



ClimateBenchPress (v1.0): a benchmark for lossy compression of climate data

Tim Reichelt^{1,★}, Juniper Tyree^{2,★}, Milan Klöwer¹, Peter Dueben³, Bryan N. Lawrence^{4,5}, Allison H. Baker⁶, Sara Faghieh-Naini³, Torsten Hoefler⁷, and Philip Stier¹

¹University of Oxford, Oxford, United Kingdom

²INAR, University of Helsinki, Helsinki, Finland

³European Centre for Medium-Range Weather Forecasts, Reading, United Kingdom

⁴Department of Meteorology, University of Reading, Reading, UK

⁵National Centre for Atmospheric Science (NCAS), Reading, UK

⁶The NSF National Center for Atmospheric Research, Boulder, CO, United States of America

⁷ADIA Lab Fellow & ETH Zurich, Zurich, Switzerland

★These authors contributed equally to this work.

Correspondence: Tim Reichelt (tim.reichelt@physics.ox.ac.uk)

Received: 5 January 2026 – Discussion started: 9 February 2026

Revised: 16 June 2026 – Accepted: 19 June 2026 – Published: 7 July 2026

Abstract. The rapidly growing volume of weather and climate data, both from models and observations, is increasing the pressure on data centers, restricting scientific analysis, and data distribution. For example, kilometre-scale climate models can generate petabytes of data per simulated month, making it generally infeasible to store all output. To address this challenge, numerous novel compression techniques have been proposed to ease data storage requirements. However, there exist no well-defined benchmarks for rigorously evaluating and comparing the performance of these compressors, including their impact on the data's properties. The lack of benchmarks makes it difficult to design and standardize compressors for weather and climate data, and for scientists to trust that compression errors have no significant impact on their analysis. Here, we address this gap by presenting ClimateBenchPress, a benchmark suite for lossy compression of climate data, which defines both data sets and evaluation techniques. The benchmark covers climate variables following various statistical distributions at medium to very high resolution in time and space, from both numerical models and satellite observations. To ensure a fair comparison between different compressors, each variable comes with a set of maximum error bound checks that the lossy compressors need to pass. By evaluating an initial set of baseline compressors on the benchmark, we gather practical insights for

effective application of lossy compression. Our benchmark is open source and extensible: users can easily add new compressors, data sources, and evaluation metrics depending on their own specific use cases.

1 Introduction

Datasets quantifying Earth's weather and climate are rapidly growing in size due to the advent of next-generation km-scale Earth system models (ESMs) (Stevens et al., 2019; Bauer et al., 2021; Rackow et al., 2025; Segura et al., 2025) and an expanding suite of satellite missions monitoring the Earth's atmospheric state (Rossow and Schiffer, 1991; Schmit et al., 2017; Stephens et al., 2002; Illingworth et al., 2015). These datasets can easily reach petabyte scale (Bauer et al., 2021), requiring very large data centers with significant energy consumption and slowing down data analysis and sharing. Hence, there is an urgent need for novel compression algorithms that can handle the unique characteristics of climate data.

Lossless compressors cannot provide sufficiently large compression ratios due to the high entropy in the mantissa of the floating-point representation of the data (Klöwer et al., 2021), limiting compressibility following Shannon's

source coding theorem (Shannon, 1948). However, climate data comes with inherent uncertainty associated with each data point owing to measurement, model, and simulation errors that vary in space, time, and for different variables. Additionally, the precision required from the input data varies widely depending on the downstream analysis a user might want to conduct. As an example, computing the expected average global mean temperature rise from ensembles of ESM simulations is robust to uncorrelated spatial errors that will cancel upon averaging. Therefore, a lower level of precision is required than computing the expected power generation of wind farms from high-resolution storm-resolving models which requires precise knowledge of wind speeds at specific locations. In climate data there is generally a *tolerable compression error*, though defining exact, or at least practical, error tolerances for all data and downstream analyses remains an open problem.

The existence of a tolerable compression error motivates the usage of *lossy compression* algorithms. As the name implies, in contrast to lossless compression, lossy compression no longer guarantees that the original input and the reconstructed output after decompression are identical. However, by allowing for a non-zero reconstruction error, lossy compression is generally able to achieve significantly larger compression ratios compared to lossless compression (Baker et al., 2016; Underwood et al., 2022). There exists already a wide body of work on developing lossy compressors tailored to scientific datasets (Ballester-Ripoll et al., 2019; Underwood et al., 2022; Liang et al., 2022; Lindstrom, 2014; Silver and Zender, 2017; Zhao et al., 2020) and an emerging field are *neural compression* schemes for climate data (Huang and Hoefler, 2023; Han et al., 2024; Mirowski et al., 2024)¹.

However, at the time of writing, no well-established and easily accessible benchmark exists that is focused specifically on the task of compressing climate and weather data. The issue is complicated by the fact that compression algorithm developers tend not to be experts in climate science, and although abundant data exist, much of it is either difficult to access or stored in formats unfamiliar to researchers outside of climate science. This leads a majority of research to focus on evaluating compressors on datasets derived from a single model or data source, as can be seen in Fig. 1. Without questioning the importance of any single dataset, it is important to validate compressors across a variety of data sources to test their general application. Each dataset, from varying models or observations, poses different challenges for compression due to differing resolution in space and time, statisti-

cal distributions of variables, variable characteristics, associated uncertainty, and scientific objectives. For example, temperature fields are generally smooth with strong correlations in both space and time, precipitation fields often have large gradients, contain a large number of zeros and have a long tail of extreme values, whereas sea surface temperature fields are only defined over ocean regions and therefore contain large number of missing values. The differences between individual variables can be so stark that they might require different compression approaches altogether (Baker et al., 2017). However, a sufficiently general compressor should be able to deal with the diversity of variable characteristics found in climate datasets. It is important that a compression benchmark adequately captures this diversity.

Surveying the literature on lossy compression for climate data, we find that the lack of a common benchmark makes it difficult to rigorously compare the performance of different compression algorithms. Figure 1 provides an overview of the different compression ratios as reported in published works. Even for the same compressor, different studies may report drastically varying compression ratios because different datasets, different error metrics, or very different levels of a tolerable compression error are being used. Even the way that the compression ratio is measured can change between authors. Some directly measure the size reduction in the binary data of the data fields, while others measure the difference in the final file size on disk, making the compression ratio dependent on the used file format.

To provide a standardized framework for comparing compression algorithms, we therefore present *ClimateBenchPress*, a novel *climate data benchmark* for compression algorithms. The benchmark contains data on structured grids from a variety of input sources covering land, ocean and atmospheric variables of the Earth system at different spatial and temporal resolutions. Crucially, the benchmark comes with a pre-defined list of error bounds for each input variable that provides a guideline to make compression performance comparable. These error bounds are calculated with an automated algorithm and validated against bounds provided by experts. The code for data processing and evaluation procedures is fully open source and easily extensible, allowing users to add new data sources, compressors, or evaluation metrics.

In summary, our contributions are:

- Introducing ClimateBenchPress, a standardized open-source benchmark to evaluate the performance of lossy compression algorithms on climate datasets, available at <https://github.com/ClimateBenchPress> (last access: 2 July 2026).
- Evaluating an initial set of baseline compressors on ClimateBenchPress to provide a reference for the compression performance that downstream users can expect in practice.

¹In contrast to “traditional” compressors, which use expert-crafted transformations and heuristics to compress the data, neural compression approaches learn compressors from the data itself, sometimes based on large pre-training datasets and sometimes based on overfitting onto the specific data that is about to be compressed.

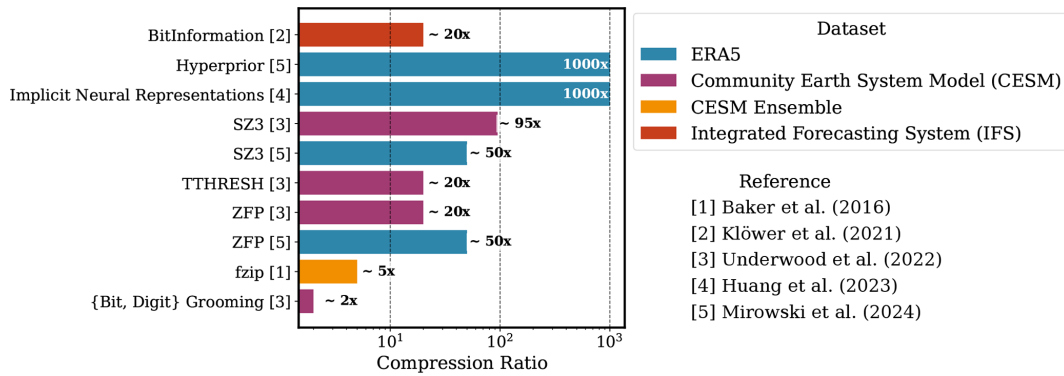


Figure 1. Compression ratios (uncompressed size / compressed size) of different compressors applied to climate data reported in the literature. There are large variations in the reported compression ratios due to differences in: the data sources and variables that are being compressed; the exact techniques used to compute the compression ratio; and the error tolerances. Overall, these differences make it difficult to directly compare the reported compression ratios.

- Discussing the trade-offs between different compressors, providing actionable insights for compression users. Namely, compressors based on bit rounding or stochastic rounding achieve modest compression ratios of around 10 but they are very robust to failures. More advanced compressors can reach higher compression ratios but they are more prone to fail on edge cases so they require more careful use-case specific validation. Compressors generally have limited support for point-wise relative errors and different compressors might use slightly varying definitions of “relative” which end users should be aware of.

2 A Benchmark Dataset for Evaluating Climate Model Compressors

2.1 Compression Tasks

The goal of the benchmark is to provide data that comes in the form of multi-dimensional arrays $X \in \mathbb{R}^{T \times N_{(\text{lon})} \times N_{(\text{lat})} \times L \times V}$ with T giving the number of time steps, $N_{(\text{lon})}$ the number of longitude coordinates, $N_{(\text{lat})}$ the number of latitude coordinates, L the number of vertical levels, and V the number of variables/fields. Crucially, T , $N_{(\text{lon})}$, $N_{(\text{lat})}$, L and V are all allowed to vary across datasets and not assumed constant. Hence, compression algorithms for this benchmark need to be able to deal with multiple grid resolutions in space, time, and number of variables. *Lossy compression* algorithms encode the data into a compressed representation such that, after decoding, the reconstructed array \hat{X} is an approximation to the original input, i.e. $X \approx \hat{X}$.

For the purpose of this benchmark, we made the simplifying assumption that all data is defined on regular longitude-latitude grids. Of course, there are many data sources which do not natively lie on a regular grid, with many ESMs using their own unstructured grids. Regridding the data to a

regular grid before passing it to a compressor is a common choice to deal with unstructured grids, hence measuring the performance of lossy compressors on regular grids still provides a useful signal for performance on unstructured grids. Moreover, many scientific compressors implicitly or explicitly assume a regular grid and therefore perform better if the input is transformed to a regular grid. However, the regridding from unstructured to regular grids is often a lossy process in itself that can affect the downstream compression performance, with the exact behavior depending on the details of the source and target grids, the regridding algorithms, and the variables that are being regridded. How to generalize compressors to arbitrary unstructured grids is still an open research problem, and outside the scope of this benchmark, as the additional complexity conflicts with the comparability we want to achieve. Similarly, there exist many relevant data modalities that do not lie on a structured grid, e.g. data from weather stations, weather balloons, ocean drifters or satellite swaths. Designing compressors for these modalities can require fundamentally different design decisions, which is why we exclude them here.

To keep the results of the benchmark easily interpretable, we only include a small representative subset of all available variables in climate data sets in this benchmark. Furthermore, we restrict the overall size of the benchmark dataset to relatively small subsets of data so that it can be easily downloaded onto a researcher’s laptop. When selecting the input datasets and variables for our benchmark, we ensured that they cover the various dimensions of the climate data compression problem, including: varying spatial resolution (100 m to 150 km); varying temporal resolution (hourly to monthly); variables with a single vertical layer and variables with multiple vertical layers; data generated from a diversity of numerical models and from observations; different statistical distributions induced by different variables; different aspects of the Earth system, including land, ocean, and atmo-

sphere variables. The specific datasets and variables in the benchmark, chosen to span the aforementioned dimensionalities of the compression problem, are:

- *Sea-level Pressure, 10m Wind Vector, and Humidity* from the Integrated Forecasting System (IFS). Crucially, we use uncompressed IEEE-754 Standard 64-bit floating point outputs (<https://apps.ecmwf.int/ifs-experiments/rd/hplp/>, last access: 2 July 2026) The IFS model underpins the popular ERA5 reanalysis dataset (Hersbach et al., 2020) but this data distributed through the Copernicus Climate Change Service (C3S) Climate Data Store has already been compressed using GRIB’s internal linear quantization that encodes floating point data into 24-bit unsigned integers (Dey et al., 2007; Klöwer et al., 2021).
- *Air Temperature* from the Coupled Model Intercomparison Project Phase 6 (CMIP6) (Eyring et al., 2016) as 4-dimensional datasets including the vertical and time dimension. Notably, the data also contains NaNs due to topography;
- *Sea Surface Temperature (SST)* fields from CMIP6 contain large regions of NaN values, due to SSTs being undefined on land, providing a significant challenge to compression algorithms.
- *Precipitation and Outgoing Longwave Radiation (OLR)* fields from the Next Generation Earth Modelling Systems (NextGEMS) project (Koldunov et al., 2023; Wieners et al., 2024; Segura et al., 2025) contain data from high-resolution km-scale climate models. The OLR – heavily impacted by the presence of clouds – and precipitation fields provide a test of whether compression algorithms can preserve high-frequency information and zero values in the input.
- *Nitrogen Dioxide* reanalysis data from the Copernicus Atmospheric Monitoring Service (CAMS) includes anthropogenic sources from industrial activity and transportation in the form of visual artifacts such as tracks and point emissions (Inness et al., 2019). We use CAMS data that is stored in 32-bit floating point precision and has not undergone any other compression (<https://apps.ecmwf.int/ifs-experiments/rd/hej6/>, last access: 2 July 2026).
- *Above-Ground Biomass (AGB)* (Santoro et al., 2024) as captured by the European Space Agency’s Climate Change Initiative is a measure of the total weight of all living trees in a unit area. Derived from a multitude of satellite observations and provided at a resolution of 100 m, it is relevant to climate modeling, as forests are a major source and sink of atmospheric CO₂.

Table 1 succinctly summarizes the different data sources and their dimensionalities, and Appendix A contains visualizations of the different data sources. Note that most of the data sources contain a single variable with $V = 1$. This is motivated by the fact that most climate and weather datasets are currently stored in chunked representations with each variable being stored separately. While there are often strong correlations between different variables that a compressor might be able to exploit, for large archived datasets like ERA5 users will rarely want to download all variables at the same time but rather only a small subset of the variables. Therefore, it is important that compressors perform well in the single-variable compression case which is the main focus of our benchmark. Nevertheless, for compression developers wishing to explore cross-variable compression we include the IFS dataset with sea-level pressure and the 10 m north- and eastward wind components, as well as, the NextGEMS dataset with the precipitation and outgoing longwave radiation variables.

2.2 Error Bounds

Compression inherently involves trade-offs between different performance metrics: compression ratio (input data size divided by encoded size), speed, and error. If we are willing to tolerate a high error, we are able to compress more and vice versa. For example, just replacing all the input data in an array with their mean will lead to a high compression ratio – we only need to store a single floating point number, the mean, instead of the entire array – but will incur a large error. This is also known in the compression literature as the *rate-distortion trade-off* (Thomas and Joy, 2006).

Therefore, in designing our benchmark it is important to specify error bounds for each variable that the compressors should satisfy. Otherwise, compressors can report large compression ratios by allowing for large errors. Unfortunately, there are no well-defined, agreed-upon, standard error bounds for each variable. This is also largely due to the fact that the level of compression error that is tolerable is inherently application dependent, as mentioned in the introduction. Consequently, we do not select a single error bound but rather a set of three error bounds, which allows us to explore the Pareto front of optimal compressors, trading off higher compression ratios with higher errors. Some compressors achieve high compression ratios only with high errors, others produce low errors but consequently also lower compression ratios.

Additionally, we distinguish between *pointwise absolute* and *relative* error bounds.

Definition 1. Let \mathbb{F}_p denote the set of numbers representable with precision p in the IEEE 754 Standard for Floating-Point Arithmetic. For input data $X \in \mathbb{F}_p^{T \times N_{(\text{lon})} \times N_{(\text{lat})} \times L \times V}$, decompressed output $\hat{X} \in \mathbb{F}_p^{T \times N_{(\text{lon})} \times N_{(\text{lat})} \times L \times V}$, and a variable index $0 \leq v < V$, let $X_{i,v}$ and $\hat{X}_{i,v}$ denote the entries corre-

Table 1. Description of datasets contained in ClimateBenchPress. The total size of the evaluation set is around 2.72 GB in v1.0. Dimensions are given as time (T), longitude ($N_{(\text{lon})}$), latitude ($N_{(\text{lat})}$), vertical levels (L), and variables (V). The data source abbreviations are: Integrated Forecasting System (IFS), Climate Model Intercomparison Project Phase 6 (CMIP6), Next Generation Earth Modelling Systems Project (NextGEMS), Copernicus Atmospheric Monitoring Service (CAMS), European Space Agency’s Climate Change Initiative (ESA CCI; observation).

Variable(s)	Data Source	Dimensions ($T \times N_{(\text{lon})} \times N_{(\text{lat})} \times L \times V$)	Time Resolution	Data Type	Data Size
Sea-level Pressure, North- and Eastward 10 m Wind	IFS	$21 \times 1440 \times 721 \times 1 \times 3$	3-Hourly	float64	523.27 MB
Humidity	IFS	$1 \times 1440 \times 721 \times 137 \times 1$	N/A	float64	1137.91 MB
Air Temperature	CMIP6	$12 \times 192 \times 144 \times 19 \times 1$	Monthly	float32	25.21 MB
Sea Surface Temperature	CMIP6	$12 \times 360 \times 300 \times 1 \times 1$	Monthly	float32	5.18 MB
Precipitation, Outgoing Longwave Radiation	NextGEMS	$8 \times 7200 \times 3600 \times 1 \times 2$	3-Hourly	float32	265.42 MB
Nitrogen Dioxide	CAMS	$8 \times 480 \times 241 \times 25 \times 1$	3-Hourly	float32	92.54 MB
Biomass	ESA CCI	$1 \times 16987 \times 9900 \times 1 \times 1$	Yearly	float32	672.69 MB

N/A – not applicable.

spending to the i th element obtained by flattening the first four dimensions (time, longitude, latitude, level) of X and \hat{X} onto a single axis. Then

1. the pointwise absolute error bound $b_{\text{abs}} > 0$ is satisfied if $|X_{i,v} - \hat{X}_{i,v}| \leq b_{\text{abs}}$ or $X_{i,v} \equiv \hat{X}_{i,v}$,²
2. the pointwise relative error bound $b_{\text{rel}} > 0$ is satisfied if $|X_{i,v} - \hat{X}_{i,v}| \leq b_{\text{rel}} \cdot |X_{i,v}|$ or $X_{i,v} \equiv \hat{X}_{i,v}$,

for all entries $i \in \{1, \dots, M\}$ with $M = T \cdot N_{(\text{lon})} \cdot N_{(\text{lat})} \cdot L$ denoting the total number of entries.

Relative error bounds are useful because absolute error bounds are not necessarily suitable for fields that vary largely in their magnitude, as e.g. many tracer concentrations. This specific formulation of the relative error bound also ensures it is defined for $X_i = 0$, compared to the alternative formulation of the relative error bound in terms of $|X_i - \hat{X}_i|/|X_i|$. Therefore, if the true value, X_i , is 0 then the pointwise relative error will only be satisfied if and only if the corresponding reconstructed value, \hat{X}_i , is also 0. Importantly, there is no single, standard definition of the relative error convention in the compression literature and other relative error conventions might not be defined around 0 or do not require exact preservation of zeros. For example, the pointwise range-relative / normalised absolute error which is defined as $|X_i - \hat{X}_i| \leq b_{\text{range-rel}} \cdot r$ with $r = \max_i X_i - \min_i X_i$, denoting the range of the data, does not require exact preservation of zeros.

²The equivalence condition ensures that the bound is still satisfied for matching infinity and NaN values. For example, without this condition the case $X_{i,v} = \infty$ and $\hat{X}_{i,v} = \infty$ would violate the bound because $\infty - \infty = \text{NaN}$ under IEEE 754.

The type of bound we use for each variable is listed in Table 2. By default, we select an absolute error bound for each variable, but we resort to relative error bounds for the specific humidity, biomass, nitrogen dioxide, and precipitation variables, because these four variables have large variations in magnitude. Additionally, the biomass and precipitation fields contain many entries with 0s, and our definition of a relative error bound ensures that these values have to be preserved exactly.

Rather than selecting the error bounds for each variable by hand, we leverage an automated mechanism that relies on the uncertainty information derived from the official ensemble statistics published along with ERA5 (Hersbach et al., 2020), which are based on a reduced resolution 10-member ensemble that incorporates the uncertainty from the data assimilation process and the physics parameterizations. The ERA5 ensemble data provides an ensemble mean $\mu_{i,v}$ and ensemble spread $\sigma_{i,v}$ for variable v at grid point i , where the index i ranges over the time, longitude, latitude, and level dimensions³. Hence, $\{\sigma_{i,v}\}_{i=1}^M$ for $M = T \cdot N_{(\text{lon})} \cdot N_{(\text{lat})} \cdot L$ represents the collection of ensemble spread data for variable v over all time steps, latitude-longitude points, and a specific vertical level. For pressure-level (not single-level) variables, we first compute the error bounds for the 1000, 850, 500, and 50 hPa levels as below and then use the average error bound for this benchmark.

³We use the ensemble mean and spread data as published on the Copernicus Climate Data Store (Hersbach et al., 2023) which contains ensemble data at 3-hourly intervals. To compute the bounds we leverage all the data available for June and December 2023.

Table 2. Individual error bounds for different input variables in ClimateBenchPress. Definition 1 describes the meaning of absolute and relative error bound, and Eqs. (1) and (2) define how the three different error bounds $b^{(low)}$, $b^{(mid)}$, and $b^{(high)}$ are computed in practice. “Non-Zero Range” indicates the smallest and largest non-zero absolute value, and the “Expert” column indicates bounds obtained by consulting domain scientists at ECMWF on what the maximum tolerable error for the variable should be, entries marked with an asterisk (*) are average rather than point-wise error bounds. Conventional abbreviations for variable names in parentheses.

Variable	Non-Zero Range	SI Unit	Type	$b^{(low)}$	$b^{(mid)}$	$b^{(high)}$	Expert
10 m u Wind Component (10u)	$[3 \times 10^{-7}, 29]$	m s^{-1}	Absolute	0.0056	0.0620	0.1000	0.01*
10 m v Wind Component (10v)	$[3.8 \times 10^{-7}, 32]$	m s^{-1}	Absolute	0.0039	0.0610	0.1000	0.01*
Mean Sea Level Pressure (msl)	$[9.5 \times 10^4, 10.5 \times 10^5]$	Pa	Absolute	1.50	6.00	7.90	1.0
Specific Humidity (q)	$[1.3 \times 10^{-11}, 2.6 \times 10^{-2}]$	kg kg^{-1}	Relative	0.17 %	0.94 %	1.42 %	1 %
Biomass (agb)	[1.0, 400]	Mg ha^{-1}	Relative	14.29 %	30.77 %	33.33 %	N/A
Nitrogen Dioxide (no_2)	$[1.7 \times 10^{-14}, 4.6 \times 10^{-7}]$	kg kg^{-1}	Relative	2.44×10^{-2} %	3.12 %	12.50 %	N/A
Precipitation Rate (pr)	$[1.4 \times 10^{-45}, 0.03]$	$\text{kg m}^{-2} \text{s}^{-1}$	Relative	1.10 %	10.71 %	17.52 %	1 %*
Outgoing Longwave Radiation (rlut)	[72, 352]	W m^{-2}	Absolute	0.0280	0.2100	0.4900	0.1*
Air Temperature (t_a)	[150, 310]	K	Absolute	0.0119	0.0740	0.1038	0.05
Sea Surface Temperature (t_{os})	$[3.6 \times 10^{-5}, 35]$	$^{\circ}\text{C}$	Absolute	6.10×10^{-5}	9.80×10^{-5}	1.80×10^{-4}	0.01

N/A – not available.

For each variable, we compute three error bounds, $b^{(low)}$, $b^{(mid)}$, and $b^{(high)}$ that are based on the percentiles of the ensemble spread data $\{\sigma_{i,v}\}_{i=1}^M$. These three bounds are meant to capture a variety of use-cases, from high error tolerance (e.g. visualization) to low tolerance (e.g. computing a function over the data). To derive absolute bounds, we use the 0 % (the minimum), 1 %, and 5 % percentiles of the non-zero ensemble spreads $\Theta = \{\sigma_{i,v} | \sigma_{i,v} > 0\}_{i=1}^M$, i.e.

$$\begin{aligned} b_{\text{absolute}}^{(low)} &= P_0\%(\Theta), & b_{\text{absolute}}^{(mid)} &= P_1\%(\Theta), \\ b_{\text{absolute}}^{(high)} &= P_5\%(\Theta). \end{aligned} \quad (1)$$

Relative error bounds are calculated in a similar manner, with the only change that the ensemble spread values are normalized by the ensemble mean, again filtering out zero spreads and zero means $\Theta^* = \{(\sigma_{i,v}/\mu_{i,v}) | \sigma_{i,v} > 0, |\mu_{i,v}| > 0\}_{i=1}^M$ in

$$\begin{aligned} b_{\text{relative}}^{(low)} &= P_0\%(\Theta^*), & b_{\text{relative}}^{(mid)} &= P_1\%(\Theta^*), \\ b_{\text{relative}}^{(high)} &= P_5\%(\Theta^*). \end{aligned} \quad (2)$$

Intuitively, this process makes the heuristic assumption that variables with larger ensemble spreads require larger error bounds because there is more uncertainty in estimating their exact value.

The bounds for the sea surface temperature (ERA5 parameter name: sst), air temperature (t), outgoing longwave radiation (avg_tnlwrf), humidity (q), precipitation (avg_tprate), sea level pressure (msl), and wind vector variables (u10, v10) are all computed in this manner from the ERA5 ensemble data. For the biomass and nitrogen dioxide variable, there exists no matching ERA5 variable, and therefore, we revert to another mechanism to compute the error bounds. The biomass dataset also publishes standard deviation estimates, which represent the precision in the biomass point estimate, so we instead plug those standard deviations into Eq. (2) to compute relative error bounds. For

nitrogen dioxide, we revert to using relative error bounds derived from the bitwise real information published in Klöwer et al. (2021), which are computed using information theoretic analysis.

Table 2 lists the exact error bounds we use for each variable. These bounds are not meant to represent general recommendations for which errors are tolerable or which error bounds should be used in practice. Instead, these bounds serve as a mechanism to enable a fair comparison of compression ratios. Users of lossy compression should always validate separately what error bound is acceptable for their specific use case.

To check whether the derived bounds are realistic, we additionally consulted domain scientists at ECMWF. Provided with existing errors from standard GRIB compression, the scientists drew on their experience with ERA5 data to specify the maximal absolute or relative errors they would consider acceptable. To not bias their judgements, scientists did not get access to the computed error bounds from Table 2. With this process, we were able to obtain “expert bounds” for the sea level pressure, humidity, precipitation, outgoing longwave radiation, air temperature and sea surface temperature variables. However, some of the expert provided bounds (marked with a * in the table) are *average* rather than *point-wise* error bounds which can sometimes make a direct comparison difficult. Notably, pointwise error bounds are more conservative because they restrict the maximum pointwise error. In comparison to our computed bounds, the expert bounds mostly lie between $b^{(low)}$ and $b^{(mid)}$. The exceptions are the sea level pressure, specific humidity, precipitation, and sea surface temperature variables. For sea level pressure the expert provided bound (1.0 Pa) is just slightly below $b^{(low)} = 1.50$ Pa. For specific humidity, the expert bound (1 %) is between $b^{(mid)} = 0.94$ % and $b^{(high)} = 1.42$ %. For precipitation the derived error bound of $b^{(low)} = 1.10$ % is arguably more conservative than the the expert provided bound

of 1 % because the derived error bound is a pointwise bound whereas the expert bound is an average bound. Hence, the expert average bound permits pointwise relative errors that exceed 1.1 % while the derived bound does not. The sea surface temperature variable is the only case for which the expert bound is multiple magnitudes larger than the derived bounds. They therefore provide bounds that are much more conservative. However, in general, the comparison to the expert bounds shows that our selected bounds are values that could realistically be picked in practice and are therefore suitable to be used as a base for comparing lossy compression algorithms.

2.3 Quantitative Evaluation Metrics

While the presented error bounds make the compression ratios between different compressors comparable, we still need a wider set of evaluation metrics to meaningfully compare their performance. As is standard in the compression literature, in this benchmark we will distinguish between two different *types of evaluation measures: distortion metrics and compression metrics*. Distortion metrics measure how much the reconstructed data differs from the original input, while compression metrics measure how much the data has been compressed and under what computational cost, e.g. encoding/decoding throughput. The next two sections introduce the specific metrics we compute as part of our benchmark. Notably though, our benchmark is easily extensible and users can readily add their own metrics for their own use cases.

2.3.1 Distortion metrics

Any distortion metric will only capture certain aspects of information loss, e.g. the maximum error bounds provided to the compressors do not take into account any spatial correlations of the errors, which is why we need multiple metrics to develop a more holistic picture of compressor performance. We selected the metrics so that each metric measures a distinct qualitative aspect of the compression performance. We purposefully made an effort to restrict the total number of metrics to avoid information overload when analyzing the benchmark results. Additionally, the distortion metrics need to be easy to understand for climate scientists, to help them inform their decision on which compressor to use in practice. Overall, the distortion metrics in the benchmark are as follows.

Mean absolute error (MAE) defined as $\frac{1}{N} \sum_i |X_i - \hat{X}_i|$ provides a measure of the average finite pointwise reconstruction error incurred by the compressor. Other metrics that are averages of pointwise errors, such as the peak signal-to-noise ratio (PSNR), could be used as well but we chose the MAE because it allows us to measure the error in the same units as the input data.

Maximum absolute error (MaxAbsError) defined as $\max_i |X_i - \hat{X}_i|$, provides a mechanism to investigate whether

the provided error bound is exceeded for any data point, or not.

Spectral Error defined as $SL(X, \hat{X}) = \sum_l (S(X, l) - S(\hat{X}, l))^2$ where $S(X, l)$ denotes the radially averaged power spectral density (RAPSD) at wavelength l . This metric allows us to check whether the compressors preserve high-frequency signal information. We compute the RAPSD for each 2D latitude longitude field separately using the conventions of the pysteps library (Pulkkinen et al., 2019) making the simplifying assumption that the data lies on a Cartesian grid. For a field X of shape $(N_{(\text{lon})}, N_{(\text{lat})})$, we apply the 2D discrete Fourier transform, shift the zero-frequency component to the array center, and form the squared-magnitude power spectrum normalized by the total number of grid points. $S(X, l)$ is then defined as the mean of all power-spectrum values whose distance from the center equals l (rounded to the nearest integer). For data with multiple time steps and vertical levels, the SL error is computed for each 2D latitude longitude field independently and then averaged over all time steps and vertical levels.

Data Structural Similarity Index Measure (DSSIM) (Baker et al., 2023) aims to measure the visual similarity between the compressed and uncompressed data as it would be perceived by the human eye. Compared to pointwise error metrics it also takes into account the spatial correlations of the data. In particular, DSSIM is an adaption of the SSIM measure, which is widely used in the compression literature, to scientific floating point data. DSSIM is only defined on 2D fields, to be able to handle data with multiple time steps and vertical levels we follow the convention of Baker et al. (2023) and compute the DSSIM for each vertical level and time step separately, then take the minimum over vertical levels and average over time steps.

2.3.2 Compression Metrics

The standard measure to compare compression performance is the *compression ratio*, which is computed by dividing the size of the input data by the size of the encoding; hence, larger compression ratios are preferable because they correspond to smaller compressed files. For the benchmark we define the input data size as the number of bytes necessary to store the raw input arrays without any metadata. An input array $X \in \mathbb{R}^{T \times N_{(\text{lon})} \times N_{(\text{lat})} \times L \times V}$ with $M = T \times N_{(\text{lon})} \times N_{(\text{lat})} \times L \times V$ entries stored in 32-bit floating point precision will therefore be defined as having an input size of $4 \cdot M$ bytes. This definition of the input size ensures that our compression ratio measurements are independent of the data format that is used to store the data and that we are measuring the pure compression performance of the compression algorithm and not the ability of a compressor to deal efficiently with a specific data format.

Another important aspect is the computational cost of applying a compressor, which we measure in two different

ways. Firstly, for each compressor we keep track of the *compression and decompression throughput*, i.e. what is the wall clock time for compressing a fixed chunk of data? This metric is relevant because it allows us to extrapolate the time it would take to compress/decompress datasets of arbitrary size. On the other hand, raw throughput measurements are generally flawed because they are noisy and depend on the specific hardware and software environment which the benchmark code was executed in, including what other processes were executed at the same time. Hence, we additionally record the *instruction count* for both compressing and decompressing the data. Our benchmark ensures that the baseline compressors will reproducibly execute with exactly the same instruction counts on different machines. Therefore, instruction counts are a more objective measure of compressor complexity.

Similarly, other performance aspects such as memory usage or support for multi-threaded execution and hardware accelerators are often relevant in practice but are hard to measure objectively which is why we do not consider them here. Additionally, as most climate and weather datasets are stored in chunked representations, even compressors that only support single-threaded compression can easily be parallelized over large datasets by compressing separate chunks independently. Hence, the instruction count for single-threaded execution provides an objective first-order estimate of a compressor's computational cost and their relative ranking.

The prioritization between different compression metrics might also change depending on the different compression tasks, use cases, and compute and storage resources available. Not only that, but the prioritization between encoding and decoding complexity might change depending on the use case as well. For example, if we have a large archival dataset, such as ERA5, that gets accessed by thousands of users, then the compression ratio and decompression speed should be maximized. In this case, it is important that scientists are able to download and decompress the data quickly on their personal machines or institutional clusters. However, it is less important that the data can be compressed quickly because this only has to be done once. Therefore, compression times of multiple hours or even days might be acceptable. Of course, other use cases could have different priorities. For example, in operational weather forecasting, the time and computational resources that can be allocated to compression are constrained by the rate at which new forecasts need to be produced.

2.4 Implementation Details

All the code to download the data and reproduce the experimental results for the benchmark is accessible open-source at <https://github.com/ClimateBenchPress>. The data from the different sources outlined in Table 1 are converted into Zarr arrays (Miles et al., 2020) with canonicalised dimensions $V \times T \times L \times N_{(\text{lat})} \times N_{(\text{lon})}$ to provide a common data for-

mat for the benchmark. To compare a new compressor, the minimum requirement is that it can be made to compress a Zarr array (e.g. by individually compressing the numpy arrays for each variable), measure the byte size of the compressed representation, and output a Zarr array with the decompressed output. In the benchmark, we use the numcodecs (<https://github.com/zarr-developers/numcodecs>, last access: 2 July 2026) compression framework, which provides a minimal API for compressing n -dimensional arrays. To add a new compressor to the reproducible benchmark baseline, it must be wrapped inside a numcodecs codec.

For the results presented in this paper, we additionally wanted to ensure the fairness, reproducibility, and comparability of the performance of different compressors. To compress the input data, we use the numcodecs-wasm (<https://numcodecs-wasm.readthedocs.io>, last access: 2 July 2026) project, which adapts many popular scientific compressors to the numcodecs compression API (<https://github.com/zarr-developers/numcodecs>, last access: 2 July 2026). To guarantee cross-platform reproducibility, the numcodecs-wasm project compiles its compressors to WebAssembly bytecode inside an isolated Nix build environment and publishes them as pure Python packages that are compatible with the numcodecs API. WebAssembly (WASM) is a cross-platform bytecode format that was initially designed to safely execute compiled programs on web pages. Given its focus on security and cross-web-browser compatibility, WebAssembly is designed to facilitate performant and fully sandboxed execution (Spies and Mock, 2021). In our experimental evaluation, we use the Wasmtime (<https://github.com/bytecodealliance/wasmtime>, last access: 2 July 2026) WebAssembly reference engine. While the core WebAssembly instruction set is not entirely deterministic, non-deterministic instructions are very limited by design (WebAssembly Working Group, 2019), and any single-threaded core WebAssembly program can be transformed to have bit-reproducible execution. The WebAssembly 3.0 draft specification even defines a deterministic profile for this use case (WebAssembly Community Group, 2025). It is furthermore possible to instrument any WebAssembly program with a virtual bytecode instruction counter, which can be used as a deterministic, fully reproducible, user-space performance metric that does not suffer from measurement noise or platform differences (Binder and Hulaas, 2006). Since the Python packages provided by numcodecs-wasm ship with the pre-compiled cross-platform WebAssembly bytecode and only use deterministic instructions, the wrapped compressors execute exactly the same on any platform, thus allowing this benchmark to report reproducible, platform-independent, and noise-free results. Overall, the usage of numcodecs-wasm guarantees that, apart from wall-time execution speed measurements, all results and metrics are reproducible on any CPU architecture, from large HPC systems to interactive demonstrations on local machines.

For the comparison of the baseline compressors in Sect. 3, different variables are compressed separately. Evaluating different multi-variable compression approaches is left for future extensions to the benchmark.

3 Results

In order to gain an understanding of the current state of lossy compression algorithms for climate data, we evaluate a set of baseline compressors on ClimateBenchPress. For readers who are new to lossy compression, the presented results here should be seen as a helpful guideline for how to interpret the results of the benchmark and understand the different trade-offs involved when choosing between different codecs. For readers who are interested in using lossy compression for their own work, the results here can help narrow down the compressors they might want to consider. As mentioned earlier though, the benchmark is mainly designed to be a tool to *compare* different compressors and *not to validate* them. It is important that all users of lossy compression run their own use case-specific validations before applying lossy compression. This is especially important because different compressors can have substantially different qualitative behavior that can be hard to capture using general-purpose error metrics, as we will discuss in Sect. 3.3.

3.1 Baseline Compressors

As baselines, we mainly consider compressors that were specifically developed for floating-point data and scientific applications. A key feature of lossy compressors designed for scientific applications is a mechanism to control the maximum pointwise error (relative and/or absolute error) incurred during the compression process. Some compressors also support bounding the peak signal-to-noise ratio (PSNR) which also bounds the mean squared error. Bounding the pointwise absolute error is a stronger bound and therefore also bounds an average error but many compressors do not provide this as a specific control. A notable exception in our list of baseline codecs is the JPEG2000 codec, which was developed for image compression but has been increasingly applied to scientific data as well (Woodring et al., 2011; Silver and Zender, 2017; Pellicer-Valero et al., 2025). Since it has seen decades of development and continuous improvement, it can provide a reference point for what compression ratios may be possible with traditional compression schemes. For now, we have not included any neural compressors in the benchmark because they are either developed only for a specific dataset (Mirowski et al., 2024; Han et al., 2024), generally ERA5, or do not come with any built-in mechanisms to control their error bound (Huang and Hoeffler, 2023). We hope this benchmark encourages developments in neural compressors that overcome these shortcomings. We now briefly describe our baseline compressors.

Bit Rounding (Zender, 2016; Delaunay et al., 2019) rounds the floating point mantissa to a specified number of bits. This gives a transparent mechanism to control the desired precision and relative error in the data and provides an interpretable mechanism for understanding what information is lost in the compression stage. Bit Rounding can bound the absolute error by varying the number of mantissa bits but this is not considered here. Similarly, *Stochastic Rounding* uses random noise to round the input data to the nearest specified decimal precision. The goal of the added noise is to avoid any systematic biases in the compressed data due to the rounding process, which can otherwise remove small gradients in the data. Stochastic Rounding can be formulated to round to a constant number of mantissa bits, effectively bounding a relative error but this is not considered here. We combine both Bit Rounding and Stochastic Rounding with a lossless compressor. For each rounding mechanism we will evaluate two choices of lossless compressors: *Zstd*, a widely used byte compression codec designed for web data, and *PCO* (Loncaric et al., 2025), a novel lossless compressor designed specifically for numerical data. *SZ3* (Liang et al., 2022) is a prediction-based error-bounded lossy compressor that uses Lorenzo predictors (Ibarria et al., 2003) and spline-based interpolators to exploit local correlations in the input data; in our experiments we use version 3.2.2. *EBCC* (Huang et al., 2026) uses JPEG2000 as a base compression layer and uses residual encodings based on wavelet transforms and SPIHT (Set Partitioning In Hierarchical Trees) to ensure adherence to error bounds. *ZFP* (Lindstrom, 2014) instead decorrelates the data by applying an orthogonal transform to $4 \times \dots \times 4$ sub-blocks of X . We additionally evaluate a more recent variant of ZFP called *ZFP-ROUND* (Hammerling et al., 2019) that reduced bias with an improved rounding mechanism for the transform coefficients based on feedback from the climate science community. *JPEG2000* (Skodras et al., 2001) uses a wavelet-based transform to decorrelate the data and then quantizes individual entries in the transformed input space. JPEG2000 does not support floating-point data; instead, we linearly quantize the data to operate on integers. JPEG2000 does not provide a mechanism to bound the maximum pointwise error; instead, we convert the absolute or relative error bounds to peak signal-to-noise ratio (PSNR), which can be passed to the compressor as a target. Notably though, PSNR is a measure of the *average* rather than maximum error. Finally, *SPERR* (Li et al., 2023) is also a wavelet-based transform compressor but with the added feature of allowing for bounding the maximum pointwise error. The exact versions of the compressors that we tested in the benchmark are given in Table B1 in Appendix B.

As highlighted in Table 3, most compressors do not have support for specifying both absolute and relative error bounds. Therefore, we sometimes need to convert between absolute and relative error bounds in order to use the error bounds from Table 2 for a specific compressor. Specifically, according to Definition 1, a pointwise relative error bound,

Table 3. Supported types of error bounds for different compressors. The support for relative error bounds for EBCC, SZ3, ZFP, and ZFP-ROUND is in brackets because while these compressors have some mechanism to bound the relative error, we found in our experiments that they tend not to strictly adhere to the provided error bounds. In particular, the “relative” error bounds provided by EBCC and SZ3 are range-relative, not pointwise relative. Bit and Stochastic Rounding have errors in brackets as it is possible to reformulate these compressors towards these bounds which however is ignored here. Some compressors also support bounding the peak signal-to-noise ratio, PSNR. SZ3 is in brackets because it translates the PSNR bound into a pointwise absolute error bound internally.

Compressor	Pointwise Abs. Error	Pointwise Rel. Error	PSNR
Bit Rounding	(✓)	✓	×
Stochastic Rounding	✓	(✓)	×
EBCC	✓	(✓)	×
JPEG2000	×	×	✓
SPERR	✓	×	✓
SZ3	✓	(✓)	(✓)
ZFP	✓	(✓)	×
ZFP-ROUND	✓	(✓)	×

b_{rel} , satisfies $|X_i - \hat{X}_i| \leq b_{\text{rel}} \cdot |X_i|$ for all indices i , whereas a pointwise absolute error bound satisfies $|X_i - \hat{X}_i| \leq b_{\text{abs}}$. For relative error bounds the right-hand side value of the inequality varies depending on the input value $|X_i|$, whereas the right-hand side value is fixed for the absolute error bound. To convert a relative error bound, b_{rel} , to an absolute error bound, we find the smallest non-zero right-hand side value that appears in the dataset, i.e. $b'_{\text{abs}} = \min_i |X_i| \cdot b_{\text{rel}}$, where the minimum is taken over all *non-zero* values. Similarly, to convert an absolute error bound, b_{abs} , to a relative error bound b'_{rel} we want to ensure that every possible right-hand side value of the inequality is bounded by the original absolute error bound, i.e. $b'_{\text{rel}} \cdot |X_i| \leq b_{\text{abs}}$ for all i , which is satisfied by picking $b'_{\text{rel}} = b_{\text{abs}} / (\max_i |X_i|)$.

These error bound transformations are intentionally quite conservative, meaning that if a compressor only supports absolute error bounds, passing b'_{abs} as an error bound will ensure the original relative error bound is satisfied on all non-zero entries. At the same time, if the input data has many near-zero or subnormal values the converted absolute error bound b'_{abs} might be very restrictive. Other approaches to converting relative to absolute error bounds and vice versa are feasible, e.g. Liang et al. (2018) describe an alternative method for bounding the ratio-relative error using a logarithmic transform.

Overall, if an error bound transformation was applied for a compressor, the achieved compression ratios should be seen as a conservative lower bound estimate and higher compression ratios could be achievable if a more permissible error bound transformation is used. Both the SZ3 and the EBCC compressor allow the user to configure a relative error bound. However, they use a range-relative error (as defined in Sect. 2.2) which in most cases is less strict than our pointwise relative error bound. To assess the impact of these different relative error bound definitions we also consider the two compressor variants *SZ3-Abs* and *EBCC-Abs* that do not use the built-in relative error functionality of these compressors.

Instead these variants convert all pointwise relative errors to pointwise absolute errors using the conservative transform described above.

3.2 Quantitative Results

3.2.1 Scorecards

A single evaluation sweep over all baseline compressors, input variables, error bounds, and evaluation metrics produces a large set of results that can be difficult to analyze. Our two main tools to summarize our benchmark results will be: (1) Scorecards, inspired by their use to evaluate different forecasting models at ECMWF (<https://sites.ecmwf.int/ifs/scorecards/>, last access: 2 July 2026) and in WeatherBench (Rasp et al., 2024); and (2) rate-distortion plots, which are common in the compression literature.

The scorecards in Fig. 2 provide a quick visual overview of which compressors perform well in the different performance metrics. The wavelet-based compressors JPEG2000, SPERR, and EBCC are all able to achieve large compression ratios but they behave differently in how they control errors. JPEG2000 tends to fail the provided error bound as it has no in-built mechanism to constrain the pointwise error. SPERR does not always adhere to the error bound due to known issues in encoding the error bound correction (Fallin and Burtscher, 2024). EBCC tends to provide smaller average errors compared to SPERR but often fails on relative error bounds which is discussed more below. SZ3 is also able to provide compression ratios that are often on the same order of magnitude as those of the wavelet-based compressors. However, the benchmark also exposes some interesting failure cases for SZ3 v3.2.2. For example, both the air and sea surface temperature variables (“ta” and “tos” columns in the scorecard, respectively) contain NaN values in the input data, but instead of preserving these NaN values, SZ3 fills in these

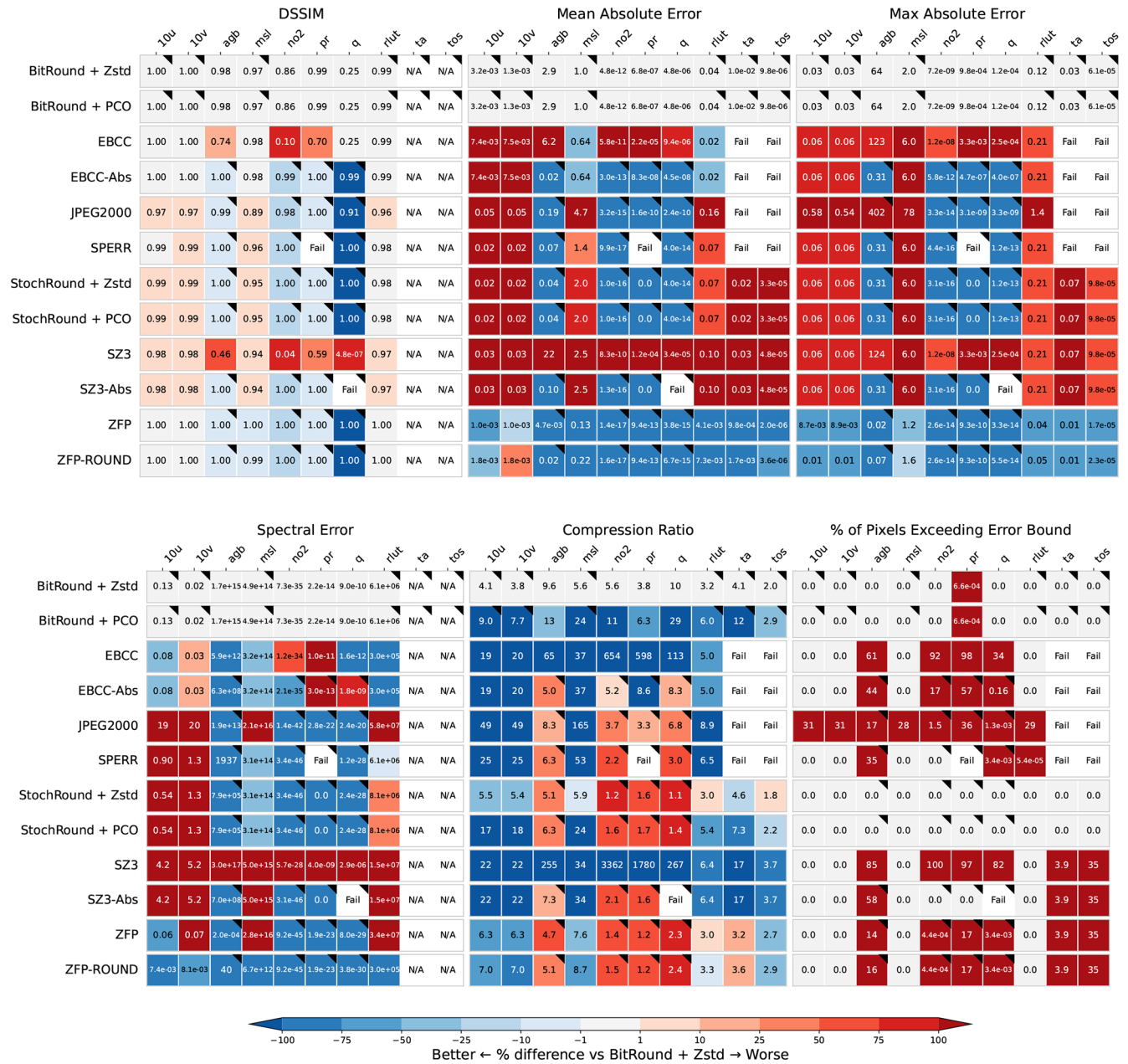


Figure 2. Overview scorecard summarizing the benchmark results. Each *panel* shows a different metric, each *row* a different compressor, and each *column* a different variable. Black triangles indicate compressor-variable pairings for which an error bound transformation was used. The scorecard allows us to compare the performance of compressors across different metrics: for example, SZ3 provides uniformly high compression ratios but sometimes exceeds the provided error bounds and leads to generally higher average errors. This scorecard shows the specific metrics evaluated on the “mid” level error bound; scorecards for the other error bounds are listed in Appendix C. “Fail” entries denote cases in which running the compressor raised an error. We exclude DSSIM and Spectral Error metrics for “ta” and “tos” variables, indicated by “N/A” entries, because these metrics are not robust to large regions of NaN values. See Table 3 for variable abbreviations.

regions in the input data with interpolated numerical values.⁴

⁴Since our discovery of this issue, we have submitted a patch to SZ3, <https://github.com/szcompressor/SZ3/pull/91> (last access: 2 July 2026), which has been published in v3.3. However, v3.3 introduced an out-of-bounds access bug, which we reported in <https://github.com/szcompressor/SZ3/issues/119> (last access: 2 July

ZFP and ZFP-ROUND exhibit the same failure case, which is especially problematic because it is a silent failure; no error was raised during the compression process. JPEG2000

2026). At the time of writing, SZ3 v3.4 that would include the fix has not yet been published.

and SPERR fail on NaN inputs as well, but they raise an error instead of infilling any values⁵. In the case of JPEG2000, the failure can be attributed to the quantisation from floating-point numbers to integers, which does not account for NaN values.

Additionally, many compressors struggle to strictly adhere to relative error bounds; see Table 2 for the list of variables with relative error bounds. As discussed in Sect. 3.1, SZ3 and EBCC use alternative definitions of relative errors which leads to large percentages of pixels exceeding the error bound. The SZ3-Abs variant that utilises the conservative relative to absolute error transformation mostly remedies this issue, however it still fails on inputs that require preservation of 0 values⁶. The EBCC-Abs variant reduces the failure rate for relative error bounds but still has a significant percentage of pixels not adhering to the bound. ZFP, ZFP-ROUND⁷, and SPERR also struggle with relative error bounds as they all fail to preserve 0 values, e.g. above-ground biomass is zero over the ocean but compressed here to non-zero values⁸.

Even Bit Rounding fails to adhere to the relative error bound on the precipitation data. Upon investigation of the ICON precipitation data, we identified that the failure is due to subnormal floating-point numbers being present in the ICON data. Bit Rounding fails to adhere to the error bound because the implementation that we are using computes the number of mantissa bits to keep as $k = -\lceil \log_2(b_{\text{rel}}) \rceil - 1$ from the relative error bound b_{rel} , but this formula fails to hold for subnormal values. This failure case provides an illuminating example of why we need rigorous evaluation checks for lossy compressors. In many ways, this is a benign failure. The values are so small, smaller than $\approx 1.1754942107 \times 10^{-38} \text{ kg m}^{-2} \text{ s}^{-1}$, that they are physically implausible and a numerical artifact from the model. Furthermore, Bit Rounding will round these values to 0, so the impact on calculating any downstream statistics, e.g. total precipitation, will be insignificant. At the same time, exactly because these values are numerical artifacts from the model, it is important that lossy compression algorithms do not “mask/round away” these unphysical values, or at least make users aware of this special case. If this particular implementation of Bit Rounding were deployed in a way that all model outputs are lossily compressed before being written to disk, the model developers would have no way of knowing that the model produces subnormal values as outputs⁹.

⁵Notably, the SPERR HDF5 (<https://github.com/NCAR/H5Z-SPERR>, last access: 2 July 2026) plugin explicitly checks for NaNs and does not have this failure case.

⁶SZ3-Abs runs out of memory on the IFS humidity datasets.

⁷For both version of ZFP, we always run in “fixed-accuracy” mode and transform relative errors to absolute errors.

⁸Note that in our definition of the relative error bound, it implicitly also checks that 0 values are preserved in the data (see Definition 1).

⁹SPERR also fails on the precipitation data because the existence of subnormal numbers in the data means the converted abso-

In general, for variable and compressor combination that require a relative to absolute error bound transformation, we observe markedly lower compression ratios – this is true for the Stochastic Rounding, EBCC-Abs, JPEG2000, SPERR, SZ3-Abs, ZFP, and ZFP-ROUND compressors on the biomass (agb), nitrogen dioxide (no2), precipitation (pr), and humidity (q) variables. As noted earlier, this is due to the fact that we use a very conservative error bound transformation. Higher compression ratios could be achieved with a less conservative error bound transformation (e.g. Liang et al., 2018) or alternative relative error bound definitions, such as the range-relative error mentioned in Sect. 2.2.

Overall, our results demonstrate the need for a benchmark as a testing ground for compression developers. None of the failure cases of the compressors are necessarily difficult to overcome from an algorithm design perspective (failure on NaNs and adherence to relative error bounds), but the lack of an existing benchmark means that, currently, compression developers often do not consider these edge cases as requirements.

3.2.2 Rate-Distortion Plots

While scorecards provide a detailed overview of the compressor performance across different metrics and inputs, they make it difficult to visually assess the trade-off between compression ratio and distortion errors. Rate-distortion plots, as those shown in Fig. 3, allow us to assess exactly this trade-off. With rate-distortion plots, we can get an understanding of what compressors lie on the Pareto front of giving both high compression ratios and low levels of distortion, for different definitions of distortion. Normalization is applied to the compression ratios and error metrics for each variable, averaged across all datasets. This is done because different variables have different units and averaging their errors without normalizing would lead to the variables with the largest magnitudes dominating the average. Because the plotted quantities are averages, the rate-distortion plots in Fig. 3 should always be read in combination with the scorecards because even if a compressor does well on average across datasets there is still significant variation in their performance across the different datasets and variables. Specifically, for a set of metric values $\{m_{v,b,c}\}$ with input variable v , error bound b , and compressors c we compute the mean and standard deviation over all compressors and error bounds to obtain *normalized scores* as $n_{v,b,c} = (m_{v,b,c} - \text{mean}_{c,b}(\{m_{v,b,c}\})) / \text{SD}_{b,c}(\{m_{v,b,c}\})$. In Fig. 3 the normalized scores are for a given compressor and error bound averaged over all variables, i.e. $n_{c,b} = \text{mean}_v(\{n_{v,b,c}\})$. These averaged normalized scores represent the average performance of a compressor *relative to the other compressors*. Values larger than 0 indicate that a compressor, on average across variables, achieves a larger score than the other compressors.

lute error bound that is passed to the compressor is very small which causes SPERR to error.

sors whereas a score below 0 indicates that the compressor achieves a lower score than the other compressors.

Figure 3 shows that SZ3 and the wavelet-based compressors (SPERR, JPEG2000, EBCC) generally provide the highest compression ratios across the different input variables. This, however, needs to be seen in the context of the results from the scorecards that showed these compressors often fail to strictly adhere to the provided error bound. Bit Rounding in combination with the PCO lossless compressor, “BitRound + PCO” in the plots, is surprisingly competitive and often provides compression ratios that are even competitive with more sophisticated compressors such as SPERR, despite the well respected error bounds. The notable jump in the compression ratios between “BitRound + Zstd” and “BitRound + PCO” highlights that we can still get quite significant improvements in the compression performance by using lossless compressors that are specifically designed for numerical data. Additionally, this makes a case for modular compressors, as the compression ratios are increased with no change to the lossy compression method and its errors. Overall, these results suggest that the simple combination of BitRound + PCO is a reliable and competitive alternative for users who are interested in not exceeding pointwise error bounds.

3.2.3 Instruction count

Finally, Fig. 4 displays the last aspect of compressor performance that we have left out so far: compression cost. We measure cost in terms of bytecode instruction count per encoded/decoded byte of data. We focus here on instruction count in favor of wall clock throughput measurements because the platform-independent bytecode instruction counts are reproducible between different CPU hardware architectures. Throughput measurements, on the other hand, tend to be noisy due to their dependence on, e.g. hardware details and what other processes are running on the same machine at the same time. Wall clock throughput measurements are shown in Fig. E1 in Appendix E¹⁰. EBCC is the most expensive compressor in encoding cost which is to be expected because it relies as JPEG2000, the second most expensive compressor for encoding, as a base compressor. This is a deliberate trade-off made by EBCC though because it was developed with the use on large archival datasets in mind that only need to be compressed once. Decoding time for EBCC is still competitive with the other compressors. Generally, encoding is more costly than decoding for all compressors except for both versions of ZFP. Both encoding and decoding costs

¹⁰The wall clock throughput measurements in Fig. E1 should only be used as a relative comparison between different compressors because our compilation of each compressor to WebAssembly incurs an overhead (Jangda et al., 2019). Effectively, our compilation to WebAssembly trades off execution speed in favor of reproducibility and portability, which we prioritize in this benchmark to better compare compressors.

are largely error independent. While some compressors, like ZFP, are known to be faster for very high error tolerances, within the provided error ranges low to high this is of little practical relevance. Even though Bit Rounding and Stochastic Rounding are simple lossy compression mechanisms, the lossless codecs PCO and Zstd add significant complexity to the entire compression procedure so that their compression cost is comparable to more complex lossy compressors such as SZ3, ZFP, and ZFP-Round.

3.2.4 Sensitivity of Compressor Performance to Chunking

As a default, we are passing the input data to compressors as a single chunk because of the relatively small overall size of our datasets; an exception is the humidity data which has 2 chunks by default because of its large size. Popular file formats, like NetCDF and Zarr, have explicit support for storing data in chunks, which allows for efficient access of subsets of large datasets because it avoids the need to load the full dataset into memory.

As most climate datasets are stored in a chunked representation, we conducted a sensitivity analysis to study the impact that chunking has on compression performance. Determining good chunk sizes for a dataset is mainly application-specific because it depends on the typical access patterns to the data. Here, we chose a simple heuristic that ensures the chunk dimensions are roughly proportional and clean divisors of the full dataset dimensions. Additionally, the heuristic ensures each chunk is at least 4 MB in size, following recent guidance for CMIP7 datasets (Hassell and Cimadevilla Alvarez, 2025). The actual chunking used for each dataset is given in Table G1 in Appendix G.

Figures 5 and 6 show that the overall impacts of chunking on the compression ratio and mean absolute error are small. Across datasets and compressors, while overall for the majority of compressor-variable combinations the unchunked datasets provide marginally better compression ratios, for most compressors there is no uniform trend on whether chunking improves the compression ratio or not. Furthermore, the difference in compression ratios between the chunked and the unchunked data is generally < 1 , demonstrating that the overall effect is small. Even though the absolute change is small, for the ZFP and ZFP-ROUND compressors the impact of chunking is more significant because the base compression ratios are quite small so even small absolute changes have a noticeable impact. Further, on variables with a relative error bound, the EBCC and SZ3 compressors have large absolute changes. This is due to the aforementioned fact that EBCC and SZ3 internally transform the relative error bound into an absolute error bound constraint with the relationship $b'_{\text{abs}} = b_{\text{rel}} \cdot (\max_i X_i - \min_i X_i)$. If the data is passed in chunks, the range of the data in the chunk, $(\max_i X_i - \min_i X_i)$, will vary between chunks and therefore the compressors will in effect apply different error bounds

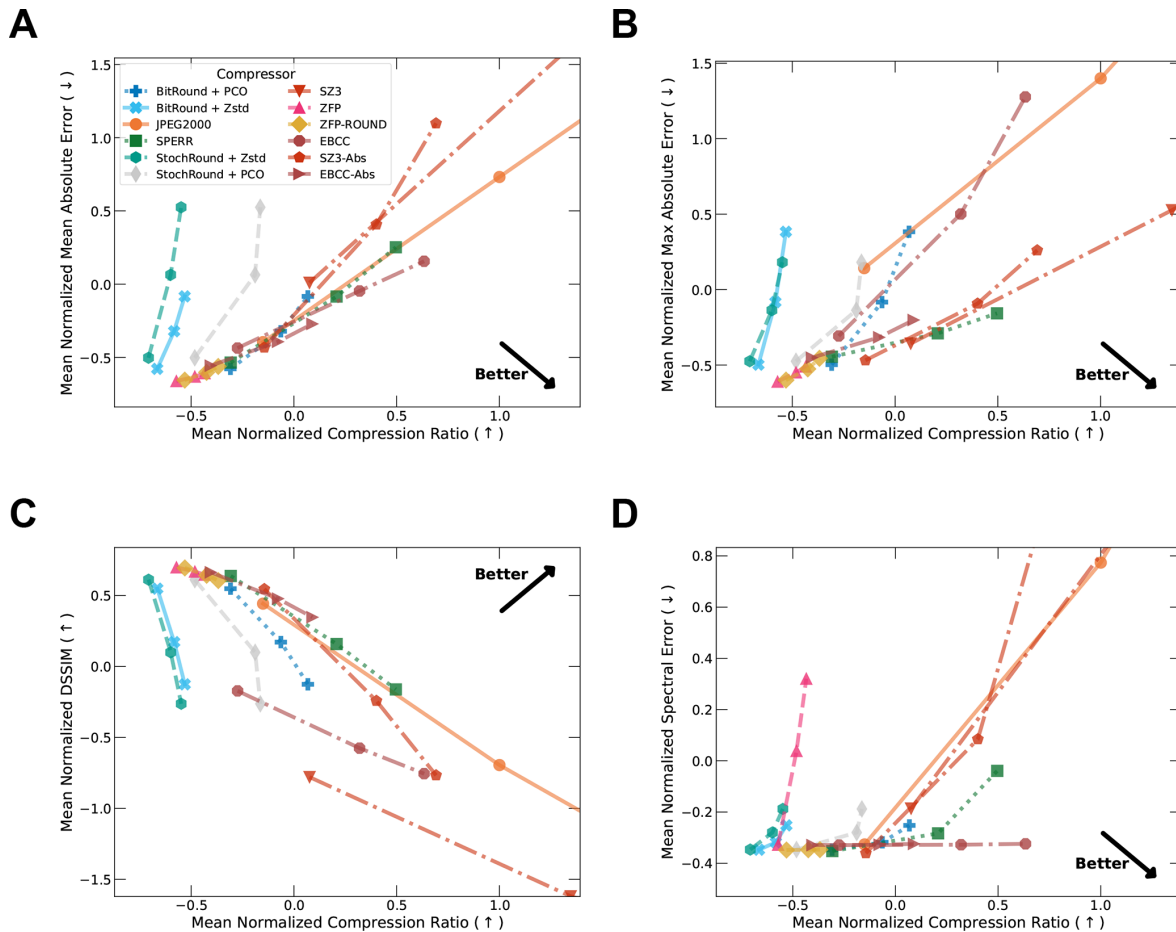


Figure 3. Rate-distortion plots for multiple distortion metrics: (A) normalized mean absolute error; (B) normalized maximum absolute error; (C) normalized DSSIM; (D) normalized spectral error, each on the y-axis, respectively. Normalization is applied to make compressor performances across metrics and datasets comparable, see text. Axes limits were selected to exclude outlier values; the high compression ratios of SZ3 and JPEG2000 are achieved through exceeding the provided error bounds. Appendix D shows the uncropped rate-distortion curves.

to different chunks. This leads to the large observed difference in some compression ratios for EBCC and SZ3 in the nitrogen dioxide, precipitation, and humidity variables.

Overall, the impact of chunking on compressor performance depends on the specific characteristics of both the input data and the used compressor. Chunking has not been found to yield large improvements to compression ratios but remains attractive to parallelize the encoding across chunks. While we considered only one explicit chunking scheme here, we encourage benchmark participants to further explore different chunking choices and report how they affect compressor performance. For example, some chunking strategies might include artifacts at the chunk boundaries that could negatively affect the reconstruction error.

3.3 Qualitative Results

While the quantitative evaluation metrics from the previous section provide a good initial comparison of the different

compressors, we found significant qualitative differences in the behavior of the baseline compressors.

Error Distributions. Different lossy compressors generally have different pointwise error distributions, as highlighted previously in e.g. Lindstrom (2017). Figure 7 illustrates the distribution of pointwise errors for the IFS sea level pressure variable, revealing that different compressors lead to substantially different error distributions. Compressing data with SZ3, SPERR, and StochRound leads to roughly uniform error distributions within the provided error bound; this also explains why in the scorecard in Fig. 2, these compressors often produce higher mean and maximum absolute errors. A uniform error distribution theoretically increases compression ratios compared to distributions with tails (e.g. SPERR or EBCC) as compressors can exploit the full range of the provided error bounds.

EBCC, ZFP and ZFP-ROUND tend to lead to roughly normally distributed errors (Lindstrom, 2017). The distribu-

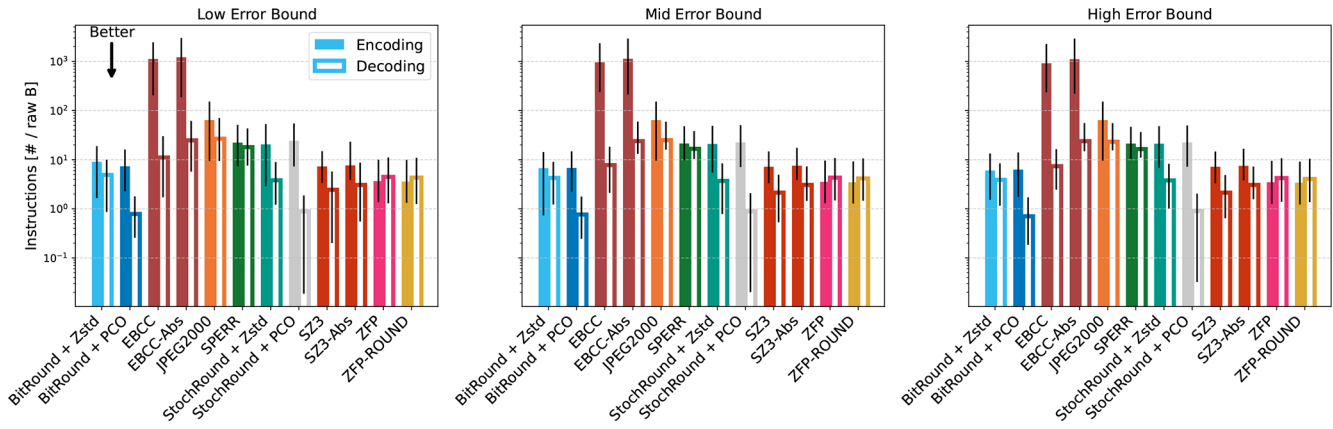


Figure 4. Comparison of encoding and decoding cost, measured in number of instructions per byte in the original data [# per raw B], for individual compressors and error bound levels. Solid bars indicate the encoding cost and hollow bars the decoding cost, bar height shows the median value across all variables and the uncertainty bars indicate the 25% and 75% quantiles. Note that the instruction count metric has no measurement noise – the variance only comes from the combination across several variables of different sizes. The instruction count in the JPEG2000 compressor currently does not include the cost of a per-element cast to float64 and uint32 during linear quantization preprocessing, which add two instructions per 4 (float32) or 8 (float64) bytes during both encoding and decoding.

	Compression Ratio									% of Pixels Exceeding Error Bound								
	10u	10v	msl	no2	pr	q	rlut	ta	tos	10u	10v	msl	no2	pr	q	rlut	ta	tos
BitRound + Zstd	-1.0e-03	-1.2e-03	-3.6e-04	-1.0e-03	1.1e-03	-2.6e-03	-1.3e-03	-0.03	-6.8e-04	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
BitRound + PCO	2.4e-05	1.8e-05	1.8e-04	-0.21	5.5e-04	-7.9e-03	1.1e-03	-0.54	-4.1e-03	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
EBCC	6.2e-05	6.6e-05	2.4e-04	424	0.12	52	8.4e-06	Fail	Fail	0.0	0.0	0.0	13	0.0	14	0.0	Fail	Fail
EBCC-Abs	6.2e-05	6.6e-05	2.4e-04	8.2e-04	2.5e-05	7.9e-04	8.4e-06	Fail	Fail	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Fail	Fail
JPEG2000	4.1e-04	4.2e-04	4.7e-03	2.0e-05	3.7e-06	8.2e-05	2.7e-05	Fail	Fail	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Fail	Fail
SPERR	1.1e-04	1.1e-04	4.9e-04	-0.18	Fail	0.02	1.4e-05	Fail	Fail	0.0	0.0	0.0	0.0	Fail	0.0	0.0	Fail	Fail
StochRound + Zstd	1.3e-04	6.6e-05	-4.6e-04	3.8e-04	2.5e-04	5.3e-05	-5.1e-04	0.10	-1.1e-03	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
StochRound + PCO	1.3e-03	1.2e-03	7.3e-04	-1.5e-03	-1.4e-05	4.6e-03	-0.13	-0.36	-5.6e-03	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
SZ3	0.02	-0.11	0.17	3170	64	160	-0.02	-1.1	0.32	0.0	0.0	0.0	5.6	1.6	39	0.0	0.0	0.0
SZ3-Abs	0.02	-0.11	0.17	0.18	-3.0e-05	Fail	-0.02	-1.1	0.32	0.0	0.0	0.0	0.0	0.0	Fail	0.0	0.0	0.0
ZFP	0.39	0.40	0.63	0.38	0.61	-0.07	1.4	-0.75	0.61	0.0	0.0	0.0	4.3e-05	6.4	1.3e-04	0.0	0.0	0.0
ZFP-ROUND	0.47	0.48	0.78	0.40	0.61	-0.07	1.5	-0.98	0.65	0.0	0.0	0.0	4.3e-05	6.4	1.1e-04	0.0	0.0	0.0

■ Unchunked Better ■ Chunked Better

Figure 5. Difference in compression ratio and percentage of pixels exceeding the error bound between the unchunked and the chunked data. For a given metric, let m_{nc} denote the score for the data that is unchunked and m_c denote the score for the data that is chunked, then the scorecard here shows the difference $d = m_{nc} - m_c$. We exclude the above-ground biomass variable because the chunking algorithm keeps the whole dataset as a single chunk and therefore, the difference in the metrics is always 0.

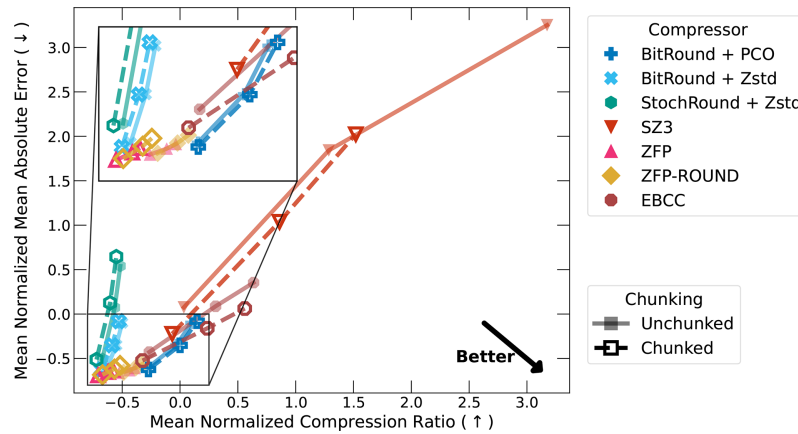


Figure 6. Impact of chunking on normalized mean absolute error and compression ratio. Hollow markers indicate results computed on chunked data and full markers with faded lines indicate results computed on unchunked data. The plot excludes StochRound + PCO, SPERR, EBCC-Abs, JPEG2000, and SZ3-Abs because for these compressors the visual difference is small. The plot with all compressors can be found in Appendix F.

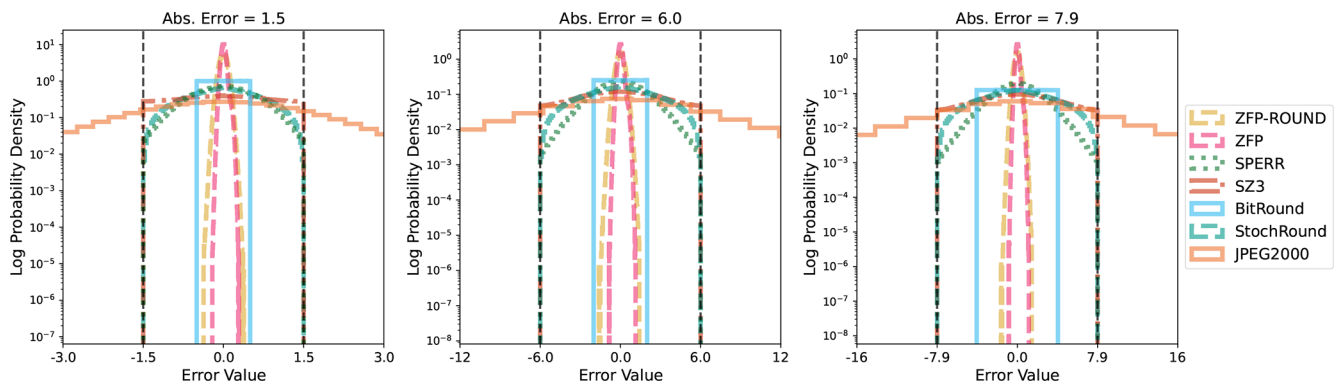


Figure 7. Error distributions for IFS sea level pressure for the three error bound levels. The black dashed vertical lines indicate the values of the error bounds. We do not differentiate between different combinations of StochRound and BitRound with lossless compressors because, by definition, the lossless compressors do not change the error distribution. Similarly, EBCC-Abs and SZ3-Abs are omitted because they behave identically to EBCC and SZ3 in this case.

tions of ZFP and ZFP-ROUND are significantly more concentrated around 0 though compared to EBCC, out of the three only EBCC exploits the full error bound range. Our implementation of BitRound relies on the conservative transformation of absolute to relative errors mentioned in Sect. 3.1 which leads to fairly conservative error distributions. The range of values in sea level pressure is less than a power of 2 (typically not exceeding 900–1100 hPa), so bounding the relative error easily bounds the absolute error too. A clear outlier is JPEG2000 because it has a much wider error distribution exceeding the bounds due to only optimizing for a global average error, as discussed before.

The heterogeneous error distributions further emphasize the need for compression users to conduct their own application-specific validation procedures, for example, by considering the errors in compressing the tails of the input data distribution or by applying statistical tests on whether

the input and output distributions of the data match. Whether one shape of error distribution is preferable to another is again very use-case specific but compression users should be aware of the differences in behavior. This also highlights the fact that rather than fixing an error bound for a given variable ahead of time, users might want to explore the compression behavior under different error bounds and consider custom error metrics.

NaN Behavior. We mentioned in Sect. 3.2 that several compressors struggle on inputs that contain NaN values. In particular, EBCC, JPEG2000 and SPERR raise errors and refuse to compress any data with NaN entries. EBCC and SPERR are the only compressors that do this natively; the JPEG2000 failure is caused by our wrapper when converting the input floating-point data to integer data (JPEG2000 was not designed to work with non-integer data). As shown in Fig. 8, the SZ3, ZFP, and ZFP-ROUND compressors actu-

ally replace some of the NaN values with finite floating-point values. In the case of SZ3, the values are smooth interpolations of the sea-surface temperatures. While the differences clearly stand out for sea surface temperature fields, this behavior might be more difficult to spot if the input fields only contain a small number of NaN values, e.g. when representing temporarily missing observation values. Similarly, ZFP infills some finite floating-point values at the edges of NaN boundaries.

Spatial Distribution of Errors. Figure 8 also demonstrates that the different compressors not only lead to different global error distributions, as shown in Fig. 7, the different compressors also have different *spatial error patterns*. Bit Rounding is implemented here to bound the relative error and therefore the magnitude of its reconstruction error will vary with the magnitude of the data, as we can clearly see in Fig. 8. The stochastic nature of Stochastic Rounding leads to noisier error distributions. As Stochastic Rounding is implemented here to bound an absolute error, the spatial distribution of its errors is uncorrelated white noise. For SZ3 and ZFP, the spatial error distribution is determined by their algorithm design. While there exist error metrics that explicitly take into account the spatial distribution of errors (Poppick et al., 2018), whether any given spatial pattern in errors is problematic is fundamentally dependent on the specific downstream analyses a user wants to conduct. It is therefore important for lossy compression users to run their own validation that is specific to their use case.

4 Conclusions

ClimateBenchPress addresses the need for a standardized benchmark of lossy compression algorithms for weather and climate data sets. The benchmark covers a diversity of different data sources (CMIP6, IFS, NextGEMS, CAMS, ESA CCI) with grid resolutions ranging from ~ 100 m to ~ 150 km. We select a small subset of variables from these data sources that cover land, ocean, and atmosphere components of the Earth system. For each variable in the benchmark, we derive quantitative error bounds with an automated algorithm that analyses the ensemble spread of each variable in the ERA5 ensemble. Our benchmark design keeps the total size of the input data at around 3 GB for easier evaluation and reproducibility, downloads to laptops, and compression without chunking on memory-constrained platforms.

Our evaluation of a set of twelve different compressor variants revealed that there are significant differences in their quantitative and qualitative performance. The benchmark uncovered edge cases that existing compressors struggle with, such as properly preserving NaN values. Bit Rounding in combination with the lossless compressor PCO generally respects error bounds and edge cases across different variables at modest compression ratios of around 10 and therefore is a robust compressor. Higher compression ratios are achieved

by SZ3, JPEG2000, EBCC, and SPERR, often reaching compression ratios of 20–100 or more, but these compressors have various failure cases, which means care needs to be taken when they are applied in practice. Specifically, SZ3 silently filled in NaN values, a bug that has now been addressed due to our feedback, and struggles with relative error bounds; JPEG2000 does not provide support for NaN values or point-wise error bounds at all; and SPERR sometimes exceed the user-provided error bound. None of the technical issues are infeasible to fix, but their existence demonstrates the need for a robust benchmark to guide the development of compressors for climate data.

We emphasize that the benchmark is not intended as an exhaustive validation suite, but rather as a tool for assessing the comparative performance of lossy compressors. It is designed to expose the trade-offs that each compression algorithm makes and to provide an estimate of the rough order of magnitude compression ratio that users can expect in practice. The benchmark does not replace data- and model-specific validations of lossy compression algorithms. For example, as shown in Sect. 3.3, different compressors exhibit qualitatively distinct behaviors, often leading to varying spatial patterns in their errors, which has also been previously highlighted in the literature (Hammerling et al., 2019). Whether these spatial patterns are significant is inherently dependent on the specific analyses that users intend to conduct on the data.

We kept the total size of the benchmark dataset small to make it accessible to a wide variety of researchers. Furthermore, while many climate and weather datasets can exceed terabytes or even petabytes in scale, most datasets of that size are stored in a chunked representation with chunk sizes rarely exceeding 100 MB. Hence, even though our total benchmark dataset is small in size it still provides a useful testing ground for compressors because the task of compressing a large dataset will often reduce to compressing many small chunks. Nevertheless, an important avenue for future work is to provide an additional benchmark that challenges users to compress the full output of a high-resolution climate model run, possibly terabytes of data, and measure the maximum achievable compression ratio and key performance metrics. Such a benchmark would complement ClimateBenchPress because the sheer size of the output data creates its own set of engineering challenges (for example, optimizing I/O, exploiting hardware accelerators and parallelism, etc.).

ClimateBenchPress is intended to serve as a guideline for compression developers designing the next generation of algorithms. It serves as an honest and balanced review of currently available compressors and highlights difficulties when used for weather and climate data in practice. Many of the compressors evaluated here are still under active development, therefore our experimental results should only be seen as a snapshot of the compressor capabilities at the time of writing this article. This is why all code and data for ClimateBenchPress are openly accessible at

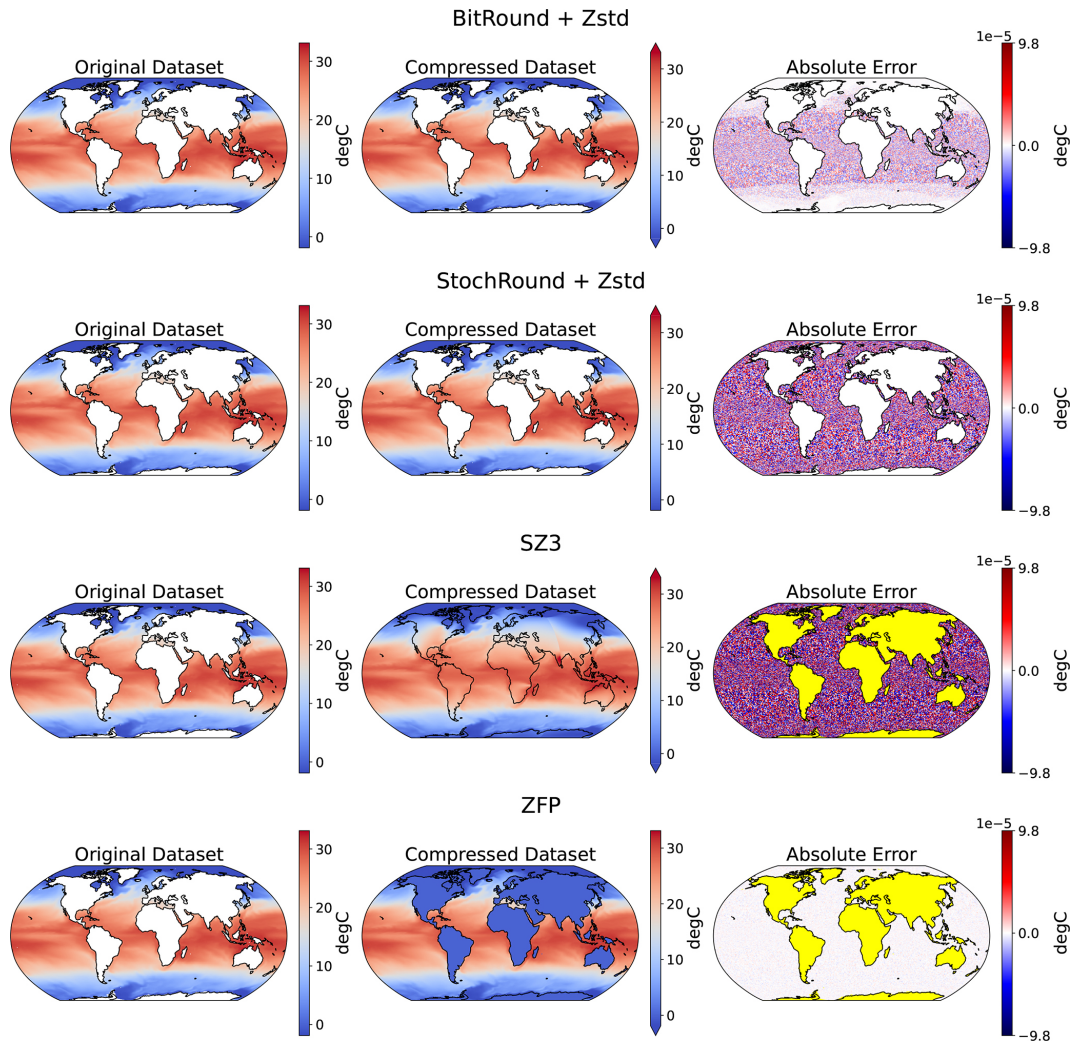


Figure 8. CMIP6 Sea Surface Temperature Errors. Original input field (left column), decompressed field (middle column), absolute error (right column) for different compressors (rows). SPERR, EBCC, and JPEG2000 are excluded here because they fail due to the presence of NaNs. We also only show one of the lossless compressors combined with rounding because, by definition, the lossless compressors do not change the error patterns. Yellow indicates regions where input values are NaNs but the compressor produces non-NaN values. Note that ZFP errors are non-zero but significantly below the error bounds; best viewed digitally.

<https://github.com/ClimateBenchPress> to enable anyone to run the benchmark whenever new compressor versions are released, and we specifically encourage compression developers to add their own compressors to the benchmark. The codebase is also easily extensible, allowing the addition of new data sources and metrics, thereby enabling the wider community to validate compression algorithms on their own datasets.

Appendix A: Sample Benchmark Data

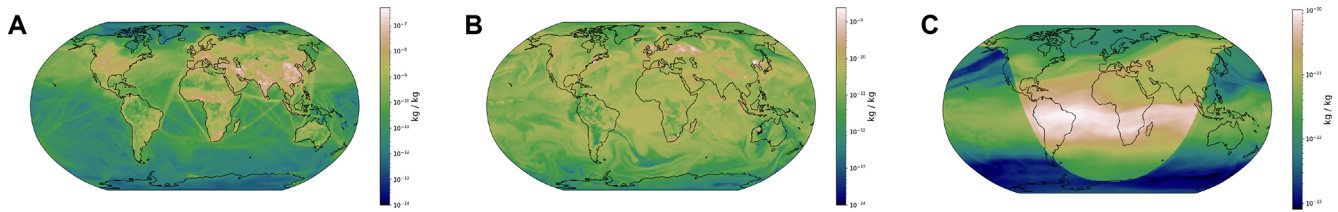


Figure A1. Nitrogen Dioxide reanalysis data from the Copernicus Atmospheric Monitoring Service (CAMS). (A) pressure level 1000 hPa; (B) pressure level 500 hPa; (C) pressure level 1 hPa. Ship tracks and near-ground point sources are visible in (A), mid-tropospheric advection and mixing in weather systems in (B) and chemical processes in the stratosphere following solar radiation in (C).

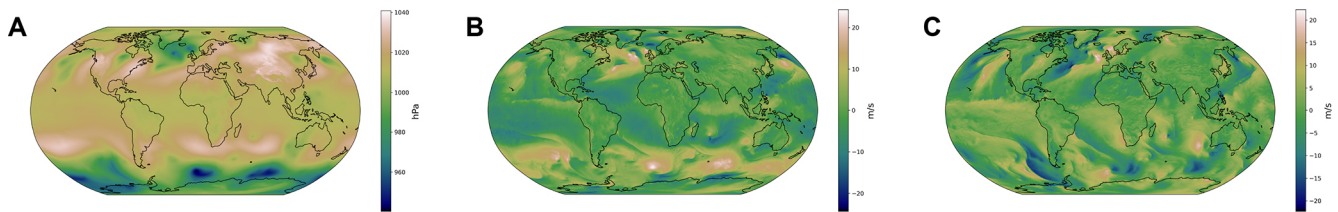


Figure A2. Variables from the IFS dataset. (A) sea-level Pressure; (B) 10 m u component of wind; (C) 10 m v component of wind.

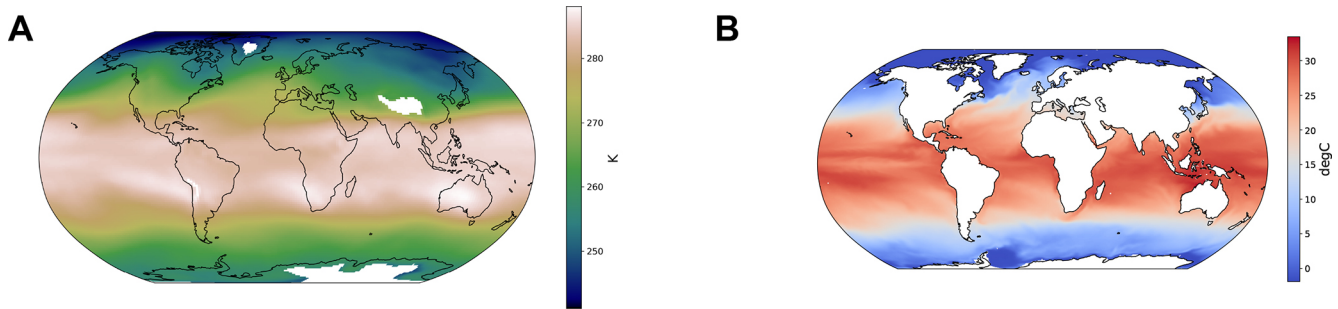


Figure A3. CMIP6 data for the ACCESS-ESM1-5 model. (A) air temperature at 700 hPa; (B) sea surface temperature.



Figure A4. Sample output fields from the ICON model in the NextGEMS intercomparison project. (A) outgoing longwave radiation; (B) precipitation.

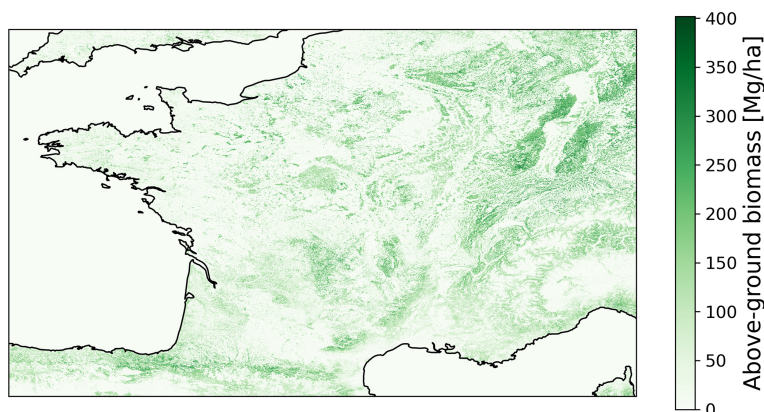


Figure A5. Sample output of the ESA above-ground biomass dataset covering France and adjacent countries.

Appendix B: Compressor Versions

Table B1. Versions of the compressors used in this benchmark. For each compressor, we provide the published version, if applicable, and the name and version of the corresponding numcodecs-wasm Python package. We use numcodecs-wasm version 0.2.2. The rounding-based compressors are widely reimplemented and no standard implementation exists.

Compressor	Version	Python package
Bit Rounding	–	numcodecs-wasm-bit-round==0.4.0
EBCC	0.1.5-alpha	numcodecs-wasm-ebcc==0.3.1a0
JPEG2000	OpenJPEG 2.5.3	numcodecs-wasm-jpeg2000==0.3.0
PCO	0.4.6	numcodecs-wasm-pco==0.3.0
SPERR	0.8.2	numcodecs-wasm-sperr==0.2.0
Stochastic Rounding	–	numcodecs-wasm-stochastic-rounding==0.2.0
SZ3	3.2.2	numcodecs-wasm-sz3==0.7.0
ZFP	1.0.1	numcodecs-wasm-zfp-classic==0.4.0
ZFP-ROUND	1.0.1	numcodecs-wasm-zfp==0.6.0
Zstd	1.5.7	numcodecs-wasm-zstd==0.4.0

Appendix C: Scorecards

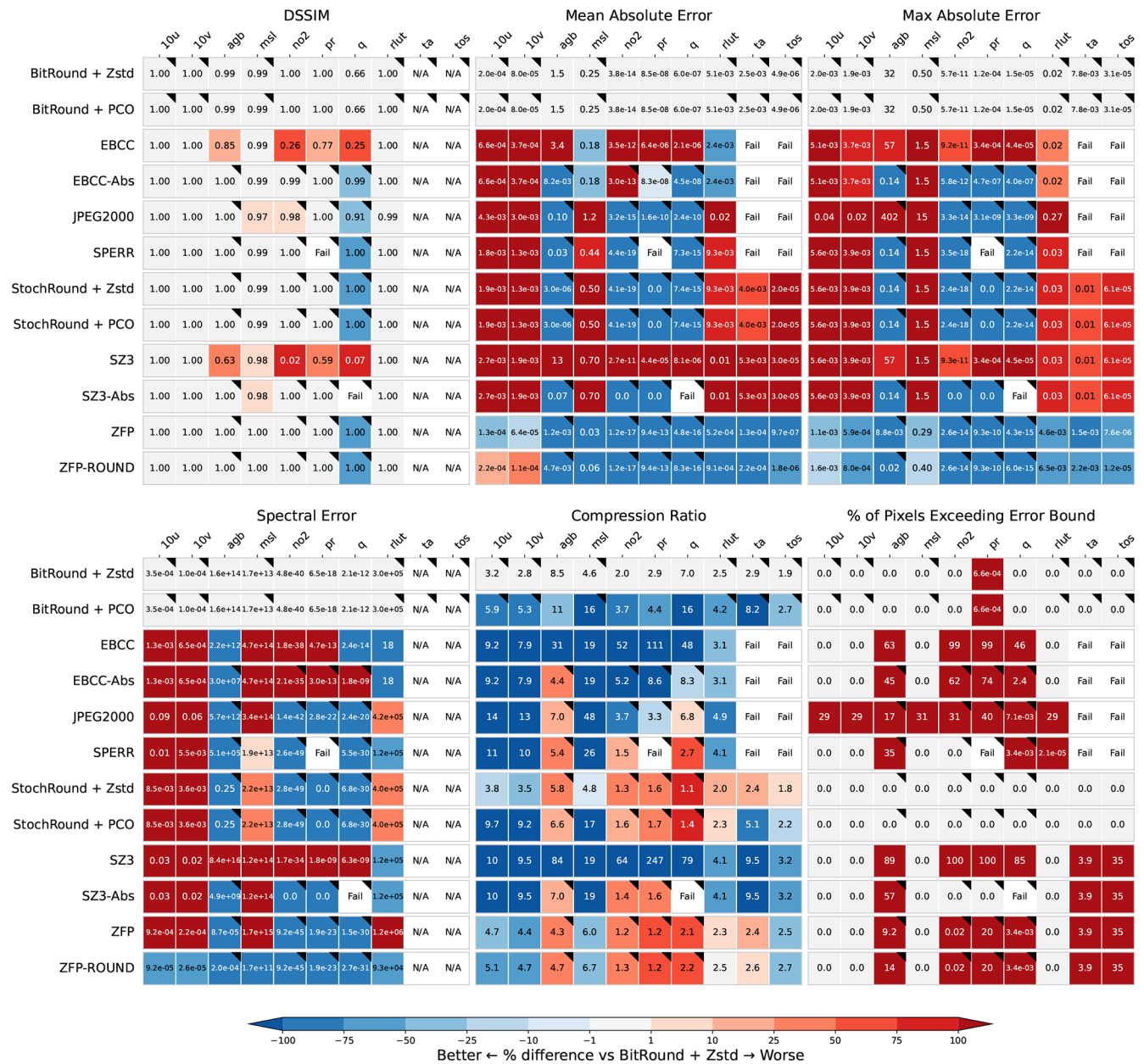


Figure C1. Scorecard for the low error bound. Figure conventions as in Fig. 2.

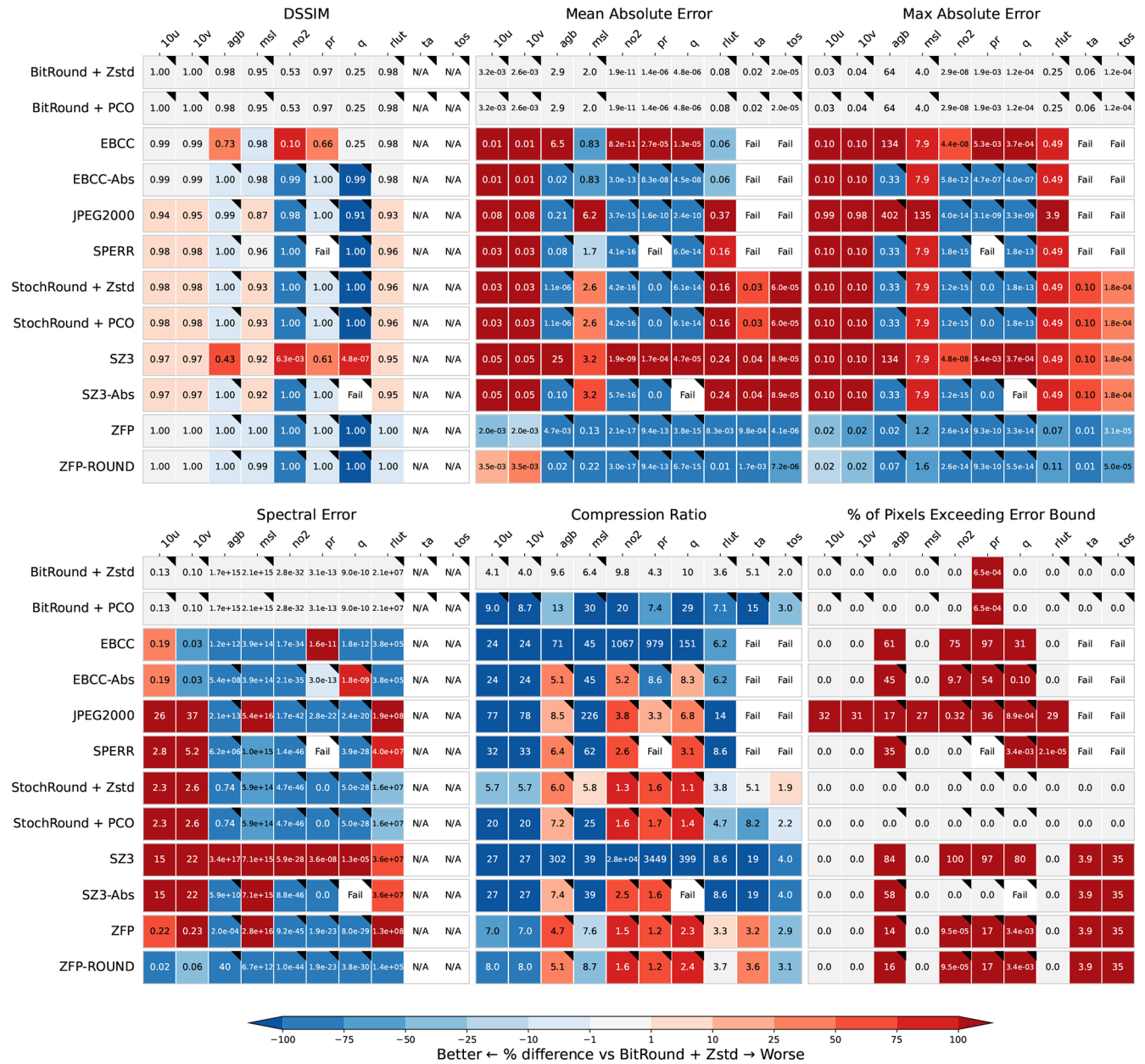


Figure C2. Scorecard for the high error bound. Figure conventions as in Fig. 2.

Appendix D: Full Rate-Distortion Plots

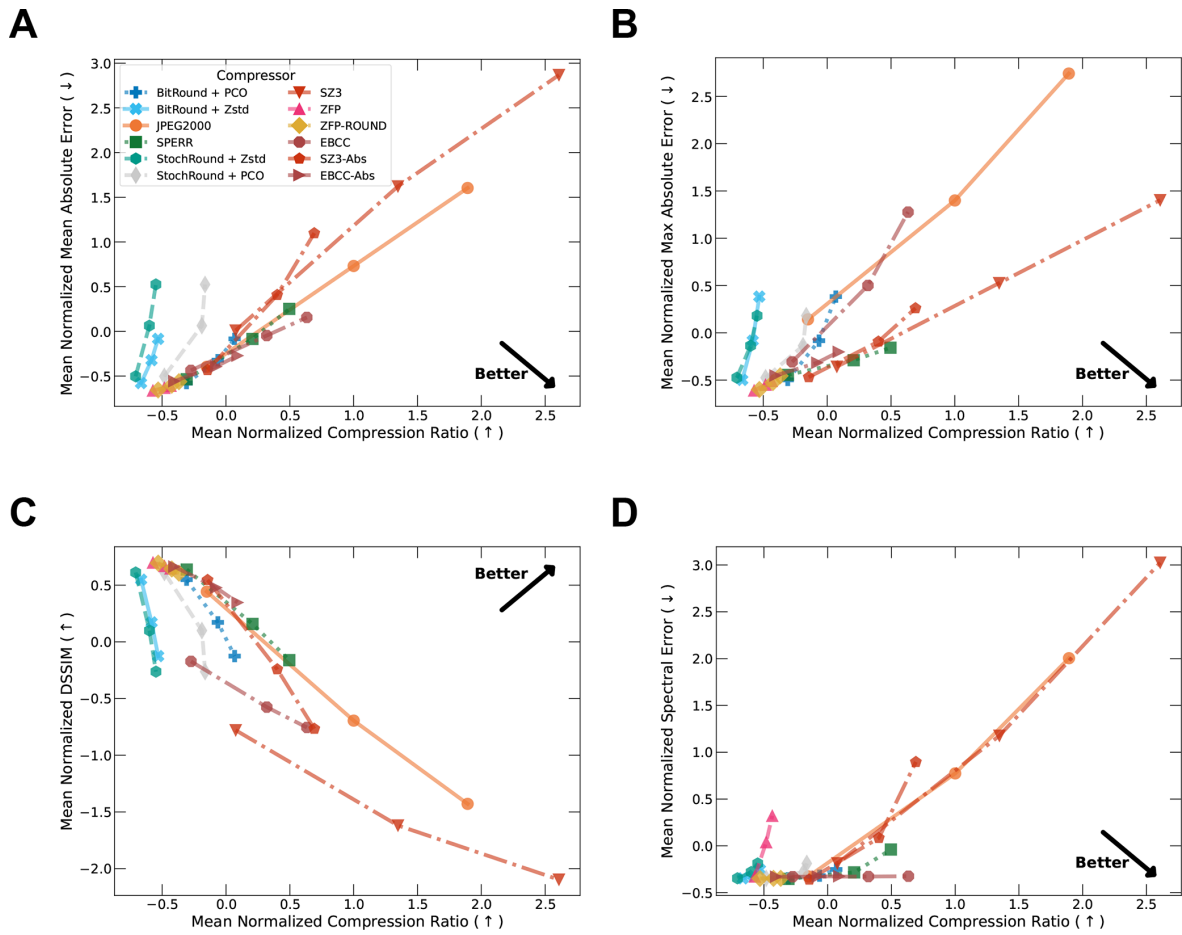


Figure D1. Full Rate-Distortion curves for multiple metrics without the cropped axes: (A) normalized mean absolute error; (B) normalized maximum absolute error; (C) normalized DSSIM; (D) normalized spectral error. Conventions as in Fig. 3.

Appendix E: Throughput Measurements

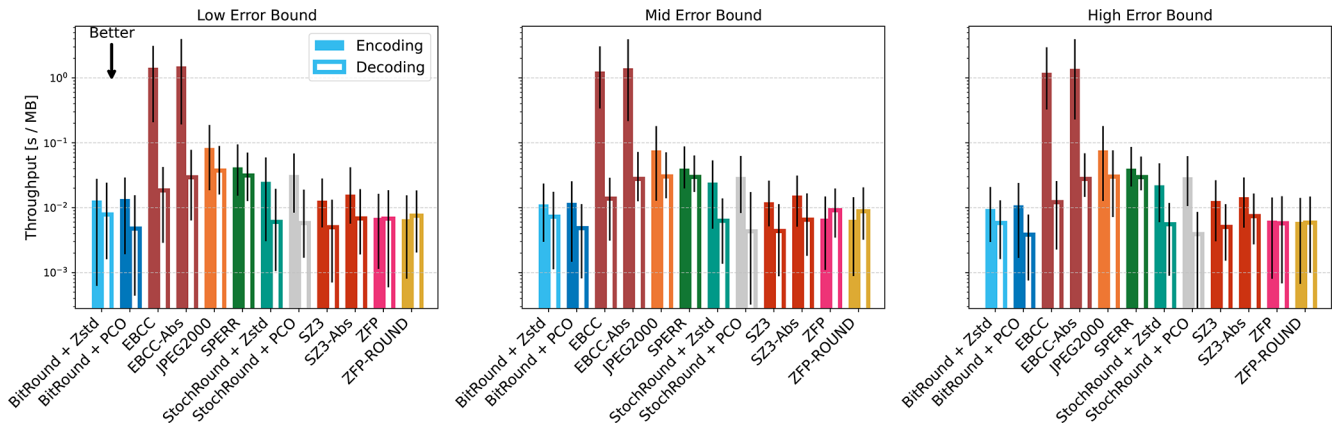


Figure E1. Comparison of encoding and decoding throughput, measured in [s MB⁻¹], for individual compressors and error bound levels. Solid bars indicate the encoding time and hollow bars the decoding time, bar height is showing the median value across all variables, and the uncertainty bars indicate the 25% and 75% quantiles. Measurements are run on a AMD EPYC 9654 CPU and utilize a single thread. Note that these throughput measurements should only be seen as a relative comparison. Our benchmarking code incurs a significant amount of overhead (Jangda et al., 2019) through the compilation to WebAssembly and to ensure reproducibility. Therefore, the numbers reported here should be seen as unoptimized, lower bound estimates of the throughput that might be achieved in practice.

Appendix F: Full Chunking Rate-Distortion Plot

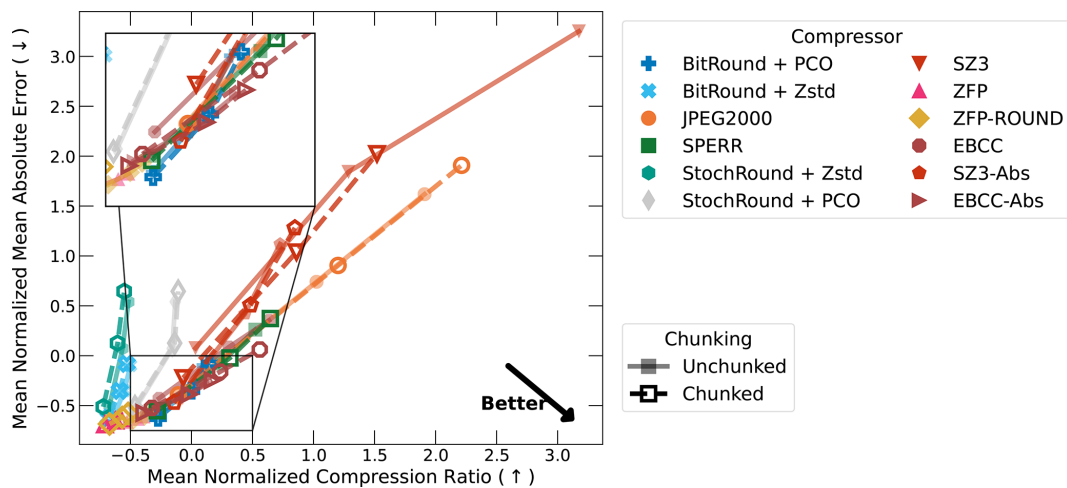


Figure F1. Impact of chunking on normalized mean absolute error and compression ratio. Hollow markers indicate results computed on chunked data and full markers with faded lines indicate results computed on unchunked data.

Appendix G: Chunk Sizes

Table G1. Chunk shapes used in the experiments from Sect. 3.2.4. Dimensions are given as time (T), longitude ($N_{(\text{lon})}$), latitude ($N_{(\text{lat})}$), vertical levels (L), and variables (V). “Chunk Size” reports the size of one chunk of a single variable under the computed chunking. “Total Size” reports the full size of the dataset. For “Chunked”, the bold numbers indicate the dimensions that have been chunked.

Variable(s)	Original Dimensions ($T \times N_{(\text{lon})} \times N_{(\text{lat})} \times L \times V$)	Chunked ($T \times N_{(\text{lon})} \times N_{(\text{lat})} \times L \times V$)	Chunk Size	Total Size
Eastward 10 m Wind, Northward 10 m Wind, Sea-level Pressure	$21 \times 1440 \times 721 \times 1 \times 3$	$7 \times 1440 \times 721 \times 1 \times \mathbf{1}$	58.14 MB	523.27 MB
Humidity*	$1 \times 1440 \times 721 \times 137 \times 1$	$1 \times 1440 \times 721 \times \mathbf{1} \times 1$	8.31 MB	1137.91 MB
Air Temperature	$12 \times 192 \times 144 \times 19 \times 1$	$\mathbf{6} \times 192 \times 144 \times \mathbf{1} \times 1$	0.66 MB	25.21 MB
Sea Surface Temperature	$12 \times 360 \times 300 \times 1 \times 1$	$\mathbf{6} \times 360 \times 300 \times 1 \times 1$	2.59 MB	5.18 MB
Precipitation, Outgoing Longwave Radiation	$8 \times 2880 \times 1440 \times 1 \times 2$	$\mathbf{2} \times 2880 \times 1440 \times 1 \times \mathbf{1}$	33.18 MB	265.42 MB
Nitrogen Dioxide	$8 \times 480 \times 241 \times 25 \times 1$	$\mathbf{4} \times 480 \times 241 \times \mathbf{5} \times 1$	9.25 MB	92.54 MB
Biomass	$1 \times 16987 \times 9900 \times 1 \times 1$	$1 \times 16987 \times 9900 \times 1 \times 1$	672.69 MB	672.69 MB

* By default Humidity uses a chunking of $1 \times 1440 \times 721 \times 69 \times 1$.

Code and data availability. All code and data is available open source at <https://github.com/ClimateBenchPress>. Frozen code versions that were used to produce the results presented in this paper are available at <https://doi.org/10.5281/zenodo.18015682> (Reichelt and Tyree, 2025), to download the datasets used in the benchmark, and <https://doi.org/10.5281/zenodo.20666484> (Reichelt and Tyree, 2026), to run and evaluate the compressors.

Author contributions. TR, JT, MK, PS led the benchmark design and conceptualization. JT developed the numcodecs-wasm compressor wrappers. TR and JT both contributed to the software development of the benchmark. TR led drafting the manuscript with input from JT, MK, PS. PD, AB, SFN, TH, BL provided feedback on the benchmark design and reviewed and edited the manuscript.

Competing interests. The contact author has declared that none of the authors has any competing interests.

Disclaimer. Publisher’s note: Copernicus Publications remains neutral with regard to jurisdictional claims made in the text, published maps, institutional affiliations, or any other geographical representation in this paper. The authors bear the ultimate responsibility for providing appropriate place names. Views expressed in the text are those of the authors and do not necessarily reflect the views of the publisher.

Acknowledgements. Juniper Tyree wishes to acknowledge CSC – IT Center for Science, Finland, for computational resources.

Financial support. Tim Reichelt and Philip Stier acknowledge funding from the EU’s Horizon Europe program under grant agreement number 10113184 and also acknowledge funding from UK Research and Innovation (UKRI). Tim Reichelt also received funding from ARIA and DSIT and Pillar VC under the Encode: AI for Science Fellowship. Juniper Tyree and Sara Faghih-Naini are funded by the ESiWACE3 Centre of Excellence, funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) under grant agreement number 101093054. Milan Klöwer acknowledges funding from the Natural Environment Research Council under grant number UKRI191. Peter Dueben acknowledges funding from the WeatherGenerator project, funded by the European Union under grant agreement No. 101187947.

Review statement. This paper was edited by Patrick Jöckel and reviewed by two anonymous referees.

References

- Baker, A. H., Hammerling, D. M., Mickelson, S. A., Xu, H., Stolpe, M. B., Naveau, P., Sanderson, B., Ebert-Uphoff, I., Samarasinghe, S., De Simone, F., Carbone, F., Gencarelli, C. N., Dennis, J. M., Kay, J. E., and Lindstrom, P.: Evaluating lossy data compression on climate simulation data within a large ensemble, *Geosci. Model Dev.*, 9, 4381–4403, <https://doi.org/10.5194/gmd-9-4381-2016>, 2016.
- Baker, A. H., Xu, H., Hammerling, D. M., Li, S., and Clyne, J. P.: Toward a Multi-method Approach: Lossy Data Compression for Climate Simulation Data, in: *High Performance Computing*, edited by: Kunkel, J. M., Yokota, R., Taufer, M., and Shalf, J., 30–42, Springer International Publishing, Cham, ISBN 978-3-319-67630-2, https://doi.org/10.1007/978-3-319-67630-2_3, 2017.
- Baker, A. H., Pinard, A., and Hammerling, D. M.: On a structural similarity index approach for floating-point data, *IEEE T. Vis. Comput. Gr.*, <https://doi.org/10.1109/TVCG.2023.3332843>, 2023.
- Ballester-Ripoll, R., Lindstrom, P., and Pajarola, R.: TTHRESH: Tensor compression for multidimensional visual data, *IEEE T. Vis. Comput. Gr.*, 26, 2891–2903, 2019.
- Bauer, P., Stevens, B., and Hazeleger, W.: A digital twin of Earth for the green transition, *Nat. Clim. Change*, 11, 80–83, 2021.
- Binder, W. and Hulaas, J.: Using Bytecode Instruction Counting as Portable CPU Consumption Metric, *Electronic Notes in Theoretical Computer Science*, 153, 57–77, <https://doi.org/10.1016/j.entcs.2005.10.032>, 2006.
- Delaunay, X., Courtois, A., and Gouillon, F.: Evaluation of lossless and lossy algorithms for the compression of scientific datasets in netCDF-4 or HDF5 files, *Geosci. Model Dev.*, 12, 4099–4113, <https://doi.org/10.5194/gmd-12-4099-2019>, 2019.
- Dey, C., Sanders, C., Clochard, J., Hennessy, J., and Elliott, S.: Guide to the WMO table driven code form used for the representation and exchange of regularly spaced data in binary form: FM 92 GRIB, World Meteorological Organization, <https://www.yumpu.com/en/document/read/11723135/guide-to-wmo-table-driven-code-forms/1> (last access: 2 July 2026), 2007.
- Eyring, V., Bony, S., Meehl, G. A., Senior, C. A., Stevens, B., Stouffer, R. J., and Taylor, K. E.: Overview of the Coupled Model Intercomparison Project Phase 6 (CMIP6) experimental design and organization, *Geosci. Model Dev.*, 9, 1937–1958, <https://doi.org/10.5194/gmd-9-1937-2016>, 2016.
- Fallin, A. and Burtscher, M.: Lessons Learned on the Path to Guaranteeing the Error Bound in Lossy Quantizers, *arXiv [preprint]*, <https://doi.org/10.48550/arXiv.2407.15037>, 2024.
- Hammerling, D. M., Baker, A. H., Pinard, A., and Lindstrom, P.: A collaborative effort to improve lossy compression methods for climate data, in: *2019 IEEE/ACM 5th International Workshop on Data Analysis and Reduction for Big Scientific Data (DRBSD-5)*, 16–22, IEEE, <https://doi.org/10.1109/DRBSD-549595.2019.00008>, 2019.
- Han, T., Chen, Z., Guo, S., Xu, W., and Bai, L.: CRA5: Extreme Compression of ERA5 for Portable Global Climate and Weather Research via an Efficient Variational Transformer, *arXiv*, <https://doi.org/10.48550/arXiv.2405.03376>, 2024.
- Hassell, D. and Cimadevilla Alvarez, E.: *cmip7repack: Repack CMIP7 netCDF-4 datasets*, Zenodo, <https://doi.org/10.5281/zenodo.17550920>, 2025.
- Hersbach, H., Bell, B., Berrisford, P., et al.: The ERA5 global reanalysis, *Q. J. Roy. Meteor. Soc.*, <https://doi.org/10.1002/qj.3803>, 2020.
- Hersbach, H., Bell, B., Berrisford, P., Biavati, G., Horányi, A., Muñoz Sabater, J., Nicolas, J., Peubey, C., Radu, R., Rozum, I., Schepers, D., Simmons, A., Soci, C., Dee, D., and Thépaut, J.-N.: ERA5 hourly data on single levels from 1940 to present, *Climate Data Store*, <https://doi.org/10.24381/cds.adbb2d47>, 2023.
- Huang, L. and Hoefler, T.: Compressing Multidimensional Weather and Climate Data into Neural Networks, *arXiv*, <https://doi.org/10.48550/arXiv.2210.12538>, 2023.
- Huang, L., Fusco, L., Scheidl, F., Zibell, J., Sprenger, M. A., Schemm, S., and Hoefler, T.: Error bounded compression for weather and climate applications, *arXiv*, <https://doi.org/10.48550/arXiv.2510.22265>, 2026.
- Ibarria, L., Lindstrom, P., Rossignac, J., and Szymczak, A.: Out-of-core compression and decompression of large n-dimensional scalar fields, in: *Computer Graphics Forum*, vol. 22, 343–348, Wiley Online Library, <https://doi.org/10.1111/1467-8659.00681>, 2003.
- Illingworth, A. J., Barker, H. W., Beljaars, A., Ceccaldi, M., Chepfer, H., Clerbaux, N., Cole, J., Delanoë, J., Domenech, C., Donovan, D. P., Fukuda, S., Hiraoka, M., Hogan, R. J., Huenerbein, A., Kollias, P., Kubota, T., Nakajima, T., Nakajima, T. Y., Nishizawa, T., Ohno, Y., Okamoto, H., Oki, R., Sato, K., Satoh, M., Shephard, M. W., Velázquez-Blázquez, A., Wandinger, U., Wehr, T., and van Zadelhoff, G.-J.: The EarthCARE satellite: The next step forward in global measurements of clouds, aerosols, precipitation, and radiation, *B. Am. Meteorol. Soc.*, 96, 1311–1332, 2015.
- Inness, A., Ades, M., Agustí-Panareda, A., Barré, J., Benedictow, A., Blechschmidt, A.-M., Dominguez, J. J., Engelen, R., Eskes, H., Flemming, J., Huijnen, V., Jones, L., Kipling, Z., Massart, S., Parrington, M., Peuch, V.-H., Razinger, M., Remy, S., Schulz, M., and Suttie, M.: The CAMS reanalysis of atmospheric composition, *Atmos. Chem. Phys.*, 19, 3515–3556, <https://doi.org/10.5194/acp-19-3515-2019>, 2019.
- Jangda, A., Powers, B., Berger, E. D., and Guha, A.: Not so fast: analyzing the performance of webassembly vs. native code, in: *Proceedings of the 2019 USENIX Conference on Usenix Annual Technical Conference*, USENIX ATC '19, 107–120, USENIX Association, USA, *arXiv*, <https://doi.org/10.48550/arXiv.1901.09056>, 2019.
- Klöwer, M., Razinger, M., Dominguez, J. J., Düben, P. D., and Palmer, T. N.: Compressing Atmospheric Data into Its Real Information Content, *Nature Computational Science*, 1, 713–724, <https://doi.org/10.1038/s43588-021-00156-2>, 2021.
- Koldunov, N., Kölling, T., Pedruzo-Bagazgoitia, X., Rackow, T., Redler, R., Sidorenko, D., Wieners, K.-H., and Ziemens, F. A.: nextGEMS: output of the model development cycle 3 simulations for ICON and IFS, WDC Climate, https://doi.org/10.26050/WDC/nextGEMS_cyc3, 2023.
- Li, S., Lindstrom, P., and Clyne, J.: Lossy scientific data compression with sperr, in: *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 1007–1017, IEEE, <https://doi.org/10.1109/IPDPS54959.2023.00104>, 2023.

- Liang, X., Di, S., Tao, D., Chen, Z., and Cappello, F.: An Efficient Transformation Scheme for Lossy Data Compression with Point-Wise Relative Error Bound, in: 2018 IEEE International Conference on Cluster Computing (CLUSTER), 179–189, <https://doi.org/10.1109/CLUSTER.2018.00036>, 2018.
- Liang, X., Zhao, K., Di, S., Li, S., Underwood, R., Gok, A. M., Tian, J., Deng, J., Calhoun, J. C., Tao, D. and Chen, Z.: Sz3: A modular framework for composing prediction-based error-bounded lossy compressors, *IEEE Transactions on Big Data*, 2022.
- Lindstrom, P.: Fixed-rate compressed floating-point arrays, *IEEE T. Vis. Comput. Gr.*, <https://doi.org/10.1109/TVCG.2014.2346458>, 2014.
- Lindstrom, P.: Error distributions of lossy floating-point compressors, Tech. rep., Lawrence Livermore National Lab. (LLNL), Livermore, CA (United States), 2017.
- Loncaric, M., Jeppesen, N., and Zinberg, B.: Pcodec: Better Compression for Numerical Sequences, *arXiv*, <https://doi.org/10.48550/arXiv.2502.06112>, 2025.
- Miles, A., Kirkham, J., Durant, M., Bourbeau, J., Onalan, T., Hamman, J., Patel, Z., shikharsg, Rocklin, M., raphael dussin, Schut, V., de Andrade, E. S., Abernathy, R., Noyes, C., sbalmer, pyup.io bot, Tran, T., Saalfeld, S., Swaney, J., Moore, J., Jevnik, J., Kelleher, J., Funke, J., Sakkis, G., Barnes, C., and Banihirwe, A.: zarr-developers/zarr-python: v2.4.0, Zenodo, <https://doi.org/10.5281/zenodo.3773450>, 2020.
- Mirowski, P., Warde-Farley, D., Rosca, M., Grimes, M. K., Hasson, Y., Kim, H., Rey, M., Osindero, S., Ravuri, S., and Mohamed, S.: Neural Compression of Atmospheric States, *arXiv*, <https://doi.org/10.48550/arXiv.2407.11666>, 2024.
- Pellicer-Valero, O. J., Aybar, C., and Valls, G. C.: Video Compression for Spatiotemporal Earth System Data, *arXiv [preprint]*, <https://doi.org/10.48550/arXiv.2506.19656>, 2025.
- Poppick, A., Nardi, J., Feldman, N., Baker, A., and Hammerling, D.: A statistical analysis of compressed climate model data, *Proc. DRBSD*, 1–6, 2018.
- Pulkkinen, S., Nerini, D., Pérez Hortal, A. A., Velasco-Forero, C., Seed, A., Germann, U., and Foresti, L.: Pys-eps: an open-source Python library for probabilistic precipitation nowcasting (v1.0), *Geosci. Model Dev.*, 12, 4185–4219, <https://doi.org/10.5194/gmd-12-4185-2019>, 2019.
- Rackow, T., Pedruzo-Bagazgoitia, X., Becker, T., Milinski, S., Sandu, I., Aguridan, R., Bechtold, P., Beyer, S., Bidlot, J., Boussetta, S., Deconinck, W., Diamantakis, M., Dueben, P., Dutra, E., Forbes, R., Ghosh, R., Goessling, H. F., Hadade, I., Hegewald, J., Jung, T., Keeley, S., Kluft, L., Koldunov, N., Koldunov, A., Kölling, T., Kousal, J., Kühnlein, C., Maciel, P., Mogensen, K., Quintino, T., Polichtchouk, I., Reuter, B., Sármany, D., Scholz, P., Sidorenko, D., Streffing, J., Sützl, B., Takasuka, D., Tietsche, S., Valentini, M., Vannière, B., Wedi, N., Zampieri, L., and Ziemann, F.: Multi-year simulations at kilometre scale with the Integrated Forecasting System coupled to FESOM2.5 and NEMOV3.4, *Geosci. Model Dev.*, 18, 33–69, <https://doi.org/10.5194/gmd-18-33-2025>, 2025.
- Rasp, S., Hoyer, S., Merose, A., Langmore, I., Battaglia, P., Russell, T., Sanchez-Gonzalez, A., Yang, V., Carver, R., Agrawal, S., Chantry, M., Ben Bouallegue, Z., Dueben, P., Bromberg, C., Sisk, J., Barrington, L., Bell, A., Sha, F.: WeatherBench 2: A benchmark for the next generation of data-driven global weather models, *J. Adv. Model. Earth Sy.*, 16, e2023MS004019, <https://doi.org/10.1029/2023MS004019>, 2024.
- Reichelt, T. and Tyree, J.: ClimateBenchPress data-loader, Zenodo [data set], <https://doi.org/10.5281/zenodo.18015682>, 2025.
- Reichelt, T. and Tyree, J.: ClimateBenchPress compressor, Zenodo [code], <https://doi.org/10.5281/zenodo.20666484>, 2026.
- Rossow, W. B. and Schiffer, R. A.: ISCCP cloud data products, *B. Am. Meteorol. Soc.*, 72, 2–20, 1991.
- Santoro, M., Cartus, O., Quegan, S., et al.: Design and performance of the Climate Change Initiative Biomass global retrieval algorithm, *Science of Remote Sensing*, 10, 100169, <https://doi.org/10.1016/j.srs.2024.100169>, 2024.
- Schmit, T. J., Griffith, P., Gunshor, M. M., Daniels, J. M., Goodman, S. J., and Lehair, W. J.: A closer look at the ABI on the GOES-R series, *B. Am. Meteorol. Soc.*, 98, 681–698, <https://doi.org/10.1175/BAMS-D-15-00230.1>, 2017.
- Segura, H., Pedruzo-Bagazgoitia, X., Weiss, P., Müller, S. K., Rackow, T., Lee, J., Dolores-Tesillos, E., Benedict, I., Aengenheyster, M., Aguridan, R., Arduini, G., Baker, A. J., Bao, J., Bastin, S., Baulenas, E., Becker, T., Beyer, S., Bockelmann, H., Brüggemann, N., Brunner, L., Cheedela, S. K., Das, S., Denissen, J., Dragaud, I., Dziekan, P., Ekblom, M., Engels, J. F., Esch, M., Forbes, R., Frauen, C., Freischem, L., García-Maroto, D., Geier, P., Gierz, P., González-Cervera, Á., Grayson, K., Griffith, M., Gutjahr, O., Haak, H., Hadade, I., Haslehner, K., ul Hasson, S., Hegewald, J., Kluft, L., Koldunov, A., Koldunov, N., Kölling, T., Koseki, S., Kosukhin, S., Kousal, J., Kuma, P., Kumar, A. U., Li, R., Maury, N., Meindl, M., Milinski, S., Mogensen, K., Niraula, B., Nowak, J., Praturi, D. S., Proske, U., Putrasahan, D., Redler, R., Santuy, D., Sármany, D., Schnur, R., Scholz, P., Sidorenko, D., Spät, D., Sützl, B., Takasuka, D., Tompkins, A., Uribe, A., Valentini, M., Veerman, M., Voigt, A., Warnau, S., Wachsmann, F., Waclawczyk, M., Wedi, N., Wieners, K.-H., Wille, J., Winkler, M., Wu, Y., Ziemann, F., Zimmermann, J., Bender, F. A.-M., Bojovic, D., Bony, S., Bordoni, S., Brehmer, P., Dengler, M., Dutra, E., Faye, S., Fischer, E., van Heerwaarden, C., Hohenegger, C., Järvinen, H., Jochum, M., Jung, T., Jungclaus, J. H., Keenlyside, N. S., Klocke, D., Konow, H., Klose, M., Malinowski, S., Martius, O., Mauritsen, T., Mellado, J. P., Mieslinger, T., Mohino, E., Pawłowska, H., Peters-von Gehlen, K., Sarré, A., Sobhani, P., Stier, P., Tuppi, L., Vidale, P. L., Sandu, I., and Stevens, B.: nextGEMS: entering the era of kilometer-scale Earth system modeling, *Geosci. Model Dev.*, 18, 7735–7761, <https://doi.org/10.5194/gmd-18-7735-2025>, 2025.
- Shannon, C. E.: A mathematical theory of communication, *Bell Syst. Tech. J.*, 27, 379–423, 1948.
- Silver, J. D. and Zender, C. S.: The compression–error trade-off for large gridded data sets, *Geosci. Model Dev.*, 10, 413–423, <https://doi.org/10.5194/gmd-10-413-2017>, 2017.
- Skodras, A., Christopoulos, C., and Ebrahimi, T.: The JPEG 2000 still image compression standard, *IEEE Signal Proc. Mag.*, 18, 36–58, 2001.
- Spies, B. and Mock, M.: An Evaluation of WebAssembly in Non-Web Environments, in: 2021 XLVII Latin American Computing Conference (CLEI), IEEE, 1–10, <https://doi.org/10.1109/CLEI53233.2021.9640153>, 2021.
- Stephens, G. L., Vane, D. G., Boain, R. J., Mace, G. G., Sassen, K., Wang, Z., Illingworth, A. J., O’Connor, E. J., Rossow, W. B., Durden, S. L., Miller, S. D., Austin, R. T., Benedetti, A.,

- Mitrescu, C., and the CloudSat Science Team: The CloudSat mission and the A-Train: A new dimension of space-based observations of clouds and precipitation, *B. Am. Meteorol. Soc.*, 83, 1771–1790, 2002.
- Stevens, B., Satoh, M., Auger, L., Biercamp, J., Bretherton, C. S., Chen, X., Düben, P., Judt, F., Khairoutdinov, M., Klocke, D., Kodama, C., Kornbluh, L., Lin, S.-J., Neumann, P., Putman, W. M., Röber, N., Shibuya, R., Vanniere, B., Vidale, P. L., Wedi, N., Zhou, L.: DYAMOND: the DYNAMICS of the Atmospheric general circulation Modeled On Non-hydrostatic Domains, *Prog. Earth Planet. Sc.*, <https://doi.org/10.1186/s40645-019-0304-z>, 2019.
- Thomas, M. and Joy, A. T.: Elements of information theory, Wiley-Interscience, <https://doi.org/10.1002/047174882X>, 2006.
- Underwood, R., Bessac, J., Di, S., and Cappello, F.: Understanding the Effects of Modern Compressors on the Community Earth Science Model, in: 2022 IEEE/ACM 8th International Workshop on Data Analysis and Reduction for Big Scientific Data (DRBSD), 1–10, IEEE, Dallas, TX, USA, ISBN 978-1-66546-337-9, <https://doi.org/10.1109/DRBSD56682.2022.00006>, 2022.
- WebAssembly Community Group: WebAssembly Specification Release 3.0 (Draft 2025-09-16), <https://webassembly.github.io/spec/versions/core/WebAssembly-3.0-draft.pdf> (last access: 5 December 2025), 2025.
- WebAssembly Working Group: WebAssembly Core Specification, <https://www.w3.org/TR/2019/REC-wasm-core-1-20191205> (last access: 23 January 2024), 2019.
- Wieners, K.-H., Rackow, T., Aguridan, R., Becker, T., Beyer, S., Cheedela, S. K., Dreier, N.-A., Engels, J. F., Esch, M., Frauen, C., Klocke, D., Kölling, T., Pedruzo-Bagazgoitia, X., Putrasahan, D., Sidorenko, D., Schnur, R., Stevens, B., and Zimmermann, J.: nextGEMS: output of the production simulations for ICON and IFS, https://www.wdc-climate.de/ui/entry?acronym=nextGEMS_prod (last access: 2 July 2026), 2024.
- Woodring, J., Mniszewski, S., Brislawn, C., DeMarle, D., and Ahrens, J.: Revisiting wavelet compression for large-scale climate data using JPEG 2000 and ensuring data precision, in: 2011 IEEE symposium on large data analysis and visualization, 31–38, IEEE, <https://doi.org/10.1109/LDAV.2011.6092314>, 2011.
- Zender, C. S.: Bit Grooming: statistically accurate precision-preserving quantization with compression, evaluated in the netCDF Operators (NCO, v4.4.8+), *Geosci. Model Dev.*, 9, 3199–3211, <https://doi.org/10.5194/gmd-9-3199-2016>, 2016.
- Zhao, K., Di, S., Liang, X., Li, S., Tao, D., Bessac, J., Chen, Z., and Cappello, F.: SDRBench: Scientific Data Reduction Benchmark for Lossy Compressors, in: 2020 IEEE International Conference on Big Data (Big Data), 2716–2724, <https://doi.org/10.1109/BigData50022.2020.9378449>, 2020.