2021 IEEE International Conference on Cluster Computing (CLUSTER) | 978-1-7281-9666-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/Cluster48925.2021.00034

Exploring Autoencoder-based Error-bounded Compression for Scientific Data

Jinyang Liu,* Sheng Di,† Kai Zhao,* Sian Jin,‡ Dingwen Tao,‡ Xin Liang,§ Zizhong Chen,* Franck Cappello†¶
*University of California, Riverside, CA, USA
†Argonne National Laboratory, Lemont, IL, USA
‡Washington State University, Pullman, WA, USA
§ Missouri University of Science and Technology, Rolla, MO, USA
¶ University of Illinois at Urbana-Champaign, Urbana, IL, USA
jliu447@ucr.edu, sdi@anl.gov, kzhao016@ucr.edu, sian.jin@wsu.edu,
dingwen.tao@wsu.edu, xliang@mst.edu, zizhong.chen@ucr.edu, cappello@mcs.anl.gov

***Abstract*—Error-bounded lossy compression is becoming an indispensable technique for the success of today's scientific projects with vast volumes of data produced during the simulations or instrument data acquisitions. Not only can it significantly reduce data size, but it also can control the compression errors based on user-specified error bounds. Autoencoder (AE) models have been widely used in image compression, but few AE-based compression approaches support error-bounding features, which are highly required by scientific applications. To address this issue, we explore using convolutional autoencoders to improve error-bounded lossy compression for scientific data, with the following three key contributions. (1) We provide an in-depth investigation of the characteristics of various autoencoder models and develop an error-bounded autoencoder-based framework in terms of the SZ model. (2) We optimize the compression quality for main stages in our designed AE-based error-bounded compression framework, fine-tuning the block sizes and latent sizes and also optimizing the compression efficiency of latent vectors. (3) We evaluate our proposed solution using five real-world scientific datasets and comparing them with six other related works. Experiments show that our solution exhibits a very competitive compression quality from among all the compressors in our tests. In absolute terms, it can obtain a much better compression quality (100%∼800% improvement in compression ratio with the same data distortion) compared with SZ2.1 and ZFP in cases with a high compression ratio.**

I. Introduction

Today's scientific applications are producing extremely large amounts of data during simulation or instrument data acquisition. Advanced instruments such as the Linac Coherent Light Source (LCLS) [1] and Advanced Photon Source [2], for example, may produce vast amounts of data with a very high data acquisition rate (250 GB/s [3]). Consequently, reducing the data volumes with user-tolerable data distortion is critical to the efficient data storage and transfer.

Error-bounded lossy compression is arguably the most efficient way to significantly reduce the data volumes for scientific applications with big data issues. Unlike lossless compressors [4]–[7] that suffer from very low compression ratios (generally ∼2:1) on floating-point datasets, error-bounded lossy compressors can obtain fairly high compression ratios (10+ or even several hundreds [3], [8], [9]). Moreover, error-bounded lossy compressors are able to keep a high fidelity of the reconstructed data for the user's post hoc analysis based on the user's required bounds on data distortion, as verified by many recent studies [10]–[12].

Error-bounded lossy compressors can be split into two models: prediction-based and transform-based models. Prediction-based compressors (such as SZ [8], [9]) may suffer from low reconstructed data quality at high compression ratios because they have to predict each data point using the reconstructed data values nearby instead of original data, in order to guarantee the bounded errors during the decompression. To obtain a high compression ratio, therefore, one has to set the error bound relatively large; and as a result, the data prediction accuracy can be degraded significantly because of large errors in the reconstructed data, leading to limited compression ratios in turn. For transform-based compressors (such as ZFP [13]), large compression ratio means that a very limited number of coefficients or bit-planes can be preserved for reconstructing the data and thus will considerably lower the data reconstruction quality.

As a classic type of deep learning model, the Autoencoder (AE) has been gaining more and more attention. Such a deep neural network (DNN) architecture is composed of both an encoder (encoding the input data) and a decoder (decoding the encoded data) and is trained to minimize the error between the reconstructed data and the initial data. In general, because the trained encoder and decoder can be used separately, AE can be used to learn efficient data representation (or coding), typically for dimensionality reduction. The corresponding DNN will be trained to reconstruct the main patterns in the dataset effectively based on the reduced information generated from the original data. Recently, several variations of the AE have been developed with different model frameworks and training paradigms for improving the effectiveness of data reconstruction and for handling more tasks such as data generation. Nevertheless, although AE has been widely used in the image compression domain, few studies explored the possibility of leveraging it for error-bounded compression

Corresponding author: Sheng Di, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 Cass Avenue, Lemont, IL 60439, USA

978-1-7281-9666-4/21/$31.00 ©2021 IEEE
DOI 10.1109/Cluster48925.2021.00034

models for scientific datasets.

In this paper we explore the possibility of leveraging the AE model to improve the error-bounded lossy compression significantly. Such a study faces several challenges. First, many types of autoencoders exist, each with different architectures or training methods, so that determining the most effective AE model is challenging. Second, applying AE in the error-bounded model with a proper configuration setting is nontrivial. Third, latent vectors from AE need to be stored in the compressed data, so minimizing the latent vector overhead while maintaining a high reconstruction quality is challenging and critical to getting a good rate distortion in the high-compression cases.

In this work we propose a novel error-bounded lossy compressor, AE-SZ, which combines the classic prediction-based error-bounded compression framework SZ [8], [9] and the Sliced-Wasserstein Autoencoder (SWAE) model with convolutional neural network implementations. The key contributions of the paper are summarized as follows:

- Our autoencoder-based error-bounded compression framework is designed on the basis of a blockwise model, which can adapt to diverse data changes in a dataset well. To the best of our knowledge, AE-SZ is the first AE-based error-bounded lossy compressor that exhibits a better rate distortion than the three state-of-the-art models SZauto [14], SZ [15], and ZFP [13].
- We investigate various autoencoder models and identify the most effective one for the error-bounded lossy compression model and also carefully optimize the related configurations, such as block sizes and strategies of compressing latent vectors.
- We evaluate the proposed AE-SZ by using the scientific datasets generated by five different real-world high-performance computing (HPC) applications across different domains. We identify the effectiveness of AE-SZ by comparing it with two other AE-based lossy compression methods and four other state-of-the-art error-bounded lossy compressors. Our experiments show that AE-SZ is the best compression method in the category of AE-based compressors. AE-SZ also exhibits competitive rate distortion compared with existing state-of-the-art error-bounded lossy compressors. Specifically, when the compression ratio is greater than 100, AE-SZ can get 100%$\sim$800% higher compression ratios than can SZ2.1 and ZFP, with the same peak signal-to-noise ratio (PSNR).

The rest of this paper is organized as follows. In Section II we discuss related work. In Section III we formulate the research problem. In Section IV we present the overall design of AE-SZ as well as the detailed optimization strategies. In Section V we evaluate our solution using multiple real-world scientific simulation datasets. In Section VI we conclude the paper with a vision of the future work.

II. Related Work

Error-bounded lossy compression techniques have been studied for years, since lossless compression suffer from very low compression ratios (generally 2:1 [16]). Many error-bounded lossy compressors have been developed for compressing scientific datasets [13], [17]–[30]. These can be categorized into prediction-based models (e.g., SZ [8], [9] and FPZIP [17]) and transform-based models (e.g., ZFP [13]). Among these compressors, SZ2.1 [8], [9], [15] and ZFP [13] are the two main state-of-the-art works with wide public usage. Several works have also been developed based on SZ2.1. For example, SZauto [14] merges second-order regression/Lorenzo and automatic parameter tuning in the SZ framework; and SZinterp [31] applies dynamic spline interpolation into data prediction and achieves significant improvement in the prediction accuracy. For assessing the lossy compressors, [32] provides an effective framework.

With the fast growth of deep learning, a recent research trend is leveraging deep learning models such as autoencoders on data compression tasks. Successful works for AE-based image compression include [33]–[39], which design different convolutional networks for image feature extracting and reconstruction and combine them with quantization and encoding algorithms. Unlike scientific lossy compressors, however, those autoencoder-based image compression models are not designed to compress floating-point data and do not provide a strict error-controlling scheme based on scientific user's requirements on post hoc analysis. Recently, a few works have used an autoencoder to compress scientific data. Glaws et al. [40] presented a convolutional autoencoder for lossy compression of turbulence flow simulation data (with a fixed compression ratio of 64). The authors proposed an AE model including 12 residual blocks (i.e., skip connections [41]) to extract features and 3 compression layers to reduce features in both the encoder and decoder. However, different from our work that can provide a strict control of local error (e.g., relative/absolute error) and can be adapted to any scientific datasets, this AE-based scientific compressor is not error-bounded and is designed only for turbulence data. The fixed compression ratio is also its limitation.

Choi et al. [42] proposed another specific variational autoencoder approach for physics plasma simulation data compression. The proposed AE model focuses on minimizing loss of information under physics constraints (e.g., mass, energy, moment) by adopting physics-informed optimization functions and refinement layers. However, unlike this work using quantized latent vectors (integer based), our solution applies lossy compression to floating-point latent vectors, which provides a highly flexible tradeoff between compression and accuracy. In [42] the authors presented a brief version that lacks some important details for replication of their experiments, so we did not compare their performances in our paper.

Liu et al. [43] developed an autoencoder method for scientific data compression. Their proposed AE model includes three fully connected layers for both encoder and decoder (i.e.,

total of seven layers including the latent vector), and the size of each layer is reduced/increased by $8\times$ compared with its previous layer, thus leading to an overall compression ratio of $512\times$. The limitations of this work are that the autoencoder in their model processes only 1-D data, and the experiments are also based mainly on small-scale 1D scientific data. Our AE-SZ framework overcomes the limitations by designing and training convolutional autoencoders that are aware of dimensional information and are well adapted to large-scale data with relatively high prediction speed and accuracy.

III. Background and Problem Formulation

In this section we describe the research background and formulate the research problem.

A. Research Background – Autoencoder

We describe autoencoder briefly as follows. A stereotype autoencoder model is composed of an encoder network and decoder network. The former encodes the input data to a latent vector in reduced size, and the latter decodes the latent vector to an approximate reconstruction of data. The latent vector stands as a compressed representation of the input data, and different autoencoders have different technical details for computation of the latent vector.

The nature of the autoencoders grants them the potential for being leveraged for data reduction, in that the reconstructed data based on the latent vector can approximate the original data to a certain extent. Figure 1 shows visualization of the reconstructed data versus the original data with the autoencoder [40] (reduction ratio = $64\times$) on a turbulence dataset.

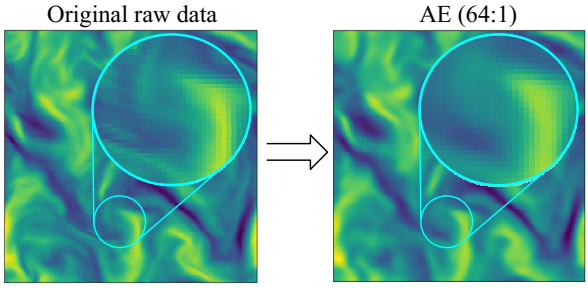


Fig. 1. Reconstructed data of AE ($64\times$) on a turbulence dataset (original value range: $[-3.06$, 2.64], max pointwise absolute error = 1.2)

As a historical and well-researched neural network model, multiple variations have been proposed for the AE model. In what follows, we mainly present SWAE, which is to be used as the fundamental AE model in our designed AE-SZ compressor.

SWAE [44] is a derivation of Wasserstein Autoencoder (WAE) [45], and regularizes the autoencoder loss with the sliced-Wasserstein distance between the distribution of the encoded training samples and a predefined samplable distribution. From [44], marking ϕ as the encoder and ψ as the decoder, given a latent dimension d, a regularization coefficient λ, a number of random projections L, and a predefined latent distribution q_Z, SWAE optimizes the following loss function:

$$\begin{aligned}\mathcal{L}(\phi,\psi) = &\tfrac{1}{M}\textstyle\sum_{m=1}^{M} c\left(x_m, \psi\left(\phi\left(x_m\right)\right)\right) \\ &+\tfrac{\lambda}{LM}\textstyle\sum_{l=1}^{L}\sum_{m=1}^{M} c\left(\theta_l\cdot\tilde{z}_{i[m]}, \theta_l\cdot\phi\left(x_{j[m]}\right)\right)\end{aligned} \tag{1}$$

in which

$\{x_1,\ldots,x_M\}$ is sampled from training set (i.e. p_X),

$\{\tilde{z}_1,\ldots,\tilde{z}_M\}$ is sampled from q_Z,

$\{\theta_1,\ldots,\theta_L\}$ is sampled from S^{d-1} (K-dimensional unit sphere),

$i[m]$ and $j[m]$ are the indices of sorted $\theta_l\cdot\tilde{z}_m$ s and $\theta_l\cdot\phi\left(x_m\right)$, respectively, and

$$c(x,y) = ||x-y||_2^2. \tag{2}$$

Kolouri et al. [44] proved that optimizing this loss function is equal to optimizing

$$\mathrm{argmin}_{\phi,\psi}\, W_c\left(p_X, p_Y\right) + \lambda SW_c\left(p_Z, q_Z\right), \tag{3}$$

in which $W_c\left(p_X, p_Y\right)$ is the Wasserstein distance from p_X (distribution of input data X) to p_Y (distribution of decoded data Y) and $SW_c\left(p_Z, q_Z\right)$ is the sliced-Wasserstein distance from p_Z (distribution of encoded latent Z) to q_Z. Kolouri et al. [44] also show the efficiency of computing Eq. 1.

B. Problem Formulation

1) Leveraging Autoencoders in Error-bounded Scientific Data Compression: The autoencoder itself cannot bound the compression errors, which is a significant gap to scientific user's demand for error controls. As shown in Figure 1, the maximum point-wise compression error is up to 1.2, which is about 20% of the original data value range $(-3.06, 2.64]$. In comparison, scientists often need to control the point-wise errors to a much smaller bound such as 1% of the original value range [9], [15].

In this work we aim to develop a deep learning based error bounded lossy compressor. Specifically, for some scientific applications, we train neural networks based on a certain amount of training data, and then apply the trained networks to compress the testing data generated by the same applications. We separate the training data and testing data because we expect that the pre-trained networks can be used to compress new data for the same applications, such that the training time and model size can be excluded from the compression time and size. In our experiments, the training and test data are from different time steps or the simulation running with different configuration settings in the same application.

2) Math Formulations for Error-bounded Lossy Data Compression: The compression ratio (denoted by ρ) is defined as $\frac{|D|}{|D'|}$, where $|D|$ and $|D'|$ denote the original data size and compressed data size (both in bytes), respectively.

Error-bounded lossy compression has one important constraint, namely, that the reconstructed data respect a user-specified error bound (denoted by e) strictly. Under this constraint, the rate distortion often serves as a criterion to assess the compression quality, which involves two critical

terms: bit rate and data distortion. The bit rate is defined as the average number of bits used to represent one data point after the compression; hence, the lower the bit rate, the higher the compression ratio. In the lossy compression community rate distortion is often evaluated by the PSNR, defined as shown below:

$$PSNR = 20\log_{10}(vrange(D)) - 10\log_{10}(mse(D,D')), \quad (4)$$

where D' is the reconstructed dataset after decompression (i.e., decompressed dataset), vrange(D) represents the value range of the original dataset D (i.e., the difference between its highest value and lowest value), and *mse* refers to mean squared error. The higher the PSNR value is, the smaller the mean squared error, which means higher precision of the decompressed data.

Our objective is to obtain higher compression ratios than other related works obtain (including other deep-learning-based compressor and traditional error-bounded lossy compressors) with the same PSNR value, while also strictly respecting the user's error bound, especially aiming at optimizing the use cases with high compression ratio. We can write the research problem formulations as follows:

$$\begin{aligned} &maximize \ \ \rho \\ &s.t. \quad PSNR(D,D') = \lambda \ , \\ &\quad\ \ |d_i - d_i'| \le e \end{aligned} \quad (5)$$

where λ is a particular PSNR value representing a specific data distortion level and d_i and d_i' refer to any data point in the original dataset D and decompressed dataset D', respectively.

IV. AE-SZ: Autoencoder-based Error-bounded Lossy Compression Framework

In this section we present the design overview of AE-SZ and describe the detailed optimization strategies for AE-SZ.

A. Design Overview of AE-SZ

We present the overall framework of our designed autoencoder-based error-bounded lossy compression framework AE-SZ as shown in Figure 2. The overall compression involves two stages: offline training and online compression.

During the offline training, we split the training data snapshots into multiple small fixed-size blocks (such as 32×32 for a 2D data field or 8×8×8 for a 3D data field) and train the network with numerous small blocks. The advantage of such a design is twofold: (1) the AE model works more efficiently on the divided data blocks, which can catch fine-grained data features; (2) such a data-splitting design creates numerous training samples (i.e., data blocks), so that the AE model is tractable.

During the online compression, AE-SZ executes four steps as shown in Figure 2: (1) splitting the input data to be compressed into many small blocks (with the same block size as during the training stage), (2) prediction, (3) linear-scale quantization, and (4) entropy/dictionary encoding. Specifically, in each block, the data are predicted by a predictor (either autoencoder or Lorenzo), and the prediction errors will be quantized based on the user's error bound, followed by Huffman encoding and Zstd [5]. The Lorenzo predictor is similar to the one used in SZ2.1. Specifically, under the Lorenzo predictor, the ith data point d_i is predicted by three nearby data values in 2D data ($d_{i,j} \leftarrow d_{i,j-1} + d_{i-1,j} - d_{i-1,j-1}$) or by 7 nearby values in 3D data ($d_{i,j,k} \leftarrow d_{i-1,j,k} + d_{i,j-1,k} + d_{i,j,k-1} - d_{i-1,j-1,k} - d_{i-1,j,k-1} - d_{i,j-1,k-1} + d_{i-1,j-1,k-1}$). We refer the readers to read our prior work [9] for more details. We note that the only difference between the Lorenzo predictor in AE-SZ and [9] is that AE-SZ makes the selection between classic Lorenzo and mean-Lorenzo separately on each block instead of a global switching mechanism as in [9]. That is, if a data block can be better predicted by its mean value than by classic Lorenzo, AE-SZ will use the mean value for prediction, and all the involved mean values will be saved losslessly. We find that this mean-Lorenzo predictor can make up for the deficiencies of classic Lorenzo and AE under extremely high error-bounds (such as ∼1E-1).

The compressed data generated by AE-SZ consists of three parts: a header containing metadata (with trivial space cost), lossy compressed latent vectors from autoencoders, and quantization bins (losslessly encoded).

The main difference between AE-SZ and SZ2.1 [15] is that SZ2.1 includes two data predictors for compression: linear regression [15] and Lorenzo [46], whereas AE-SZ replaces the linear regression predictor by a pre-trained autoencoder. For scientific datasets in which the data changes could be diverse, autoencoders can overcome the limitation of linear regression, which can only approximate the data using flat hyperplanes.

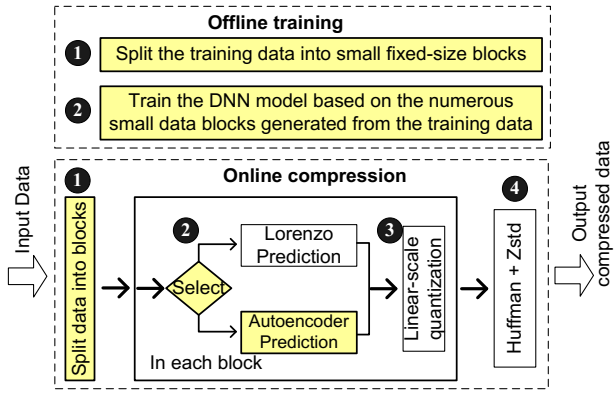


Fig. 2. Design overview of AE-SZ (highlighted parts include our specifically optimized design compared with SZ compressor)

The pseudo code of the AE-SZ compression procedure is presented in Algorithm 1. As mentioned before, AE-SZ compresses the input data block by block, and the compression of each block follows the same routine. Thus, in the following, we describe the compression procedure mainly on a single data block (i.e., line 2∼16), without loss of generality. For any block, AE-SZ first generates predicted data based on two predictors (Lorenzo and autoencoder) for this block respectively (line 3∼8). Then, the predictor with lower element-wise $l1$-loss is selected out for this block (line 9∼13).

The reason is that the smaller the prediction errors are, the more uneven the distribution of quantization bins in general, and hence the higher compression ratio of quantization bins. Then, AE-SZ uses linear-scale quantization to quantize the prediction errors based on the user-specified error bound e (line 14). Similar to SZ2.1 [9], [15], we need to set a maximum number of quantization bins (65,536 by default) for the linear quantization, in order to keep high performance. The total quantization range may not cover all predicted values as the prediction errors may be large. The corresponding data points, called *unpredictable* data, will be saved separately (denoted as U in line 14). For more details about linear-scale quantization, we refer readers to read our paper [9].

Algorithm 1 AE-SZ Compression Algorithm

Input: Input data D, block size S, error-bound e, latent error-bound e_l.
Output: Compressed data D'=$\{\hat{z}, \hat{Z}, U\}$.

1: Split D into blocks of Size S(1D), S×S(2D), or S×S×S(3D).
2: **for** (each block B in the data) **do**
3: $z \leftarrow Eec(B)$. /*Encode B with the encoder network Eec.*/
4: $z' \leftarrow f(z,e_l)$. /*Get decompressed latent vector z' based on e_l*/
5: $B' = Dec(z')$. /*Get Decoded B' using decoder network Dec*/
6: $loss_1 = ||B - B'||_1$. /*Compute $l1$ loss of B' vs. B.*/
7: $B'' = Lorenzo(B)$. /*Predict B with Lorenzo.*/
8: $loss_2 = ||B - B'||_1$. /*Compute $l1$ loss of Lorenzo predictor.*/
9: **if** $loss_2 \leq loss_1$ **then**
10: $B_p = B''$. /*Select Lorenzo-predicted values*/
11: **else**
12: $B_p = B'$. /*Select Autoencoder-predicted values*/
13: **end if**
14: $Q, U = Quantize(B, B_p, e)$. /*linear-scale quantization with e, to get quantization codes Q and unpredictable data U.*/
15: **end for**
16: Compress all saved coefficients from AE and Lorenzo.
17: $H \leftarrow$ Huffman_Encode(Q). /*Huffman encoding*/
18: $\hat{Z} \leftarrow Zstd(H)$. /*Zstd compression*/

In the following subsections, we present several critical optimization strategies for AE-SZ, which are developed in terms of fundamental takeaways we summarized from our in-depth analysis or comprehensive experimental evaluation.

B. Design Detail: AE network structure in AE-SZ

The structure of our designed autoencoder network used in AE-SZ is illustrated in Figure 3. Like most of the autoencoders, it consists of an encoder network to generate the latent vectors as the compressed representation of input original data and a decoder network to reconstruct the data from latent vectors. The input of the network are (batchs of) data blocks, which will be linearly normalized to the range of [-1, 1] based on the global maximum and minimum of data before being put in the network, and the output of the network needs to be denormalized to generate the final prediction values.

The encoder and decoder networks are both formed with several convolutional/deconvolutional blocks and a fully connected layer for resizing latents, and their structures are mirror-symmetric except for an additional final output layer-set in the decoder network.

As shown in Figure 4, the convolutional blocks in encoder network are composed of the layer sequence of Convolution(Stride 1)-Convolution(Stride 2)-GDN, and the ones in

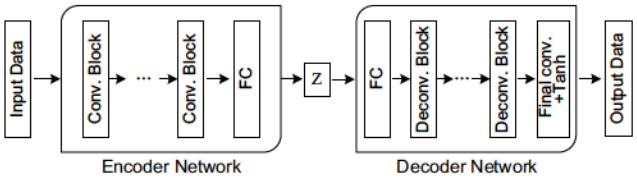


Fig. 3. Our Designed Blockwise Convolutional AE network for Compression

decoder network are composed of the layer sequence of Deconvolution(Stride 1)-Deconvolution(Stride 2)-iGDN. The size of each (de)convolutional kernel in the network is 3×3 (2D case) or 3×3×3 (3D case). We take some experiments for the block design from the image compressive autoencoder in [34] and [35]. The reason we apply stride-1 convolutions before stride-2 convolutions is to increase the number of parameters of the network without fast reducing the size of feature maps. As reported in [33] and [34], consecutively stacking stride-2 convolutions will harm the performance of the network.

In our AE-SZ autoencoder, we do not use traditional activation functions, but use Generalized Divisive Normalization (GDN) [47] as the activation function. In fact, according to Balle et al.'s work [48], GDN can provide better image reconstruction quality with trivial additional parameters compared with traditional activation and normalization functions such has Relu, LeakyRelu [49] and Batch Normalization [50]. Several existing lossy image compression autoencoder models [35], [37]–[39], [51], [52] have leveraged GDN and proved its advantages. Our primary experiments also confirm that GDN outperforms other tested activation functions on scientific data lossy compression task. More details about GDN can be found in [47] and [48]. Following the common configurations, we apply original GDN in convolutional blocks and apply its reverse iGDN in deconvolutional blocks.

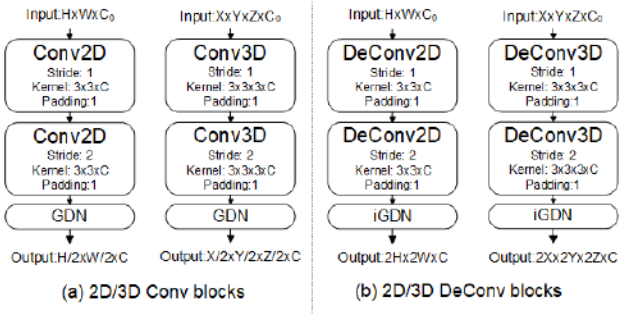


Fig. 4. (a) The Convolutional blocks used in AE-SZ encoder network. (b) The Deconvolutional blocks used in AE-SZ decoder network.

To adapt to different datasets, the number of Convolutional blocks and the number of channels in each block may vary, but the overall structure remains the same. For example, The main difference between autoencoders used for 2D/3D datasets is just the dimension (2D or 3D) of convolutional/deconvolutional operation in the network layers (see Figure 4).

The network model in AE-SZ is saved separately against the compressed data because it can be reused by different time steps or other simulations with different parameter settings, which is verified in our experiments (see Section V).

C. Design Details: Choosing the Autoencoder Type

Takeaway 1: *Sliced-Wasserstein Autoencoder is particularly suitable for data prediction in scientific data compression compared with other AE models.*

A key point in designing AE-SZ is that we need to select the most appropriate model for scientific data prediction from multiple variations of autoencoder models. In AE-SZ, we select sliced-Wasserstein autoencoder (SWAE) for the AE compressor and predictor. The advantages of SWAE in data compression are as follows:

- Compared with the other tested autoencoders, SWAE shows less reconstruction loss on scientific data.
- Different from traditional variational autoencoders (VAEs), the encoding and decoding computation in SWAE are both determinant. VAEs such as [53]–[57] actually compute means and variances with input data and sample latent vectors with the means and variances from the prior distribution. Therefore, in multiple runs with the same input, the latent vector as the output of encoder in a VAE will differ, which makes the VAE being unstable for data compression tasks.
- Compared with Wasserstein autoencoders (WAE), the computation of training loss in SWAE is more numerically efficient. Similar to SWAE, WAE computes Wasserstein distances for training losses, and its computation cost is higher than the computation of sliced-Wasserstein distances. With both n samples from the training set and prior distribution, the computational cost of the Wasserstein distance is $O(n^2)$ whereas the computational cost of the sliced-Wasserstein distance is $O(n \log n)$.

Table I presents the reconstruction quality (PSNR) on different types of autoencoders that we explored. We trained 8 types of autoencoders on a split of snapshots of the CESM-CLDHGH data field: a vanilla autoencoder, vanilla variational autoencoder [53], β-VAE [54], DIP-VAE [55], Info-VAE [56], LogCosh-VAE [57], WAE [45], and SWAE [44]. After training, the AEs were tested by using another split of data snapshots. From this table, we observe that SWAE has the best prediction accuracy with highest PSNR, which motivates us to use it as our final predictor in AE-SZ.

TABLE I
AVERAGE PREDICTION PSNR OF DIFFERENT TYPES OF AUTOENCODERS ON CESM-CLDHGH DATA FIELD

AE type	PSNR	AE type	PSNR
AE	42.2	Info-VAE	26.5
VAE	36.2	LogCosh-VAE	39.0
β-VAE	40.1	WAE	42.4
DIP-VAE	32.2	SWAE	**43.9**

D. Design Detail: Optimizing AE Configurations

Takeaway 2: *The performance of AE may differ a lot with different configurations under the same model structure. Optimizing the AE configurations, especially the input block size and the latent vector size, is critical to the final performance of AE-SZ.*

As presented in Section IV-B, the AE network in AE-SZ has a flexible structure, which can accept various configurations such as different input block sizes, latent vector sizes, block numbers, and channel numbers. From those, the input block size and latent vector size are two critical hyperparameters for the performance, corresponding to the scale of learned data patterns and the representation compactness of latent vectors. We need to optimize the input block size to have the best scope of data, and we need to optimize the latent vector size to balance prediction accuracy and latent overhead.

Table II shows the average prediction PSNR and AE-SZ compression ratio (error bound = 1E-2) of different input block sizes under the same latent ratio (input block size divides latent vector size, 64 for CESM-CLDHGH and 32 for NYX-baryon_density). We conclude that optimizing the input block size is of great importance because autoencoders can achieve apparently different performances under the same latent overhead but various input block sizes. In our work we optimize the input block size of the autoencoder in AE-SZ separately for each field, and we find that 32×32 input block fits most of the 2D data fields tested and that 8×8×8 input block fits most of the 3D data fields tested.

TABLE II
AVERAGE PREDICTION PSNR AND AE-SZ COMPRESSION RATIO (1E-2 ERROR BOUND) OF DIFFERENT INPUT BLOCK SIZES

Blocksize	CESM-CLDHGH		Blocksize	NYX-baryon_density	
	PSNR	CR(1e-2)		PSNR	CR(1e-2)
16×16	42.5	55.5	8×8×8	**46.6**	**71.1**
32×32	**43.9**	**60.9**	16×16×16	35.7	23
64×64	41.7	50.1	32×32×32	28.9	23.9

Table III presents the final compression ratio of AE-SZ under the error bound of 1E-2 with AEs of different latent sizes on the Hurricane-U data field. The input block size is 8×8×8, and the rest part of the network remains the same for different latent sizes. We can see that different latent sizes bring a 40%+ difference in final compression ratios, which motivates us to choose an appropriate latent size in our design. In what follows, we discuss how we reduce the latent vector size while maintaining high prediction accuracy of AEs.

TABLE III
COMPRESSION RATIO OF AE-SZ UNDER THE ERROR BOUND OF 0.01 WITH AES OF DIFFERENT LATENT SIZES ON THE HURRICANE-U DATA FIELD

Latent size	Latent ratio	CR (1E-2)
4	128	123.4
6	85.3	137.4
8	64	**149.1**
12	45.7	127.7
16	32	106

E. Design Detail: Lossy compression of AE latent vectors

Takeaway 3: *Predicting the data with error-bounded lossy decompressed latent vectors can maintain a very small loss of prediction accuracy, while greatly reducing the latent vector size (i.e., latent overhead).*

One main disadvantage of autoencoders is the overhead of storing latent vectors, which can be reduced but cannot be eliminated. To maximize the compression ratio with autoencoders, instead of using the original encoder output latent vectors for compression, AE-SZ compresses the latent vectors with a built-in customized compressor and then uses the decompressed latent vectors for decoding. In this approach, the compressed latents are to be stored. The computation of compressed latents and autoencoder predictions in AE-SZ is shown in Figure 5. For the original latent vector z as the encoder network, a lossy compressor generates the compressed latent z_c in reduced size and the decompressed z_d (which can also be directly computed from z_c); then the decoder network computes the prediction with z_d as its input.

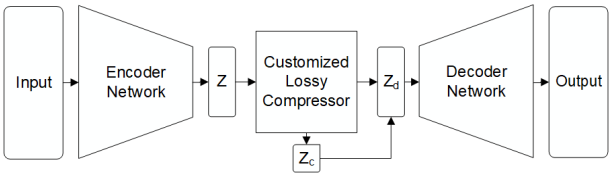


Fig. 5. The autoencoder with customized latent compressor

We customize an efficient method for compressing AE-SZ latent vectors (called *customized* or *custo.* for short) with two steps: (1) quantize the original value using error bound of $0.1e$, where e is the user-specified error bound (the error bounds are value-range based) for the dataset; and (2) use Huffman + Zstd to compress the quantization codes. The advantage of such a design is twofold. First, this can get better compression ratios than SZ2.1, as shown in Table IV. The key reason is that the latent vector data are not quite smooth across adjacent elements, based on our observation, while SZ2.1 strongly relies on the spatial smoothness. Second, the custo. design is consistent with an important constraint required in the AE-SZ: the compression of each data block must be independent of other data blocks, which is explained as follows. Note that we select the better prediction method between AE and Lorenzo based on their prediction accuracy for each block. After this step, all the blocks (either AE-predicted blocks or Lorenzo-predicted blocks) have corresponding predicted data, which can be applied with the quantization directly. Obviously, in order to minimize the latent overhead, we should not store the AE latents for the Lorenzo-predicted blocks, but that requires that the compression of latents be independent across data blocks. SZ2.1 has data dependency across blocks, which makes it unsuitable for the latent vector compression here.

Through masses of experiments using different datasets, we note that choosing a reasonable error bound can achieve a relatively high compression ratio of latents with small loss of prediction accuracy. Figure 6 presents two rate-distortion

TABLE IV
COMPRESSION RATIOS OF OUR CUSTOMIZED COMPRESSOR VS. SZ2.1 ON LATENT VECTORS UNDER DIFFERENT ERROR BOUNDS ϵ

ϵ	RTM		NYX_darkmatterdensity		EXAFEL	
	Custo.	**SZ2.1**	**Custo.**	**SZ2.1**	**Custo.**	**SZ2.1**
1E−2	**6.9**	5.9	**7.1**	6.2	**6.6**	5.7
1E−3	**3.9**	3.4	**4.1**	3.6	**3.6**	2.9
1E−4	**2.5**	2.0	**3.2**	2.5	**1.9**	1.4

plots of the AE prediction values with different compression ratios of latent vectors. The prediction accuracy (w.r.t. PSNR) does not degrade at all when the latent vectors are compressed with a ratio such as 4 (corresponding to bit-rate 0.25 in the figure as the original latent size is $\frac{1}{32}$ of the input size). That is, compressing latent vectors with a relatively high compression ratio (under a certain error bound) does not affect the compression of quantization bins much.

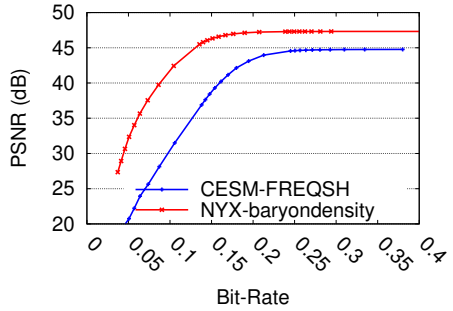


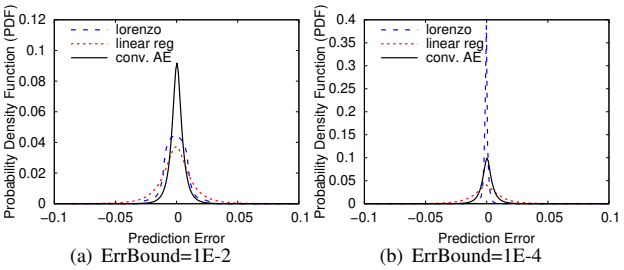
Fig. 6. Rate distortion of SWAE (without quantization)

F. Design Detail: Combination of AE and Lorenzo

Takeaway 4: *The autoencoder model has a high ability to represent the data roughly with a high reduction ratio, but it is not as effective as Lorenzo in high-precision use cases. Therefore, a combination of AE and Lorenzo can effectively mitigate their own particular limitations in data prediction.*

Although AE has a great ability in learning the distribution of data, it still has two critical drawbacks that prevent it from being directly used as a data predictor especially for high-precision error-bounded compression use-cases. The first drawback is that, similar to linear regression, the latent vectors generated by AE for decompression sometimes bring cost due to redundancies of space. Specifically, we observe that quite a few data blocks may have constant or approximately constant values in scientific data. For these blocks, applying a simple and low-cost predictor is accurate enough, while being able to reduce the storage size as much as possible. Second, to maintain the learning effectiveness and efficiencies, the reconstructed data blocks from the autoencoder always suffer from certain noises, making it inadequate for extremely high-precision compression. By comparison, we note that the Lorenzo predictor outperforms the autoencoder especially when a relatively small error bound is used.

We use Figure 7 to illustrate the pros and cons of the autoencoder and Lorenzo predictor under different error bounds.



(a) ErrBound=1E-2 (b) ErrBound=1E-4

Fig. 7. Distribution of prediction errors on CESM-FREQSH data field

This figure demonstrates the prediction error distributions of Lorenzo predictor, linear regression predictor, and our trained autoencoder under an error bound of 1E-2 and 1E-4, respectively (the input data is a snapshot of CESM-FREQSH data field). One can clearly observe that under the large error bound 1E-2, the autoencoder has a better (sharper) prediction error distribution. In contrast, the prediction accuracy of the Lorenzo predictor grows rapidly as the error bound decreases to a small value 1E-4.

During the online compression, AE-SZ selects a predictor between autoencoder and Lorenzo for each data block. The selection criterion is checking which predictor has lower prediction errors (i.e., *loss*) for the given block. The details can be found in Algorithm 1 (see line 6∼13).

V. Performance Evaluation

In this section we present the experimental setup and then discuss the results.

A. Experimental Setup

1) Experiment Environment: We perform the experiments on the gpu_v100_smx2 nodes of the Argonne National Laboratory Joint Laboratory for System Evaluation computation cluster. Each node is driven by two Intel Xeon GOLD 6152 processors with 188 GB of DRAM and NVIDIA TESLA V100 GPUs.

2) Data Used in Experiments: We perform the evaluation using five real-world application datasets in different domains that are commonly used in testing lossy compressors. Most of the datasets such as CESM, NYX, Hurricane can be downloaded from SDRBench [58].

- CESM [59]: A well-known climate simulation package. We use its atmosphere model [58] in our experiments. These datasets are 2D, although some fields exhibit three dimensions in their metadata. For the CLOUD field (26×1800×3600), for instance, SZ2.1 has a better compression ratio (31.1 vs. 22.6) if we compress it with the range-based error bound 1E-3 in 2D mode instead of 3D mode.
- RTM: Reverse time migration (RTM) code for seismic imaging in areas with complex geological structures [60].
- NYX [61]: An adaptive mesh, hydrodynamics code designed to model astrophysical reacting flows on HPC systems. Two separate simulations are performed for generation of training and test data.
- Hurricane [62]: A simulation of a hurricane from the National Center for Atmospheric Research in the United States.
- EXAFEL [63]: An Exascale Computing Project for analyzing molecular structure X-ray diffraction data generated by the LCLS [1]. The data contains groups of 32 2D arrays of size 185×388. We discard the groups with nearly uniform data points; and following [64], we concatenate the 2D arrays in each group to form a single 5920×388 2D array for each group.

More detailed information of the datasets (all in single precision) is shown in Table V. The fields of NYX are transformed to their logarithmic value before compression for better visualization, as suggested by domain scientists.

TABLE V
Basic information about application datasets

App.	# Files and fields	Dimensions	Fields used	Domain
RTM	1 field, 3600 files	449×449×235	snapshot	Seismic Wave
CESM	26 fields,62 files	1800×3600	CLDHGH FREQSH	Weather
EXAFEL	1 field, 352 files	5920×388	raw data	Crystallography
NYX	6 fields, 5 files	512×512×512	bd,t,dmd	Cosmology
Hurricane	13 fields, 48 files	100×500×500	U, QVAPOR	Weather

3) Comparison Lossy Compressors in Our Evaluation: In our experiment, we compare AE-SZ with six other lossy compressors. The first four are classic error-bounded compressors: SZ2.1 [8], [9], [15] and ZFP0.5.5 [13], which have been widely used in the community, and two recent works based on the SZ framework and developed from SZ2.1: SZauto [14] and SZinterp [31]. The fifth one is a recent work of an autoencoder-based scientific data compressor [43], called *AE-A* in our evaluation. The sixth one is a pure convolutional autoencoder model [40], called *AE-B*, proposed for compressing turbulence data, which is not error bounded.

4) Experimental Configurations: For SZ2.1, ZFP0.5.5, SZauto, and SZinterp we adopt value-range-based error bounds and use default configurations for other parameters.

For the training phase of the autoencoders in AE-SZ, we train different autoencoders for different data fields on selected parts of the data, then test and compare all compressors on the remaining parts. Table VI shows the input block size, length of latent vectors, number of the convolutional blocks in encoder network, and number of channels in convolutional blocks of encoder network. The number of deconvolutional blocks is the same as the encoder's, and the channel numbers of the decoder network are symmetric with those in the encoder network. Table VII shows the training-test split for all datasets. All autoencoders in AE-SZ are trained for 100 epochs.

TABLE VI
Autoencoder Configurations for Each Data Field

Data field	Input block	Latent size	Block num.	Channels
CESM-CLDHGH	32x32	16	4	[32,64,128,256]
CESM-FREQSH	32x32	32	4	[32,64,128,256]
EXAFEL	32x32	16	4	[32,64,128,256]
RTM	16x16x16	16	4	[32,64,128,256]
NYX (all fields)	8x8x8	16	3	[32,64,128]
Hurricane-U	8x8x8	8	3	[32,64,128]
Hurricane-QVAPOR	8x8x8	16	3	[32,64,128]

TABLE VII
TRAIN-TEST SPLIT FOR EACH DATASET

Dataset	Train split	Test split
CESM	[0,49]	[50,62]
EXAFEL	[0,299]	[300,351]
RTM	[1400,1499]	1510 to 1600 step 10
NYX	redshift [54,42]	another simulation at redshift 42
Hurricane	[1,40]	[41,48]

For *AE-A* [43], we download their from https://github.com/tobivcu/autoencoder, which supports only double-precision floating data originally. We improve the code by enabling it to compress single-precision floating data, in that most of datasets in our test are stored in single-precision. We trained its model using the same training data split for 100 epochs. The .dvalue files generated by the model are compressed by SZ2.1 following the instruction of [43]. After fine-tuning, we applied the same value-range-based relative error bound to compress the .dvalue file.

For *AE-B*, since Glaws et al. [40] does not provide enough details for training from scratch, following the paper's recommendation, we fine-tuned a pretrained autoencoder (from https://github.com/NREL/AEflow indicated by [40]) on different data fields for 5 epochs each.

5) Evaluation Metrics: We evaluate the seven lossy compressors based on the critical metrics described below.

- *Rate distortion*: Rate distortion is the most commonly used metric by the lossy compression community to assess compression quality. Rate distortion involves and plots with two critical metrics: peak signal-to-noise ratio (PSNR) and bit rate. The definition of PSNR is introduced in section III, and bit rate is defined as the average number of bits used per data point in the compressed data. Generally speaking, Bit rate equals $Sizeof(datatype)/cr$, in which $Sizeof(datatype)$ is the byte size of input data (32 for single-precision data for example), and cr is the compression ratio. Therefore, smaller bit rate means better compression ratio, and vice versa.
- Visualization with the same compression ratio (CR): Compare the visual quality of the reconstructed data based on the same CR.
- Compression speed and decompression speed: $\frac{original\,size}{compression\,time}$ (MB/s) and $\frac{reconstructed\,size}{decompression\,time}$ (MB/s). In the following experimental results, when it comes to error bound values, without loss of generality, we adopt value-range-based error bounds (denoted as ϵ), which takes the same effect with absolute error bound (denoted e) because $e = \epsilon \cdot (\max(D) - \min(D))$.

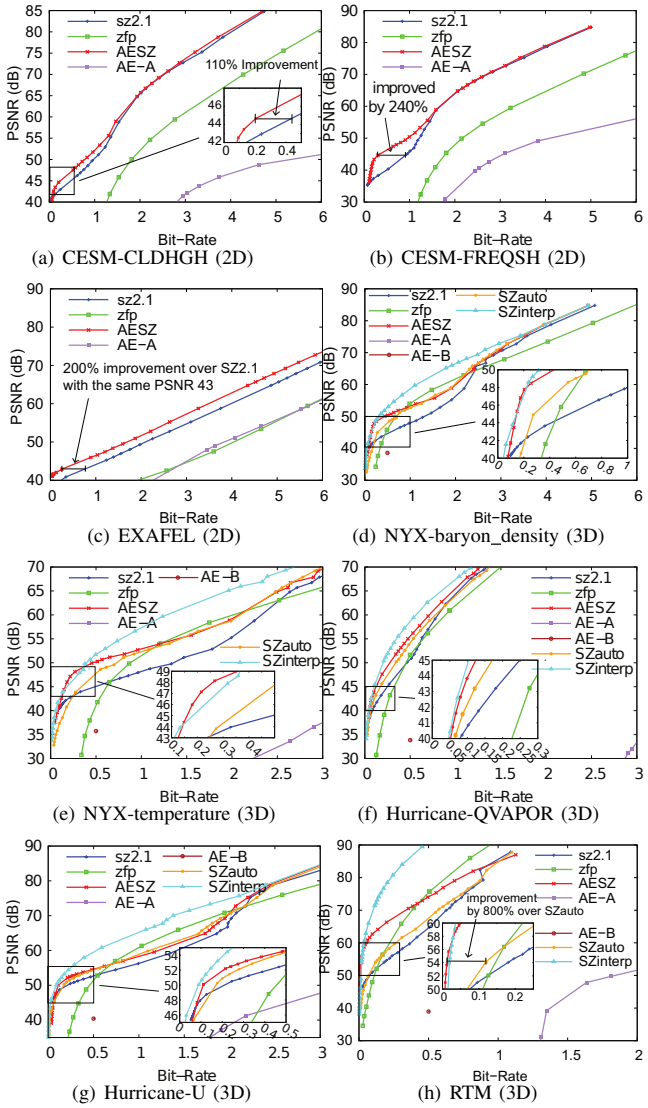
B. Evaluation Results and Analysis

1) Rate distortions of different lossy compressors: We present the rate distortion results of all seven lossy compressors on all tested data fields, illustrating the PSNR of final decompression results with bit rates. Figure 8 shows the rate distortion plots for each lossy compressor on eight data fields. Only four compressors are shown in Figure 8 (a), (b), and (c) because the other three compressors (SZauto, SZinterp, and *AE-B*) support only 3D data, while CESM and EXAFEL are both 2D datasets. We observe that AE-SZ is significantly better than the other two AE-based lossy compressors (*AE-A* and *AE-B*) in term of rate distortion. That is, our developed AE-SZ compression method is arguably the best AE-based lossy compressor to date. We also compare the most competitive error-bounded lossy compressors (to the best of our knowledge): SZinterp [31], SZauto [14], SZ2.1 [15], and ZFP [13]. Generally speaking, AE-SZ obtains much better rate distortions than SZauto, SZ2.1 and ZFP do under low bit rates (i.e., in high-compression-ratio cases) and have a comparable quality with SZ2.1 for high bit rates. We observe that AE-SZ generally has 100%$\sim$800% higher compression ratios than SZ2.1 has in the high-compression-ratio cases on both 2D and 3D datsets. In the 2D datasets, for example, AE-SZ exhibits the best rate distortion (240% higher compression ratio than the second best at the same PSNR around 44) for the CESM-FREQSH data field. On the EXAFEL dataset, AE-SZ has a 200% higher compression ratio than the second best (SZ2.1) in the high-compression cases. On the 3D datasets, AE-SZ also exhibits very competitive rate distortions from among all the seven compressors. Its compression quality is close to that of SZinterp in the low-bit-rate range (e.g., [0,1]) and may also exhibit the best rate distortion in a few cases (e.g., Figure 8 (e)).

2) Decompression data visualizations of different lossy compressors: We present data visualizations in Figure 9 on the NYX-baryon_density field to verify the effectiveness of the reconstructed data of AE-SZ at high compression ratio use cases. We clearly observe that the reconstructed data at the PSNR of 46.8 under AE-SZ has a very good visual quality. Other prior works [15], [65] show that PSNR in the range of [30,60] is good enough to have a high visual quality for different scientific applications. Moreover, Figure 9 demonstrates that with the same compression ratio of 180, AE-SZ has a much better visual quality compared with that of the three state-of-the-art lossy compressors, SZauto, SZ2.1, and ZFP0.5, and is also better than SZinterp.

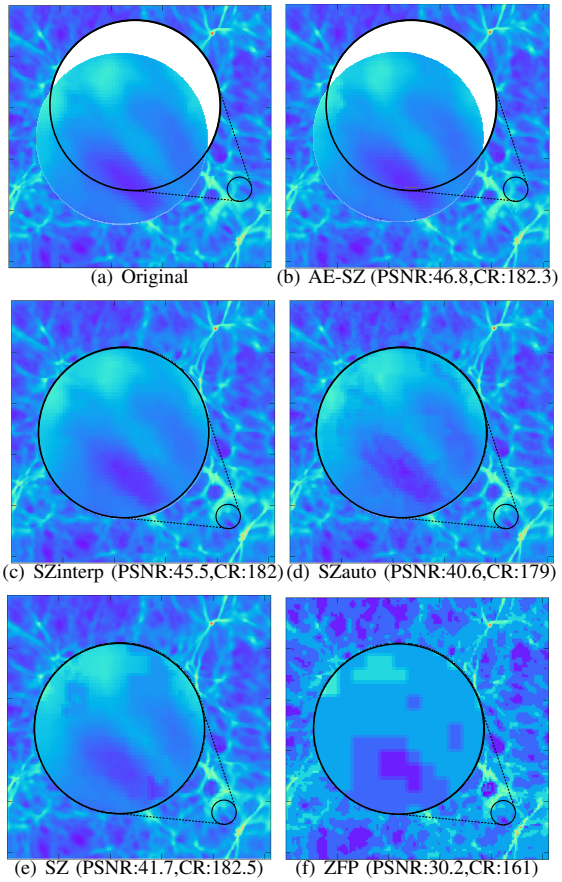
3) Performances of AE-SZ predictors under different error bounds: To better understand how the autoencoder and Lorenzo predictor in AE-SZ cooperatively contribute to the compression ratios, we record the percentage of data blocks predicted by AE-SZ autoencoders on three different data fields, as shown in Figure 10. For better vision of the plots, the x-axis is logged error bounds. The plots show that autoencoders in AE-SZ achieve advantages over Lorenzo under a range of medium error bounds (about 5E-3 to 2E-2), under which most of the data blocked can be better predicted by autoencoders. As the error bound decreases, the Lorenzo predictor becomes better than autoencoders on more data blocks. When the error bound becomes very high, the latents need to be compressed with a high error bound, so the prediction error of autoencoders may drop rapidly, and Lorenzo may also turn better.

To understand the effectiveness of our adaptive prediction design, we present the rate distortion in three situations: predicting data with only AE, predicting data with only Lorenzo,



(a) CESM-CLDHGH (2D) (b) CESM-FREQSH (2D)
(c) EXAFEL (2D) (d) NYX-baryon_density (3D)
(e) NYX-temperature (3D) (f) Hurricane-QVAPOR (3D)
(g) Hurricane-U (3D) (h) RTM (3D)

Fig. 8. Rate distortion of different compressors



(a) Original (b) AE-SZ (PSNR:46.8,CR:182.3)
(c) SZinterp (PSNR:45.5,CR:182) (d) SZauto (PSNR:40.6,CR:179)
(e) SZ (PSNR:41.7,CR:182.5) (f) ZFP (PSNR:30.2,CR:161)

Fig. 9. Visualization of reconstructed data (NYX-baryon_density)

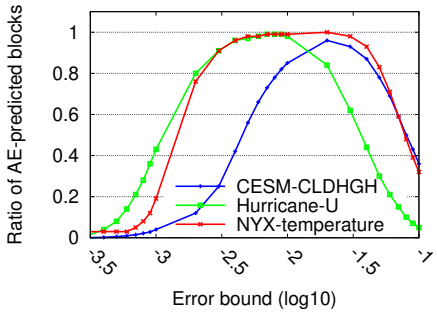
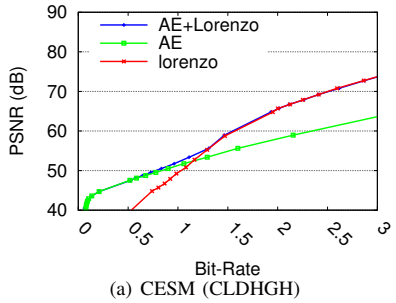


Fig. 10. Percentage of blocks predicted by AE under different error bounds

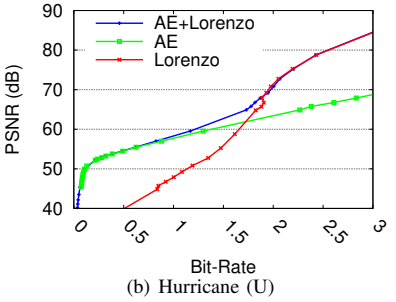
and combining both, as shown in Figure 11. The figure shows that AE+Lorenzo achieves the best quality at all bit rates since it can take advantage of both predictors adaptively.

4) Compression speeds and autoencoder training speeds: The average compression speed of each error-bounded lossy compressor on all the tested datasets under the error bound of 1E-3 are shown in Table VIII in units of Mb/s (SZauto, SZinterp, and *AE-B* have speeds only on 3D data because they currently do not support 2D data). Because of the relatively high computation cost of neural networks, AE-SZ cannot achieve comparable compression throughput with traditional lossy compressors (its speed is about 10%-40% as fast as that of SZ2.1 and SZinterp). In fact, the current version of AE-SZ code is in the experimental stage, so it is not as optimized as the off-the-shelf compressors such as SZ2.1 and ZFP. We believe that with further optimization AE-SZ can be much accelerated. In fact, the throughput of AE-SZ has been significantly better than the other autoencoder-based compressors such as *AE-A* [43] by 30× to 200× (mainly due to the complicated data preprocessing and postprocessing procedures in *AE-A*) and *AE-B* [40] by up to 4× speedup in compression and 9× speedup in decompression (note that *AE-B* is not error bounded so its running speeds include only the AE prediction process).

Second, table IX shows the training time of autoencoders in AE-SZ and *AE-A* [43] using the same training data and the same number of training epochs. We can conclude that the autoencoders in AE-SZ outperforms *AE-A* [43] with similar



(a) CESM (CLDHGH)



(b) Hurricane (U)

Fig. 11. AE-SZ rate distortion comparison of predicting data by AE+Lorenzo, AE only, or Lorenzo only

TABLE VIII
COMPRESSION/DECOMPRESSION SPEEDS (MB/S): ERROR BOUND=1E-3

Type	Dataset	SZ 2.1	ZFP 0.5	SZ auto	SZ interp	AE-SZ	AE-A [43]	AE-B [40]
Comp	CESM	145.0	174.7	N/A	N/A	26.7	0.4	N/A
	RTM	196.4	432.4	231.4	183.2	68.2	0.4	15.6
	Hurricane	166.2	91.8	198.7	168.6	22.0	0.4	15.5
	NYX	142.9	171.7	152.9	99.6	15.7	0.5	16.0
	EXAFEL	162.3	213.7	N/A	N/A	12.2	0.4	N/A
Decomp	CESM	249.7	222.7	N/A	N/A	58.0	0.6	N/A
	RTM	430.3	941.3	516.1	452.6	135.3	0.7	14.1
	Hurricane	264.9	243.1	328.6	357.4	48.2	0.6	14.7
	NYX	217.7	310.4	197.9	115.6	34.6	0.7	14.4
	EXAFEL	219.1	224.7	N/A	N/A	26.1	0.6	N/A

or shorter training time. For *AE-B*, the tested networks are only fine-tuned so we are unable to present its training time.

TABLE IX
AUTOENCODER TRAINING TIME (IN HOURS)

Dataset	AE-SZ	AE-A [43]
CESM	1.0	1.5
RTM	3.4	21.4
NYX	5.5	4.7
Hurricane	2.4	2.5
EXAFEL	2.2	3.5

VI. CONCLUSION AND FUTURE WORK

In this paper we explored leveraging convolutional autoencoders to improve error-bounded lossy compression. To this end, we developed an efficient method called AE-SZ, by integrating autoencoders in the SZ compression model with a series of optimizations. We comprehensively evaluated AE-SZ by comparing it with six related works on five real-world simulation datasets, with the following key findings.

- AE-SZ is competitive in the low-bit-rate range (i.e., high-compression-ratio cases). Specifically, it exhibits the best rate distortion results in 2D datasets. On 3D datasets, it obtains a much better rate distortion than SZ2.1 and ZFP do (about 100%∼800% improvement with the same PSNR). AE-SZ also exhibits very close rate distortions with those of SZinterp in high compression cases, demonstrating its great potential in error-bounded lossy compression.
- AE-SZ has a higher visual quality at the same compression ratio compared with SZauto, SZ2.1, and ZFP.
- AE-SZ is slower than SZ2.1 and ZFP, but is 30×∼200× faster than other autoencoder-based error-bounded lossy compressors.

In the future, we plan to improve AE-SZ in several ways, including (1) optimizing the network structure and the hyperparameters of autoencoders in AE-SZ and (2) speeding up the compression and decompression speeds for AE-SZ.

ACKNOWLEDGMENTS

This research was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations – the Office of Science and the National Nuclear Security Administration, responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering and early testbed platforms, to support the nation's exascale computing imperative. The material was supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357, and supported by the National Science Foundation under Grant OAC-2003709 and OAC-2003624/2042084. We acknowledge the computing resources provided on Bebop (operated by Laboratory Computing Resource Center at Argonne) and on Theta and JLSE (operated by Argonne Leadership Computing Facility).

REFERENCES

[1] SLAC National Accelerator Laboratory, "Linac coherent light source (lcls-ii)," https://lcls.slac.stanford.edu/, 2017, online.
[2] T. E. Fornek, "Advanced Photon Source Upgrade Project preliminary design report," 9 2017.
[3] F. Cappello, S. Di, S. Li, X. Liang, G. M. Ali, D. Tao, C. Yoon Hong, X.-c. Wu, Y. Alexeev, and T. F. Chong, "Use cases of lossy compression for floating-point data in scientific datasets," *International Journal of High Performance Computing Applications (IJHPCA)*, vol. 33, pp. 1201–1220, 2019.
[4] L. P. Deutsch, "GZIP file format specification version 4.3," 1996.
[5] Y. Collet, "Zstandard – real-time data compression algorithm," *http://facebook.github.io/zstd/*, 2015.
[6] Zlib, https://www.zlib.net/, online.
[7] M. Burtscher and P. Ratanaworabhan, "FPC: A high-speed compressor for double-precision floating-point data," *IEEE Transactions on Computers*, vol. 58, no. 1, pp. 18–31, Jan 2009.
[8] S. Di and F. Cappello, "Fast error-bounded lossy HPC data compression with SZ," in *IEEE International Parallel and Distributed Processing Symposium*, 2016, pp. 730–739.
[9] D. Tao, S. Di, Z. Chen, and F. Cappello, "Significantly improving lossy compression for scientific data sets based on multidimensional prediction and error-controlled quantization," in *2017 IEEE International Parallel and Distributed Processing Symposium*. IEEE, 2017, pp. 1129–1139.
[10] A. H. Baker, H. Xu, J. M. Dennis, M. N. Levy, D. Nychka, S. A. Mickelson, J. Edwards, M. Vertenstein, and A. Wegener, "A methodology for evaluating the impact of data compression on climate simulation data," in *Proceedings of the 23rd International Symposium on High-performance Parallel and Distributed Computing*, ser. HPDC '14. NY, USA: ACM, 2014, pp. 203–214.

[11] N. Sasaki, K. Sato, T. Endo, and S. Matsuoka, "Exploration of lossy compression for application-level checkpoint/restart," in *2015 IEEE International Parallel and Distributed Processing Symposium*. IEEE, 2015, pp. 914–922.

[12] A. H. Baker, H. Xu, D. M. Hammerling, S. Li, and J. P. Clyne, "Toward a multi-method approach: Lossy data compression for climate simulation data," in *High Performance Computing*. Springer International Publishing, 2017, pp. 30–42.

[13] P. Lindstrom, "Fixed-rate compressed floating-point arrays," *IEEE transactions on visualization and computer graphics*, vol. 20, no. 12, pp. 2674–2683, 2014.

[14] K. Zhao *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*, ser. HPDC '20, 2020, pp. 89–100.

[15] X. Liang, S. Di, D. Tao, S. Li, S. Li, H. Guo, Z. Chen, and F. Cappello, "Error-controlled lossy compression optimized for high compression ratios of scientific datasets," in *2018 IEEE International Conference on Big Data*. IEEE, 2018.

[16] S. W. Son, Z. Chen, W. Hendrix, A. Agrawal, W.-k. Liao, and A. Choudhary, "Data compression for the exascale computing era-survey," *Supercomputing frontiers and innovations*, vol. 1, no. 2, pp. 76–88, 2014.

[17] P. Lindstrom and M. Isenburg, "Fast and efficient compression of floating-point data," *IEEE transactions on visualization and computer graphics*, vol. 12, no. 5, pp. 1245–1250, 2006.

[18] J. Tian, S. Di, C. Zhang, X. Liang, S. Jin, D. Cheng, D. Tao, and F. Cappello, "Wavesz: A hardware-algorithm co-design of efficient lossy compression for scientific data," in *Proceedings of the 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, ser. PPoPP '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 74–88.

[19] M. Ainsworth, O. Tugluk, B. Whitney, and S. Klasky, "Multilevel techniques for compression and reduction of scientific data—the multivariate case," *SIAM Journal on Scientific Computing*, vol. 41, no. 2, pp. A1278–A1303, 2019.

[20] S. Li, N. Marsaglia, V. Chen, C. M. Sewell, J. P. Clyne, and H. Childs, "Achieving portable performance for wavelet compression using data parallel primitives," in *EGPGV*, 2017, pp. 73–81.

[21] X. Delaunay, A. Courtois, and F. Gouillon, "Evaluation of lossless and lossy algorithms for the compression of scientific datasets in netCDF-4 or HDF5 files," *Geoscientific Model Development*, vol. 12, no. 9, pp. 4099–4113, 2019.

[22] C. S. Zender, "Bit grooming: statistically accurate precision-preserving quantization with compression, evaluated in the netCDF Operators (NCO, v4. 4.8+)," *Geoscientific Model Development*, vol. 9, no. 9, pp. 3199–3211, 2016.

[23] S. Lakshminarasimhan, N. Shah, S. Ethier, S.-H. Ku, C.-S. Chang, S. Klasky, R. Latham, R. Ross, and N. F. Samatova, "ISABELA for effective in situ compression of scientific data," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 4, pp. 524–540, 2013.

[24] X.-C. Wu, S. Di, E. M. Dasgupta, F. Cappello, H. Finkel, Y. Alexeev, and F. T. Chong, "Full-state quantum circuit simulation by using data compression," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '19. New York, NY, USA: Association for Computing Machinery, 2019.

[25] A. M. Gok, S. Di, Y. Alexeev, D. Tao, V. Mironov, X. Liang, and F. Cappello, "PaSTRI: Error-bounded lossy compression for two-electron integrals in quantum chemistry," in *2018 IEEE International Conference on Cluster Computing (CLUSTER)*, Sep. 2018, pp. 1–11.

[26] D. Tao, S. Di, X. Liang, Z. Chen, and F. Cappello, "Fixed-psnr lossy compression for scientific data," in *2018 IEEE International Conference on Cluster Computing (CLUSTER)*, 2018, pp. 314–318.

[27] J. Tian *et al.*, "CuSZ: An efficient gpu-based error-bounded lossy compression framework for scientific data," in *Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques*, ser. PACT '20, 2020, p. 3–15.

[28] D. Tao, S. Di, Z. Chen, and F. Cappello, "In-depth exploration of single-snapshot lossy compression techniques for n-body simulations," in *2017 IEEE International Conference on Big Data (Big Data)*, 2017, pp. 486–493.

[29] S. Jin, S. Di, X. Liang, J. Tian, D. Tao, and F. Cappello, "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*, ser. HPDC '19. New York, NY, USA: ACM, 2019, pp. 159–170.

[30] X. Liang, S. Di, D. Tao, Z. Chen, and F. Cappello, "An efficient transformation scheme for lossy data compression with point-wise relative error bound," in *2018 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2018, pp. 179–189.

[31] K. Zhao, S. Di, M. Dmitriev, T.-L. D. Tonellot, Z. Chen, and F. Cappello, "Optimizing error-bounded lossy compression for scientific data by dynamic spline interpolation," in *37th IEEE International Conference on Data Engineering*, 2021.

[32] D. Tao, S. Di, H. Guo, Z. Chen, and F. Cappello, "Z-checker: A framework for assessing lossy compression of scientific data," *The International Journal of High Performance Computing Applications*, vol. 33, no. 2, pp. 285–303, 2019.

[33] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders," *arXiv preprint arXiv:1703.00395*, 2017.

[34] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Deep convolutional autoencoder-based lossy image compression," in *2018 Picture Coding Symposium (PCS)*. IEEE, 2018, pp. 253–257.

[35] L. Zhou, C. Cai, Y. Gao, S. Su, and J. Wu, "Variational autoencoder for low bit-rate image compression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 2617–2620.

[36] T. Chen, H. Liu, Z. Ma, Q. Shen, X. Cao, and Y. Wang, "Neural image compression via non-local attention optimization and improved context modeling," *arXiv preprint arXiv:1910.06244*, 2019.

[37] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," 2017.

[38] J. Balle, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," *arXiv preprint arXiv:1802.01436*, 2018.

[39] N. Johnston, E. Eban, A. Gordon, and J. Ballé, "Computationally efficient neural image compression," *arXiv preprint arXiv:1912.08771*, 2019.

[40] A. Glaws, R. King, and M. Sprague, "Deep learning for in situ data compression of large turbulent flow simulations," *Physical Review Fluids*, vol. 5, no. 11, p. 114602, 2020.

[41] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[42] J. Choi, Q. Gong, D. Pugmire, S. Klasky, M. Churchill, S.-H. Ku, C. Chang, J. Lee, A. Rangarajan, and S. Ranka, "Neural data compression for physics plasma simulation."

[43] T. Liu, J. Wang, Q. Liu, S. Alibhai, T. Lu, and X. He, "High-ratio lossy compression: Exploring the autoencoder to compress scientific data," *IEEE Transactions on Big Data*, 2021.

[44] S. Kolouri, P. E. Pope, C. E. Martin, and G. K. Rohde, "Sliced Wasserstein auto-encoders," in *International Conference on Learning Representations*, 2018.

[45] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf, "Wasserstein auto-encoders," *arXiv preprint arXiv:1711.01558*, 2017.

[46] L. Ibarria, P. Lindstrom, J. Rossignac, and A. Szymczak, "Out-of-core compression and decompression of large n-dimensional scalar fields," in *Computer Graphics Forum*, vol. 22, no. 3. Wiley Online Library, 2003, pp. 343–348.

[47] J. Balle, V. Laparra, and E. P. Simoncelli, "Density modeling of images using a generalized normalization transformation," 2016.

[48] J. Balle, "Efficient nonlinear transforms for lossy image compression," 2018.

[49] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, no. 1. Citeseer, 2013, p. 3.

[50] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[51] C. Jia, Z. Liu, Y. Wang, S. Ma, and W. Gao, "Layered image compression using scalable auto-encoder," in *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, 2019, pp. 431–436.

[52] L. Xiao, H. Wang, and N. Ling, "Image compression with deeper learned transformer," in *2019 Asia-Pacific Signal and Information Processing*

Association Annual Summit and Conference (APSIPA ASC). IEEE, 2019, pp. 53–57.
[53] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.
[54] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-VAE: Learning basic visual concepts with a constrained variational framework," 2016.
[55] A. Kumar, P. Sattigeri, and A. Balakrishnan, "Variational inference of disentangled latent concepts from unlabeled observations," 2018.
[56] S. Zhao, J. Song, and S. Ermon, "InfoVAE: Information maximizing variational autoencoders," 2018.
[57] P. Chen, G. Chen, and S. Zhang, "Log hyperbolic cosine loss improves variational auto-encoder," 2018.
[58] K. Zhao, S. Di, X. Lian, S. Li, D. Tao, J. Bessac, Z. Chen, and F. Cappello, "SDRBench: Scientific data reduction benchmark for lossy compressors," in *2020 IEEE International Conference on Big Data (Big Data)*, 2020, pp. 2716–2724.
[59] J. E. Kay and et al., "The Community Earth System Model (CESM) large ensemble project: A community resource for studying climate change in the presence of internal climate variability," *Bulletin of the American Meteorological Society*, vol. 96, no. 8, pp. 1333–1349, 2015.
[60] S. Kayum *et al.*, "GeoDRIVE – a high performance computing flexible platform for seismic applications," *First Break*, vol. 38, no. 2, pp. 97–100, 2020.
[61] NYX simulation, https://amrex-astro.github.io/Nyx, 2019, online.
[62] Hurricane ISABEL simulation data, http://vis.computer.org/vis2004contest/data.html, 2004, online.
[63] E. project, https://www.exascaleproject.org/project/exafel-data-analytics-exascale-free-electron-lasers/, 2019, online.
[64] F. Cappello, S. Di, S. Li, X. Liang, A. M. Gok, D. Tao, C. H. Yoon, X.-C. Wu, Y. Alexeev, and F. T. Chong, "Use cases of lossy compression for floating-point data in scientific data sets," *The International Journal of High Performance Computing Applications*, vol. 33, no. 6, pp. 1201–1220, 2019.
[65] X. Liang, S. Di, S. Li, D. Tao, B. Nicolae, Z. Chen, and F. Cappello, "Significantly improving lossy compression quality based on an optimized hybrid prediction model," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2019, pp. 1–26.