comput. Surv.* **2025**, *57*, 287.
12. Claggett, S.; Azimi, S.; Burtscher, M. SPDP: An Automatically Synthesized Lossless Compression Algorithm for Floating-Point Data. In Proceedings of the 2018 Data Compression Conference, Snowbird, UT, USA, 27–30 March 2018; pp. 335–344. https://doi.org/10.1109/dcc.2018.00042.
13. Tao, D.; Di, S.; Liang, X.; et al. Optimizing Lossy Compression Rate-Distortion from Automatic Online Selection between SZ and ZFP. *IEEE Trans. Parallel Distrib. Syst.* **2019**, *30*, 1857–1871. https://doi.org/10.1109/tpds.2019.2894404.
14. Liang, X.; Di, S.; Li, S.; et al. Exploring Best Lossy Compression Strategy by Combining SZ with Spatiotemporal Decimation. In Proceedings of the 4th International Workshop on Data Reduction for Big Scientific Data, Dallas, TX, USA, 11 November 2018.
15. Lu, T.; Liu, Q.; He, X. Understanding and Modeling Lossy Compression Schemes on HPC Scientific Data. In Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS), Vancouver, BC, Canada, 21–25 May 2018; pp. 348–357.
16. Wang, J.; Liu, T.; Liu, Q.; et al. Compression Ratio Modeling and Estimation Across Error Bounds for Lossy Compression. *IEEE Trans. Parallel Distrib. Syst.* **2020**, *31*, 1621–1635. https://doi.org/10.1109/tpds.2019.2938503.
17. Liu, T.; Wang, J.; Liu, Q. High-Ratio Lossy Compression: Exploring the Autoencoder to Compress Scientific Data. *IEEE Trans. Big Data* **2023** , *9*, 22–36.
18. Liu, J.; Di, S.; Jin, S.; et al. SRN-SZ: Deep Learning-Based Scientific Error-Bounded Lossy Compression with Super-Resolution Neural Networks. *arXiv* **2023**, arXiv:2309.04037.
19. Liu, J.; Di, S.; Zhao, K.; et al. Exploring Autoencoder-Based Error-Bounded Compression for Scientific Data. In Proceedings of the 2021 IEEE International Conference on Cluster Computing (CLUSTER), Portland, OR, USA, 7–10 September 2021; pp. 294–306. https://doi.org/10.1109/cluster48925.2021.00034.
20. Jia, W.; Hu, Z.; Liu, Y.; et al. NeurLZ: An Online Neural Learning-Based Method to Enhance Scientific Lossy Compression. In Proceedings of the 39th ACM International Conference on Supercomputing, Salt Lake City, UT, USA, 8–11 June 2025; pp. 26–42. https://doi.org/10.1145/3721145.3725763.
21. Jin, S.; Di, S.; Liang, X.; et al. DeepSZ: A Novel Framework to Compress Deep Neural Networks by Using Error-Bounded Lossy Compression. In Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing, Phoenix, AZ, USA, 22–29 June 2019; pp. 159–170. https://doi.org/10.1145/3307681.3326608.
22. Li, G.; Alhumaidi, M.; Skiadopoulos, S.; et al. Extreme Error-bounded Compression of Scientific Data via Temporal Graph Autoencoders. *IEEE Trans. Knowl. Data Eng.* **2026**, *38*, 3597–3610. https://doi.org/10.1109/tkde.2026.3681654.
23. Li, X.; Lee, J.; Rangarajan, A.; et al. Foundation Model for Lossy Compression of Spatiotemporal Scientific Data. In Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, 10–13 June 2025; pp. 368–380. https://doi.org/10.1007/978-981-96-8295-9_27.
24. Collet, Y. Zstd GitHub Repository from Facebook. Available online: https://github.com/facebook/zstd (accessed on 28 April 2026).
25. Alted, F. BLOSC GitHub Repository. 2009. Available online: https://github.com/Blosc/c-blosc (accessed on 28 April 2026).
26. Deutsch, P. GZIP File Format Specification Version 4.3. 1996. Available online: https://www.rfc-editor.org/rfc/rfc1952 (accessed on 28 April 2026).
27. Gailly, J.L. GZIP: The Data Compression Program. 2016. Available online: https://www.gnu.org/software/gzip/manual/gzip.pdf (accessed on 28 April 2026).
28. Klus, L.; Klus, R.; Torres-Sospedra, J. EWOk: Towards Efficient Multidimensional Compression of Indoor Positioning Datasets. *IEEE Trans. Mob. Comput.* **2024**, *23*, 3589–3604. https://doi.org/10.1109/tmc.2023.3277333.
29. Qin, Z.; Wang, J.; Liu, Q.; et al. Estimating Lossy Compressibility of Scientific Data Using Deep Neural Networks. *IEEE Lett. Comput. Soc.* **2020**, *3*, 5–8. https://doi.org/10.1109/locs.2020.2971940.
30. Mumenin, K.M.; Dai, D.; Wang, J.; et al. QualityNet: Error-Bounded Lossy Compression Quality Prediction via Deep Surrogate. In Proceedings of the IEEE International Conference on Big Data (BigData), Washington, DC, USA, 15–18 December 2024; pp. 252–261.
31. Zhang, Y.; Lin, H.; Sun, J.; et al. Learning to Predict Object-Wise Just Recognizable Distortion for Image and Video Compression. *IEEE Trans. Multimed.* **2024**, *26*, 5925–5938. https://doi.org/10.1109/tmm.2023.3340882.
32. Li, C.; Yin, S.; Jia, C.; et al. Multirate Progressive Entropy Model for Learned Image Compression. *IEEE Trans. Circuits Syst. Video Technol.* **2024**, *34*, 7725–7741. https://doi.org/10.1109/tcsvt.2024.3376704.
33. Zhang, J.; Chen, T.; Ding, D.; et al. Neural Compression System for Point Cloud Video Streaming. *IEEE Trans. Image Process.* **2025**, *34*, 8032–8045. https://doi.org/10.1109/tip.2025.3638126.
34. Yang, Y.; Jiao, C.; Gao, X.; et al. Adaptive Volumetric Data Compression Based on Implicit Neural Representation. In Proceedings of the 17th International Symposium on Visual Information Communication and Interaction, Hsinchu, Taiwan, 11–13 December 2024; pp. 1–8. https://doi.org/10.1145/3678698.3678703.

35. Han, J.; Zheng, H.; Bi, C. KD-INR: Time-Varying Volumetric Data Compression via Knowledge Distillation-Based Implicit Neural Representation. *IEEE Trans. Vis. Comput. Graph.* **2024**, *30*, 6826–6838. https://doi.org/10.1109/tvcg.2023.3345373.
36. Bai, Y.; Zhao, Y.; Wang, K. Practical Lossless Volumetric Medical Image Compression via Tri-plane Context Tree Learning. *IEEE Trans. Image Process.* **2026**, *35*, 5938–5953.
37. Li, Y.; Xia, M.; Liang, X.; et al. Preserving Discrete Morse–Smale Complexes in Error-Bounded Lossy Compression. *IEEE Trans. Vis. Comput. Graph.* **2026**, *32*, 6593–6609. https://doi.org/10.1109/tvcg.2026.3684385.
38. Dai, W.; Fan, J.; Miao, Y.; et al. Deep Learning Model Compression with Rank Reduction in Tensor Decomposition. *IEEE Trans. Neural Netw. Learn. Syst.* **2025**, *36*, 1315–1328. https://doi.org/10.1109/tnnls.2023.3330542.
39. Cappello, F.; Ainsworth, M. Scientific Data Reduction Benchmarks. 2018. Available online: https://sdrbench.github.io/ (accessed on 28 April 2026).
40. Heaton, J. *Introduction to Neural Networks with Java*; Heaton Research, Inc.: St. Louis, MO, USA, 2008.
41. Apley, D.W.; Zhu, J. Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **2020**, *82*, 1059–1086. https://doi.org/10.1111/rssb.12377.
42. Hinton, G.; Vinyals, O.; Dean, J. Distilling the Knowledge in a Neural Network. *arXiv* **2015**, arXiv:1503.02531.
43. Yang, J.; Martinez, B.; Bulat, A.; et al. Knowledge Distillation via Softmax Regression Representation Learning. In Proceedings of the International Conference on Learning Representations (ICLR), Virtual, 3–7 May 2021.