**Computing&AI Connect**
ISSN: 3104-4719
2024, Article ID. 2024.0001, Cite as: https://www.doi.org/10.69709/CAIC.2024.193132

SCIFINITI PUBLISHING

Research Article

OPEN ACCESS

# Hierarchical Autoencoder-Based Lossy Compression for Large-Scale High-Resolution Scientific Data

**Hieu Le**[✉,1,*] **Jian Tao**[✉,2]

[1] Electrical and Computer Engineering, Texas A&M University, College Station, Texas, USA
[2] School of Performance, Visualization, and Fine Arts, Texas A&M University, College Station, Texas, USA

**Article History**
Received: April 07, 2024 Accepted: May 15, 2024 Published: June 13, 2024

**Abstract**
Lossy compression has become an essential technique to reduce data size in many domains. This type of compression is especially valuable for large-scale scientific data, whose size ranges up to several petabytes. Although Autoencoder-based models have been successfully leveraged to compress images and videos, such neural networks have not widely gained attention in the scientific data domain. Our work presents a neural network that not only significantly compresses large-scale scientific data, but also maintains high reconstruction quality. The proposed model is tested with scientific benchmark data available publicly and applied to a large-scale high-resolution climate modeling data set. Our model achieves a compression ratio of 140 on several benchmark data sets without compromising the reconstruction quality. 2D simulation data from the High-Resolution Community Earth System Model (CESM) Version 1.3 over 500 years are also being compressed with a compression ratio of 200 while the reconstruction error is negligible for scientific analysis.

**Keywords:**
scientific data; compression; deep learning; neural networks; autoencoder; vector quantization

## 1. Introduction

Over the past few decades, the amount of information available for analysis has increased significantly. Scientific instruments and related computation systems, such as the Linac Coherent Light Source [1], the Very Large Array Radio Telescope [2], and high-resolution climate modeling [3], have produced massive amounts of data, putting a huge burden on the existing storage system. Therefore, it is important to design efficient compression models that are able to reduce the data size for storage while maintaining the key information for analysis.

Data compression can be lossless and lossy. Lossless compression, whose reconstruction is exactly the same as the original data, suffers from a low compression ratio (around 2:1 [4]) in floating point data sets [5,6]. Meanwhile, lossy compression removes imperceptible details to achieve a much higher compression ratio. Despite the loss of information, the quality of the data reconstructed by lossy compression schemes is generally acceptable and usable [7]. The nature of lossy compression has led scientists and engineers to implement many compression algorithms and methods to substantially reduce the size of scientific data [8,9], whose size is often enormous (up to 32 exabytes [10]). Furthermore, recent studies by [11,12], and [13] showed that lossy compression reconstruction data can be used for post hoc analyses.

In recent years, significant attention from scientific and engineering communities has been directed towards the advancement of neural network models across various domains such as computer vision [14], natural language processing [15,16], and compression [17]. Among nu-

* **Corresponding Author:**
Hieu Le, Electrical and Computer Engineering, Texas A&M University, College Station, Texas, USA, hieult@tamu.edu

merous types of deep learning architectures, Autoencoder (AE) has gained tremendous attention because of its capability to learn data representation. AE demonstrates proficiency in unsupervised learning of data representations for reconstruction purposes. Internally, the network contains a bottleneck layer, whose representation is much smaller than its inputs in terms of size. Consequently, AE finds primary application in dimensionality reduction and feature extraction tasks. Numerous AE adaptations have surfaced to enhance the fidelity of reconstructed data [18,19]. While AE has exhibited success in lossy compression of images and videos [20], its utilization in scientific data compression remains relatively under-explored. Despite the limited literature on scientific data compression, AE-based compression methodologies have demonstrated notable compression ratios, surpassing many traditional non-machine learning-based techniques [21].

Inspired by the studies by [21] and [22], our objective is to develop a machine learning approach to substantially reduce the storage footprint of scientific data, known for its substantial space requirements. Thus, we investigate the potential of employing a lossy AE for this task. Our objective is to attain high data reconstruction quality at an ultra-low bit rate, specifically aiming for values below 0.40. We present an AE-based model engineered to significantly reduce the size of the data while preserving the fidelity of the data. The primary contributions of this study are outlined below.

- Targeting a very low bit rate region, we implement our own architecture to significantly compress our simulation data from high-resolution HR-CESM1.3 data sets. Benchmark results are also presented to ensure the performance of our model.
- We incorporate masking layers and several preprocessing techniques to significantly improve the compression performance of our model.

The remainder of this paper is organized as follows. In Section 2, related work is discussed. In Section 3, we describe important concepts and techniques that are implemented in our proposed models. Section 4 provides available resources for this study. Section 5 describes compression experiments on the benchmark data and our large-scale simulation data. We evaluate and analyze our results in Section 6. Section 7 concludes our findings with directions for future work.

## 2. Related Work

Traditional lossy compression for scientific data could be categorized into two types: prediction-based and transform-based. Transform compression (e.g. ZFP [23]) transformed the data before applying other techniques, e.g. embedded coding, to truncate the transformed data. Coefficients and bit-plans determined by the model were used to decompress the data. Increasing the number of coefficients and bit-planes improved the quality of reconstructed data but decreased the compression ratio.

On the other hand, prediction-based models, such as SZ [8,9], and FPZIP [24], predicted the target data using previously reconstructed data points [8,9]. Similar to transform-based models, the authors of [25] found that the fidelity of the reconstructed data degraded when a high compression ratio was required. Prediction-based models have been shown to have high reconstruction quality at a high compression ratio, which has led to more studies to improve the performance of this type of compression.

Recently, deep learning models have been leveraged to compress many types of data. Many AE-based models showed remarkable results in image and volumetric compression tasks.

Balle et al. [26] introduced an effective end-to-end AE-based model to compress images. The authors trained their models to optimize the performance of rate-distortion. To balance the trade-off between the quality of reconstructed data and compression ratio, both the losses for reconstruction and compression rate were minimized simultaneously. Since the quantization layer of their compression models prevented the gradients from flowing through the networks, independently and identically distributed uniform noise was used to replace the quantization layer during training. The added noise allowed backpropagation without significantly deteriorating the performance of the quantization layer when compressing images.

Models with two levels of quantization were also investigated in [27]. The second layer not only provided fine-grained quantization but also acted as a prior for the first quantization layer. In addition, arithmetic encoding [28,29] was implemented instead of variants of Huffman encoding [30]. Integer quantization, proposed by the authors of [31], was applied for quantization layers to eliminate the dependence on hardware-platform floating-point implementation, which varied from machine to machine, during compression.

Using the idea of two-level quantization, several studies have been conducted to improve the capabilities of neural networks to compress images. Minnen et al. [32] built an autoregressive model. The first quantization layer, which received input from the prior given by the second quantization and from the encoder, autoregressively processed data representation to produce high-quality images. Their neural networks were also among the first machine-learning models that outperformed the state-of-

2024, Vol. 1, Article ID. 2024.0001
https://www.doi.org/10.69709/CAIC.2024.193132

the-art BPG compression [33]. However, autoregression by its nature prevented neural networks from computing in parallel. The models created by [20] eliminated the autoregressive layer and replaced it with multiple splitting layers, allowing the decoder to fully learn different sets of channels in parallel. Furthermore, optimization for compression speed using neural networks was also addressed by [17], which suggested several methods to improve compression performance.

Compression on audio signals using AE-based neural networks has also experienced much progress. The work of [34] outperformed MP3 in both compression ratio and audio quality. Their models adopted the vector quantization techniques proposed by [35]. The authors not only optimized signal losses in the time domain, but also minimized reconstruction losses in the frequency domain. Furthermore, the coupling of AE and Generative Adversarial Networks (GAN) [36] was leveraged to achieve a high-quality compression model.

Neural networks have also been implemented to compress volumetric scene data. Kim et al. [34] replaced fine-grained compression layers in their tree-based models with neural networks, greatly enhancing the performance on volumetric representation. Coordinate networks by [37] not only focused on learning the scene representation but also provided great compression capability.

However, image and video compression models mainly reconstructed integer pixels (or voxels), which were only a subset of scientific data, where data types ranged from integer to floating-point. As a result, several studies have been conducted using neural networks to enhance scientific data compression. Glaws et al. [38] proposed an AE model, which was built upon 12 residual blocks of convolution layers. The authors incorporated three compression layers to reduce the data dimensions in their AE's architecture. The model was trained to compress turbulence data at a fixed compression ratio of 64.

Liu et al. [22] introduced a seven-layer AE model to compress scientific data. The encoder consisted of three fully connected layers, each of which compressed input data by eight folds. Theoretically, the encoder could compress data by 512x ($8^3$). Similar to the encoder, the decoder had three fully connected layers to decompress the compressed data. A bottleneck Between the encoder and decoder contained latent variables much smaller in size than the inputs. However, this work mainly focused on small-scale 1D data, whereas our models learned data representation in higher dimensions, particularly in 2D and 3D. Another limitation of this model was that it only utilized CPUS for compression, failing to fully utilize the parallel computing power offered by GPUs [39].

Recently, a compression method proposed by Liu et al. [21] achieved great results for 2D and 3D data. Their AE-SZ framework consisted of a Lorenzo prediction model and an AE model, each of which compressed the data independently. The compression results from both models were then compared for the framework to select the model for the data being compressed. The compression ratio of their proposed framework on many scientific data sets surpassed results from other hand-engineered models and AE-based models. However, instead of optimizing one particular model for each input, the framework employed two distinct models to compress the same data.

Slightly different from traditional deep learning models, physics-informed neural networks (PINNs) [40] have been successfully developed to extrapolate data and solve many scientific problems. Choi et al. [41] combined PINN and variational autoencoder (VAE) [18] to compress plasma simulation data. Unlike other types of neural networks, this PINN model optimized several physics constraints, such as mass, moment, and energy, along with the reconstruction loss, i.e., L2 distance. Similar to our work, the authors used integer quantized latent variables, which could be reliably transmitted between different hardware and software platforms as studied by [31].

AE-based structures are applied in various scientific investigations. The authors of [42] utilize AE to capture the latent representation of the input data. Following this, they employ long-short-term-memory (LSTM) models for forecasting sequential data across subsequent time intervals. This strategy reduces computational demands by minimizing the need for full dynamic simulations of drop interactions within microfluidics devices. Similarly, the authors of [43] present a methodology for predicting wildfire occurrences in future years based on input data from the previous year. Leveraging an AE model, input data undergo encoding via the AE's encoder, mapping them to their latent representation. This latent representation is then fed into a recurrent network network (RNN) model to obtain a representation of the subsequent time step, subsequently which is decoded to derive predictions for that step. Both studies employ Latent Assimilation (LA) techniques to improve prediction accuracy. Additionally, the work of [44] adopts the concept of sequence prediction using RNN within the latent space generated by an AE model. Furthermore, they present Variational Generalized LA to refine the accuracy of sequential prediction within the RRN-AE pipeline. These investigations indicate the potential of utilizing the latent representation to forecast data across subsequent time intervals, presenting an intriguing avenue for our future exploration.

2024, Vol. 1, Article ID. 2024.0001
https://www.doi.org/10.69709/CAIC.2024.193132



# 3. Methods

Our proposed model is built on three main components: an encoder network (**E**), a quantizer (**Q**), and a decoder network (**D**). The encoder network encodes data inputs to output latent variables $z_e$. The quantizer then processes $z_e$ to produce a quantized latent representation $z_q$. Finally, the decoder network reconstructs data from the compressed representation $z_q$ to output $\hat{x}$. The whole model is trained in an end-to-end fashion to minimize a reconstruction loss and constraint losses imposed by the codebook of the quantizer. The model architecture is depicted in Figure 1.

The detailed implementation of the model is given in Table 1. Each stage (EncRes) of the Encoder is connected to an intermediate convolution layer. The intermediate layer acts as a bridge to map the number of channels to the desired vector dimension of the quantization layer. The output representation is then quantized using the corresponding codebook.

**Table 1:** The implementation of the model architecture.

| Network | Stage | Operator | Stride | #Channels | #Layers |
|---|---|---|---|---|---|
| Encoder | Norm | N/A | N/A | N/A | 1 |
| | PreBlock | Conv4x4 | 1 | 32 | 2 |
| | EncRes_0 | EncRes | 2 | 64 | 3 |
| | EncRes_1 | EncRes | 2 | 128 | 3 |
| Decoder | DecRes_1 | DecRes | 2 | 64 | 3 |
| | DecRes_0 | DecRes | 2 | 32 | 3 |
| | PostBlock | Conv4x4 | 1 | 1 | 2 |
| | De-norm | N/A | N/A | N/A | 1 |
| Quantizer | VQ_0 | Quantization | N/A | N/A | 1 |
| | VQ_1 | Quantization | N/A | N/A | 1 |

## 3.1. Encoder & Decoder Architecture

As mentioned above, the encoder is trained to extract data representation into latent spaces, whereas the decoder decodes the latent variables to reconstruct the given data. There are two most widely used reconstruction errors, mean-squared error (MSE) and multi-scale structural similarity (MS-SSIM). Depending on targeting criteria, either measure can be used to achieve desirable outcomes. Both measures have been shown to be effective metrics as they typically result in high-quality generated images [20,26,27].

The encoder network **E** is created hierarchically. The first level aggressively reduces the dimensions of the inputs and learns their data representation. The second level also performs slight dimension reduction. Data representation from the second level is quantized by its corresponding vector quantizer. These quantized values are then fed into the first-level vector quantizer. The second-level quantization acts as a prior to the first-level quantization. The additional information from the second-level quantization improves the capability of the first-level quantization, which leads to a better reconstruction quality. Although the second level creates slightly more bits during compression, the improvements in reconstruction quality significantly outweigh a slight decrease in compression ratio.

The network **E** comprises several 2D convolution layers and blocks of residual connections. The first two convolution layers map inputs to a higher number of channels using a kernel size of 4. It is followed by a couple of residual blocks, which consist of strided convolutions with a kernel size of 5. Components of a residual block are illustrated in Figure 2. The construction of residual blocks is inspired by the model proposed by [14], which achieves high performance for classification tasks without significantly increasing the size of the model. Moreover, residual blocks alleviate the vanishing gradient problem and enable the implementation of deeper models, which improves the expressiveness of our networks [45]. We

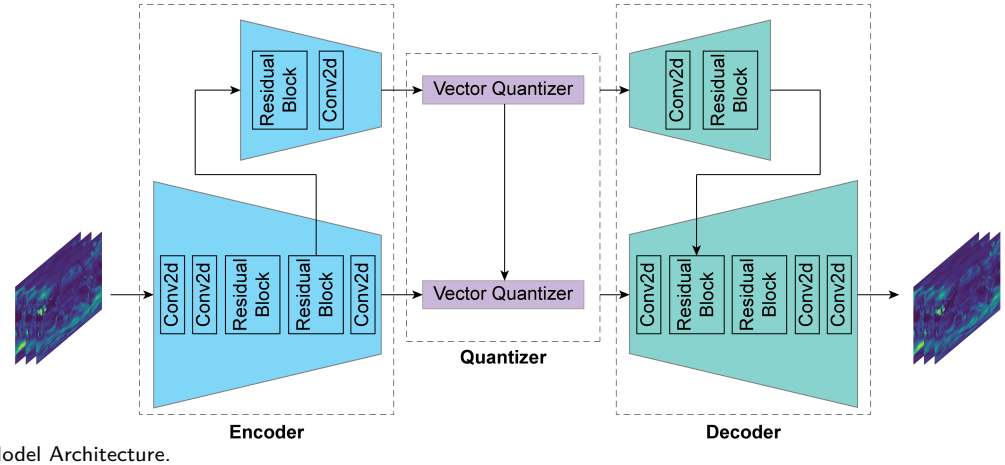


**Figure 1:** Model Architecture.

2024, Vol. 1, Article ID. 2024.0001
https://www.doi.org/10.69709/CAIC.2024.193132

use non-linear GELU functions as our activation functions [46]. A generalized divisive normalization (GDN) is used to normalize the outputs of the residual block and transform their distribution to be more Gaussian [47]. GDN has been shown to be effective for both image compression [48] and scientific data compression [21]. The encoder network can be simply represented as a mapping function, as shown in (1).

$$z_e = \mathbf{E}(x), \tag{1}$$

where **E** is the encoder network.

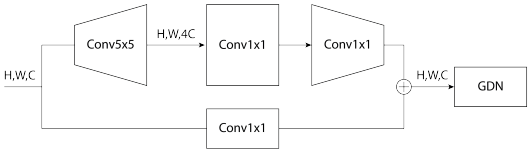


**Figure 2:** Components of a residual block.

The decoder network **D** is a mirror of the encoder network **E**. Transposed convolution layers are used to replace strided convolutions. At the beginning of each hierarchy, transposed convolutions alter the input of the decoder to acquire a suitable representation with C channels for the following residual blocks. In general, all blocks of the decoder loosely reverse the operations performed by the encoder. Network **D** maps the latent representation back to the original dimension, outputting reconstructed data. The decoder can also be considered to be a mapping function as shown in (2).

$$\hat{x} = \mathbf{D}(\text{decoder\_inputs}), \tag{2}$$

where **D** is the encoder network.

## 3.2. Vector Quantizer

Although vanilla AE can perform dimension reduction, it cannot flexibly generate data given fixed inputs. Variational Autoencoder (VAE) [18] and its variants are implemented to improve reconstruction performance. VAEs not only minimize the reconstruction loss but also learn the distribution of the latent variable by optimizing the Kullback–Leibler (KL) divergence. As a result, a more diverse set of images can be generated with much higher quality [19,49].

Based on the idea of VAE, we impose slightly different criteria on the objective function. Following a proposed approach implemented in Vector Quantized Variational Autoencoder (VQ-VAE) [50], our model is trained to minimize the reconstructed loss, i.e., L2 distance, as well as optimize discrete codebooks. The latent representation encoded by the encoder is projected onto codebook vectors. The vector with the smallest Euclidean distance to the encoded latent variable, is selected to become the decoder's input as shown in (3).

$$z_q = \mathbf{Q}(z_e) = \arg\min_{q_k \in \mathbf{Q}}(||z_e - q_k||), \tag{3}$$

where $q_k$ is a vector in codebook **Q** that has the smallest Euclidean distance with the output of the decoder, $z_e$.

The quantizer outputs a set of integer values, which are indices of quantized vectors. These indices are then further compressed using a lossless compression scheme, i.e. Huffman coding-based algorithms. The size of compressed quantized data is significantly reduced because quantized values are in the form of integers, which are efficiently compressed by any lossless compression algorithm.

Our training procedure for our codebooks is similar to the method described in [50]. Each codebook in the model is updated using an exponential moving average with a decay of 0.99. The update is measured based on changes in codebook entries after each training iteration. A straight-through estimator [51] is implemented to overcome the discontinuity of gradients created by discrete codebooks. The estimator acts as an identity function that identically maps the gradients of the decoder to the encoder in the backward propagation.

## 3.3. Preprocessing Large-Scale Data

### 3.3.1. Data Standardization

In this work, we focus on compressing large-scale high-resolution scientific data obtained from Earth's simulation. Since each set of data has its own data distribution, it is important to preprocess raw data prior to training. Statistical measures of data can be analyzed based on each specific data type. The availability of statistics enables us to use Gaussian standardization for data whose distribution is Gaussian. The technique is also applicable to distribution that approaches the Gaussian distribution. The standardization method is shown in (4).

$$x_{st} = \frac{x - \mu}{\sigma}, \tag{4}$$

where $x$ is a data value, $\mu$ is the mean of the data, and $\sigma$ is the data standard deviation.

The inverse of standardization is required for converting the reconstructed data back to the actual value range. The inverse is formulated in (5).

$$x = \mu + x_{st} * \sigma \tag{5}$$

However, if the data distribution is not Gaussian, directly applying standardization does not improve compression performance. In this scenario, logarithm scaling is a technique used to transform the original data to its corresponding logarithmic scale. This technique usually transforms the data distribution to be close to Gaussian, which enables us to effectively use the standardization method on the data.

#### 3.3.2. Missing Value Handling

Data masking is necessary for data compression in many cases. In many scientific simulations, there are regions not of interest to the researchers conducting experiments. Those areas are generally assigned values that are extremely negative or easily distinguished from actual simulation values. Therefore, we use masking layers to indicate valuable values and ignore unwanted regions in our model. Although masking increases the storage size, this redundancy is negligible as it consists of several integer values, which can be significantly compressed by any standard lossless compression algorithm such as Huffman coding-based compression schemes.

Missing values in data are also replaced by a different value. The replacement values can be the mean or the median of the entire available data. For simplicity, we assign missing values with the data mean since data statistics are readily available. After cleansing missing values and masking the data, the data and their corresponding masks are partitioned into small blocks.

#### 3.3.3. Data Partitioning

Machine learning models generally cannot handle raw scientific data because each dimension of any data is large and cannot fit into the system's memory. To address this issue, data are partitioned into small blocks before training or compression. Each dimension of a block is a power of two. Specifically, we restrict the block to having a height and width of 64 for the training process, as we observe that this setting achieves the best reconstruction quality. Additionally, a power of two in each block dimension makes up-sampling and down-sampling efficient. No padding or trimming is required for the outputs, saving additional computing power.

However, the shapes of raw data are not always divisible by two, which is a requirement to have a block size as a power of 2. In such cases, data whose size is not a multiple of the block size are padded. Padding is performed at the edges of each dimension. For Earth's simulation data, we cyclically replicate data values at one edge and concatenate them at the other end. For example, to pad the left edge of 2D data, values on the right edge are copied and appended to the opposite side. This padding pattern is especially helpful for treating continuous simulation data with periodical boundary conditions, e.g., climate modeling data.

The partitioning technique mentioned above works well in general. However, since all partitioned blocks are discrete, the whole set of partitions does not include any transition from one block to its adjacent neighbors. To smooth out the boundary and make the transition from one block to another more accurate, an overlapping block partition technique is implemented [34]. Instead of creating mutually exclusive blocks of data, adjacent blocks are partitioned in a way that they overlap with each other in a small area. Specifically, assuming each block is of size 64 and there is an overlap of eight, the second block is created to contain the last eight values of the first block as well as the next 56 values. The data overlapping technique is implemented only during data training, while the discrete data partitioning technique without overlapping is used for testing and compression.

### 3.4. Objective Function

#### 3.4.1. Reconstruction Loss

The reconstruction loss is the discrepancy between the reconstructed and original data. We minimize the L2 distance of the target and compressed data, i.e. $l_{recon}(x, \hat{x}) = ||x - \hat{x}||_2$. The minimization simply matches the compressed data to the original data as closely as possible.

#### 3.4.2. VQ Commitment Loss

The commitment loss accounts for the difference between the quantized codebook vectors and outputs of the encoder. Since quantization distorts the data, decreasing the distance between the quantized vectors and the original data reduces the distortion. We impose an L2 distance constraint on the codebook vectors and their corresponding inputs. The commitment loss, $l_q$, is defined as in Equation (6).

$$l_q(z_e, z_q) = ||z_e - z_q||_2 = ||z_e - \mathbf{Q}(z_e)||_2, \tag{6}$$

where $\mathbf{Q}$ is the quantizer; $z_e$ and $z_q$ are outputs of the en-

2024, Vol. 1, Article ID. 2024.0001
https://www.doi.org/10.69709/CAIC.2024.193132

coder and their corresponding quantization values, respectively.

Overall, the model is trained to optimize the following objective.

$$L = \lambda_{recon} * mask * l_{recon} + \lambda_q * l_q, \quad (7)$$

where $mask$ is a masking layer, that indicates which data points should be taken into account in optimization; $\lambda_{recon}$ and $\lambda_q$ are constant coefficients of the reconstruction and commitment losses, respectively. The constant $\lambda_q$ is set to be 0.25 following the suggestion by [35]. Meanwhile, $\lambda_{recon}$ is set to 2 because the parameter slightly affects the trade-off between reconstruction quality and compression ratio. The objective function in Equation (7) is acquired based on the assumption that quantization values are uniformly distributed. Uniform distribution leads to the removal of an additional KL term in the objective because the term becomes a constant with respect to encoder parameters [35].

### 3.5. Error-Bounded Technique

Reconstructed data from neural networks sometimes have large distortions from the original data. To counteract the large distortion of some reconstructed values, a straight-through technique is introduced. The straight-through technique classifies reconstructed values into two groups, predictable and unpredictable. Reconstructed data that meet the tolerance constraints are called predictable values. In other words, predictable data have error values less than or equal to a predefined threshold. Otherwise, they are unpredictable values. Unlike predictable values, which can be used directly as final reconstructed values, unpredictable values have errors that exceed the threshold. Thus, corresponding true values and their locations are saved separately in a file to replace unpredictable values during reconstruction.

## 4. Availability of Data and Materials

This paper uses existing, publicly available data from SDRBench (https://sdrbench.github.io/) for benchmarking the performance of our model. Regarding compression on our application data, the model compresses the High-Resolution Earth System Prediction (iHESP) data. The iHESP data have been deposited at https://ihesp.github.io/archive/ and are publicly available. All original code has been deposited on GitHub at https://github.com/hieutrungle/data-slim and is publicly available as of the date of publication. Any additional information required to reanalyze the data reported in this paper is available from the lead contact upon request.

## 5. Experiments

### 5.1. Benchmark Data: SDRBench

Our proposed models are initially tested on published scientific benchmark data, SDRBench [52]. This benchmark provides numerous simulation data from many different fields, ranging from electronic structures of atoms, molecules to weather, and cosmology. The benchmark is publicly available for different scientific purposes.

Even though we focus on compression 2D data, a couple of 3D data sets are also being compressed to verify the possibility of generalizing our architecture to higher-dimension data. Table 2 summarizes several data sets and some fields we use for our compression.

The 3D CESM data include comprehensive attributes of cloud properties for many different altitudes, which can be viewed as many 2D data stacking on top of each other. Therefore, we use the 3D CESM data as a training set for CESM cloud data, whereas all snapshots of 2D CESM data are our testing data.

**Table 2:** Basic information of benchmark data from SDRBench.

| Dataset | Dimension | Domain | Field |
|---|---|---|---|
| CESM 2D | 1800x3600 | Weather | CLDHGH<br>CLDMED<br>CLDLOW<br>CLDTOT |
| CESM 3D | 26x1800x3600 | Weather | CLOUD |
| NYX 3D | 512x512x512 | Cosmology | Temperature<br>Baryon density |

### 5.2. High-Resolution Earth System Prediction (iHESP) Data

The International Laboratory for High-Resolution Earth System Prediction (iHESP) [3] was a project aimed at developing more advanced modeling frameworks for predicting high-resolution multiscale Earth systems to improve simulation and prediction of future changes in extreme events. iHESP also provides numerous global and regional high-resolution simulation data spanning hundreds of years. The global climate was simulated using different high-resolution configurations of CESM version 1.3 for atmosphere, land, ocean, and sea-ice. Meanwhile, regional data were generated from the ocean model ROMS (Regional Ocean Modelling System) with the atmospheric model WRF (Weather Research and Forecast

model) using the CESM/CIME coupling infrastructure. All data are also publicly accessible.

Among a large array of ocean properties provided by iHESP, sea surface temperature (SST) is one of the most important attributes. The property is simulated over hundreds of years, necessitating a substantial amount of storage for the data. However, the large amount of available data also enables us to leverage machine learning for compression.

Basic information about SST data is presented in Table 3. The first dimension of the data represents the evolution in time. The next two dimensions are the height and width of the data, respectively. General ocean information, such as simulation history and climate coefficients, is also included in the dataset metadata. Latitudes and longitudes are also available to scale the data back to the global coordinate system when it is required.

**Table 3:** Basic information of SST data.

| Data Size | Dimension | Lowest Value | Highest Value |
|---|---|---|---|
| 111.90 GB | 3240x1800x3600 | -1.95 °C | 34.84 °C |

Data preprocessing is crucial for SST in both training and compression. Temperature values are only available where sea water is present, whereas undefined values are assigned to continents. In order to deal with missing values, a masking layer is created to differentiate between these regions.

The data are divided into two sets: a training set and a testing set. The training set contains nearly ∼100GB of SST data while the testing set consists of temperature data of the last 120 consecutive months in the simulation. Data in the training set are partitioned using the overlapping technique, while the discrete partitioning technique is applied to the testing set. Both training and testing sets contain blocks of data of size 64. During compression, data are partitioned into blocks of size 256 for better resolution.

# 6. Results and Discussion

## 6.1. Compression of Benchmark Data

### 6.1.1. 2D Data

The compression performance of our models on different data sets is compared to other compression models, namely SZ2.1 [53], ZFP [23] and AESZ [21]. Figure 3 shows that our proposed model outperforms other compression schemes when bit-rates are below 0.40, which are equivalent to compression ratios of greater than 80. At a very low bit-rate of 0.22, the reconstructed data of our model has a PSNR of 46.35 dB. This is an improvement over the hybrid AESZ model, which requires a bit-rate of around 0.37 to obtain the same PSNR (Table 4).

**Table 4:** Compression efficiency evaluated on CESM 2D CLDHGH data with a target PSNR of approximately 46 dB.

| Method | Bit-Rate | PSNR |
|---|---|---|
| Ours | 0.22 | 46.35 |
| AESZ | 0.37 | 46.03 |
| SZ2.1 | 0.61 | 46.22 |
| zfp | 1.54 | 45.84 |

However, the PSNR of the proposed model does not follow the same trend as the compression performance of other compression models. Our trained model has a fixed set of parameters. To increase PSNR without training a different model, we apply the straight-through method to restrict the error-bound of the reconstructed data. There is a possibility of training different models with larger latent variables and codebooks to achieve much higher PSNR at any given bit rate. However, with the goal of targeting the low bit-rate regime, exhaustively exploring all possible combinations of neural networks over a wide range of bit-rates is not in the scope of this work.

Our compression model is also used to compress different 2D data. The compression performance on many different cloud data is illustrated in Figure 4. Since the CESM 3D CLOUD data should be treated as 2D data as suggested by domain scientists [52], its compression results are presented together with other 2D data. It is worth mentioning that compression on all CESM cloud data uses the same model architecture with the exact same weights. Even when applying the model to these data, it obtains high PSNR while maintaining a very low bit rate. The compression performance indicates that this particular model for CESM cloud data achieves a good generalization.

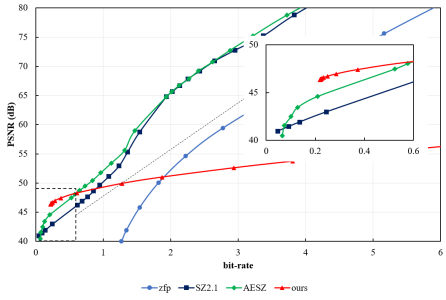


**Figure 3:** Compression performance on CESM 2D CLDHGH data.

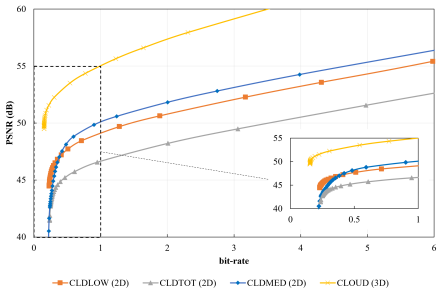


**Figure 4:** Compression performance of the proposed model on different CESM cloud data.

### 6.1.2. 3D Data

The proposed model achieves reasonable compression on 3D benchmark data. As can be seen in Figure 5, at low bit-rates, our model surpasses SZ2.1 and ZFP in terms of performance. However, the quality of reconstruction from the hybrid model - AESZ - is higher than our model. One of many possible reasons for the weaker performance of our compressor is that our model is designed using primarily 2D convolution layers. Therefore, it does not have the extensive capability to learn data representation in 3D. On the other hand, when compressing 3D data, AESZ changes its machine learning architecture to 3D convolution neural networks. This change is one of the factors that increase compression performance for volumetric data.

## 6.2. Compression of iHESP Sea Surface Temperature (SST) Data

The compression results for the testing dataset of high-resolution SST data show that the model can reconstruct the data with high quality while maintaining a high compression ratio even for large-scale simulation data. As shown in Figure 6, after being compressed by a factor of 240, the reconstruction achieves a PSNR of 50.16. Moreover, in terms of visualization, it is unlikely that differences between the original and reconstructed data are detected. However, there are some slightly noticeable distortion areas, especially along coastal lines between oceans and continents. Since data are only available for sea water, data points on continents are set to be a suitable constant. The assignment of the constant creates large variations in values along the edges of continents, which hinders the reconstruction ability of the model in those particular regions.

Table 5 presents the compression performance of the model on the testing data. The quality of reconstruction, PSNR, of each snapshot, varies from 48.58 to 51.5. The reason for the differences in PSNR is that the data distribution of each snapshot differs from time to time, which leads to a variation in the quantization values from codebooks; hence changes in the reconstruction quality. Nevertheless, the deviation of the snapshots' PSNR does not vary much from the average of 50.04, indicating that our model achieves stable performance across all data sequences.

**Table 5:** Compression performance of the proposed model on testing SST data (data size: 4,144.60 MB).

| Metrics | Results |
|---|---|
| Compression ratio | 231.54 |
| PSNR (dB) | 50.04 |
| Compression speed (MB/s) | 96.53 |
| Decompression speed (MB/s) | 87.43 |

Compression and decompression speeds are also acceptable. The compression speeds on HPC nodes are presented in Table 5. On average, it takes around 45 sec-

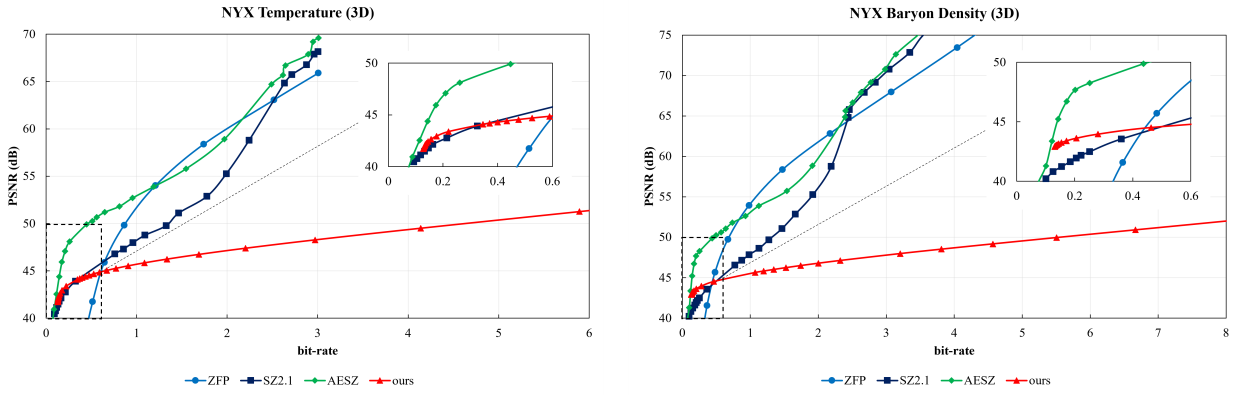


**Figure 5:** Compression performance on NYX 3D data. Left: Compression on NYX temperature. Right: Compression on NYX Baryon density.

2024, Vol. 1, Article ID. 2024.0001
https://www.doi.org/10.69709/CAIC.2024.193132



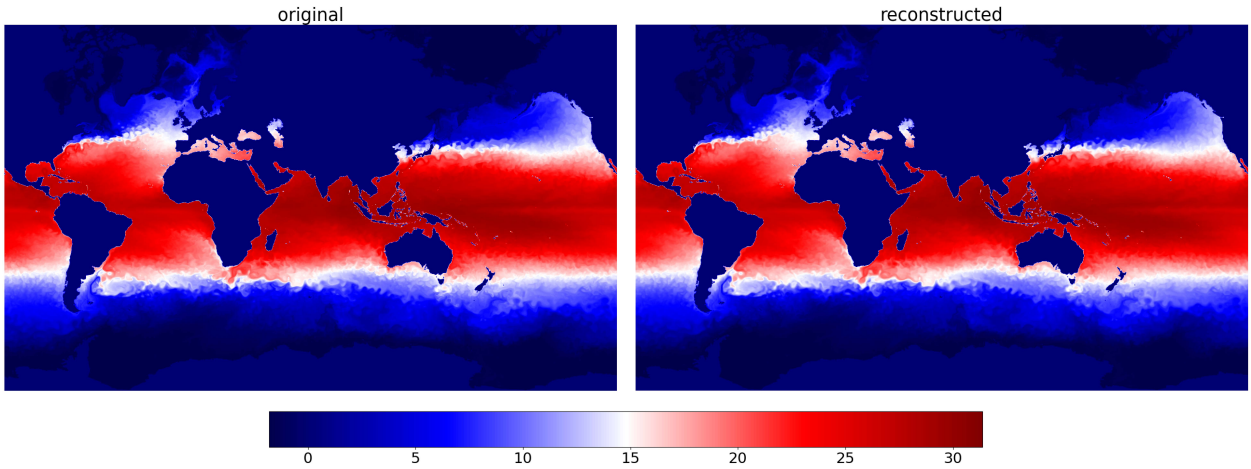


**Figure 6:** A snapshot of the sea surface temperature (SST) data (original value range: [-1.82°C; 31.38°C]; PSNR: 51.31)

onds to complete compression or decompression for 4GB of data. On a personal computer with an NVIDIA 3060 Ti accelerator, compression and decompression both take around one and a half minutes on the same data. The small difference between the two platforms indicates that the compression pipeline is primarily bottlenecked by the data transferring between CPUs and GPUs. However, compression speed on the personal computer shows promising results that the model is also suitable for compression on small devices.

## 6.3. Ablation Study on iHESP Sea Surface Temperature (SST) Data

We conduct experiments with different levels of quantization to obtain the best trade-off between compression ratio and reconstruction quality. Generally, increasing the number of quantization layers should improve the quality of the reconstruction while reducing the compression ratio. However, as shown in Table 6, our results demonstrate that the two-stage architecture achieves the lowest MSE on the testing SST data. Although our implementation can scale up to an arbitrary number of quantization layers, the reconstruction quality does not always improve. One possible explanation for this phenomenon is that when the architecture becomes too deep, it might overfit the training data, which leads to worse performance.

In terms of data pre-processing techniques, using the two-stage model, the combination of the uniform masking and overlapping partitioning method achieves the best performance over other techniques (Table 7). For the weighted masking model, we use larger weights for regions, which are more important during simulation, while lower the weights for other sections. However, the performance of the weighted version reduces the testing MSE since the model focuses more on those targeted regions with the compromise for other areas.

**Table 6:** Compression performance of different model architectures on SST data.

| Model | Testing MSE |
|---|---|
| Single Stage Quantization | 0.119 |
| Two Stage Quantization | 0.012 |
| Three Stage Quantization | 0.030 |

**Table 7:** Compression performance of the two-stage quantization model for different preprocessing techniques on SST data.

| Model | Testing MSE |
|---|---|
| masking + overlapping partition | 0.012 |
| weighted masking + overlapping partition | 0.021 |
| masking + discrete partition | 0.020 |

## 6.4. Limitation

One of many limitations of the proposed model, and possibly the biggest limitation, is the training of neural networks (NNs). NNs perform best on data that have a similar distribution to the data being trained on. Therefore, if out-of-distribution data are given to a machine learning model, the results might not be reliable and, oftentimes, incorrect. In our case, which is the Earth system, each distinct attribute requires a different model. This limitation may restrict the usability of our proposed model. However, transfer learning techniques can be used to reuse the architecture for different types of data [54].

Secondly, one architecture might not perform well for different types of data because of the difference in data distributions. Each distribution has a different property which makes it difficult to select the correct architecture

2024, Vol. 1, Article ID. 2024.0001
https://www.doi.org/10.69709/CAIC.2024.193132

for that data. As a result, exploring an optimal architecture for a data type might require a tremendous amount of effort.

## 7. Conclusion

Our proposed model proves to be effective in compressing floating-point scientific data, both on 2D benchmark data and our large-scale high-resolution data. It achieves a high compression ratio while preserving a high quality of reconstruction. The model outperforms other state-of-the-art models in some benchmark data sets, particularly 2D simulation data. However, there is room for further improvement. Other lossless compression schemes, such as arithmetic coding, which offers better compression performance, can be used to replace Huffman coding. The model can also be further improved by optimizing the rate loss term, potentially leading to a better compression ratio. Furthermore, the compression pipeline of the proposed model can be optimized to improve compression speed. Since scientific data compression using neural networks is still in its early stages, there is so much more potential improvement that can be achieved for future research along this line.

### List of Abbreviations

| | |
|---|---|
| **AE** | Autoencoder |
| **CESM** | Community Earth System Model |
| **2D** | Two-Dimensional |
| **DL** | Deep Learning |
| **VQ** | Vector Quantization |
| **MSE** | Mean Squared Error |
| **PSNR** | Peak Signal-to-Noise Ratio |
| **QoR** | Quality of Reconstruction |

## Author Contributions

Conceptualization: Jian Tao and Hieu Le; Methodology: Hieu Le and Jian Tao; Investigation: Hieu Le and Jian Tao; Writing – Original Draft: Hieu Le and Jian Tao; Writing – Review & Editing: Hieu Le and Jian Tao; Funding Acquisition: Jian Tao; Visualization: Hieu Le; Resources: Jian Tao and Hieu Le; Supervision: Jian Tao. All authors have read and agreed to the published version of the manuscript.

## Conflicts of Interest

The authors declare no competing interests.

## Funding

This work is partially supported by the TAMIDS Career Initiation Fellow Program and NSF grants OAC-2112356, OAC-2019129, and OAC-1925764.

## Acknowledgments

The authors would like to thank Dr. Chao Tian, Dr. Jaison Kurian, and Dr. Ping Chang from Texas A&M University for their suggestions and comments on this work. The authors gratefully acknowledge the helpful support provided by the School of Performance, Visualization and Fine Arts, Texas A&M High-Performance Research Computing (HPRC) and Texas A&M Institute of Data Science (TAMIDS). Portions of this research were conducted with the advanced computing resources provided by Texas A&M High Performance Research Computing. The authors confirm that no content in this manuscript was generated using artificial intelligence (AI) tools and that they take full responsibility for the accuracy and integrity of the work.

## Appendix A. Additional Experiments

### Appendix A.1. Frequency Loss Term

We conduct experiments that take into account loss terms in the frequency domain of the data. Both the input and reconstruction undergo transformation into the frequency domain using the Fast Fourier Transform (FFT). The L2 distance between the two transformed sets is then calculated using Equation (A1). This added term forces the model to directly minimize errors between high- and low-frequency components of the L2 distance.

$$l_{fft}(x, \hat{x}) = ||\mathbf{FFT}(x) - \mathbf{FFT}(\hat{x})||_2, \qquad \text{(A1)}$$

where $x$ and $\hat{x}$ are inputs and reconstructed data, respectively.

The FFT loss, $l_{fft}$, is added with other losses to create an objective function of the model as shown in Equation (A2).

$$L = \lambda_{recon} * mask * l_{recon} + \lambda_q * l_q + \lambda_{fft} * l_{fft}, \qquad \text{(A2)}$$

where $\lambda_{recon}$, $\lambda_q$, and $\lambda_{fft}$ are constant coefficients of the

2024, Vol. 1, Article ID. 2024.0001
https://www.doi.org/10.69709/CAIC.2024.193132



reconstruction, commitment losses, and FFT loss, respectively.

## Appendix A.2. Results

Our model trained with the added FFT loss performs reasonably well for the iHESP sea surface temperature (SST) data set. At a compression ratio of 221.63, the model achieves a PSNR of 47.04 for the reconstruction. Despite having a good quality of reconstruction, its performance is surpassed by the model trained without the added FFT loss, as discussed in Section 6, which achieves an average PSNR of 50.04 at a compression ratio of 231.54. One possible explanation for the lower reconstruction quality is that there is a trade-off between the MSE terms in the time domain and the frequency domain during training. While the MSE loss term in the time domain learn data representation in a particular region, the FFT loss term focuses on different regions. As a result, the quantitative result, PSNR, of the "FFT model" is outperformed by its counterpart.

# Appendix B. Additional Visualization Results

Additional results of the compression using our proposed model are provided in this section. Figures A1–A7 show visualization results for compression on different data sets.

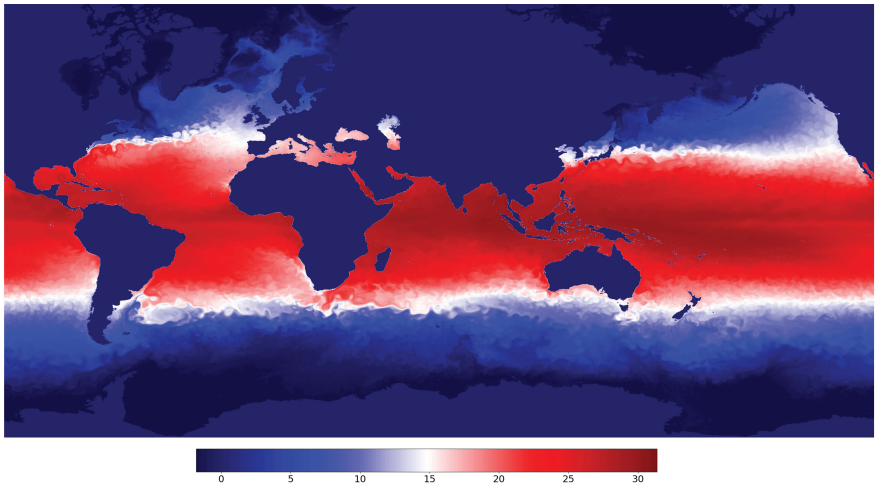


**Figure A1:** A snapshot of the sea surface temperature (SST) original data.

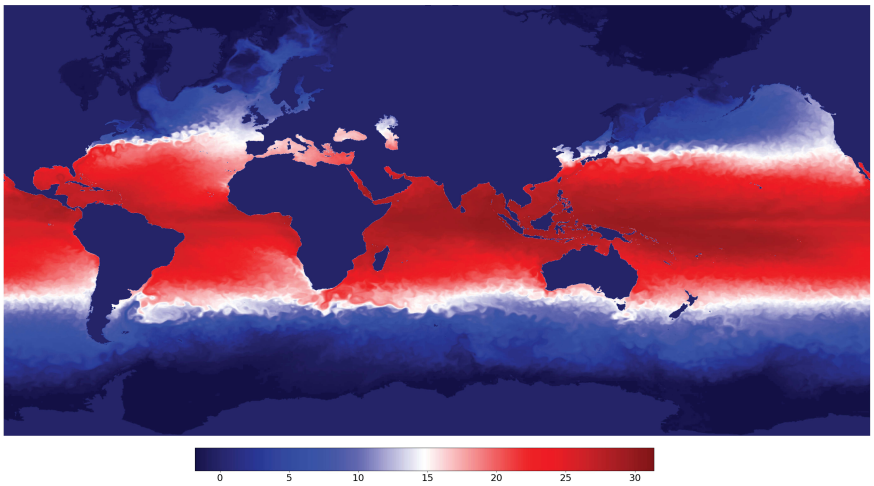


**Figure A2:** A snapshot of the sea surface temperature (SST) reconstructed data of the data in Figure A1 (PSNR: 51.31, compression ratio: 231.54).

2024, Vol. 1, Article ID. 2024.0001
https://www.doi.org/10.69709/CAIC.2024.193132

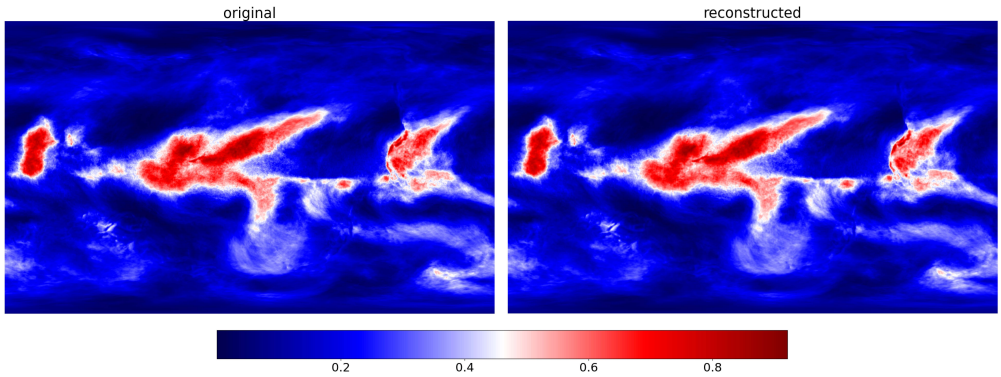


**Figure A3:** Compression for CESM CLDHGH 2D data (original value range: [3.38e-07;0.92]; compression ratio: 122.73; PSNR: 46.35).

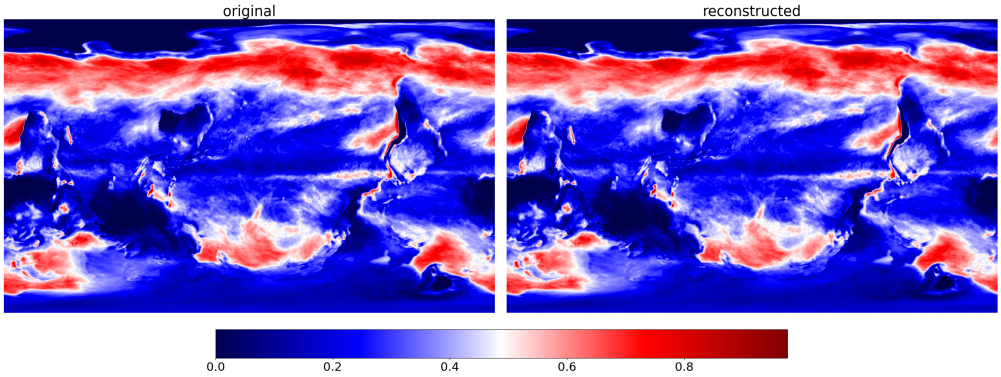


**Figure A4:** Compression for CESM CLDLOW 2D data (PSNR: 44.87, compression ratio: 136.41).

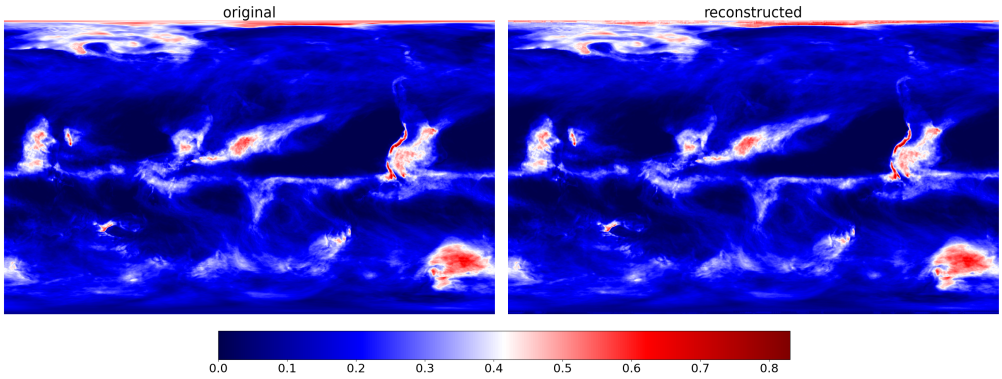


**Figure A5:** Compression for CESM CLDMED 2D data (PSNR: 42.63, compression ratio: 133.99).

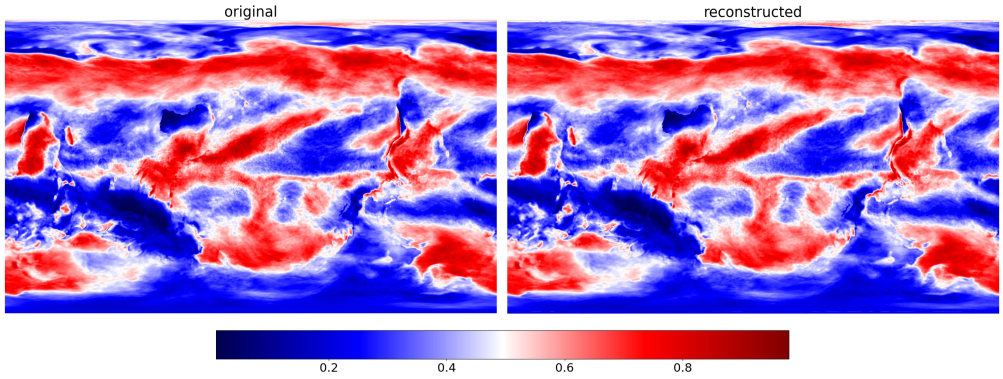


**Figure A6:** Compression for CESM CLDTOT 2D data (PSNR: 42.71, compression ratio: 127.87).

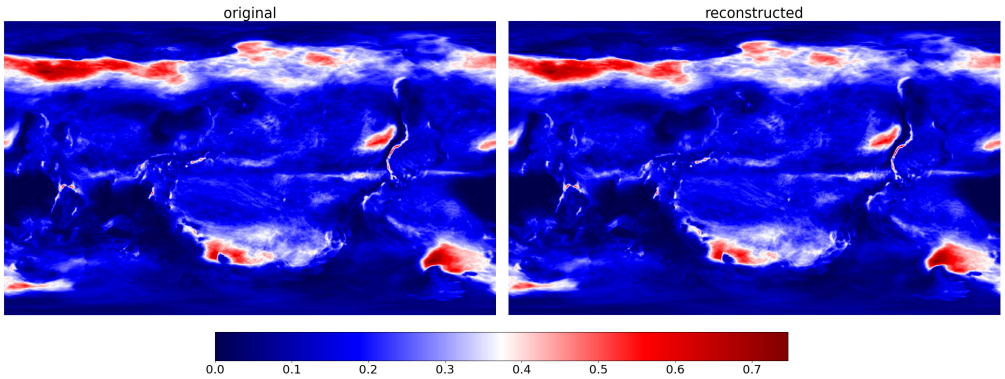


**Figure A7:** Compression for a snapshot of CESM CLOUD 3D data (PSNR: 48.51, compression ratio: 210.97).

## References

[1] SLAC National Accelerator Laboratory, "Linac coherent light source (LCLS-II)," 2023. [Online]. Available: https://lcls.slac.stanford.edu/

[2] The National Radio Astronomy Observatory, "The very large array radio telescope," 2023. [Online]. Available: https://public.nrao.edu/

[3] P. Chang et al., "An unprecedented set of high-resolution earth system simulations for understanding multiscale interactions in climate variability and change," *J. Adv. Model. Earth Syst.*, vol. 12, no. 12, 2020, Art. no. e2020MS002298. [CrossRef]

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

[5] P. Deutsch, "Gzip file format specification version 4.3," *Tech. Rep.*, 1996. [CrossRef]

[6] Y. Collet and E. Kucherawy, "Zstandard-real-time data compression algorithm," 2015.

[7] G. K. Wallace, "The jpeg still picture compression standard," *Commun. ACM*, vol. 34, no. 4, pp. 30–44, 1991. [CrossRef]

[8] S. Di and F. Cappello, "Fast error-bounded lossy hpc data compression with SZ," in *Proc. 2016 IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, Chicago, IL, 2016, pp. 730–739. [CrossRef]

[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 *Proc. 2017 IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, Orlando, FL, 2017, pp. 1129–1139. [CrossRef]

[10] J. Kim et al., "Qmcpack: An open source ab initio quantum monte carlo package for the electronic structure of atoms, molecules and solids," *J. Phys. Condens. Matter*, vol. 30, no. 19, 2018, Art. no. 195901. [CrossRef]

[11] A. H. Baker et al., "A methodology for evaluating the impact of data compression on climate simulation data," in *Proc. 23rd Int. Symp. High Perform. Parallel Distrib. Comput.*, Vancouver, BC, 2014, pp. 203–214. [CrossRef]

[12] N. Sasaki, K. Sato, T. Endo, and S. Matsuoka, "Exploration of lossy compression for application-level checkpoint/restart," in *Proc. 2015 IEEE Int. Parallel Distrib. Process. Symp.*, Hyderabad, 2015, pp. 914–922. [CrossRef]

[13] 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 *Proc. Int. Conf. High Perform. Comput.*, Genoa, 2017, pp. 30–42. [CrossRef]

[14] M. Tan and Q. Le, "Efficientnetv2: Smaller models and faster training," in *Proc. Int. Conf. Mach. Learn.*, Virtual, 2021, pp. 10096–10106. [CrossRef]

[15] A. Vaswani et al., "Attention is all you need," *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017. [CrossRef]

[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv*, 2018. [CrossRef]

[17] N. Johnston, E. Eban, A. Gordon, and J. Ballé, "Computationally efficient neural image compression," *arXiv*, 2019. [CrossRef]

[18] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv*, 2013. [CrossRef]

[19] A. Vahdat and J. Kautz, "Nvae: A deep hierarchical variational autoencoder," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 19667–19679, 2020. [CrossRef]

[20] D. Minnen and S. Singh, "Channel-wise autoregressive entropy models for learned image compression," in *Proc. 2020 IEEE Int. Conf. Image Process. (ICIP)*, Bordeaux, 2020, pp. 3339–3343. [CrossRef]

[21] J. Liu et al., "Exploring autoencoder-based error-bounded compression for scientific data," in *Proc. 2021 IEEE Int. Conf. Clust. Comput. (CLUSTER)*, Portland, OR, 2021, pp. 294–306. [CrossRef]

[22] 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 Trans. Big Data*, vol. 9, no. 1, pp. 22–36, 2021. [CrossRef]

[23] P. Lindstrom, "Fixed-rate compressed floating-point arrays," *IEEE Trans. Vis. Comput. Graph.*, vol. 20, no. 12, pp. 2674–2683, 2014. [CrossRef]

[24] P. Lindstrom and M. Isenburg, "Fast and efficient compression of floating-point data," *IEEE Trans. Vis. Comput. Graph.*, vol. 12, no. 5, pp. 1245–1250, 2006. [CrossRef]

[25] 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 *Proc. 2021 IEEE 37th Int. Conf. Data Eng. (ICDE)*, Chania, 2021, pp. 1643–1654. [CrossRef]

[26] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," *arXiv*, 2016. [CrossRef]

[27] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," *arXiv*, 2018. [CrossRef]

[28] J. J. Rissanen, "Generalized kraft inequality and arithmetic coding," *IBM J. Res. Dev.*, vol. 20, no. 3, pp. 198–203, 1976. [CrossRef]

[29] J. Rissanen and G. G. Langdon, "Arithmetic coding," *IBM J. Res. Dev.*, vol. 23, no. 2, pp. 149–162, 1979. [CrossRef]

[30] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proc. IRE*, vol. 40, no. 9, pp. 1098–1101, 1952. [CrossRef]

[31] J. Ballé, N. Johnston, and D. Minnen, "Integer networks for data compression with latent-variable models," in *Proc. Int. Conf. Learn. Represent.*, Vancouver, BC, 2018. [Online]. Available: https://openreview.net/forum?id=S1zz2i0cY7

[32] D. Minnen, J. Ballé, and G. D. Toderici, "Joint autoregressive and hierarchical priors for learned image compression," *Adv. Neural Inf. Process. Syst.*, vol. 31, 2018. [CrossRef]

[33] F. Bellard, "Bpg image format," 2023. [Online]. Available: https://bellard.org/bpg/

[34] D. Kim, M. Lee, and K. Museth, "Neuralvdb: High-resolution sparse volume representation using hierarchical neural networks," *arXiv*, 2022. [CrossRef]

[35] A. Van Den Oord, and O. Vinyals, "Neural discrete representation learning," *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017. [CrossRef]

[36] I. Goodfellow et al., "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, 2020. [CrossRef]

[37] J. N. Martel, D. B. Lindell, C. Z. Lin, E. R. Chan, M. Monteiro, and G. Wetzstein, "Acorn: Adaptive coordinate networks for neural scene representation," *arXiv*, 2021. [CrossRef]

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

[39] A. Nasari et al., "Benchmarking the performance of accelerators on national cyberinfrastructure resources for artificial intelligence/machine learning workloads," in *Proc. Pract. Exp. Adv. Res. Comput.*, Boston, MA, 2022, pp. 1–9. [CrossRef]

[40] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *J. Comput. Phys.*, vol. 378, pp. 686–707, 2019. [CrossRef]

[41] J. Choi et al., "Neural data compression for physics plasma simulation," in *Proc. Neural Compress. Inf. Theory Appl.–Workshop@ ICLR 2021*, 2021. [Online]. Available: https://openreview.net/forum?id=eEp5uad8bM

[42] Y. Zhuang et al., "Ensemble latent assimilation with deep learning surrogate model: Application to drop interaction in a microfluidics device," *Lab Chip*, vol. 22, no. 17, pp. 3187–3202, 2022. [CrossRef]

[43] C. Zhong, S. Cheng, M. Kasoar, and R. Arcucci, "Reduced-order digital twin and latent data assimilation for global wildfire prediction," *Nat. Hazards Earth Syst. Sci.*, vol. 23, no. 5, pp. 1755–1768, 2023. [CrossRef]

[44] S. Cheng et al., "Generalised latent assimilation in heterogeneous reduced spaces with machine learning

surrogate models," *J. Sci. Comput.*, vol. 94, no. 1, 2023, Art. no. 11. [CrossRef]

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, 2016, pp. 770–778. [CrossRef]

[46] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv*, 2016. [CrossRef]

[47] J. Ballé, V. Laparra, and E. P. Simoncelli, "Density modeling of images using a generalized normalization transformation," *arXiv*, 2015. [CrossRef]

[48] J. Ballé, "Efficient nonlinear transforms for lossy image compression," in *Proc. 2018 Pict. Coding Symp. (PCS)*, San Francisco, CA, 2018, pp. 248–252. [CrossRef]

[49] C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther, "Ladder variational autoencoders," *Adv. Neural Inf. Process. Syst.*, vol. 29, 2016. [CrossRef]

[50] A. Razavi, A. Van den Oord, and O. Vinyals, "Generating diverse high-fidelity images with vq-vae-2," *Adv. Neural Inf. Process. Syst.*, vol. 32, 2019. [CrossRef]

[51] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv*, 2013. [CrossRef]

[52] K. Zhao et al, "Sdrbench: Scientific data reduction benchmark for lossy compressors," in *Proc. 2020 IEEE Int. Conf. Big Data (Big Data)*, Atlanta, GA, 2020, pp. 2716–2724. [CrossRef]

[53] X. Liang et al., "Error-controlled lossy compression optimized for high compression ratios of scientific datasets," in *Proc. 2018 IEEE Int. Conf. Big Data (Big Data)*, Seattle, WA, 2018, pp. 438–447. [CrossRef]

[54] N. Wang, T. Liu, J. Wang, Q. Liu, S. Alibhai, and X. He, "Locality-based transfer learning on compression autoencoder for efficient scientific data lossy compression," *J. Netw. Comput. Appl.*, vol. 205, 2022, Art. no. 1103452. [CrossRef]