sensors

MDPI

Article

A Data Compression Method for Wellbore Stability Monitoring Based on Deep Autoencoder

Shan Song [1], Xiaoyong Zhao [2], Zhengbing Zhang [1,*] and Mingzhang Luo [1]

[1] School of Electronic Information and Electrical Engineering, Yangtze University, Jingzhou 434023, China; songs_yzu@163.com (S.S.); lmz@yangtzeu.edu.cn (M.L.)
[2] Directional Drilling Branch of China National Petroleum Corporation Bohai Drilling Engineering Co., Ltd., Tianjin 300280, China; xiaoyong2810@163.com
* Correspondence: zhangzb@yangtzeu.edu.cn

Abstract: The compression method for wellbore trajectory data is crucial for monitoring wellbore stability. However, classical methods like methods based on Huffman coding, compressed sensing, and Differential Pulse Code Modulation (DPCM) suffer from low real-time performance, low compression ratios, and large errors between the reconstructed data and the source data. To address these issues, a new compression method is proposed, leveraging a deep autoencoder for the first time to significantly improve the compression ratio. Additionally, the method reduces error by compressing and transmitting residual data from the feature extraction process using quantization coding and Huffman coding. Furthermore, a mean filter based on the optimal standard deviation threshold is applied to further minimize error. Experimental results show that the proposed method achieves an average compression ratio of 4.05 for inclination and azimuth data; compared to the DPCM method, it is improved by 118.54%. Meanwhile, the average mean square error of the proposed method is 76.88, which is decreased by 82.46% when compared to the DPCM method. Ablation studies confirm the effectiveness of the proposed improvements. These findings highlight the efficacy of the proposed method in enhancing wellbore stability monitoring performance.

Keywords: wellbore safety monitoring; deep autoencoder; well trajectory; data compression; logging while drilling (LWD)

check for updates

Citation: Song, S.; Zhao, X.; Zhang, Z.; Luo, M. A Data Compression Method for Wellbore Stability Monitoring Based on Deep Autoencoder. *Sensors* **2024**, *24*, 4006. https://doi.org/10.3390/s24124006

Academic Editor: Francesco Carlo Morabito

Received: 21 May 2024
Revised: 17 June 2024
Accepted: 18 June 2024
Published: 20 June 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Wellbore instability is a significant challenge encountered during drilling operations in diverse oil and gas reservoirs [1,2]. It encompasses issues such as collapse, shrinkage, diameter enlargement, and fracture, all of which can impede drilling efficiency and, if left unchecked, result in serious incidents. Therefore, it is imperative to monitor wellbore stability to ensure safe drilling practices.

At present, the primary method for monitoring wellbore health [1,3] involves the use of logging while drilling (LWD) tools to gather real-time data on the wellbore's condition. The data are then evaluated using established rock mechanics principles and correlations with logging data. However, this approach is limited by the low data transmission rates of LWD. The most common data transmission technology used in LWD is mud pulse telemetry (MPT), which operates at a rate of 0.5 bit/s to 1.0 bit/s [4–6]. Consequently, it is challenging to meet the real-time logging engineering requirements with this transmission rate. While stress wave-based communication using piezoceramic transducers offers higher data transmission rates [7–9], this technology is still in the laboratory research stage and has not yet been commercialized [10].

Given the current constraints of MPT or stress wave-based communication bandwidth, transmitting compressed data for wellbore stability monitoring can significantly enhance real-time performance. This improvement is crucial for maintaining wellbore safety and stability while enhancing drilling efficiency. Among the parameters vital for wellbore stability

monitoring, inclination, azimuth, and depth are particularly critical [11]. During drilling operations, continuous monitoring and adjustment are essential to ensure the wellbore trajectory aligns optimally, especially in formations with narrow safety margins [12]. Consequently, researching compression methods for inclination and azimuth data becomes imperative.

Currently, the predominant compression techniques for logging while drilling (LWD) data include the methods based on Huffman coding [13], compressed sensing [14], wavelet transform [15], long short-term memory (LSTM) [16], and the Differential Pulse Code Modulation (DPCM) method [17]. Huffman coding, a classic lossless compression method, allows for complete restoration of original data from the compressed data, albeit with relatively low compression rates. For instance, Song et al. [18] employed Frame Prediction Huffman Coding (FPHC) in their 2021 study to compress various logging data, achieving a compression ratio of only 1.41. Although this ratio surpasses that of traditional Huffman and arithmetic coding, its impact on enhancing data transmission efficiency remains limited. The compressed sensing method utilizes the sparsity of logging while drilling data to collect logging data at a rate much lower than Nyquist sampling, thereby reducing the amount of data collected and achieving the effect of data compression. For instance, in 2020, Li et al. [19] proposed a compressed sensing method based on dictionary machine learning, which constructs and adjusts a dictionary through machine learning to compress the downhole density data. Compared with traditional averaging and interpolation methods, this method has a compression ratio of 3.33 and a mean square error 18 times smaller than traditional methods. However, this method requires obtaining enough sampling data (128 sampling data) before compression and transmission can begin, resulting in low real-time performance. Wavelet transform represents a signal as a scaled and shifted "mother wavelet" oscillation waveform. After matching the signal, only the corresponding parameters need to be transmitted to effectively represent the original signal, thus achieving effective compression of the signal. For instance, Jarrot et al. [20] proposed the method of directional wavelet transform to compress downhole image data in 2018. The results showed that for images with a width of 80 pixels and a height of 256 pixels, the rate reached 0.1 bits/pixel (with a compression ratio of 80), and the tilt angle could still be clearly identified. However, this method requires obtaining two-dimensional data with a width of 80 pixels and a height of 256 pixels for compression. Even if each pixel occupies an average of 0.1 bits, when transmitting compressed data, it still needs to transmit 2048 bits to express a downhole image. As a result, for the currently widely used MPT system, the waiting time for transmission would be very long. The LSTM method achieves data compression mainly by predicting the signal and then compressing and encoding the difference between the original data and the predicted data. Yan Zhidan et al. [21] used the LSTM and XGBoost method [22] to predict the main and detailed data of LWD after discrete wavelet transform, and then quantified and encoded the difference between the original data and the predicted data. The compression ratio reached two, and the distortion rate was only 0.01%. However, this method uses wavelet transform and the LSTM network, which requires a long time to obtain sufficient data volume to achieve ideal prediction results and a sufficient compression ratio. Therefore, the real-time performance of this method is also poor. The DPCM method is easy to implement, with no requirement for the amount of collected data; thus, it has good real-time performance. For instance, Zhang et al. [23] proposed in 2009 to use the DPCM method to compress gamma and resistivity data. The predictor used was a first-order predictor, while the quantizer used three-bit and four-bit quantization bits, respectively. The results showed that the compression ratio of the DPCM could reach more than two, and the distortion remained within 5%. The following year, Zhang et al. conducted further research on the DPCM method [24]. They used multiple predictor orders and multiple quantization bits to compress various logging data, such as gamma, resistivity, and temperature, for a wider range of logging parameters, achieving a compression ratio of two and a distortion rate below 5%. Despite its superior compression ratios compared to Huffman coding and good real-time performance, the DPCM method still falls short in practical engineering applications due to its relatively

low compression efficiency. Additionally, abrupt data changes often result in significant decoding errors with the DPCM method [17,24]. Given the commonplace occurrence of rapidly changing LWD data in practice, these decoding errors render the DPCM method unreliable. Among the above methods, the compression ratio of Huffman coding-based compression methods is very low and difficult to improve. Although compression sensing, wavelet transform, and LSTM-based compression methods have higher compression ratios, these methods require a large amount of collected data, often requiring hundreds or thousands of data points to achieve effective compression. In M/LWD engineering, the sampling interval of inclination, azimuth, gamma, and other logging data is usually in seconds. If these methods are used, it will take a long time to complete the compression and transmission of data, which will inevitably cause serious information lag. Relatively, the DPCM method can achieve real-time transmission of logging data, and its compression ratio is higher than the lossless compression method based on Huffman coding. However, the compression ratio of this method is still relatively low and the error is large, making it difficult to meet the needs of real-time monitoring of wellbore stability. Addressing the limitations of existing compression methods, it becomes imperative to develop novel compression methods tailored to wellbore trajectory data.

In recent years, machine learning methods have emerged as potent tools for signal processing [25–27]. Among these, the autoencoder stands out as an unsupervised artificial neural network renowned for efficiently learning data features, often utilized for data dimensionality reduction [28,29]. Comprising two main components—the encoder and the decoder—the autoencoder operates by first reducing the dimensionality of the data, thereby reducing the volume of data and facilitating data compression. Subsequently, the decoder reconstructs the data to closely resemble the input data [30,31]. Leveraging its capacity for dimensionality reduction and reconstruction, autoencoders find widespread application in data compression. For instance, Nuha et al. [32] introduced a stacked autoencoder extreme learning machine (AE-ELM) for seismic data compression. Achieving a compression ratio of 10, the normalized mean square error (NMSE) between the reconstructed seismic data and the original data was recorded at 1.28×10^{-3}, surpassing the performance of the classic Discrete Cosine Transform (DCT) method. This underscores the efficacy of autoencoder-based approaches in seismic data compression. Similarly, in 2021, Liu et al. [33] conducted a pioneering compression study on HPC (high-performance computing) scientific data using an autoencoder. Notably, the compression ratio of the HACC dataset in the scientific data reduction benchmark dataset reached 241.67, marking a four-fold increase compared to the classic Squeeze (SZ) [34] compression method and a fifty-fold increase over the ZFP [35] method. This study underscores the compelling compression capabilities of autoencoders for HPC scientific data. Moreover, autoencoders have found substantial applications in image compression for both research and practical purposes [36–38].

Compared to classical autoencoders, the deep autoencoder proposed by Hinton et al. [39] demonstrates superior performance in learning data features and reducing data dimensionality. Structurally akin to autoencoders, deep autoencoders comprise an encoder and decoder, yet each component boasts multiple neural network layers. This architecture allows for more effective learning of data feature representations and dimensionality reduction, making these autoencoders promising candidates for data compression across various domains. For instance, Yildirim et al. [40] employed deep convolutional autoencoders to compress electrocardiogram (ECG) signals, achieving a compression ratio of 32.25 with a percentage RMS difference (PRD) of 2.73%. This outperforms classic methods such as DWT + RLE (Discrete Wavelet Transform + Run Length Encoding) and DCT + RLE + Huffman. Similarly, Kuester et al. [41] utilized a deep autoencoder to compress a representative set of spectral data in 2020, achieving a compression ratio of four with an almost lossless compression process.

The dynamic ranges of inclination and azimuth data in wellbore trajectory parameters are relatively large, typically requiring 11 bits for encoding. Consequently, traditional compression methods like Huffman coding and DPCM are not suitable for this task. Deep

autoencoders, renowned for their adeptness at learning data features, offer a promising alternative. By leveraging their capability to extract key features from high-dimensional data and obtain low-dimensional representations, deep autoencoders can effectively compress inclination and azimuth data. In light of these considerations, we propose a data compression method based on deep autoencoders specifically tailored for compressing inclination and azimuth data. This approach aims to enhance the compression ratio, addressing the issue of low compression ratios encountered with existing methods.

Our proposed method involves subtracting the reconstruction data of the autoencoder from the original data to obtain residual data, which is then compressed. As the residual data typically exhibits low correlation characteristics, quantization coding is applied for compression, complementing the compressed data from the autoencoder. This integrated approach effectively reduces the error between the reconstruction data and the source data. Furthermore, mean filtering based on a standard deviation threshold is employed to further mitigate errors between the reconstructed data at the decoding end and the source data. However, it is essential to note that not all data in the reconstructed data are suitable for mean filtering. Data with significant changes, indicated by large standard deviations, may actually incur increased errors when subjected to mean filtering. To address this, we propose utilizing a standard deviation threshold to control the mean filter, filtering only the data below this threshold. To optimize the selection of the standard deviation threshold and minimize reconstruction data errors, we employ the Root Mean Square Propagation (RMSProp) [42] optimization algorithm. This approach ensures efficient parameter tuning, thereby enhancing the overall performance of the compression method. The contributions of this paper are summarized as follows:

- We propose an efficient and real-time compression method for wellbore safety monitoring-related data, which effectively improves the compression efficiency of wellbore inclination and azimuth data while greatly reducing the error between the reconstructed data and the original data. This solves the problems of low real-time performance, low compression efficiency, and large reconstruction data error in existing methods, and can effectively improve the performance of wellbore stability monitoring;
- We propose for the first time the use of deep autoencoders to compress inclination and azimuth data, achieving significant compression of inclination and azimuth data, effectively solving the problem of low compression ratio in existing methods;
- We propose a mean filtering method based on the optimal standard deviation threshold to filter the reconstructed data after compensation of the residual, further effectively reducing the error between it and the original data.

The rest of the paper is organized as follows: Section 2 introduces the basic principles and provides a detailed process of the proposed method. Section 3 introduces the experimental data and experimental setup. Section 4 showcases the results of the simulation experiments. Finally, Section 5 offers concluding remarks to summarize the key findings and implications of the study.

2. Proposed Method

2.1. The Overall Framework of the Proposed Method

2.1.1. Block Diagram of Compressed Data Transmission System for LWD

In order to improve the transmission efficiency of the data about wellbore stability monitoring and enhance real-time monitoring, it is necessary to embed the compression method into the MPT system, forming a compressed data transmission system for LWD. The structural diagram of the compressed data transmission system for LWD is shown in Figure 1, which is mainly divided into downhole measurement systems, mud pulse transmission systems, and surface signal processing systems.

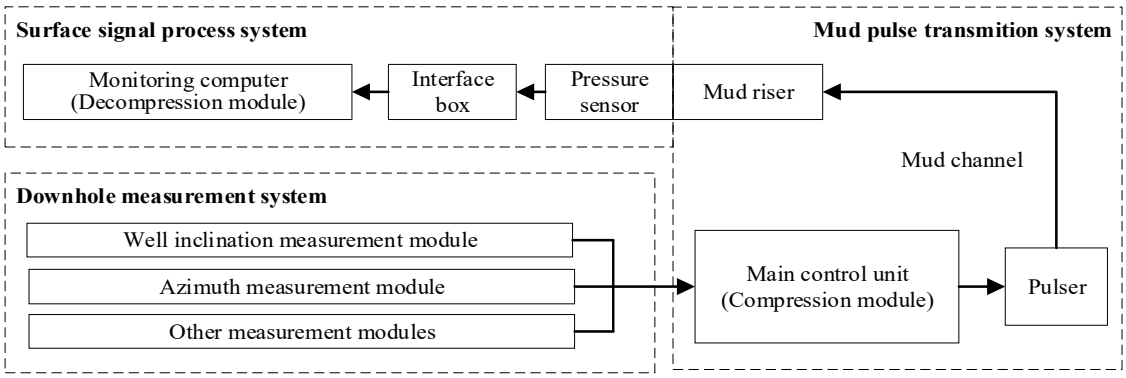


Figure 1. Block diagram of compressed data transmission system for LWD.

The downhole measurement system comprises various modules for measuring parameters like inclination, azimuth, and other logging parameters. In particular, the inclination and azimuth measurement modules acquire data regarding the downhole equipment's inclination and azimuth angles, which need to be compressed and transmitted to improve the performance of monitoring wellbore stability.

The mud pulse transmission system is composed of the main control unit, pulser, mud channel and mud riser. Embedded within the main control unit, the data compression module—whether in software or hardware form—compresses and encodes inclination and azimuth data. This compression reduces their code length, thereby enhancing transmission efficiency and ultimately improving wellbore stability monitoring performance. During operation, the main control unit acquires data from the downhole measurement system via the bus. Subsequently, it compresses and encodes the inclination and azimuth data, integrating them with other measurement data. After encoding and packaging, the pulser emits pulses to alter the circulating mud pressure within the drill string in the mud channel, transmitting the signal to the surface as mud pressure waves. Finally, the signal reaches the surface signal processing system through a mud riser.

The surface signal processing system comprises pressure sensors, interface boxes, and a monitoring computer. Within the monitoring computer, the decompression module is integrated into the signal processing software. This module decompresses the compressed inclination and azimuth data, generating their reconstruction data, which reflects the inclination and azimuth information of the downhole equipment. During operation, the pressure sensors detect changes in mud fluid pressure, generating signals that are transmitted to the monitoring computer via the interface box. Within the monitoring computer, the signal processing software processes these pressure signals, removing noise and decoding them to obtain compressed inclination and azimuth data. Subsequently, through the decompression module, the wellbore inclination and azimuth information are reconstructed and displayed, facilitating real-time monitoring of wellbore stability.

To enhance the monitoring performance of wellbore stability, it is necessary to effectively compress and decompress inclination and azimuth data through compression and decompression modules. However, existing compression methods suffer from low real-time capability, low compression ratios, and significant errors between the reconstructed and raw data. To address these issues and enhance the effectiveness of wellbore stability monitoring, a novel approach to data compression must be proposed.

2.1.2. Structural Diagram of the Proposed Method

To boost the compression ratio of inclination and azimuth data, and considering the efficient data dimensionality reduction ability of deep autoencoders, we leverage the dimensionality reduction capability of deep autoencoders, enabling significant compression of the raw data. However, there is an error between the reconstructed data of the deep autoencoder and the original data. The error is primarily due to the complete discarding of the residual between the reconstructed data and the original data; therefore, to reduce

the error, it is necessary to compress and transmit the residual data. To this end, we use quantization coding and Huffman coding to compress the residual data, and then compensate for the reconstructed data of the deep autoencoder. Quantization coding can effectively reduce the error between the reconstructed data of the deep autoencoder and the original data, but it will also lead to a reduction in the compression ratio. To minimize the reduction in the compression ratio, we use Huffman coding to further compress the data encoded by quantization coding. In addition, since quantization coding reduces the dynamic range of the residual data, the amount of codeword information required for the Huffman coding is also reduced, making Huffman coding easier to implement in downhole equipment. The introduction of quantization coding and Huffman coding can effectively reduce the error while still maintaining a high compression ratio.

Additionally, we apply mean filtering based on a standard deviation threshold to the compensated reconstructed data. This step is crucial because even after compensation, the reconstructed data still contains errors compared to the raw data, akin to noise interference. These errors primarily stem from the deep autoencoder's inability to fully extract all features of the raw data. Hence, mean filtering is necessary to mitigate this interference. However, not all compensated reconstructed data is suitable for filtering. For segments of data that exhibit stability (with a small standard deviation), the error interference is more pronounced, and mean filtering effectively reduces this interference. Conversely, for segments of data undergoing significant changes (with a large standard deviation), applying mean filtering would exacerbate errors rather than mitigate them.

We employ a standard deviation threshold as a criterion to determine whether a particular segment of the data requires mean filtering. To establish an optimal standard deviation threshold that minimizes the error between the final reconstructed data and the raw data, we leverage the RMSProp optimization algorithm. RMSProp is chosen for its stability and rapid convergence, making it an ideal candidate for this task. Figure 2 illustrates the compression method based on a deep autoencoder, incorporating the aforementioned concepts.

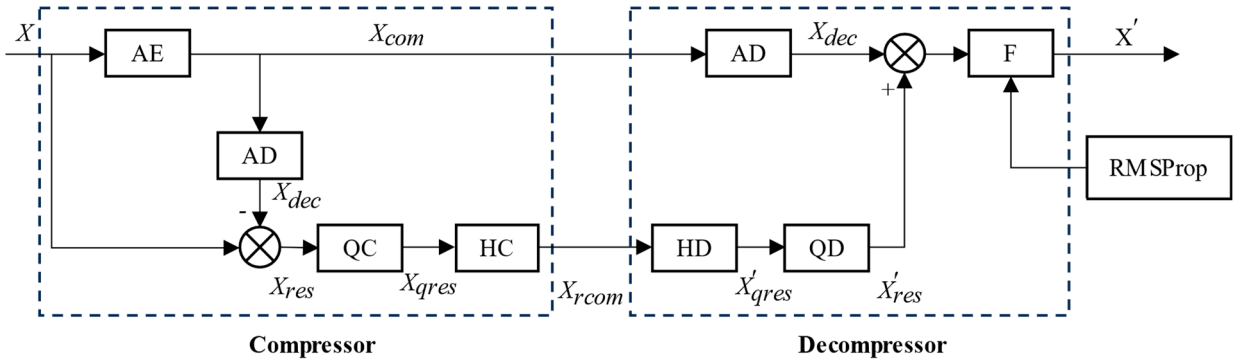


Figure 2. Diagram of data compression method based on deep autoencoder.

The compression method based on deep autoencoder proposed in this paper consists of two parts: a compressor and a decompressor. During compression, the original data and residual data are compressed separately. On one hand, the compressor compresses the source data X with the encoder of the deep autoencoder (AE) to obtain compressed data X_{com}, and X_{com} will be directly transmitted to the decompressor. On the other hand, the compressor decompresses X_{com} with the decoder of the deep autoencoder (AD) to obtain the decompressed data X_{dec}, then X_{dec} is subtracted from X to obtain residual data X_{res}. Subsequently, the quantization coding method (QC) is used to encode X_{res} and obtain X_{qres}, then compress X_{qres} with Huffman coding (HC) to obtain the compressed data (X_{rcom}) of residual X_{res} and pass it to the decompressor.

The decompressor is also divided into two parts. The first part uses the decoder of the deep autoencoder (AD) to decompress X_{com} to obtain the decoded data of the deep autoencoder (X_{dec}). The second part decompresses the compressed residual data X_{rcom} with

the Huffman decoding method (HD) to obtain the quantization encoding data of residual data X_{qres}, then decompresses X_{qres} with the quantization decoding method to obtain the reconstructed data of residual data X'_{res}. Then, X_{dec} is added to X'_{res}, and the output of the adder is filtered with filter F to obtain the reconstructed data X'. Filter F adopts the mean filtering method based on the optimal standard deviation threshold and should be trained using the RMSProp optimization algorithm with a training dataset measured in advance to obtain the optimal standard deviation threshold.

Below are the principles of deep autoencoder, quantization coding, and Huffman coding, as well as the detailed implementation process of the proposed method.

2.2. Extraction of Source Data Features

The proposed method uses a deep autoencoder to extract features from the source data, which can significantly improve the compression ratio. Deep autoencoders are developed based on autoencoders. The following section will introduce the concepts of autoencoder and deep autoencoder.

2.2.1. Autoencoder

Autoencoder is an unsupervised neural network with a symmetric structure, which can effectively learn the internal features of data [43] to obtain concise expressions of data; it is often used for data dimensionality reduction [44]. The standard autoencoder has a three-layer architecture [45], as shown in Figure 3. Autoencoder is structurally composed of an input layer, a hidden layer, and an output layer. According to its function, autoencoder can be further divided into an encoder and decoder. The encoder includes an input layer and a hidden layer, responsible for compressing high-dimensional input data $x \in \mathbb{R}^n$ to obtain its low-dimensional feature representation $h \in \mathbb{R}^{\hat{n}}$ ($\hat{n} < n$) to achieve data compression. The decoder includes the hidden layer and an output layer, responsible for reconstructing data x using feature representation h to obtain reconstructed data $\hat{x} \in \mathbb{R}^n$ and try to ensure that $\hat{x}$, as much as possible, approaches the input raw data x.

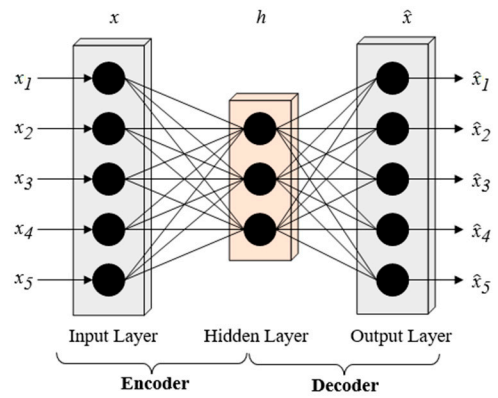


Figure 3. Autoencoder structure.

The formulas for the encoder and decoder are as follows:

$$h = f(W_h x + b_h) \tag{1}$$

$$\hat{x} = f(W_{\hat{x}} h + b_{\hat{x}}) \tag{2}$$

where x is the input data, and h is the compressed data for feature representation. $\hat{x}$ is the reconstructed data. W_h and b_h are the weights and bias between the input layer and the hidden layer. $W_{\hat{x}}$ and $b_{\hat{x}}$ are the weights and bias from the hidden layer to the output layer. $f(Wx + b)$ is an activation function, usually a nonlinear function such as a sigmoid function or a hyperbolic tangent function.

The goal of training autoencoder is to find the optimal matrix W_h, $W_{\hat{x}}$, b_h, and $b_{\hat{x}}$ to minimize the error between the input data x and the reconstructed data $\hat{x}$, as follows:

$$\underset{W_h, W_{\hat{x}}, b_h, b_{\hat{x}}}{arg\ min}\ [d(x, \hat{x})] \tag{3}$$

where $d(x, \hat{x})$ is a loss function used to characterize error between the input data x and the reconstructed data $\hat{x}$. During training, parameters are updated using backpropagation algorithms and optimization algorithms such as adaptive moment estimation (Adam). The backpropagation algorithm is used to calculate the gradient of a multivariate function, while the optimization algorithm updates W_h, $W_{\hat{x}}$, b_h, and $b_{\hat{x}}$ based on the direction in which the gradient of the loss function decreases. After the training, the loss function converges to the local or global minimum value, and the generated autoencoder model is used for feature extraction and dimensionality reduction of input data.

2.2.2. Deep Autoencoder

Compared to autoencoder, deep autoencoder has more hidden layers, and the output of each layer constitutes the input of the next layer. As shown in Figure 4, except for the input and output layers, all other layers in the figure are hidden layers, and the hidden layer in the middle has the smallest dimension, which will be used as a compressed representation.

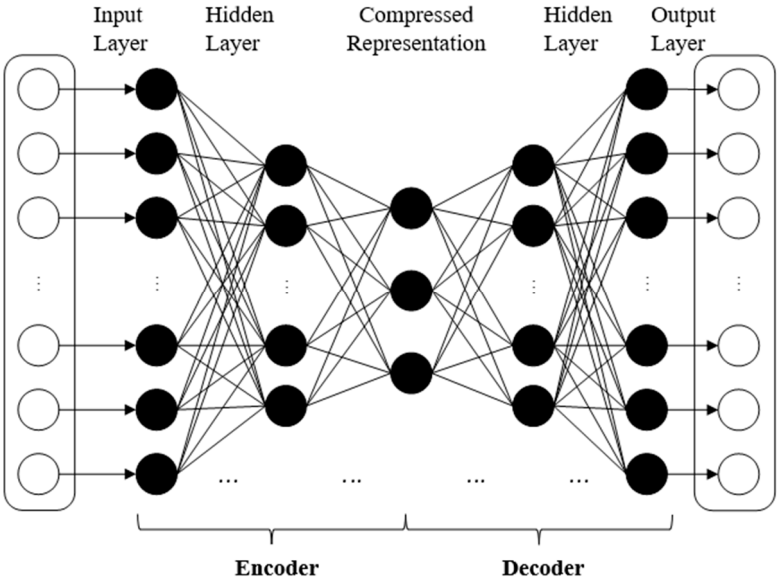


Figure 4. Deep autoencoder structure.

Deep autoencoders have a better ability to learn data features; therefore, we use a deep autoencoder to compress inclination and azimuth data. The deep autoencoder structure information used in the proposed method is listed in Table 1.

All layers are dense layers, among which the encoder includes hidden layer 1, hidden layer 2, and the bottleneck layer. The bottleneck layer has the least number of data nodes, and its output value is the compressed data. The decoder includes hidden layer 3, hidden layer 4, and hidden layer 5. The output data of hidden layer 5 is the decoded data, and its dimension is exactly the same as the input data dimensions of hidden layer 1.

The computation method for the compression ratio of deep autoencoder is as expressed by Equation (4).

$$C_R = \frac{nb_{in}}{\hat{n}b_{BN}} \tag{4}$$

where n is the dimension of the input data, and $\hat{n}$ is the dimension of the bottleneck layer. b_{in} is the bit length of the input data, and b_{BN} is the bit length of the output data of the

bottleneck layer (compressed data). The activation function of each hidden layer adopts Leaky_ReLU, and its formula is as follows:

$$\phi = f(x) = \begin{cases} ax, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{5}$$

where $a > 0$. The reason we opt for the Leaky_ReLU function is its capability to prevent gradient vanishing [46] while also introducing nonlinear transformation characteristics to the network.

Table 1. Structure of the deep autoencoder used in the proposed method.

Composition	Layer Name	Layer Type	Activation Function	Dimension
Encoder	input layer	-	-	n
	hidden layer 1	Dense	$\phi = leaky\_relu$	$n/2$
	hidden layer 2	Dense	$\phi = leaky\_relu$	$n/4$
	bottleneck	Dense	$\phi = leaky\_relu$	$\hat{n} = n/8$
Decoder	hidden layer 3	Dense	$\phi = leaky\_relu$	$n/4$
	hidden layer 4	Dense	$\phi = leaky\_relu$	$n/2$
	hidden layer 5	Dense	$\phi = leaky\_relu$	n
	output layer	-	-	n

2.3. Compression of Residual Data

After the deep autoencoder compresses the source data, further feature extraction from the remaining residual data becomes challenging. Simply discarding this residual data is not conducive to reducing the error between the reconstructed data and the source data. Hence, we employ quantization coding and Huffman coding to compress the residual data. In the decompressor, this compressed residual data is reconstructed to compensate for the reconstruction data of the deep autoencoder, thereby reducing the error between the reconstructed data and the source data. Quantization coding serves to preliminarily compress the residual data, thereby reducing its dynamic range, while Huffman coding effectively compresses the data.

2.3.1. Quantization Coding

Quantization coding uniformly uses smaller bit lengths B_q to represent data that are originally needed to be represented by B_{in} bit lengths $(1 < B_q < B_{in})$. The quantization encoding formula is as follows:

$$d'_n = Q(d_n) = \frac{b_{i-1} + b_i}{2} (b_{i-1} \leq d_n < b_i) \tag{6}$$

$$b_j = -2^{B_{\text{in}}-1} + j\prime \frac{2^{B_{\text{in}}}}{2^{B_q}} \left(j = 0,\ 1, 2 \ldots 2^{B_q} \right) \tag{7}$$

where b_i is the boundary value of the quantization interval, i is the quantization level, and $i = 1, 2 \ldots 2^{B_q}$, whose value needs to be determined through traversal. By taking $i = 1, 2 \ldots 2^{B_q}$ when $b_{i-1} \leq d_n < b_i$ is established, the value of i is the quantization level corresponding to the current data.

The quantization coding encodes the residual quantization value d'_n based on the bit length of quantization coding B_q. Since the total value of B_q is less than B_d, the dynamic range of the required encoded data is reduced, which is beneficial for further compression using Huffman coding in the subsequent step.

2.3.2. Huffman Coding

The principle of Huffman coding is to assign Huffman codes to data based on the frequency or probability distribution of the source data. Among them, data with high frequency or probability are assigned codes with fewer encoding bits, while data with low frequency or probability are assigned codes with more encoding bits. In this way, most of the source data will be encoded into Huffman code with fewer bits, thereby reducing the overall amount of encoded data and achieving compression. Huffman coding is a commonly used and easily implementable lossless data compression method, often serving as part of a more complex data compression method. Huffman coding is a relatively common, efficient, and easily implementable lossless data compression method, often used as part of a more complex data compression method. The detailed principles and implementation methods can be found in ref. [13]. This paper utilizes Huffman coding to compress the data encoded by quantization coding, in order to reduce the impact of the reduced compression ratio caused by quantization coding.

2.4. Mean Filtering of Compensated Reconstructed Data

After obtaining the compensated reconstruction data, there still exists a certain degree of error compared to the source data. Filtering can further mitigate this error between the compensated reconstruction data and the source data. Therefore, we employ mean filtering to process the compensated reconstruction data. However, the reconstructed data encompasses various data characteristics, and not all data are suitable for filtering. For data exhibiting significant changes, mean filtering may exacerbate errors, while for data with subtle changes, it can effectively reduce the error between the reconstructed data and the source data. To discern these data characteristics, we utilize standard deviation. Thus, it is essential to determine an appropriate standard deviation threshold. If the standard deviation surrounding the current data is less than this threshold, mean filtering is applied; otherwise, no processing is conducted. The value of the standard deviation threshold is correlated with the error between the reconstructed data and the source data. To select the optimal standard deviation threshold and minimize the error, we leverage the RMSProp optimization algorithm for training.

2.4.1. RMSProp Optimization Algorithm

Obtaining the standard deviation threshold to minimize the reconstruction data error is a type of optimization problem. The RMSProp algorithm is a suitable optimization algorithm, which has good convergence stability since it only solves for the average value of gradients within a certain duration [47]. At the same time, this method has a faster convergence rate by discarding the early gradients. Therefore, we adopt the RMSProp algorithm to obtain the optimal standard deviation threshold as

$$v_t = \beta \cdot v_{t-1} + (1-\beta)(g_t)^2 \tag{8}$$

$$w_{t+1} = w_t - \frac{lr \cdot g_t}{\sqrt{v_t + \varepsilon}} \tag{9}$$

where β is a hyper-parameter; the larger its value, the greater the weight of the memorized historical gradient, usually around 0.9. ε is a very small constant, mainly designed to avoid being divided by zero, usually taking a value of 10^{-6}. lr is the learning rate, which can generally be taken as 0.01. w_t is the weight of the t-th iteration, which is the value to be solved, while g_t is the gradient of the loss function relative to w_t. Their relationship is

$$g_t = \nabla_{w_t} f(w_t) \tag{10}$$

where $f(w_t)$ is the loss function. To find the optimal standard deviation threshold and minimize the error between the compensated reconstructed data and the source data, the standard deviation threshold can be used as w_t, and the mean square error MSE can be

used as the loss function. When iterating using the RMSProp formula mentioned above, the value of w_t corresponding to the convergence of the loss function value is the optimal value of standard deviation threshold.

2.4.2. Mean Filtering Method Based on Optimal Standard Deviation Threshold

After obtaining the optimal standard deviation threshold, selective filtering is performed on the compensated reconstructed data based on this threshold to further reduce the error between the reconstructed data and the source data. The filtering method used in this paper is the mean filtering, which is a linear filter that takes the average value of the data as the output value of the center point. If the optimal standard deviation threshold obtained through the optimization algorithm is T_W, the implementation process of the mean filtering method based on the standard deviation threshold T_W is as follows:

Step 1: Calculate the index of the center point and the end point data within the filtering window. The calculation method is as follows:

$$i_e = i + N_W - 1 \tag{11}$$

$$i_c = \frac{2i + N_W - 1}{2} \tag{12}$$

where N_W is the size of the filtering window; its value should not be too large because the larger the value, the more points that are required for filtering, which will delay the time for filtering. This is not favorable for the surface system to quickly obtain the filtered reconstructed data, and it may cause more significant errors at the beginning of the drilling operation. Meanwhile, its value should not be too small, as the filter will not be able to accurately distinguish between noise and collected signals, leading to poor filtering results. Generally, choosing a value close to the input data dimension of the deep autoencoder can help achieve better results.

Additionally, i is the index of the data sequence to be filtered corresponding to the starting point of the filtering window, i_c is the index of the center point of the window, and i_e is the index of the end point of the window.

Step 2: Calculate the average value of the data within the filtering window $\overline{x}_W$ using the following formula:

$$\overline{x}_W = \frac{1}{N_W} \sum_{k=i}^{i_e} x_k \tag{13}$$

Step 3: Calculate the standard deviation of data within the window $s(i_c)$ using the following formula:

$$s(i_c) = \sqrt{\frac{\sum_{k=i}^{i_e} (x_i - \overline{x}_W)^2}{N_W}} \tag{14}$$

Step 4: Obtain the value of the center point within the filtering window with the following formula:

$$x_{i_c} = \begin{cases} \overline{x}_W, s(i_c) < T_W \\ x_{i_c}, s(i_c) \geq T_W \end{cases} \tag{15}$$

Step 5: Output the value of x_{i_c} to replace the data value with index i_c in the sequence to be filtered, while keeping the data values at other positions unchanged.

Step 6: Move the window index backward by the following formula:

$$i' = i + i_{steplen} \tag{16}$$

where i' is the start point index in new window, and $i_{steplen}$ is the sliding window step length. Replace i with i' and continue to perform the mean filtering according to Equations (11)–(15) until all the required data are filtered.

2.5. Performance Evaluation

To objectively evaluate the compression performance of the method proposed in this paper, the signal-to-noise ratio (SNR), mean squared error (MSE), compression ratio (CR), and their incremental ratios (ΔSNR, ΔMSE, and ΔCR) are used to evaluate the performance of compression/reconstruction methods. The SNR is calculated as

$$SNR(dB) = 10 \cdot log_{10}\left(\frac{\sum_{i=1}^{n} {y_i}^2}{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}\right) \tag{17}$$

MSE is calculated as

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{18}$$

where y_i and $\hat{y}_i$ represent the raw and reconstructed data values, respectively.

The smaller the MSE or the larger the SNR, the smaller the decoding data error, indicating better decoding data quality. When the MSE or SNR is constant, the larger the compression ratio, the better the data compression effect, and the greater the effect on improving the equivalent transmission rate of the MPT system. The compression ratio formula is defined as

$$C_R = \frac{S_O}{S_C} \tag{19}$$

where S_O is the amount of raw data, and S_C is the amount of compressed data.

ΔSNR is the incremental ratio of SNR and is used to measure the degree of improvement in SNR between two different compression methods. The formula for ΔSNR is as follows:

$$\Delta SNR_{A2,A1} = \frac{SNR_{A2} - SNR_{A1}}{SNR_{A1}} \times 100\% \tag{20}$$

where $\Delta SNR_{A2,A1}$ represents the percentage of improvement in SNR of method A2 relative to method A1.

The corresponding ΔMSE and ΔCR are the incremental ratios of MSE and CR, respectively, used to measure the degree of performance improvement between different compression methods.

The formulas for ΔMSE and ΔCR are as follows:

$$\Delta MSE_{A2,A1} = \frac{MSE_{A2} - MSE_{A1}}{MSE_{A1}} \times 100\% \tag{21}$$

$$\Delta CR_{A2,A1} = \frac{CR_{A2} - CR_{A1}}{CR_{A1}} \times 100\% \tag{22}$$

where $\Delta MSE_{A2,A1}$ represents the percentage increase or decrease in MSE of method A2 relative to method A1, and $\Delta CR_{A2,A1}$ represents the percentage increase or decrease in CR of method A2 relative to method A1.

2.6. Implementation of the Proposed Method

The implementation process of the proposed method is divided into three stages: training, compression, and decompression.

In the training phase, the deep autoencoder is first trained to achieve efficient compression of the input data. Then, we use the RMSProp optimization algorithm to find the optimal standard deviation threshold required for the mean filtering method. The compression stage utilizes the trained deep autoencoder along with residual data compression methods to compress the measured data. In the decompression stage, the compressed data are decompressed using the deep autoencoder and residual data decompression methods. Subsequently, the reconstructed data are filtered using the optimal standard deviation threshold and mean filtering method to obtain the final decompressed data.

Below are the implementation processes of these three stages.

2.6.1. Training Process

The training process of the proposed method consists of two parts: one is the process of training the deep autoencoder, and the other is the process to find the optimal standard deviation threshold for the mean filter with the RMSProp algorithm.

The process of training the deep autoencoder is as follows (Algorithm 1).

Algorithm 1: Training steps for the deep autoencoder

Step 1 : Collect previously measured inclination and azimuth data separately as the training datasetX_{train}.
Step 2 : Set the input data dimension D_{in} and change the shape of the dataset X_{train} to include D_{in} sampling data in each sample. Then, set shuffle to "True" to shuffle the sample order.
Step 3: Use MSE as the loss function and the optimization method of Adam to train the deep autoencoder with the structure shown in Table 1.
Step 4 : When the training converges, save the corresponding deep autoencoder model parameter P_{model}.

The process to find the optimal standard deviation threshold for the mean filter is as follows (Algorithm 2).

Algorithm 2: Steps to find the optimal standard deviation threshold

Step 1: Load the deep autoencoder shown in
Table 1 and model parameter P_{model}, use the deep autoencoder to compress and decompress the dataset X_{train}, and obtain compressed data X_{train_com} and decompressed data X_{train_dec}.
Step 2 : Subtract the autoencoder decompressed data X_{train_dec} from the original dataset X_{train} to obtain the residual data X_{train_res}.
Step 3: Set the quantization bits for quantization coding to B_q, perform quantization coding on the residual data X_{train_res} according to Formulas (6) and (7), and then perform Huffman coding on the quantized encoded data according to the method in ref. [13] to obtain the compressed data of the residual data X_{train_rescom}.
Step 4: Decode the compressed data of the residual data X_{train_rescom} with Huffman decoding and quantization decoding according to the inverse process of the method in step 3 to obtain the reconstructed data of the residual data X'_{train_res}.
Step 5: Add the reconstructed residual data X'_{train_res} and decompressed data X_{train_dec} to obtain the reconstructed test dataset X'_{train}.
Step 6: Use the RMSProp algorithm (in Section 2.4.1 of this paper) to find the optimal standard deviation threshold T_W. The detailed steps are as follows:

(1) Set values for hyperparameters β, lr, and ε. Set the standard deviation threshold T to its initial value T_0. Set the mean filtering window size to N_W and the step length to $StepLen$.
(2) Starting from the first data, take N_W data points in order from the reconstructed data X'_{train} and fill them in the filtering window.
(3) Use the standard deviation threshold T, filter the center point data of the filtering window according to formulas (11) to (15), and output the center point data of the window.
(4) According to Equation (16), move the filtering window backward according to the step length $StepLen$, and then take N_W data points in order to fill the filtering window.
(5) Repeat steps (3) to (4) until the remaining number of data is less than N_W, and obtain the filtered data X'_{filter}.
(6) Calculate the MSE of the original data X_{train} and the filtered data X'_{filter} as the loss function, and use Equation (10) to calculate the gradient value of this on the standard deviation threshold T.
(7) Use the gradient value from step (6) to update the standard deviation threshold according to Equations (8) and (9).
(8) Repeat steps (2) to (7), and perform multiple iterations to update until the loss function converges to *a* certain value and the optimal standard deviation threshold T_W is obtained.

Step 7: Output the optimal standard deviation threshold T_W.

2.6.2. Data Compression Process

The data compression process is as follows (Algorithm 3).

Algorithm 3: Steps for compression encoding

Step 1: Load the deep autoencoder model with the trained model parameters P_{model}.
Step 2: Collect D_{in} data points as a set of data X, and input X into the encoder of the deep autoencoder to obtain the compressed data, denoted as X_{com}.
Step 3: Input X_{com} into the decoder of the deep autoencoder to obtain the decompressed data X_{dec}, and subtract X_{dec} from X to obtain the residual data X_{res}.
Step 4: Encode the residual data X_{res} with quantization coding and Huffman coding to obtain the compressed data of the residual data X_{rcom}.
Step 5: Output the compressed data of X (X_{com}) and compressed data of residual data (X_{rcom}).
Step 6: Repeat steps 2 to 5 until all the collected data are completed.

2.6.3. Data Decompression Process

The data decompression process is as follows (Algorithm 4).

Algorithm 4: Steps for decompression

Step 1 : Load the deep autoencoder model with the trained model parameter P_{model}, set the mean filtering window size to N_W and the step length to *StepLen*, and use the optimal threshold T_W obtained from Algorithm 2 as the standard deviation threshold required for the mean filtering method.
Step 2 : Obtain the compressed data (X_{com}) and the compressed data of residual data (X_{rcom}).
Step 3 : Input the compressed data (X_{com}) into the decoder of the deep autoencoder to obtain the decompressed data (X_{dec}).
Step 4 : Decode the compressed data (X_{rcom}) with Huffman decoding and quantization decoding to obtain the reconstructed residual data (X'_{res}).
Step 5 : Add the decompressed data (X_{dec}) and the reconstructed residual data (X'_{res}) to obtain the reconstructed collected data (X').
Step 6 : Take N_W data points of X' in order and fill them in the filtering window.
Step 7 : Use the optimal standard deviation threshold T_W and step length *StepLen*, filter the data in X' according to Equations (11)–(16), and output the filtered data.
Step 8: Repeat steps 2 to 7 until all the compressed data have been decoded.

3. Experiments

3.1. Experimental Data

To evaluate the proposed method, a dataset comprising inclination and azimuth data obtained from an LWD system was collected for experimentation. The schematic diagram of the LWD system is depicted in Figure 5. The measurement short section in the downhole component includes an inclination and azimuth measurement module, which captures information on the well's inclination and azimuth near the drill bit. This information is transmitted to the receiving short section via an antenna, encoded by the main control unit, and then sent to the ground. Finally, it is decoded by the interface box and monitoring computer in the ground section to provide inclination and azimuth data for monitoring the stable state of the wellbore by ground workers.

The monitoring computer records the decoded inclination and azimuth data while monitoring. We obtained the inclination and azimuth data of multiple wells measured at different time periods recorded by this LWD system, denoted as X_{train_test}. The X_{train_test} contains 72,328 points each for the inclination and azimuth data, with 80% (57,864 sampling data) of the data taken as the training dataset, denoted as the inclination training dataset X_{inc_train} and azimuth training dataset X_{azi_train}. The remaining 20% (14,464 sampling data) will be used as the testing dataset, denoted as the inclination testing dataset X_{inc_test} and azimuth testing dataset X_{azi_test}. Among them, X_{inc_train} and X_{azi_train} are used for Algorithm 1 to train the deep autoencoder, and for Algorithm 2 to obtain the optimal standard deviation threshold of the mean filter. In addition, X_{inc_test} and X_{azi_test} are used to test and compare the compression performance of various compression methods.

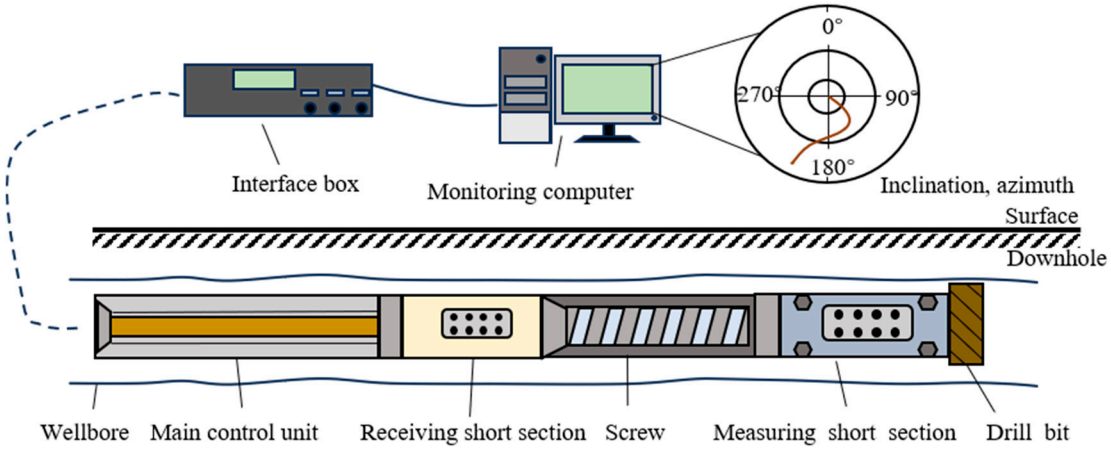


Figure 5. Schematic diagram of LWD system device operation.

3.2. Experimental Setup

To verify the effectiveness of the proposed method in this paper, we first use the datasets of X_{inc_train} and X_{azi_train} to train the deep autoencoder shown in Table 1 with Algorithm 1. The dimension of the input data (D_{in}) of the deep autoencoder is set to 8. After the training, the deep autoencoder models are obtained for inclination data and azimuth data. Algorithm 2 and training datasets X_{inc_train} and X_{azi_train} are used to find the optimal standard deviation thresholds for inclination data and azimuth data, respectively. The window size N_W of the mean filter in Algorithm 2 is set to 5, and the initial value of the standard deviation threshold is flexibly selected based on the convergence situation. The quantization bits are set to 4, 5, 6, 7, and 8, respectively, to obtain the optimal filtering threshold T_W for different compression multiples and errors.

4. Analysis and Discussion of the Experimental Results

To verify the effectiveness of the method proposed in this paper, the following compression methods were used for comparative experiments:

DeepAE+QC+HC+F (the proposed method): The wellbore stability monitoring data compression method based on deep autoencoder, consisting of the deep autoencoder (deepAE), quantization coding (QC), Huffman coding (HC), and the mean filtering method based on optimal standard deviation threshold (F).

DPCM: The DPCM method in ref. [23]. This method uses the simplest first-order predictor to compress LWD data.

DPCM-I: The DPCM method in ref. [24]. This method requires the use of previously measured LWD data, using the minimum mean square error as a criterion to determine the optimal predictor parameters, and then using these parameters to compress the current data that need to be compressed. Its performance is stronger than that of the DPCM method.

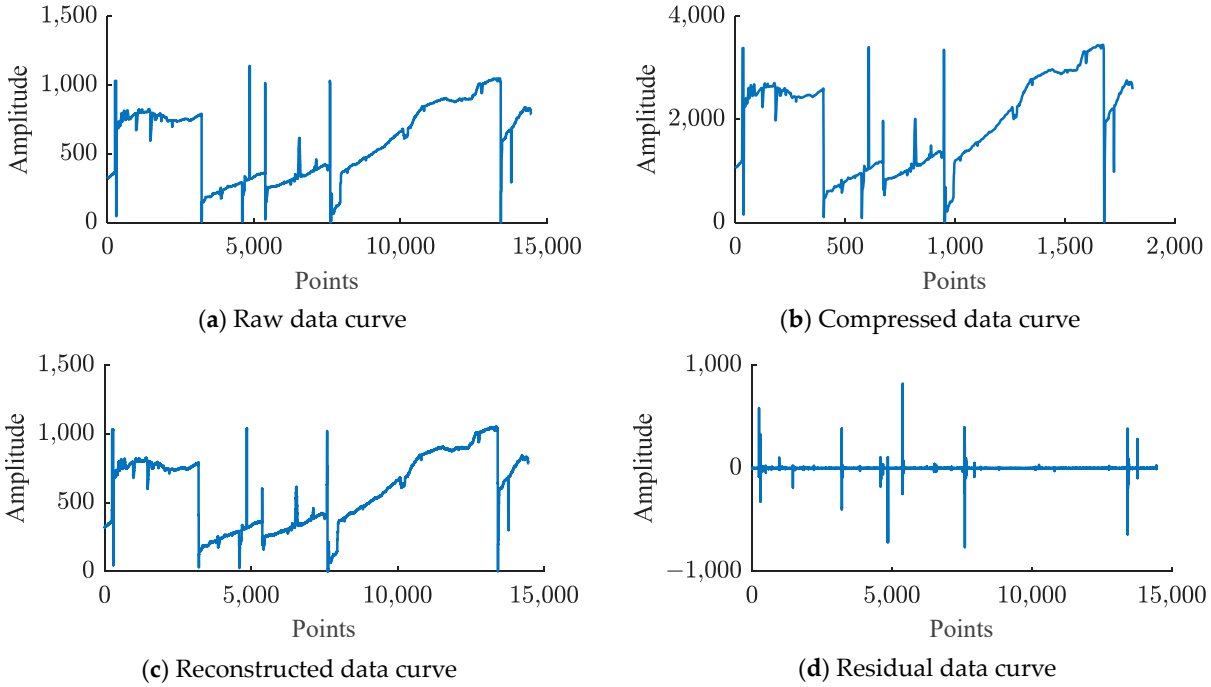
deepAE: The deep autoencoder method. Its structure adopts the structure shown in Table 1, and it is trained using Algorithm 1 in Section 2.6.1 and datasets X_{inc_train} and X_{azi_train}.

deepAE+QC+HC: The deep autoencoder (deepAE), combining the quantization encoding (QC) and Huffman coding (HC) methods. Firstly, use the deep autoencoder to compress the data, then perform quantization encoding and Huffman coding on the obtained residual data. Finally, compensate for the reconstructed data in the decoder section of the deep autoencoder.

All deepAE methods mentioned above are the same, have the same structure, and use the same training dataset for training, making it easier to verify the functionality of different components of the proposed method.

4.1. Data Feature Extraction Results

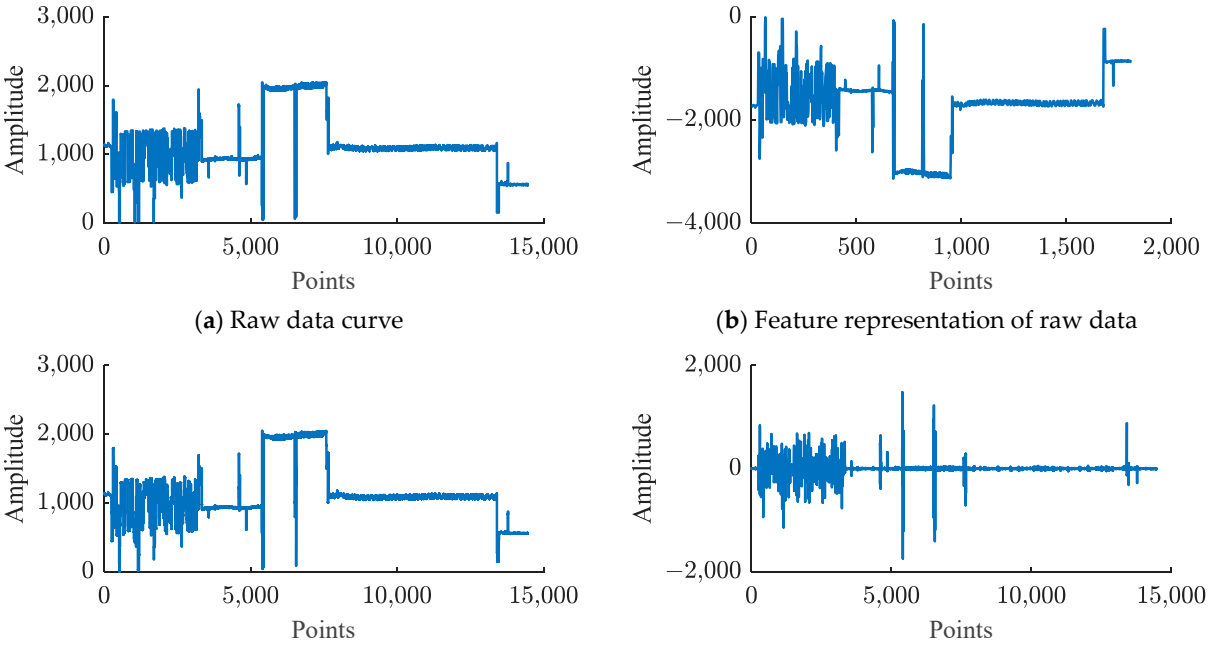
The compression results of X_{inc_test} and X_{azi_test} data using a trained deep autoencoder are shown in Figures 6 and 7.



(**a**) Raw data curve (**b**) Compressed data curve

(**c**) Reconstructed data curve (**d**) Residual data curve

Figure 6. Compression results of deep autoencoder on dataset X_{inc_test}: (**a**) represents the original data of X_{inc_test} and X_{azi_test}, (**b**) represents the features extracted by the deep autoencoder, (**c**) represents the reconstructed data of the deep autoencoder, and (**d**) represents the residual between the raw data and the reconstructed data.



(**a**) Raw data curve (**b**) Feature representation of raw data

(**c**) Reconstructed data curve (**d**) Residual data curve

Figure 7. Compression results of deep autoencoder on dataset X_{azi_test}: (**a**) represents the original data of X_{inc_test} and X_{azi_test}, (**b**) represents the features extracted by the deep autoencoder, (**c**) represents the reconstructed data of the deep autoencoder, and (**d**) represents the residual between the raw data and the reconstructed data.

In Figures 6 and 7, due to the input data dimension D_{in} of the deep autoencoder being 8, according to the structure of the deep autoencoder in Table 1, the bottleneck layer has a dimension of 1. Hence the extracted features have a dimension of 1 after compression. Therefore, (b) in Figures 6 and 7 show all the features extracted by the deep autoencoder. Comparing (a) and (b) in Figures 6 and 7 reveals that although the number of points in the proposed feature data is smaller than that of the raw data, its shape is very similar or symmetrical to the raw data, indicating that the deep autoencoder effectively extracts the main features of the raw data. The (d) figures in Figures 6 and 7 illustrate that after feature extraction by a deep autoencoder, there are still small differences between the reconstructed data and the original data. Therefore, using quantization coding and Huffman coding to compress the residuals will help reduce the error between the reconstructed data and the source data.

4.2. Comparison of Compression Results

We used the proposed method, deepAE, DPCM, and DPCM-I to compress X_{inc_test} and X_{azi_test}.

Due to the input dimension of the proposed method and deepAE method, the shape of the X_{inc_test} and X_{azi_test} data was reset to (18,088). In addition to the deepAE method, the quantization data bits of the proposed method, DPCM method, and DPCM-I method were set to 4, 5, 6, 7, and 8 to verify the error reduction performance under different quantization data bits. The compression results are shown in Table 2.

$X_{inc\_test}X_{azi\_test}X_{inc\_test}X_{azi\_test}X_{inc\_test}X_{azi\_test}X_{inc\_test}X_{azi\_test}$ According to Table 2, to verify the performance improvement of the proposed method, we used Formulas (20)~(22) to calculate the compression ratio increment ratio (ΔCR), signal-to-noise ratio increment ratio (ΔSNR), and mean square error increment ratio (ΔMSE) of the proposed method relative to deepAE, DPCM, and DPCM-I. The results are shown in Table 3.

According to Tables 2 and 3, it can be seen that compared to the DPCM and DPCM-I methods, the deep autoencoder (deepAE) shows a significant improvement in compression ratio (CR), reaching 7.33, but its error is also significant, with a signal-to-noise ratio (SNR) of only 25.25 dB and a mean square error (MSE) of 4122.73. On the basis of deepAE, our proposed method was enhanced. Although the compression ratio was reduced by 44.69%, the error was significantly reduced, the signal-to-noise ratio was improved by 79.73%, and the mean square error was reduced by 97.65%. The proposed method maintains high compression performance while significantly reducing errors, thus achieving a better balance.

In addition, it can be seen that under the same number of quantization bits, the compression ratio and signal-to-noise ratio of the proposed method are overwhelmingly higher than those of DPCM method and DPCM-I method, while MSE is overall lower than those of DPCM and DPCM-I. Among them, under the quantization bits of 4~8 bits, the compression ratio of the proposed method for the inclination angle data X_{inc_test} and azimuth data X_{azi_test} is 3.22~4.38, with an average compression ratio of 4.05. Compared to DPCM and DPCM-I, the proposed method improves the compression ratio by 53.80~203.78%, with an average improvement of 118.54%. Meanwhile, the signal-to-noise ratio of the proposed method reached 35.34~53.22 dB, with an average of 45.09 dB; compared to DPCM-I, it is improved by 6.46~81.39%, with an average improvement of 32.40%. Compared to DPCM, the signal-to-noise ratio is improved by 7.78~84.09%, with an average improvement of 35.84%. The mean square error of the proposed method reached 1.98~428.61, with an average of 76.88; compared to DPCM-I, it decreased by 51.76~98.00%, with an average decrease of 82.46%. Compared with DPCM, it decreased by 57.98~98.21%, with an average decrease of 86.31%. These results indicate that the compression ratio of the proposed method is significantly improved compared to the DPCM and DPCM-I, and the error of the reconstructed data is significantly reduced.

Table 2. Comparison results of the proposed method, deepAE, DPCM, and DPCM-I.

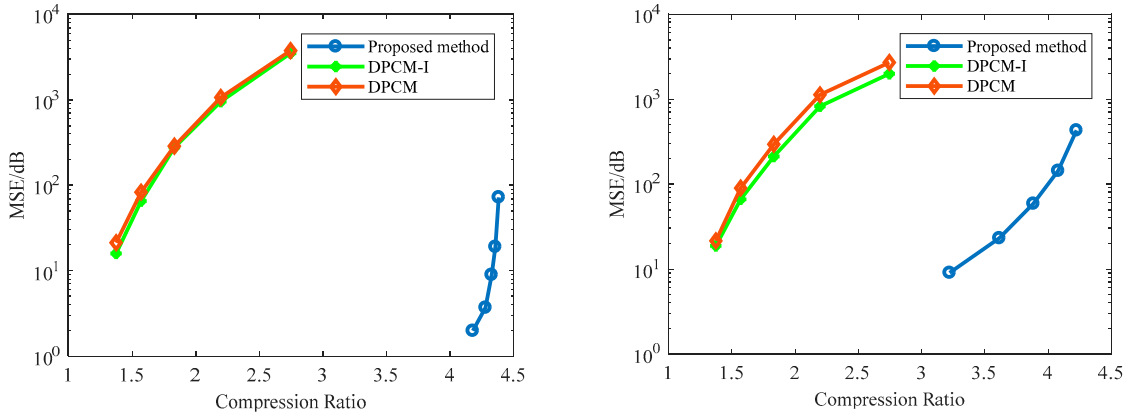
Method	Data Set	Quantization Data Bits	Tw	CR	SNR/dB	MSE
DPCM	X_{inc_test}	4	-	2.75	20.42	3767.45
		5	-	2.20	25.90	1067.62
		6	-	1.83	31.60	286.80
		7	-	1.57	36.99	83.06
		8	-	1.38	42.90	21.29
	X_{azi_test}	4	-	2.75	27.34	2704.75
		5	-	2.20	31.15	1124.74
		6	-	1.83	36.98	293.73
		7	-	1.57	42.13	89.66
		8	-	1.38	48.34	21.47
	Average			1.94	34.37	946.06
DPCM-I	X_{inc_test}	4	-	2.75	20.72	3512.95
		5	-	2.20	26.39	952.75
		6	-	1.83	31.77	276.33
		7	-	1.57	38.03	65.28
		8	-	1.38	44.16	15.92
	X_{azi_test}	4	-	2.75	28.68	1986.52
		5	-	2.20	32.50	824.54
		6	-	1.83	38.42	210.84
		7	-	1.57	43.46	66.09
		8	-	1.38	48.94	18.70
	Average			1.94	35.31	792.99
deepAE	X_{inc_test}	-	-	7.33	27.60	721.63
	X_{azi_test}	-	-	7.33	22.89	7523.82
	Average			7.33	25.25	4122.73
The proposed method	X_{inc_test}	4	53.25	4.38	37.59	72.26
		5	45.51	4.35	43.38	19.06
		6	33.61	4.33	46.66	8.96
		7	21.81	4.28	50.50	3.70
		8	11.01	4.18	53.22	1.98
	X_{azi_test}	4	55.23	4.23	35.34	428.61
		5	50.05	4.08	40.10	143.27
		6	24.50	3.88	43.96	58.82
		7	11.79	3.62	48.03	23.05
		8	6.18	3.22	52.10	9.02
	Average			4.05	45.09	76.87

Figure 8 compares the compression performance between the proposed method, DPCM-I, and DPCM. It is evident that the proposed method generally achieves a higher compression ratio than DPCM-I and DPCM, while also exhibiting a lower overall mean square error (MSE). Even in cases where the MSE is similar, the proposed method consistently achieves a higher compression ratio, indicating superior compression performance.

To intuitively demonstrate the compression performance of the proposed method, deepAE, DPCM-I, and DPCM, the reconstructed data curves of the inclination data (X_{inc_test}) and azimuth data (X_{azi_test}) using these methods under 8-bit quantization are plotted and compared with the original data curves. The results are shown in Figure 9.

Table 3. Performance improvement of the proposed method relative to DPCM, DPCM-I, and deepAE.

Method Used for Comparison	Data Set	Quantization Data Bits	ΔCR	ΔSNR	ΔMSE
DPCM	X_{inc_test}	4	59.48%	84.09%	−98.08%
		5	98.00%	67.52%	−98.21%
		6	136.08%	47.64%	−96.88%
		7	172.25%	36.54%	−95.55%
		8	203.78%	24.06%	−90.70%
	X_{azi_test}	4	53.80%	29.28%	−84.15%
		5	85.54%	28.74%	−87.26%
		6	112.01%	18.88%	−79.98%
		7	130.11%	14.00%	−74.29%
		8	134.33%	7.78%	−57.98%
	Average		118.54%	35.85%	−86.31%
DPCM-I	X_{inc_test}	4	59.48%	81.39%	−97.94%
		5	98.00%	64.38%	−98.00%
		6	136.08%	46.89%	−96.76%
		7	172.25%	32.78%	−94.33%
		8	203.78%	20.52%	−87.56%
	X_{azi_test}	4	53.80%	23.23%	−78.42%
		5	85.54%	23.40%	−82.62%
		6	112.01%	14.43%	−72.10%
		7	130.11%	10.53%	−65.12%
		8	134.33%	6.46%	−51.76%
	Average		118.54%	32.40%	−82.46%
deepAE	X_{inc_test}	4	−40.23%	36.20%	−89.99%
		5	−40.60%	57.17%	−97.36%
		6	−41.00%	69.06%	−98.76%
		7	−41.65%	82.97%	−99.49%
		8	−43.02%	92.83%	−99.73%
	X_{azi_test}	4	−42.36%	54.39%	−94.30%
		5	−44.34%	75.19%	−98.10%
		6	−47.01%	92.05%	−99.22%
		7	−50.68%	109.83%	−99.69%
		8	−56.04%	127.61%	−99.88%
	Average		−44.69%	79.73%	−97.65%



(**a**) Compression results for X_{inc_test} data (**b**) Compression results for X_{azi_test} data

Figure 8. Comparison of compression performance between the proposed method, DPCM-I, and DPCM: (**a**) represents the compression results for X_{inc_test} data, (**b**) represents the compression results for X_{azi_test} data.

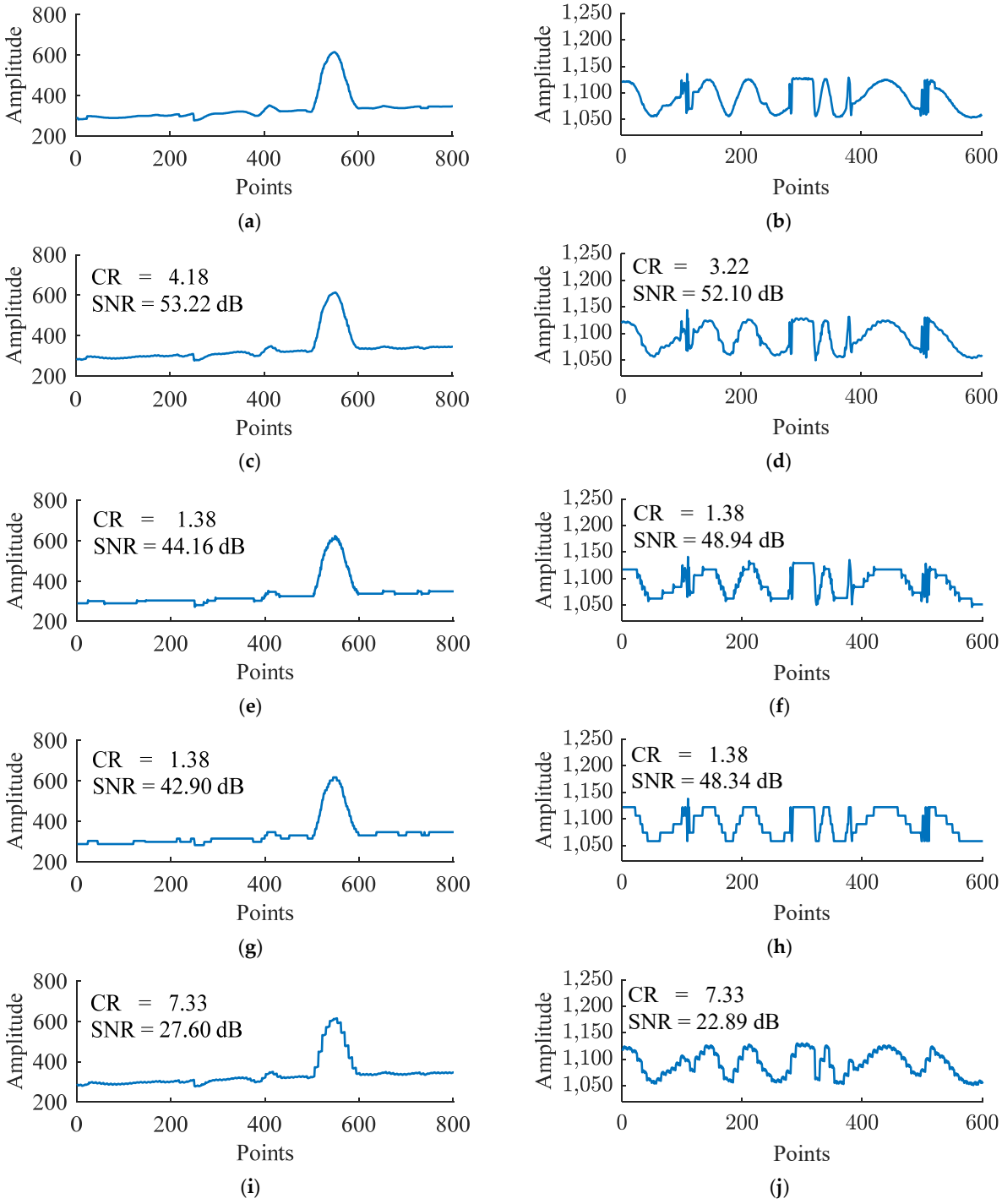


Figure 9. Comparison of the reconstructed data of the proposed method, deepAE, DPCM-I, and DPCM and the raw data: (**a**,**b**) represent a portion of the original data curves in X_{inc_test} and X_{azi_test}, respectively. (**c**,**e**,**g**,**i**) represent the reconstructed data corresponding to (**a**) after compressing and decompressing X_{inc_test} data using the proposed method, DPCM-I, DPCM and deepAE, respectively; (**d**,**f**,**h**,**j**) represent the reconstructed data corresponding to (**b**) after compressing and decompressing X_{azi_test} data using the proposed method, DPCM-I, DPCM, and deepAE, respectively.

In Figure 9, by observing and comparing Figure 9a,c,e,g,i, it can be seen that the reconstructed data curve of the proposed method is closest to the original data curve, followed by DPCM-I and then DPCM, while the reconstructed data curve of deepAE has the largest difference between the reconstructed data curve and the original data

curve, which is consistent with the compression ratio and signal-to-noise ratio reflected in Figure 9c,e,g,i. Similarly, by observing and comparing Figure 9b,d,f,h,j, the same conclusion can be drawn. The above results indicate that compared with the DPCM methods, the proposed method in this paper has a higher compression ratio for inclination and azimuth data, and at the same time, it has a smaller error in reconstructing the data curve, which is more in line with the original data curve.

4.3. Ablation Study

To quantitatively ascertain the contributory impact of individual components within the proposed method, a meticulously structured series of ablation study analyses were undertaken. The purpose of these analyses is to demonstrate the effective role of newly integrated components in terms of compression performance or reducing errors. Among them, for methods containing quantization coding, the quantization data bits are set to 4, 5, 6, 7, and 8 to fully verify the compression performance of these methods under different quantization data bits. Table 4 presents the results of the ablation experiment.

Table 4. Ablation experimental results. √ represents that the module is used. × represents that the module is not used.

deepAE	QC+HC	F	Data Set	Quantization Data Bits	Tw	CR	SNR/dB	MSE
√	×	×	X_{inc_test}	-	-	7.33	27.60	721.63
			X_{azi_test}	-	-	7.33	22.89	7523.82
				Mean		7.33	25.25	4122.73
√	√	×	X_{inc_test}	4	-	4.38	36.98	83.25
				5	-	4.35	42.17	25.15
				6	-	4.33	44.90	13.42
				7	-	4.28	48.08	6.45
				8	-	4.18	49.93	4.22
			X_{azi_test}	4	-	4.23	34.86	478.87
				5	-	4.08	39.22	175.22
				6	-	3.88	42.75	77.85
				7	-	3.62	46.49	32.86
				8	-	3.22	50.19	14.01
				Mean		4.06	43.56	91.13
√	√	√	X_{inc_test}	4	53.25	4.38	37.59	72.26
				5	45.51	4.35	43.38	19.06
				6	33.61	4.33	46.66	8.96
				7	21.81	4.28	50.50	3.70
				8	11.01	4.18	53.22	1.98
			X_{azi_test}	4	55.23	4.23	35.34	428.61
				5	50.05	4.08	40.10	143.27
				6	24.50	3.88	43.96	58.82
				7	11.79	3.62	48.03	23.05
				8	6.18	3.22	52.10	9.02
				Mean		4.06	45.09	76.87

In the ablation study, the benchmark model selected was the deepAE method; as shown in Table 4, the addition of QC+HC and F continuously reduced the error of the reconstructed data of the benchmark model, while the compression ratio was still at a relatively high level.

By introducing QC+HC, the mean square error was reduced from 4122.73 to 91.13, a decrease of 97.79%. Although the compression ratio decreased from 7.33 to 4.06 (a decrease of 44.61%), as shown in Table 3, the current compression ratio was 118.54% higher than the DPCM-I method and still at a relatively high level.

By introducing F, the mean square error was further reduced to 76.87, which is 98.14% lower than the benchmark model. The proposed mean filter based on optimal standard deviation threshold effectively further reduced errors.

The ablation experiment conducted shows that the proposed combination effectively reduces errors while maintaining a high level of compression ratio. Therefore, the proposed method achieves a better balance between compression performance and error compared to the benchmark model.

To more intuitively demonstrate the progressive effect of each component of the proposed method on reducing the error between reconstructed data and original data, the reconstructed data curves of the deepAE method, the deepAE+QC+HC method, and the proposed method (deepAE+QC+HC+F) on X_{inc_test} and X_{azi_test} data were plotted and compared with the original data curves. The results are shown in Figure 10.

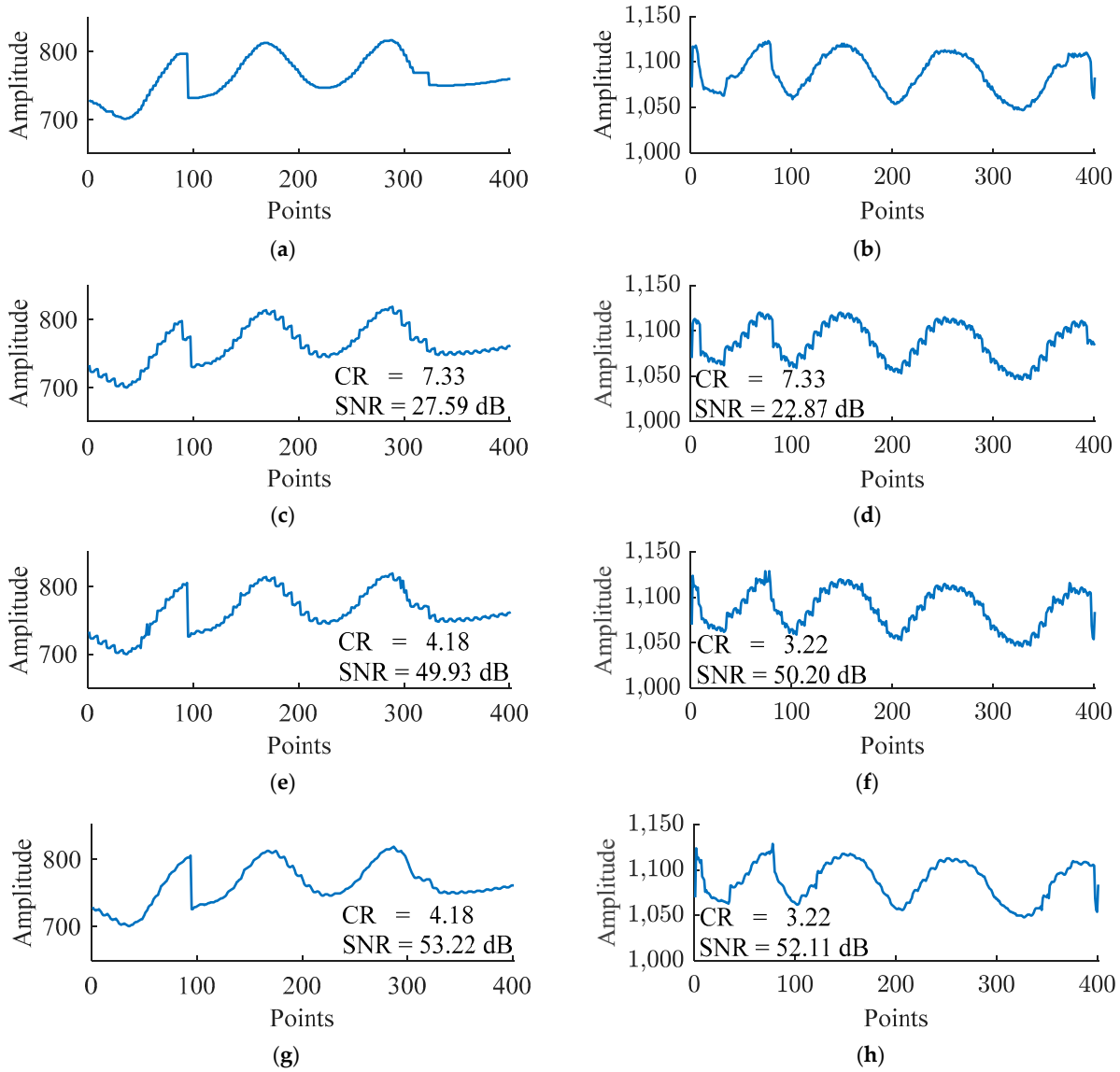


Figure 10. Comparison of differences between the reconstructed data curves of deepAE, deepAE+QC+HC, and the proposed method and the original data curves: (**a**,**b**) represent a portion of the original data curves in X_{inc_test} and X_{azi_test}, respectively; (**c**,**e**,**g**) represent the reconstructed data corresponding to (**a**) after compressing and decompressing X_{inc_test} data using deepAE, deepAE+QC+HC, and the proposed method, respectively; (**d**,**f**,**h**) represent the reconstructed data corresponding to (**b**) after compressing and decompressing X_{azi_test} data using deepAE, deepAE+QC+HC, and the proposed method, respectively.

In Figure 10, by comparing Figure 10a,c,e,g, it can be seen that the reconstructed data curve of the deepAE method shows the greatest difference from the original data curve, followed by the deepAE+QC+HC method. The reconstructed data curve of our proposed method is closest to the original data curve, which is consistent with the signal-to-noise ratio reflected in Figure 10c,e,g. By comparing Figure 10b,d,f,h, the same conclusion can be drawn. The above results indicate that as the QC+HC component and F component are sequentially added to the benchmark model deepAE, the error of reconstructed data becomes smaller and closer to the original data curve. This result further demonstrates that our proposed method effectively reduces the error of the reconstructed data and the original data and maintains a high level of compression ratio, achieving a better balance.

The above results indicate that our proposed method effectively improves the compression ratio and reduces reconstruction data errors, which is more conducive to improving the performance of wellbore stability monitoring and achieving the goal of improving production safety.

5. Conclusions

This paper introduced a novel compression method utilizing a deep autoencoder to significantly enhance the compression ratio of inclination and azimuth data during drilling operations. Additionally, residual data were encoded using quantization coding and Huffman coding, effectively minimizing the error between reconstructed data and source data. Moreover, a mean filtering technique based on the optimal standard deviation threshold was employed to further mitigate the error between the reconstructed data and the source data. Simulation testing on inclination and azimuth data demonstrates that the average compression ratio of the proposed method is 4.05; compared to the DPCM methods, it is improved by 118.54%. Meanwhile, the average mean square error of the proposed method is 76.88, which is decreased by 82.46% when compared to the DPCM method. The results of the ablation experiment also indicate that our method achieves a better balance between compression performance and error compared to deep autoencoders.

This work pioneers the use of a deep autoencoder-based methods for compressing wellbore trajectory data and integrates classical compression and filtering methods to effectively process residual and reconstructed data, thereby enhancing compression performance and reducing errors between reconstructed data and source data. The compression method proposed in this paper for wellbore stability monitoring data has real-time performance, a higher compression ratio, and a smaller reconstruction data error, solving the problems of low real-time performance, low compression ratio, and large error in the existing methods. This not only provides valuable insights for the advancement of data compression technology related to LWD data but also holds significant implications for enhancing the safety monitoring performance of wellbores in logging for drilling engineering.

Considering the deep autoencoder's excellent capability in feature extraction and the fact that the proposed method in this paper integrates neural networks with classical data compression techniques, the proposed method holds reference value for the study of other types of LWD data compression methods. Additionally, it also provides a certain degree of reference significance for the research of data compression methods in other fields.

Although the proposed method effectively improves the performance of wellbore stability monitoring, the method only discusses a simple deep autoencoder structure. In other fields, such as image data compression, many excellent neural network structures have been used for data compression. Therefore, future research can focus on the exploration and innovation of neural network structures to further improve the compression performance and reduce the error of the reconstructed data and the original data.

Author Contributions: Conceptualization, S.S. and Z.Z.; methodology, S.S., Z.Z. and X.Z.; software, S.S.; validation, S.S. and X.Z.; investigation, X.Z. and Z.Z.; writing—original draft preparation, S.S.; writing—review and editing, X.Z., S.S., Z.Z. and M.L.; visualization, X.Z. and S.S.; supervision, X.Z. and M.L.; data curation, X.Z. and Z.Z.; funding acquisition, X.Z. and M.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Natural Science Foundation of China (Grad. No. 51978079, Grad. No. 61901059) and the Hubei Provincial Outstanding Young and Middle-Aged Science and Technology Innovation Team Project (Grad. No. T2020007).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to project confidentiality.

Conflicts of Interest: Author Xiaoyong Zhao was employed by the Directional Drilling Branch of China National Petroleum Corporation Bohai Drilling Engineering Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Fouda, M.; Taher, A.; Hussein, M.; Al-Hassan, M. Advanced Techniques for Wellbore Stability Evaluation Using Logging-While-Drilling Technologies. In Proceedings of the ARMA/DGS/SEG International Geomechanics Symposium, Al Khobar, Saudi Arabia, 30 October–2 November 2023; p. ARMA–IGS-2023-0087.
2. Ciuperca, C.-L.; Di Tommaso, D.; Dawber, M.; Tidswell, J. Determining Wellbore Stability Parameters Using a New LWD High Resolution Ultrasonic Imaging Tool. In Proceedings of the SPE/IADC Drilling Conference and Exhibition, The Hague, The Netherlands, 5–7 March 2019; p. D021S009R002.
3. Stricker, K.; Schimschal, S.; Müller, B.; Wessling, S.; Bender, F.; Kohl, T. Importance of drilling-related processes on the origin of borehole breakouts—Insights from LWD observations. *Geomech. Energy Environ.* **2023**, *34*, 100463. [CrossRef]
4. Greten, A.; Brahim, I.B.; Emmerich, W.; Akimov, O. Reliable Mud-Pulse Telemetry System for High-Resolution Real-Time Logs. In Proceedings of the SPE/IADC Drilling Conference and Exhibition, The Hague, The Netherlands, 14–16 March 2017.
5. Mwachaka, S.M.; Wu, A.P.; Fu, Q.Q. A review of mud pulse telemetry signal impairments modeling and suppression methods. *J. Pet. Explor. Prod. Technol.* **2019**, *9*, 779–792. [CrossRef]
6. Li, C.; Xu, Z. A Review of Communication Technologies in Mud Pulse Telemetry Systems. *Electronics* **2023**, *12*, 3930. [CrossRef]
7. Siu, S.; Ji, Q.; Wu, W.; Song, G.; Ding, Z. Stress wave communication in concrete: I. Characterization of a smart aggregate based concrete channel. *Smart Mater. Struct.* **2014**, *23*, 125030. [CrossRef]
8. Wu, A.; He, S.; Ren, Y.; Wang, N.; Ho, S.C.M.; Song, G. Design of a new stress wave-based pulse position modulation (PPM) communication system with piezoceramic transducers. *Sensors* **2019**, *19*, 558. [CrossRef] [PubMed]
9. He, S.; Wang, N.; Ho, M.; Zhu, J.; Song, G. Design of a new stress wave communication method for underwater communication. *IEEE Trans. Ind. Electron.* **2020**, *68*, 7370–7379. [CrossRef]
10. Zhang, G.; Yang, P.; He, S.; Zheng, Y.; Song, G. A power waveform design based on OVSF-PPM for stress wave based wireless power transfer. *Mech. Syst. Signal Process.* **2021**, *147*, 107111. [CrossRef]
11. Alkamil, E.H.; Abbood, H.R.; Flori, R.E.; Eckert, A. Case study of wellbore stability evaluation for the Mishrif Formation, Iraq. *J. Pet. Sci. Eng.* **2018**, *164*, 663–674. [CrossRef]
12. Khan, K.; Altwaijri, M.; Taher, A.; Fouda, M.; Hussein, M. Real-Time Wellbore Stability and Hole Quality Evaluation Using LWD Azimuthal Photoelectric Measurements. In Proceedings of the SPE Middle East Oil and Gas Show and Conference, Sanabis, Bahrain, 28 November–1 December 2021; p. D021S008R009.
13. Huffman, D.A. A method for the construction of minimum-redundancy codes. *Proc. IRE* **1952**, *40*, 1098–1101. [CrossRef]
14. Donoho, D.L. Compressed sensing. *IEEE Trans. Inf. Theory* **2006**, *52*, 1289–1306. [CrossRef]
15. Shensa, M.J. The discrete wavelet transform: Wedding the a trous and Mallat algorithms. *IEEE Trans. Signal Process.* **1992**, *40*, 2464–2482. [CrossRef]
16. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
17. O'Neal, J., Jr. Predictive quantizing systems (differential pulse code modulation) for the transmission of television signals. *Bell Syst. Tech. J.* **1966**, *45*, 689–721. [CrossRef]
18. Song, S.; Lian, T.; Liu, W.; Zhang, Z.; Luo, M.; Wu, A. A lossless compression method for logging data while drilling. *Syst. Sci. Control Eng.* **2021**, *9*, 689–703. [CrossRef]
19. Li, J.; Dai, B.; Jones, C.M.; Samson, E.M.; Gascooke, D. Downhole Signal Compression and Surface Reconstruction Based on Dictionary Machine Learning. In Proceedings of the SPWLA Annual Logging Symposium, Virtual Online Webinar, 24 June–29 July 2020; p. D363S025R006.
20. Jarrot, A.; Gelman, A.; Kusuma, J. Wireless Digital Communication Technologies for Drilling: Communication in the Bits\/s Regime. *IEEE Signal Process. Mag.* **2018**, *35*, 112–120. [CrossRef]
21. Yan, Z.D.; Wang, J.F.; Sheng, L.; Yang, Z.Y. An effective compression algorithm for real-time transmission data using predictive coding with mixed models of LSTM and XGBoost. *Neurocomputing* **2021**, *462*, 247–259. [CrossRef]
22. Chen, T.; Guestrin, C. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM Sigkdd International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 785–794.

23. Zhang, Y.; Xiong, K.; Qiu, Z.; Wang, S.; Sun, D. A new method for real-time LWD data compression. In Proceedings of the 2009 International Symposium on Information Processing (ISIP 2009), Huangshan, China, 21–23 August 2009; p. 163.
24. Zhang, Y.; Wang, S.; Xiong, K.; Qiu, Z.; Sun, D. DPCM Compression for Real-Time Logging While Drilling Data. *J. Softw.* **2010**, *5*, 280–287. [CrossRef]
25. Kim, H.; Nam, S.; Nam, E. Estimation of Shape Error with Monitoring Signals. *Sensors* **2023**, *23*, 9416. [CrossRef]
26. Zhong, Z.; Li, H. Recognition and prediction of ground vibration signal based on machine learning algorithm. *Neural Comput. Appl.* **2020**, *32*, 1937–1947. [CrossRef]
27. Yi, H. Efficient machine learning algorithm for electroencephalogram modeling in brain–computer interfaces. *Neural Comput. Appl.* **2022**, *34*, 9233–9243. [CrossRef]
28. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
29. Abbaschian, B.J.; Sierra-Sosa, D.; Elmaghraby, A. Deep Learning Techniques for Speech Emotion Recognition, from Databases to Models. *Sensors* **2021**, *21*, 1249. [CrossRef] [PubMed]
30. Eddahmani, I.; Pham, C.-H.; Napoléon, T.; Badoc, I.; Fouefack, J.-R.; El-Bouz, M. Unsupervised Learning of Disentangled Representation via Auto-Encoding: A Survey. *Sensors* **2023**, *23*, 2362. [CrossRef] [PubMed]
31. Al-Ashwal, N.H.; Al Soufy, K.A.M.; Hamza, M.E.; Swillam, M.A. Deep Learning for Optical Sensor Applications: A Review. *Sensors* **2023**, *23*, 6486. [CrossRef] [PubMed]
32. Nuha, H.H.; Balghonaim, A.; Liu, B.; Mohandes, M.; Deriche, M.; Fekri, F. Deep neural networks with extreme learning machine for seismic data compression. *Arab. J. Sci. Eng.* **2020**, *45*, 1367–1377. [CrossRef]
33. Liu, J.Y.; Di, S.; Zhao, K.; Jin, S.; Tao, D.W.; Liang, X.; Chen, Z.Z.; Cappello, F.; Soc, I.C. Exploring Autoencoder-based Error-bounded Compression for Scientific Data. In Proceedings of the IEEE International Conference on Cluster Computing (Cluster), Electr Network, Portland, OR, USA, 7–10 September 2021; pp. 294–306.
34. 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.
35. Lindstrom, P. Fixed-rate compressed floating-point arrays. *IEEE Trans. Vis. Comput. Graph.* **2014**, *20*, 2674–2683. [CrossRef] [PubMed]
36. Jalilian, E.; Hofbauer, H.; Uhl, A. Iris Image Compression Using Deep Convolutional Neural Networks. *Sensors* **2022**, *22*, 2698. [CrossRef] [PubMed]
37. Candido de Oliveira, D.; Nassu, B.T.; Wehrmeister, M.A. Image-Based Detection of Modifications in Assembled PCBs with Deep Convolutional Autoencoders. *Sensors* **2023**, *23*, 1353. [CrossRef] [PubMed]
38. Shinde, A.B.; Bagade, J.; Bhimanpallewar, R.; Dandawate, Y.H. Image Compression of Handwritten Devanagari Text Documents Using a Convolutional Autoencoder. *Int. J. Intell. Syst. Appl. Eng.* **2023**, *11*, 449–457.
39. Hinton, G.E.; Osindero, S.; Teh, Y.-W. A fast learning algorithm for deep belief nets. *Neural Comput.* **2006**, *18*, 1527–1554. [CrossRef]
40. Yildirim, O.; San Tan, R.; Acharya, U.R. An efficient compression of ECG signals using deep convolutional autoencoders. *Cogn. Syst. Res.* **2018**, *52*, 198–211. [CrossRef]
41. Kuester, J.; Gross, W.; Middelmann, W. An Approach to Near-lossless Hyperspectral Data Compression using Deep Autoencoder. In Proceedings of the Conference on Image and Signal Processing for Remote Sensing XXVI, Electr Network, Online, 21–25 September 2020.
42. Tieleman, T.; Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Netw. Mach. Learn.* **2012**, *4*, 26–31.
43. Dhar, J. An adaptive intelligent diagnostic system to predict early stage of parkinson's disease using two-stage dimension reduction with genetically optimized lightgbm algorithm. *Neural Comput. Appl.* **2022**, *34*, 4567–4593. [CrossRef]
44. Zhao, S.; Tu, K.; Ye, S.; Tang, H.; Hu, Y.; Xie, C. Land Use and Land Cover Classification Meets Deep Learning: A Review. *Sensors* **2023**, *23*, 8966. [CrossRef] [PubMed]
45. Pei, X.L.; Zheng, X.Y.; Wu, J.L. Intelligent bearing fault diagnosis based on Teager energy operator demodulation and multiscale compressed sensing deep autoencoder. *Measurement* **2021**, *179*, 15. [CrossRef]
46. Yang, Z.; Xu, B.B.; Luo, W.; Chen, F. Autoencoder-based representation learning and its application in intelligent fault diagnosis: A review. *Measurement* **2022**, *189*, 20. [CrossRef]
47. Zou, F.; Shen, L.; Jie, Z.; Zhang, W.; Liu, W. A sufficient condition for convergences of adam and rmsprop. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 11127–11135.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.