https://doi.org/10.5194/egusphere-2026-6708
Preprint. Discussion started: 5 November 2026
© Author(s) 2026. CC BY 4.0 License.

How Much Lossy Compression of Training Data Can An Atmosphere Emulator Tolerate?

Finn O. Rebassoo[1], Peter M. Caldwell[1], Peter Lindstrom[1], Allison H. Baker[2], and Elynn Wu[3]

[1]Lawrence Livermore National Laboratory, Livermore, CA, USA
[2]NSF National Center for Atmospheric Research, Boulder, CO, USA
[3]Allen Institute for Artificial Intelligence (Ai2), Seattle, WA, USA

Correspondence: Finn O. Rebassoo (rebassoo1@llnl.gov)

Abstract. Training machine learning (ML) emulators on km-scale Earth system models (ESMs) would require more data than is practical to store. Lossy data compression is a natural solution to this problem, but adoption by the ESM community has been slow, partly because of the fear of negatively impacting scientific conclusions. In this study, we demonstrate what level of lossy compression is acceptable for an ML-based emulator of the E3SMv3 atmospheric model at 1° resolution. We specifically use the ZFP compressor, which provides state-of-the-art performance and is integrated into NetCDF4. Thus, ZFP can be applied to the E3SM data output directly and the compressed data read easily with Python tools. We measure lossy compression impact by looking at the skill of the ACE2-emulator for AMIP-like runs. We find that a compression rate of roughly 5x does not affect the physics predictions of the ACE2-emulator. As we compress the data more aggressively, some metrics start to degrade, and at a compression rate of 11x many metrics have become unacceptable.

1 Introduction

Accurate predictions of Earth system variability and change are crucial for understanding the impact on energy, infrastructure, and national security needs. Recently km-scale general circulation models (GCMs) have been developed to understand effects from processes at spatial scales smaller than traditional GCMs, which typically have a spatial resolution on the order of 100 km. These processes include deep convection, cloud formation, complex terrain effects, and intense precipitation. In parallel to the development of km-scale GCMs, machine learning (ML) emulators have recently been shown to predict decadal-to-century timescales at a huge speedup while retaining physics skill (Watt-Meyer et al., 2025; Cresswell-Clay et al., 2025; Chapman et al., 2025). However, ML-based emulators, including those in this paper, are typically trained on the coarser spatial resolutions, such as 100 km. Producing an ML-based emulator of km-scale GCMs requires an enormous volume of data that is much too large to store practically. This is because the ML-trained emulators need multiple snapshots per day to represent the diurnal cycle, and they carry 3D fields of temperature, specific total water, and wind speed. For instance, a 3-km Simple Cloud-Resolving E3SM Atmosphere Model (SCREAM) configuration has approximately 25 million columns and 128 vertical levels (Donahue et al., 2024). For a 3D variable saved at this resolution with a 6-hour timestep, the data size is approximately 20 TB per simulated year. Some reduction can be obtained since generally ML emulators use coarsened vertical levels (which itself could be seen as a form of compression). For the roughly 50 variables needed to train the ML-emulator ACE2 (Watt-Meyer et al., 2025),

https://doi.org/10.5194/egusphere-2026-6708
Preprint. Discussion started: 5 November 2026
© Author(s) 2026. CC BY 4.0 License.

50 years of data at 3 km resolution requires approximately 360 TB. However, at finer resolution compression becomes far more effective for a fixed error tolerance (Lindstrom et al., 2025), offering a path to substantially reduce this storage burden. In this paper, we examine how much lossy compression conventional-resolution emulators can tolerate, which is a necessary precursor towards providing the data needed to train a km-scale emulator for multi-year runs.

There are two categories of compression, lossless and lossy. Lossless compression reduces redundancy in the data while preserving all of the information. Conventional lossless compression has long been applied to Earth System model outputs, but typically only reduces the volume of atmospheric model data by a factor of 2x or less (Hammerling et al., 2019). To more meaningfully reduce the volume of data from km-scale GCMs, we need to compress the data more aggressively; therefore, we turn to lossy compression. Lossy compression, as the name implies, modifies the data such that the reconstructed data differs from the original, meaning data is "lost" during compression. Lossy compression is effective on numerical simulations for the same reason that lossless is not: the trailing bits are effectively random noise. The challenge with applying lossy compression is achieving a balance between data quality and data reduction, as we want to reduce the data volume as much as possible without impacting scientific results. While lossy compression has been studied for more than a decade as a means of mitigating the increasing amount of data produced by GCMs (e.g., Woodring et al. (2011); Bicer et al. (2013); Hübbe et al. (2013); Baker et al. (2014, 2016); Kuhn et al. (2016); Silver and Zender (2017)), the rise of ML emulators for GCMs has brought these problematic, massive data volumes to the forefront.

In this study, we explore how much lossy compression can be tolerated before emulator quality degrades. We train the Ai2 Climate Emulator v2 (ACE2) on output from the E3SMv3 atmosphere model, using the same dataset as Wu et al. (2025). To compress the model training data, we use ZFP (Lindstrom, 2014), a widely used compressor for scientific data that has been effectively applied to climate simulation data in previous studies (Hammerling et al., 2019; Baker et al., 2019; Poppick et al., 2020; Klöwer et al., 2021; Reichelt et al., 2026). Though a couple neural net compression algorithms have demonstrated larger compression rates than ZFP, these methods are less general as they require retraining to preserve accuracy when compressing new data with a different climate (Huang and Hoefler, 2023; Mirowski et al., 2024). We also note that other recent relevant work with training on compressed data includes applying compression to image and video (Poyser et al., 2020) as well as applying SZ (Di and Cappello, 2016; Tao et al., 2017) compression for classification tasks (Jin et al., 2021). Previous studies have looked at training an emulator on compressed data for short weather rollouts of 10 days or less (Huang and Hoefler, 2023; Han et al., 2024). However, we believe this is the first study to look at an emulator trained on compressed data for longer decadal rollouts.

2 Model Description

In this section, we describe the E3SMv3 model used to produce the training data, the ZFP algorithm used for compression, and the ML emulator (ACE2) including the training procedure.

https://doi.org/10.5194/egusphere-2026-6708
Preprint. Discussion started: 5 November 2026
© Author(s) 2026. CC BY 4.0 License.

2.1 E3SMv3

This study is based on an emulator for the atmosphere component of E3SM assuming prescribed sea surface temperature and sea ice extent. The E3SMv3 atmosphere model (Xie et al., 2025) is run on the standard ne30pg2 grid (roughly 1° horizontal resolution) with 80 vertical levels and a model top at approximately 60 km. The E3SMv3 atmosphere model has significant updates with respect to previous versions, including to gas phase chemistry, aerosols, clouds, and convection. These updates have improved the tropical variability and reduced the aerosol forcing for a more realistic simulation. The E3SMv3 training data used is an AMIP-style simulation from 1970 to 2020 and is similar to that in Wu et al. (2025). The training data is stored in single precision. In this simulation, the sea-surface temperature (SST) and sea-ice extent is time-varying from the PCMDI AMIP boundary condition dataset (Taylor et al., 2000; Hurrell et al., 2008) while the CO_2 and aerosol contributions are fixed at their 2010 level. The E3SMv3 data is regridded to a 1° lat-lon resolution before compressing.

2.2 Data compression

ZFP (Lindstrom, 2014) is a popular state-of-the-art compression method for multidimensional numerical arrays that supports both lossy and lossless compression. It works by partitioning a d-dimensional scalar array into blocks of 4^d values that are independently (de)compressed, allowing random access to compressed data at block granularity. ZFP achieves compression primarily by *decorrelating* values within a block using a near-orthogonal linear transform closely related to the *discrete cosine transform* used, for example, in JPEG image compression. The purpose of this decorrelation step is to take a dense block of similar values and sparsify it, turning large values into small coefficients that require fewer bits to represent. This step is followed by coefficient reordering and embedded coding that allows any desired compression ratio or error tolerance to be met.

In this study, we use ZFP's lossy *fixed-precision mode*, which aligns the values within a block to a single common exponent, transforms them, and then truncates the transform coefficients by keeping only a prescribed number, p, of most-significant bits, which are encoded losslessly using a variable-length code. This compression of the p most significant bits usually results in far fewer than p bits per coefficient of compressed output. Fixed-precision mode bounds compression errors to a fraction of the largest-in-magnitude value within the block, which in practice allows capping pointwise relative errors, much like in conventional floating point. Note that we use ZFP version 1.0.1 compiled with ZFP_ROUNDING_MODE=ZFP_ROUND_FIRST, wrapped by the H5Z-ZFP version 1.1.1 HDF5 filter plugin (filter plugin 32013).

Data compression with ZFP fixed-precision mode is performed after the E3SMv3 output has been processed offline to match the proper input variables for ACE2. The data compression is applied to NetCDF-4 files, and the same precision is used for every variable. For this study we used p08, p10, and p12 as the compression scenarios, where the number after the p denotes the precision of transform coefficients in number of bits. In practice, a ZFP precision of p bits translates to roughly $p-2$ bits of precision in decompressed floating-point values, i.e., $p-3$ bits of trailing significand when excluding the leading one-bit. The smaller the number the greater the compression, so for instance p08 has more compression than p10. These specific compression scenarios were chosen for emulator training based on our initial comparison of E3SMv3 data with and without a wide range of compression values. We did not believe going to a greater compression than p08 would be acceptable because

https://doi.org/10.5194/egusphere-2026-6708
Preprint. Discussion started: 5 November 2026
© Author(s) 2026. CC BY 4.0 License.

Table 1. Data size and compression for 51 years of E3SMv3 data at 100 km resolution and 6-hourly time steps. The variables compressed include four 3D variables at 8 vertical levels, and 19 2D variables. Note that pxx denotes ZFP fixed-precision mode with parameter xx. Lossless is ZFP reversible mode.

Compression	Data size	Size reduction
None	966 GB	–
Lossless	719 GB	1.3x
p12	178 GB	5.4x
p10	129 GB	7.5x
p08	87 GB	11.1x

already at p08 there were large differences in the training data (see Section 3). For the least compression, p12, there was already very good agreement with the raw E3SMv3 data, so we believed it unnecessary to consider less compression.

The reduction in storage size corresponding to the different fixed-precision compression parameters is shown in Table 1 and varies from 5x-11x, where all the variables start as floats. Table 2 shows the training time and inline inference time for each of the compression scenarios. The training with the compressed data was on average 12% faster, though there were large variations in the training time on the uncompressed data, with the runs taking 86, 99, and 106 minutes per epoch. For the uncompressed data the amount of data read is several times larger. However, even though the loader reads far fewer bytes from disk when the data is compressed, more CPU time is spent on decompression, so the large reduction in data volume translates into only a modest speed-up.

Table 2. Average training time per epoch for each compression scenario. Values are the mean over the three training runs in each scenario, with the uncertainty given as the standard deviation across those three runs. The training-only time excludes inline inference, i.e. the inference run after each training epoch, and the total time is the sum of training-only and inline inference times. Inline inference consists of sixteen 5-year rollouts, used to select the checkpoint for the 50-year standalone inferences, which run after training is complete (Sect. 2.3). Timing is calculated for 16 A100 GPUs.

Scenario	Avg. training-only time/epoch (min)	Avg. inline inference/epoch (min)	Total time/epoch (min)
No Compression	77.1 ± 8.4	19.9 ± 1.6	97.0 ± 10.0
p12	69.1 ± 0.7	16.9 ± 0.5	86.0 ± 1.2
p10	68.3 ± 0.8	17.2 ± 0.8	85.5 ± 0.7
p08	68.7 ± 1.7	16.8 ± 0.9	85.5 ± 2.5

2.3 ACE2 (ML emulator)

Machine learned emulators are crucial in order to speed up inference from GCMs. While machine-learning approaches to parameterizing components of GCMs have been explored for several years, stable autoregressive emulators capable of reproducing the behavior of full atmospheric or hybrid general circulation models over decadal to centennial timescales have only

https://doi.org/10.5194/egusphere-2026-6708
Preprint. Discussion started: 5 November 2026
© Author(s) 2026. CC BY 4.0 License.

emerged recently (Kochkov et al., 2024; Watt-Meyer et al., 2024; Cresswell-Clay et al., 2025). In this study we use ACE2, which has shown to accurately learn subseasonal to decadal atmospheric variability (Watt-Meyer et al., 2025). Both ACE and ACE2 have shown success in emulating the E3SM atmosphere; see for instance Duncan et al. (2024) and Wu et al. (2026). ACE2 is an autoregressive emulator based on a spherical fourier neural operator (SFNO) architecture. ACE2 takes forcing input and predicts 2D and 3D fields at each subsequent six hour snapshot. For this study, ACE2 was modified to allow data loading to happen on ZFP compressed data, but is otherwise unchanged. This modification to ACE2 involved installing the HDF5/netCDF/ZFP stack plus Python bindings, and the hdf5plugin package (Vincent et al., 2025). The hdf5plugin package bundles a precompiled ZFP HDF5 filter needed to read ZFP-compressed data at load time. The HDF5_PLUGIN_PATH environment variable is set to point to this filter, so no changes to ACE2's data-loading are required.

We train ACE2 on the E3SMv3 variables listed in Table 1 of Wu et al. (2025). For the training there are 6 prognostic scalar variables (input and output), two of which are 2D (skin surface temperature and surface pressure) and four of which are 3D at 8 vertical levels: temperature, specific total water, and eastward and northward winds. The training also includes ten 2D diagnostic fields such as radiative fluxes and surface variables like temperature, pressure, and heat fluxes. Lastly, the training includes five forcing variables (TOA downward shortwave, surface geopotential, and land/ocean/sea-ice fractions). In addition, TS is prescribed over ocean from the SST data. Our training protocol is exactly that as described in Section 2.1 of Wu et al. (2025), except for a slightly different inline inference that will be described below. We train on data from 1970-2020 and train three different models for each compression scenario, each differing in the random seed. We ran 50 epochs of training for each seed. We optimize losses from two 6-hourly time steps ahead, use an embedding dimension of 384, force the SSTs using historical data, and enforce global conservation of dry air mass and moisture.

In our analysis we use two kinds of inference. Inline inference is performed during training after each epoch to determine the best checkpoint (i.e. the epoch with the best performance for long rollouts). The best checkpoint cannot be determined using the training or validation loss as that is only on 12-hour rollouts and does not necessarily describe the best checkpoint for longer rollouts. The best checkpoint is the epoch with the lowest normalized root-mean-square error (RMSE) of the time mean, averaged over all fields. During inline inference the model weights are not updated. The other type of inference is standalone inference, which is performed after training is complete, using the best checkpoint determined from training. Inline inference cannot be used to determine the final metrics in this paper because they are run in parallel over 16 different five year periods from 1970-2013 to speed up the training process. In contrast, standalone inference provides a 50-year continuous run that can be used to determine climate metrics more accurately. We perform 12 different standalone inferences for each compression scenario, four initial conditions with three models with different random seeds. Compressed data is used for training and validation, whereas forcing and target data used for inference are uncompressed.

3 Results

In this section we will first look at the impact of lossy compression on E3SMv3 data before any training. Then we will look at the output of the emulator trained on the compressed data to see how well it performs on various metrics compared to an

https://doi.org/10.5194/egusphere-2026-6708
Preprint. Discussion started: 5 November 2026
© Author(s) 2026. CC BY 4.0 License.

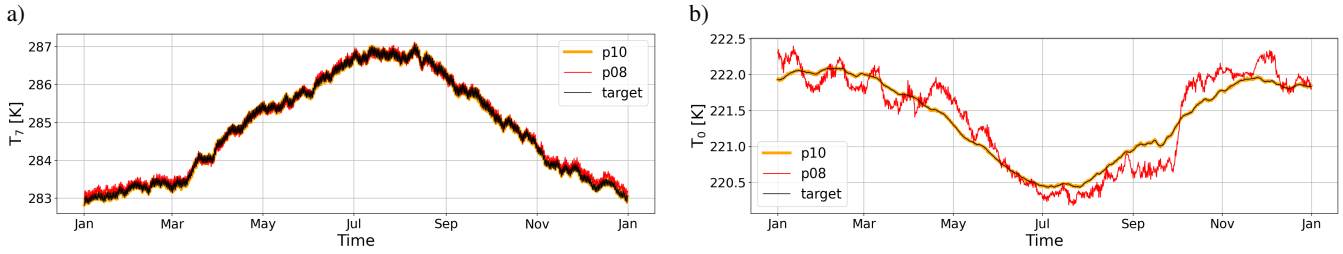


Figure 1. Global-mean temperature in 1995 for a) the lowest model level, T_7, and b) the highest model level, T_0, in Kelvin. Uncompressed E3SMv3 model output (target) is compared with ZFP p10 (orange) and p08 (red). p10 is drawn as a wider line beneath the target since there is so much overlap between the two. p12 is indistinguishable from the target (black) in both panels and is not shown.

emulator trained without compression. For these metrics, the changes due to compression should ideally be less than the spread of the emulator ensemble itself, i.e. the variability from training seeds and initial conditions. In our analysis we compare to the nominal scenario, ACE2 trained on uncompressed data, and the target, the E3SMv3 physics-based run.

3.1 Impact of compression on training data

Figure 1 shows a comparison of E3SMv3 data with and without compression. While near-surface temperature (T_7) is insensitive to compression up to p08, stratospheric temperature (T_0) cannot handle compression beyond p10. This makes it clear that certain variables can take more compression compared to others. To make this more explicit, at p12 the size reduction varies drastically based on the variable, with a minimum size reduction of 3.3x for ice fraction (ICEFRAC) and a maximum size reduction of 14.9x for the specific total water at the highest vertical layer (specific_total_water_0). This is consistent with the literature, where it has been shown that different climate variables need individualized treatment because they vary so much in terms of characteristics (Baker et al., 2014, 2016). Future work could optimize the compression settings on each variable to give the best performance. However, this optimization needs to minimize the data size while satisfying some quality metric threshold(s). What exactly those metrics are depends on the use of the emulator, but the metrics in this paper give a reasonable first starting point.

For more study of the impact of compression it is important to also look at the tails of distributions to quantify any distortion. Figure 2a) shows the effects of compression on surface temperature at all time steps and locations. For p12 the low and high tails of the distribution are described well, whereas for p10 the low tail is described better than the high tail. For p08 the discrete nature of the compression impacts the distribution significantly and causes huge disagreements even in the peak of the distribution. That discrete nature can be seen more explicitly in Figure 2b) and shows up for every compression level. As the amount of compression is increased (i.e., the precision is reduced) the gap in these discrete values of surface temperature increases.

https://doi.org/10.5194/egusphere-2026-6708
Preprint. Discussion started: 5 November 2026
© Author(s) 2026. CC BY 4.0 License.

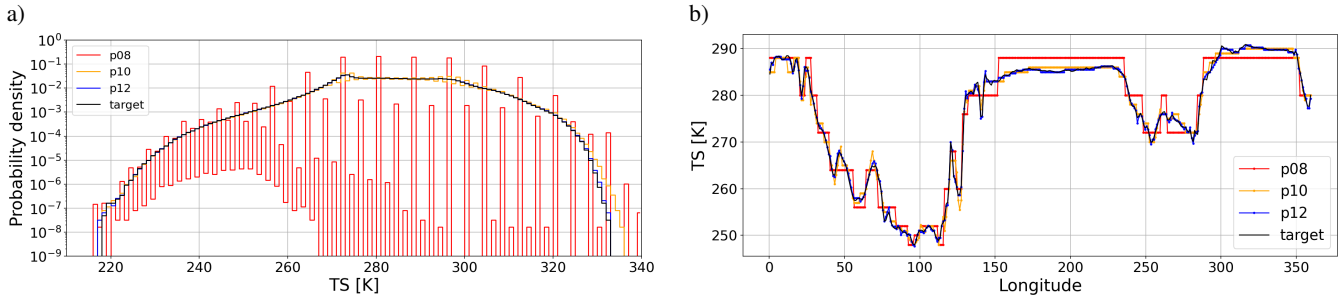


Figure 2. Surface temperature (TS) for a) the contiguous United States (CONUS) from 1970-2019 and b) a transect at 40.4°N as a function of longitude for a single timestep on January 1st, 1995. Target distributions are E3SMv3 model output with no compression.

3.2 Impact of compression on emulator

For the output of the emulator we look at many different metrics, including annual mean prediction, RMSE, El-Niño Southern Oscillation (ENSO) variability, time-averaged prediction, and rare events (tails of the distribution). Figure 3 shows the annual mean temperature at the lowest temperature level (T_7) as well as the annual mean total water path. The lowest temperature level spans about 907 to 1013 mb over the ocean with a midpoint around 960 mb. The uncertainty bands show uncertainty from different training seeds and different initial conditions. The largest uncertainty in the emulator prediction for these annual mean predictions comes from training with different random seeds. For a particular compression scenario, the spread in the R^2 (coefficient of determination) from different training seeds is far larger than the difference between compression scenarios. Overall both p12/p10 perform well compared to the target data. The emulator trained on p08 does reasonable as well though the mean prediction has less variability year to year than the target and the other compression scenarios.

Figure 4 shows the normalized RMSE over the different compression scenarios for all the ACE variables. The error here is the difference between the emulator and the target E3SMv3 run. From these plots, it is clear that the uncompressed emulator (nominal) and p12/p10 are indistinguishable when it comes to determining the best emulator. As expected for random samples from identical distributions, p10, p12, and nominal are all better and worse than each other in roughly equal proportion. The emulator trained on the most compressed data, p08, has much worse normalized RMSE than the other emulators. Part of this larger error comes from block artifacts produced by ZFP's 4×4 block structure, especially for fields that vary by only a small fraction of their magnitude within a block (for instance surface pressure). But even fields with negligible block-scale error (FLUT, LHFLX) have roughly double the RMSE of the nominal due to a shifted mean-state. The model trained on p08 is learning a different climate.

Another important metric to look at is how well the emulator describes atmospheric response to ENSO variability. To look at this we calculated the regression of various coefficients with the Niño 3.4 index, which tracks sea surface temperature anomalies in the central equatorial Pacific Ocean (specifically 5 degrees North to 5 degrees South and 170 degrees West to 120 degrees West). Figure 5 shows the global RMSE for the regression coefficients between emulators with different compression scenarios and the target data set. For both surface precipitation and longwave radiation at model top (FLUT), two variables

https://doi.org/10.5194/egusphere-2026-6708
Preprint. Discussion started: 5 November 2026
© Author(s) 2026. CC BY 4.0 License.

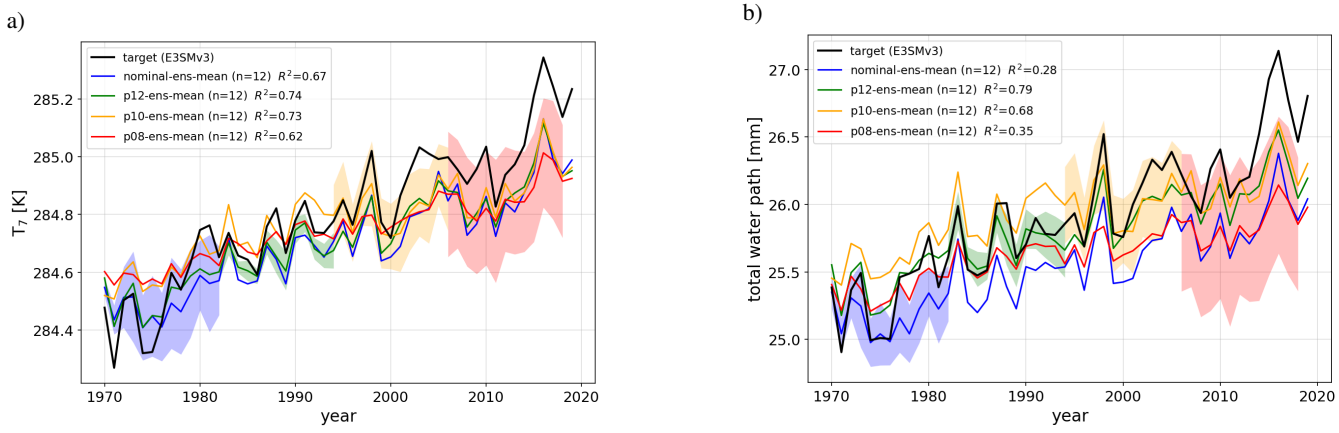


Figure 3. Global annual mean for different compression scenarios for a) T_7, the temperature at the lowest vertical layer, and b) total water path, the integrated water path per column. The uncertainty band is ± 1 standard deviation and combines two sources: the variance over training seeds (3 seeds) and the variance over initial conditions (4 ICs per training seed). The two sources are assumed to be uncorrelated and their standard deviations added in quadrature. The uncertainty varies in time (annually), but for readability the band for each compresssion scenario is shown over only 12 years to avoid overlap. The training-seed uncertainty dominates over the initial-condition uncertainty.

that are highly dependent on ENSO, there is no statistical difference between the RMSE of nominal, p12, and p10. However, the RMSE for p08 is much worse. These results are consistent across inline inferences performed during training as well as standalone inferences with 50 year rollouts (inline and standalone are described in more detail above in Sec 2.3).

To understand the performance of long-term rollouts, we look at the time-averaged global distributions for standalone inference for the different compression scenarios. Figure 6 shows the time-averaged fields for T_7 and FLUT as well as difference maps between nominal and various compression scenarios. These two variables have very different characteristics; T_7 values only vary by a factor of 1.5 over the globe while FLUT values vary by a factor of 6.5. Within a 4×4 block, the variation is on average 1.2% for T_7 and 13.5% for FLUT of the block magnitude. However, the smaller variation of T_7 with respect to the block magnitude can paradoxically cause relatively larger errors after compression than for FLUT because of the block structure for ZFP. For example, the ZFP quantization step (i.e. the discrete step shown in Figure 2) is set by the block magnitude and is similar in size for T_7 and FLUT (for p08 $\sim$8 units for blocks near 280). But because the variation across a block is much smaller for T_7 than the quantization step, at p08 the block is reconstructed as a single value 99% of the time, with an offset from the true block mean. In contrast for FLUT, the larger variations in a block are not washed out by the discrete step size and the loss of sub-block detail is the dominant error. This error changes sign from one time step to the next and thus largely averages out for a time-averaged plot.

In these plots the grid points where the difference is statistically significant at the 95% level are greyed out. For p12 roughly 5% of the grid points are greyed out in both T_7 and FLUT, which is expected by chance. However, because of spatial correlations the chance detections are not scattered randomly but appear in coherent patches. This is effectively stating that each grid point is not an independent test. For p10 there is approximately 35% greyed out grid points for T_7 and 11% for FLUT.

https://doi.org/10.5194/egusphere-2026-6708
Preprint. Discussion started: 5 November 2026
© Author(s) 2026. CC BY 4.0 License.

Lastly, for p08 there are many more grid points greyed out, especially for T_7. One feature that can be seen in the p10 and p08 difference plots for T_7 is an oscillation in the difference as a function of latitude. This is related to the discussion above about the discrete values from the ZFP compression. For cases where the differences are largest (smallest), the true field's block mean lies furthest from (closest to) one of the discrete levels in the compressed data. The emulator trained on compressed data has this periodic offset in climatology, whereas the nominal emulator, trained on uncompressed data, does not. The structure oscillates every 4 degrees in latitude consistent with a ZFP block being 4×4. This structure only really shows up as a function of latitude not longitude due to the stronger gradient in latitude. The reason this does not show up for FLUT is that the error is dominated by the loss of sub-block detail.

The last metric we looked at was whether models trained on compressed data could reproduce tails of the surface temperature over CONUS. Figure 7 shows the comparison of the target simulation (E3SMv3) to the different emulations. Remarkably, the p08 inferences do not have the discrete structure present in the compressed training data (Fig. 2). The quantization error that is introduced in the compression by only using discrete temperature values is not learned by the emulator. In addition to the discrete values being removed, the performance of p08 is quite good even in the tails. The p08 inferences do underestimate the tails at large temperature but actually perform slightly better than the nominal at the low temperature tails. In fact, for the nominal inferences the tails are slightly underrepresented, but p12 or p10 compression does not make this worse. For the low temperature tails p10 is slightly better than the nominal.

4 Discussion

In the previous section, we looked at various metrics for models trained with different compression scenarios and compared to the case of training on uncompressed data (nominal). Across all the metrics considered, p12 performs as well as the nominal. To be explicit, "performs as well" means the error due to compression is less than the error due to the emulator model, i.e. the training seed variation, and initial conditions. For instance, in Figure 5 the p08 values are not consistent within error bars with the nominal, but p12/10 are. The p10 model performs as well across all the metrics except the time-averaged mean of T_7, where roughly 30% of the noise is above noise differences with respect to the nominal. The p08 scenario performs worse across most metrics than the nominal and is clearly unacceptable. Notably, the emulator does not reproduce the discrete structure of the compressed training data so an emulator trained on compressed data gives a smoother field than the compressed data itself. The systematic part of the quantization error, the shift in each block's mean, is inherited, and it is this that produces the differences seen in the time-averaged maps.

What these results mean is that with a lossy compression factor greater than five one can retain as much performance as using uncompressed data. Thus, compressed data could start to be used for training these emulators. By compressing the data this will enable us to write out training data whenever we do a physics-based run. The logical next step is to look at higher resolution simulations to see if these results hold there, since that is where reducing storage becomes much more important. Conventional wisdom and preliminary experiments suggest that with higher resolution comes greater compression due to an increase in spatial correlation, as also borne out by the analysis and experiments in Lindstrom et al. (2025). For the training of

https://doi.org/10.5194/egusphere-2026-6708
Preprint. Discussion started: 5 November 2026
© Author(s) 2026. CC BY 4.0 License.

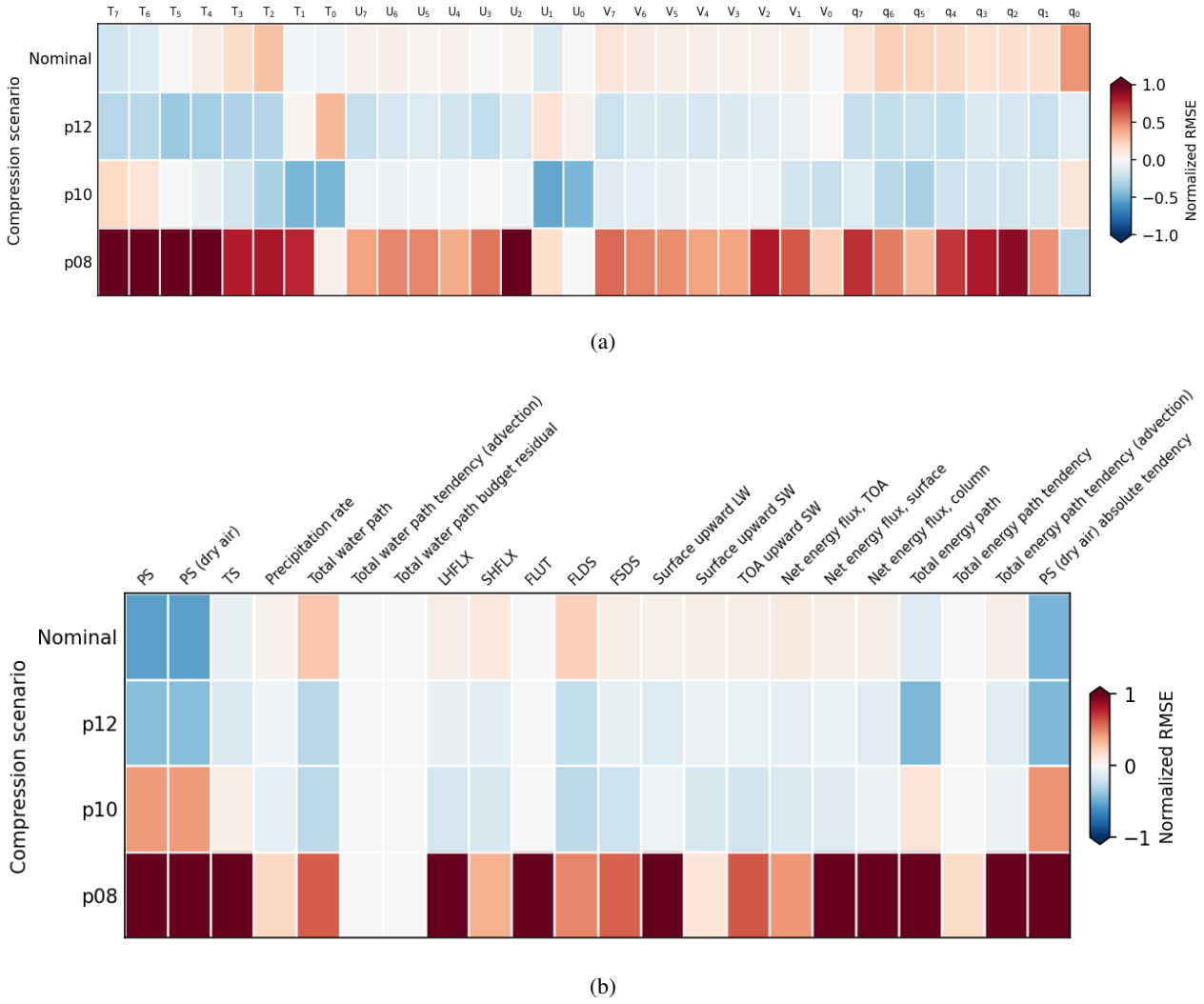


Figure 4. Normalized RMSE for 1970–2019 for different compression scenarios for a) vertically-resolved variables with the vertical level 0 at the model top and 7 at the surface (T: temperature, U and V: eastward and northward wind, q: specific total water), b) all other 2D variables in ACE. The normalized RMSE is calculated by first taking the mean RMSE for each compression scenario looping over all initial conditions and training seeds. Then this mean is compared to the median across the four compression scenarios, $NRMSE_{v,s} = \frac{E_{v,s}}{M_v} - 1$, where E is the time-mean RMSE, M is the median of E across scenarios, v is the variable and s is the compression scenario. A value of +(-) 0.2 means 20% higher (lower) than the median. The normalization is done with respect to the median because the outlier p08 values skew the mean over all compressed scenarios and make it harder to compare the other compression scenarios. The RMSE is calculated with respect to the target run.

the emulator we only compressed in latitude and longitude, not in time. However, compression in time would be interesting to

https://doi.org/10.5194/egusphere-2026-6708
Preprint. Discussion started: 5 November 2026
© Author(s) 2026. CC BY 4.0 License.

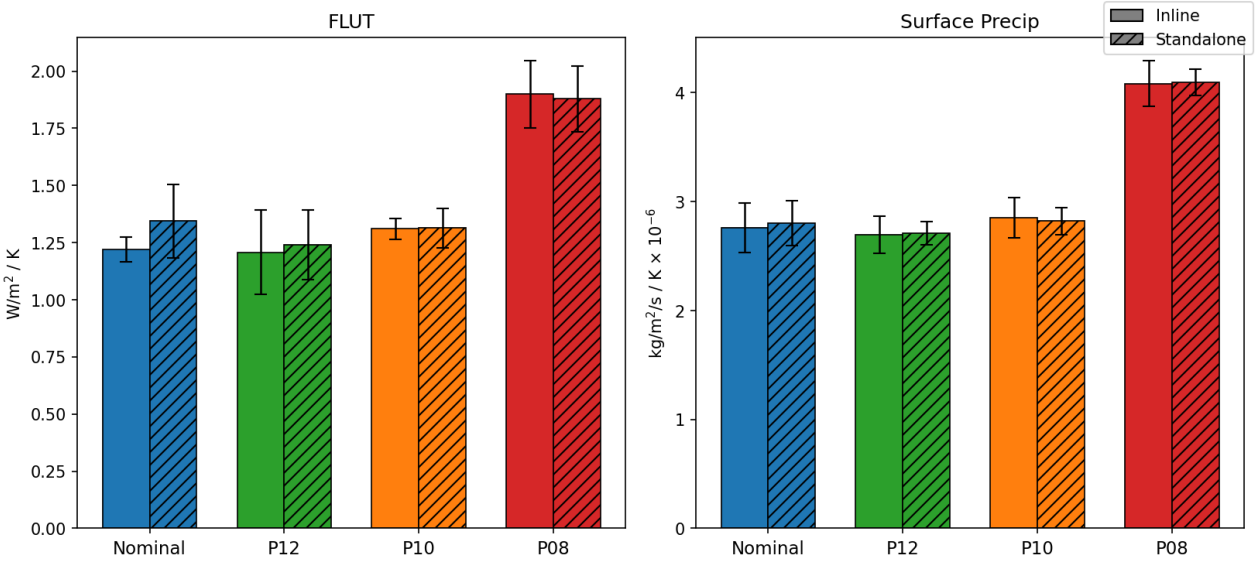


Figure 5. ENSO regression coefficients RMSE between predicted and target dataset. Regression is done against the Niño 3.4 index. For standalone inference the values are the average of twelve 50-year runs, three random seeds with four initial conditions. The error bars are the standard deviation of the twelve runs. For inline inference the 16 five-year inferences are averaged over three training seeds.

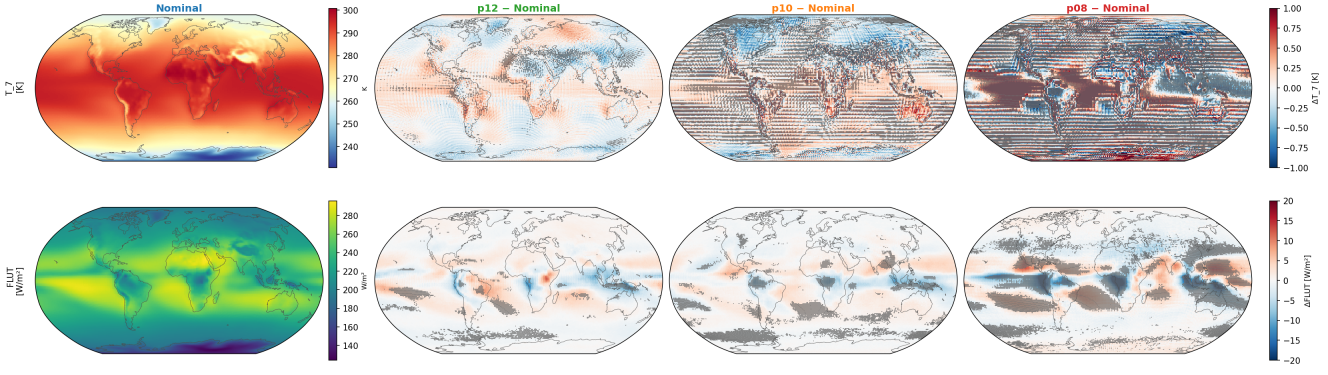


Figure 6. Time-averaged T_7 (top row) and FLUT (bottom row) over 1970–2019. FLUT is the outgoing longwave radiation at the model top. The leftmost column shows the ACE prediction with no compression, averaged over 12 runs (3 training seeds times 4 initial conditions). The other columns show the difference between compressed inferences and the nominal (no compression). Grey cells mark grid points where the difference is significant at the 95% level, tested against the spread of the three training seeds (each averaged over 4 initial conditions); ∼5% would be grey by chance.

explore for training the emulator. For the price of reading and writing output in the four-time-step blocks that ZFP compresses, compression can be increased by approximately 50% (p12) to 100% (p08).

https://doi.org/10.5194/egusphere-2026-6708
Preprint. Discussion started: 5 November 2026
© Author(s) 2026. CC BY 4.0 License.

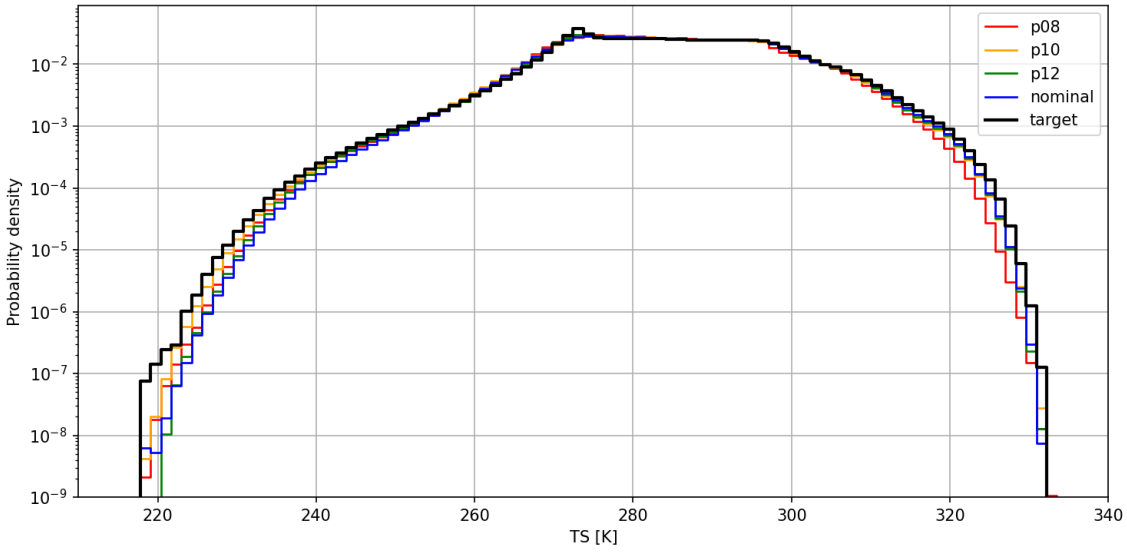


Figure 7. PDF of surface temperature (TS) for CONUS from 1970–2021 for target vs. ACE2 inferences with emulators trained on different compressed data. Each entry in the histogram comes from a single grid point at a single 6-hourly snapshot. Nominal means no compression. Each ACE2 inference plotted is the average over 12 scenarios (3 training seeds × 4 initial conditions).

5 Conclusions

In this work, we show a first study of an emulator trained on compressed data for longer decadal rollouts. Our ML-emulator is trained on ZFP-compressed E3SMv3 data without degrading the performance on metrics such as annual mean, RMSE, ENSO regression, time-averaged mean, and rare events. Our evaluation approach is not specific to ZFP and in general should carry over to other compressors. For most metrics using a compression of fixed precision 10 (p10) does not change the inference results but gives a 7.5x storage size reduction for our training data. However, the p10 results for the time-averaged mean do show some non-negligible impact on temperature. Thus, to be very safe, p12 compressed data could be used for training, which provides a factor of 5.4 in compression. The inferences with the model trained on p12 show no meaningful differences with respect to inferences with models trained on uncompressed data. Though using p08 compressed data leads to worsening of most metrics, the emulator trained on p08 does not reproduce the discrete structure present in the p08 data itself. Being able to compress the training data will reduce the storage burden and thus should increase the capability to train on more years of data. This could lead to, for instance, storing training data for every physics-based run at low resolution or enough years of training data for the emulator at high resolution.

Code and data availability. The E3SMv3 source code (v3.0.0) is available at https://github.com/E3SM-Project/E3SM and archived at 10.11578/E3SM/dc.20240301.3. The training data is the same as in Wu et al. (2026) and is publicly available via Guest Collections on Globus un-

https://doi.org/10.5194/egusphere-2026-6708
Preprint. Discussion started: 5 November 2026
© Author(s) 2026. CC BY 4.0 License.

der ACE-EAM-data (https://app.globus.org/file-manager/collections/2962fb8b-d98e-42d0-9584-9524ae0e3967/overview). This data is hosted through NERSC SHARE. The ACE2 emulator code used is commit c1008671 of https://github.com/ai2cm/ace, archived by Software Heritage at swh:1:rev:c10086710d5e8bf64b90fc070f72b3c861b84223. To be able to read compressed data the repository's Dockerfile was modified and is available along with the trained weights, configurations, and model output here: https://doi.org/10.5281/zenodo.23581685 (Rebassoo et al. (2026)). ZFP version 1.0.1 and H5Z-ZFP version 1.1.1 are available at https://github.com/LLNL/zfp and https://github.com/LLNL/H5Z-ZFP, archived by Software Heritage at swh:1:rev:f40868a6a1c190c802e7d8b5987064f044bf7812 and swh:1:rev:092190c38d5760b3212ce4d96cced767b90aa189 respectively.

Author contributions. FR ran E3SMv3 to produce the training data, compressed the data, modified ACE2 to run on compressed data, trained ACE2, and made the plots. EW helped give input for training ACE2. AB and PL gave ideas on best ways to use compression and provided comments on the compression. PC supervised experimental design and helped with interpretation of the plots. FR wrote the paper with important feedback from all authors.

Competing interests. No competing interests are present.

Acknowledgements. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Review and Release number LLNL-JRNL-2026337. This research used resources of the National Energy Research Scientific Computing Center (NERSC), a Department of Energy Office of Science User Facility. This work was partially supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, via the RAPIDS Scientific Discovery through Advanced Computing (SciDAC) program. The authors acknowledge the use of Claude Code (Anthropic) to assist with grammar correction and sentence structure improvements during manuscript preparation. In addition, most of the plotting scripts were originally written by the authors and then subsequently modified/updated by Claude Code. Figure 2 and Figure 7 were generated by Claude Code directly. For all plots the authors checked the results and also performed an independent verification. This verification removed the plots from the paper and asked Claude Code to reproduce all plots described in the paper using only the text from the paper and the location of the data. The resulting plots were consistent with the original version.

https://doi.org/10.5194/egusphere-2026-6708
Preprint. Discussion started: 5 November 2026
© Author(s) 2026. CC BY 4.0 License.

References

Baker, A., Xu, H., Dennis, J., Levy, M., Nychka, D., Mickelson, S., Edwards, J., Vertenstein, M., and Wegener, A.: A Methodology for Evaluating the Impact of Data Compression on Climate Simulation Data, in: Proceedings of the 23rd International Symposium on High-performance Parallel and Distributed Computing, HPDC '14, pp. 203–214, https://doi.org/10.1145/2600212.2600217, 2014.

Baker, A. H., Hammerling, D., Michelson, S. A., Xu, H., et al.: Evaluating lossy data compression on climate simulation data within a large ensemble, Geoscientific Model Development, 9, 2016.

Baker, A. H., Hammerling, D. M., and Turton, T. L.: Evaluating image quality measures to assess the impact of lossy data compression applied to climate simulation data, Computer Graphics Forum, 38, 517–528, https://doi.org/https://doi.org/10.1111/cgf.13707, 2019.

Bicer, T., Yin, J., Chiu, D., Agrawal, G., and Schuchardt, K.: Integrating online compression to accelerate large-scale data analytics applications, Parallel and Distributed Processing Symposium, International, pp. 1205–1216, 2013.

Chapman, W. E., Schreck, J. S., Sha, Y., Gagne II, D. J., Kimpara, D., Zanna, L., Mayer, K. J., and Berner, J.: CAMulator: Fast Emulation of the Community Atmosphere Model, arXiv preprint arXiv:2504.06007, https://doi.org/10.48550/arXiv.2504.06007, 2025.

Cresswell-Clay, N., Liu, B., Durran, D. R., Liu, Z., Espinosa, Z. I., Moreno, R. A., and Karlbauer, M.: A Deep Learning Earth System Model for Efficient Simulation of the Observed Climate, AGU Advances, 6, e2025AV001 706, https://doi.org/10.1029/2025AV001706, 2025.

Di, S. and Cappello, F.: Fast Error-Bounded Lossy HPC Data Compression with SZ, in: 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 730–739, https://doi.org/10.1109/IPDPS.2016.11, 2016.

Donahue, A. S., Caldwell, P. M., Bertagna, L., Beydoun, H., Bogenschutz, P. A., Bradley, A. M., Clevenger, T. C., Foucar, J., Golaz, C., Guba, O., Hannah, W., Hillman, B. R., Johnson, J. N., Keen, N., Lin, W., Singh, B., Sreepathi, S., Taylor, M. A., Tian, J., Terai, C. R., Ullrich, P. A., Yuan, X., and Zhang, Y.: To Exascale and Beyond—The Simple Cloud-Resolving E3SM Atmosphere Model (SCREAM), a Performance Portable Global Atmosphere Model for Cloud-Resolving Scales, Journal of Advances in Modeling Earth Systems, 16, e2024MS004 314, https://doi.org/10.1029/2024MS004314, 2024.

Duncan, J. P. C., Wu, E., Golaz, J.-C., Caldwell, P. M., Watt-Meyer, O., Clark, S. K., McGibbon, J., Dresdner, G., Kashinath, K., Bonev, B., Pritchard, M. S., and Bretherton, C. S.: Application of the AI2 Climate Emulator to E3SMv2's Global Atmosphere Model, With a Focus on Precipitation Fidelity, Journal of Geophysical Research: Machine Learning and Computation, 1, e2024JH000 136, https://doi.org/10.1029/2024JH000136, 2024.

Hammerling, D. M., Baker, A. H., Pinard, A., and Lindstrom, P.: A Collaborative Effort to Improve Lossy Compression Methods for Climate Data, in: Proceedings of the SC19 Workshop on Data Analysis and Reduction for Big Scientific Data (DRBSD-5), pp. 16–19, IEEE, Denver, CO, USA, https://doi.org/10.1109/DRBSD-51957.2019.00008, 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 preprint arXiv:2405.03376, https://doi.org/10.48550/arXiv.2405.03376, 2024.

Huang, L. and Hoefler, T.: Compressing multidimensional weather and climate data into neural networks, https://arxiv.org/abs/2210.12538, 2023.

Hübbe, N., Wegener, A., Kunkel, J. M., Ling, Y., and Ludwig, T.: Evaluating Lossy Compression on Climate Data, in: Proceedings of the International Supercomputing Conference (ISC '13), pp. 343–356, 2013.

Hurrell, J. W., Hack, J. J., Shea, D., Caron, J. M., and Rosinski, J.: A New Sea Surface Temperature and Sea Ice Boundary Dataset for the Community Atmosphere Model, Journal of Climate, 21, 5145–5153, https://doi.org/10.1175/2008JCLI2292.1, 2008.

https://doi.org/10.5194/egusphere-2026-6708
Preprint. Discussion started: 5 November 2026
© Author(s) 2026. CC BY 4.0 License.

Jin, S., Zhang, C., Jiang, X., Feng, Y., Guan, H., Li, G., Song, S. L., and Tao, D.: COMET: A Novel Memory-Efficient Deep Learning Training Framework by Using Error-Bounded Lossy Compression, CoRR, abs/2111.09562, https://arxiv.org/abs/2111.09562, 2021.

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, 2021.

Kochkov, D., Yuval, J., Langmore, I., et al.: Neural General Circulation Models for Weather and Climate, Nature, 631, 106–114, https://doi.org/10.1038/s41586-024-07744-y, 2024.

Kuhn, M., Kunkel, J., and Ludwig, T.: Data Compression for Climate Data, Supercomputing Frontiers and Innovations, 3, 75–94, https://doi.org/10.14529/jsfi160105, 2016.

Lindstrom, P.: Fixed-Rate Compressed Floating-Point Arrays, IEEE Transactions on Visualization and Computer Graphics, 20, 2674–2683, https://doi.org/10.1109/TVCG.2014.2346458, 2014.

Lindstrom, P., Hittinger, J., Diffenderfer, J., Fox, A., Osei-Kuffuor, D., and Banks, J.: ZFP: A compressed array representation for numerical computations, The International Journal of High Performance Computing Applications, 39, 104–122, https://doi.org/10.1177/10943420241284023, 2025.

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, https://arxiv.org/abs/2407.11666, 2024.

Poppick, A., Nardi, J., Feldman, N., Baker, A. H., Pinard, A., and Hammerling, D. M.: A statistical analysis of lossily compressed climate model data, Computers & Geosciences, 145, https://doi.org/https://doi.org/10.1016/j.cageo.2020.104599, 2020.

Poyser, M., Abarghouei, A. A., and Breckon, T. P.: On the Impact of Lossy Image and Video Compression on the Performance of Deep Convolutional Neural Network Architectures, CoRR, abs/2007.14314, https://arxiv.org/abs/2007.14314, 2020.

Rebassoo, F. O., Caldwell, P. M., Lindstrom, P., Baker, A. H., and Wu, E.: Trained emulators and model output for "How Much Lossy Compression of Training Data Can An Atmosphere Emulator Tolerate?", https://doi.org/10.5281/zenodo.23581685, 2026.

Reichelt, T., Tyree, J., Klöwer, M., Dueben, P., Lawrence, B. N., Baker, A. H., Faghih-Naini, S., Hoefler, T., and Stier, P.: ClimateBenchPress (v1.0): a benchmark for lossy compression of climate data, Geoscientific Model Development, 19, 5933–5960, https://doi.org/10.5194/gmd-19-5933-2026, 2026.

Silver, J. D. and Zender, C. S.: The compression–error trade-off for large gridded data sets, Geoscientific Model Development, 10, 413–423, https://doi.org/10.5194/gmd-10-413-2017, 2017.

Tao, D., Di, S., Chen, Z., and Cappello, F.: Significantly Improving Lossy Compression for Scientific Data Sets Based on Multidimensional Prediction and Error-Controlled Quantization, in: 2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 1129–1139, https://doi.org/10.1109/IPDPS.2017.155, 2017.

Taylor, K. E., Williamson, D., and Zwiers, F.: The Sea Surface Temperature and Sea Ice Concentration Boundary Conditions for AMIP II Simulations, Tech. Rep. PCMDI Report 60, Program for Climate Model Diagnosis and Intercomparison, Lawrence Livermore National Laboratory, Livermore, CA, USA, https://pcmdi.llnl.gov/report/pdf/60.pdf, 2000.

Vincent, T., Solé, V. A., Kieffer, J., Tintó Prims, O., Kittisopikul, M., et al.: silx-kit/hdf5plugin: 5.1.0, https://doi.org/10.5281/zenodo.15111354, 2025.

Watt-Meyer, O., Dresdner, G., McGibbon, J., et al.: ACE: A Fast, Skillful Learned Global Atmospheric Model for Climate Prediction, arXiv preprint arXiv:2310.02074, 2024.

https://doi.org/10.5194/egusphere-2026-6708
Preprint. Discussion started: 5 November 2026
© Author(s) 2026. CC BY 4.0 License.



Watt-Meyer, O., Henn, B., McGibbon, J., Clark, S. K., Kwa, A., Perkins, W. A., Wu, E., Harris, L., and Bretherton, C. S.: ACE2: accurately learning subseasonal to decadal atmospheric variability and forced responses, npj Climate and Atmospheric Science, 8, 205, https://doi.org/10.1038/s41612-025-01090-0, 2025.

Woodring, J., Mniszewski, S. M., Brislawn, C. M., DeMarle, D. E., and Ahrens, J. P.: Revisting Wavelet Compression for Large-Scale Climate Data Using JPEG2000 and Ensuring Data Precision, in: IEEE Symposium on Large Data Analysis and Visualization (LDAV), edited by Rogers, D. and Silva, C. T., pp. 31–38, IEEE, 2011.

Wu, E., Rebassoo, F., Paul, P., Proistosescu, C., Nugent, J., McCoy, D., Caldwell, P., and Bretherton, C. S.: Applying the ACE2 Emulator to SST Green's Functions for the E3SMv3 Global Atmosphere Model, Journal of Geophysical Research: Machine Learning and Computation, 2, e2025JH000 774, https://doi.org/https://doi.org/10.1029/2025JH000774, e2025JH000774 2025JH000774, 2025.

Wu, E., Duncan, J. P. C., Arcomano, T., McGibbon, J., Watt-Meyer, O., Bretherton, C. S., Mahfouz, N., Tebaldi, C., Van Roekel, L., Roberts, A., Lin, W., Rebassoo, F., Golaz, J.-C., and Caldwell, P. M.: Stochastic Emulation of a Fully Coupled Preindustrial E3SMv3 Simulation, arXiv preprint arXiv:2608.10277, https://doi.org/10.48550/arXiv.2608.10277, 2026.

Xie, S., Terai, C. R., Wang, H., Tang, Q., Fan, J., Burrows, S., Lin, W., Wu, M., Song, X., Zhang, Y., Taylor, M. A., Golaz, J.-C., Benedict, J. J., Chen, C.-C.-J., Feng, Y., Hannah, W. M., Ke, Z., Shan, Y., Larson, V. E., Liu, X., Prather, M. J., Richter, J. H., Shrivastava, M., Wan, H., Zhang, G. J., Zhang, K., Bradley, A. M., Cameron-Smith, P., Damiano, L., Debusschere, B. J., Donahue, A. S., Easter, R. C., Eldred, M. S., Griffin, B. M., Guba, O., Guo, Z., Huang, X., Lee, J., Lee, H.-H., Lou, S., Mahfouz, N., Moncrieff, M., Mülmenstädt, J., Qian, Y., Rasool, Q. Z., Roberts, A. F., Santos, S. P., Sargsyan, K., Shpund, J., Singh, B., Tao, C., Xie, J., Yang, Y., Zeng, X., Zhang, C., Zhang, M., Zhang, S., Zhang, T., Zheng, X., Jacob, R. L., Leung, L. R., McCoy, R. B., and Bader, D. C.: The Energy Exascale Earth System Model Version 3: 1. Overview of the Atmospheric Component, Journal of Advances in Modeling Earth Systems, 17, e2025MS005 120, https://doi.org/https://doi.org/10.1029/2025MS005120, e2025MS005120 2025MS005120, 2025.