On a structural similarity index approach for floating-point data

Allison H. Baker, Alexander Pinard, and Dorit M. Hammerling

Abstract—Data visualization is typically a critical component of post-processing analysis workflows for floating-point output data from large simulation codes, such as global climate models. For example, images are often created from the raw data as a means for evaluation against a reference dataset or image. While the popular Structural Similarity Index Measure (SSIM) is a useful tool for such image comparisons, generating large numbers of images can be costly when simulation data volumes are substantial. In fact, computational cost considerations motivated our development of an alternative to the SSIM, which we refer to as the Data SSIM (DSSIM). The DSSIM is conceptually similar to the SSIM, but can be applied directly to the floating-point data as a means of assessing data quality. We present the DSSIM in the context of quantifying differences due to lossy compression on large volumes of simulation data from a popular climate model. Bypassing image creation results in a sizeable performance gain for this case study. In addition, we show that the DSSIM is useful in terms of avoiding plot-specific (but data-independent) choices that can affect the SSIM. While our work is motivated by and evaluated with climate model output data, the DSSIM may prove useful for other applications involving large volumes of simulation data.

Index Terms—structural similarity index, floating-point data, compression, climate simulation data

1 Introduction

Given the advances in high-performance computing (HPC) in recent years, model simulation codes are advancing scientific discovery across many disciplines. Visualizations of simulation output data are important to domain scientists as they typically provide the primary means of exploring the output data. A typical post-processing data workflow often includes evaluating the differences between an image generated from the data and a reference image. Image quality assessment (IQA) measures are designed for this use-case, particularly the so-called "full reference" (FR) IQA measures that require the original (or reference) image for the comparison (e.g., [32], [40]). The two images may differ due to differences in the datasets from which they were generated: model simulation data versus experimental or observational data; two simulation datasets generated with slightly different model parameters or initial conditions; or the increasingly common scenario where one image's data have been subjected to lossy compression. Regardless of the source of the discrepancy between images, IQA measures provide an objective way of quantifying that difference. Perhaps the most well-known and commonly used IQA measure is the Structural Similarity Index (SSIM) [30], [35], though a number of such image-comparison measures have been developed (e.g., [18], [27], [34], [38], [39]).

Because lossy compression remains a powerful tool in reducing the enormous volumes of simulation data produced with modern HPC machines, we are particularly interested in the use-case in which IQAs help to discover and quantify image artifacts due to lossy compression. Recall that in contrast to lossless compression, applying lossy compression to a dataset prohibits exact reconstruction of the original data. The SSIM is useful in this context as an objective means of assessing the effects of lossy compression. For example, in the medical imaging field, images are typically compressed to reduce unmanageable data volumes, but clearly the potential loss of critical information, such as details needed to make an accurate diagnosis, is a concern. Concordantly, the SSIM has been advocated as a means of evaluating compressed-medical-image quality in many studies (e.g., see [7], [8], [16], [25], [36]).

The application area of particular interest to us is climate modelling, where simulations are well-known for producing enormous amounts of output data (e.g., terabytes or even petabytes). For a number of years, lossy data compression has been proposed as a means of mitigating the big data problem in climate research (e.g., [3], [11], [17], [37]), though its acceptance in the climate community is far from secured as more comprehensive measurements for evaluating the loss of information are still needed. Given the importance of data visualization to climate scientists interacting with model output, an objective means of assessing whether images generated from the compressed model data are noticeably different from images based on the original model data is critical. Therefore, as part of an effort to persuade climate scientists to adopt lossy compression, we included the SSIM in a suite of measures to evaluate the "quality" of compressed climate simulation data [5]. In a follow-up work [4], we proposed a minimum threshold for SSIM values to indicate when differences could be seen when comparing images. This threshold was based on a forced-choice visual evaluation study in which participants indicated whether a visual difference could be seen, with respect to the reference image that was created from uncompressed data. Note that evaluating the impact of lossy compression is a non-trivial task and depends on a number of factors such as the characteristics of the data, the type of compressor algorithm, and the intended scientific analysis of the data. Therefore, multiple evaluation approaches are typically necessary to instill confidence

- *A.H. Baker is with the National Center for Atmospheric Research. E-mail: abaker@ucar.edu*
- *A. Pinard and D.M. Hammerling are with Colorado School of Mines. E-mail: apinard@ucar.edu, hammerling@mines.edu*

in the compressed dataset. Here, we do not analyze different methods of detecting compressor artifacts, as that has been done previously (e.g., [3], [5], [24]), but rather acknowledge that the SSIM is beneficial in this context and focus on how to reduce its cost.

While the SSIM is undoubtedly useful for objectively comparing images, several shortcomings arise in the context of its use in our compression-related research on large volumes of data. While we address those issues in this manuscript in the context of climate model data, the findings may be applicable to other application areas that rely on large volumes of floating-point simulation data. First, because the SSIM calculation is based on two corresponding grids of pixel values (i.e. a reference and a modified image), rendering images from the dataset values to be compared is required before the SSIM can be computed. Generating many images, for example, from a long time series of climate data, potentially at high spatial resolution, can be quite computationally intensive. Indeed, the computational cost of generating the images from the climate model output required for the SSIM makes the SSIM a much more expensive measure of lossily compressed data quality than other data comparison measures that only require the floating-point data for identifying problematic compressor artifacts. However, we are loath to abandon the SSIM due to cost considerations given its popularity and documented usefulness in image quality assessment [34] as well as our own positive experiences with it [4], [5].

A second, albeit more minor, motivation is that the SSIM value is naturally dependent on plot parameters such as color scheme or geometric transformation (or other decisions that are not based on information contained in the data) that a scientist may make when creating a plot. As a result, for a particular floating-point dataset, two images that are considered indistinguishable based on their SSIM value given one set of plot parameters, may become distinguishable (again, based on the SSIM) if the plot parameters are changed. A simple example of a plot setting for climate data that affects the SSIM is the global map projection. However, for our use case we often do not know how an image will be generated from data, but we can make reasonable assumptions that will allow a data similarity measure to indicate whether an image created from a compressed dataset is likely to be distinguishable from the original.

Hence, the cost of generating images from the raw data as well as the possible dependence on plotting choices motivates applying the SSIM directly to the climate model's floating-point dataset values, rather than to the pixel values. Unfortunately, simply applying the standard SSIM formula without modification to floating-point data, rather than pixel values of images created from the data, does not always result in desired behavior. However, by making a few relatively simple but critical modifications, which we collectively refer to as the Data SSIM (DSSIM), we obtain a useful measure to apply directly to floating-point datasets that is conceptually similar to the SSIM and discriminates between the differences in our test datasets. In this paper, we make the following contributions:

- demonstrate the effect of basic data modifications and image generation choices on the computed SSIM value to improve our understanding of SSIM value ranges and dependencies;
- present a SSIM-like statistic that can be applied directly to floating-point data, thus avoiding the computational expense of rendering otherwise unnecessary images;
- and provide an in-depth evaluation that illustrates the method's utility in evaluating the effects of data compression on large volumes of climate data, particularly in terms of computational cost reduction.

The remainder of this paper is organized as follows. In Section 2, we review the SSIM and demonstrate its dependence on plot and parameter choices. Next, in Section 3, we discuss considerations for floating-point data and introduce the DSSIM approach. In Section 4, we discuss the applicability of the DSSIM in the context of evaluating lossy compression on climate data. We provide concluding remarks in Section 5.

2 Structural Similarity Index (SSIM)

As previously noted, full reference IQAs are a popular means for comparing two images, where one image is typically the reference image, against which the quality of the second image is being compared. The IQA value is intended to be an objective measurement of the more subjective concept of how noticeably different the two images are, say to a human observer. While the SSIM was developed to compare the encoding of natural images, we found in previous work [4] that the SSIM showed good predictive ability to gauge when experts perceive differences in images generated from climate model simulation data. In fact, while a number of other IQAs showed good predictive ability, the SSIM IQA measure performed the best. It is important to note that the plots of most interest to the climate community in diagnostic packages, for example, are typically those that smoothly map the floating-point data to RGB values. In other words, pseudocolor plots (or possibly a filled contour - depending on the transfer function) that use a smooth colormap are suitable for comparison with the SSIM. We do not consider scatter plots, data plots with a lot of white space, glyph-based techniques, or, more generally, plots for which a small change in the data causes a large or abrupt change in the image. Also note that the SSIM can be sensitive to accessories on the plot such as plot grids, labels, etc. [31], which should be removed for comparison purposes.

Figure 1 contains an example of the type of pseudocolor plots that climate scientists typically create for a commonly used climate variable. Both plots in the figure show a single time slice of surface temperature (TS) data; the top plot contains the original (not compressed) data and the bottom plot contains data that has been aggressively (lossily) compressed such that artifacts are clearly visible. The application dataset from which we obtained these data and the compressor are described in Section 4.

2.1 Method overview

The SSIM enjoys widespread use across a number of disciplines. While some recent works cast doubt on the popular notion that the SSIM truly represents human visual perception (e.g., [6], [21]), it nevertheless remains very popular in practice, due in large part to its simplicity and usefulness as a statistical measure [30]. The SSIM is the product of three factors that are intended to represent luminance, contrast, and structure. Consider comparing two 2D images $\mathbf{X}$ and $\mathbf{Y}$, each of dimension $m_x \times m_y$ with $M = m_x m_y$ pixel values. The SSIM is computed by first calculating so-called per-pixel SSIM values comparing local patches (or windows) of the images. Let $\mathbf{x}_i$ and $\mathbf{y}_i$ be local image patches (i.e., 2D arrays that have been flattened) taken from the same location in $\mathbf{X}$ and $\mathbf{Y}$,

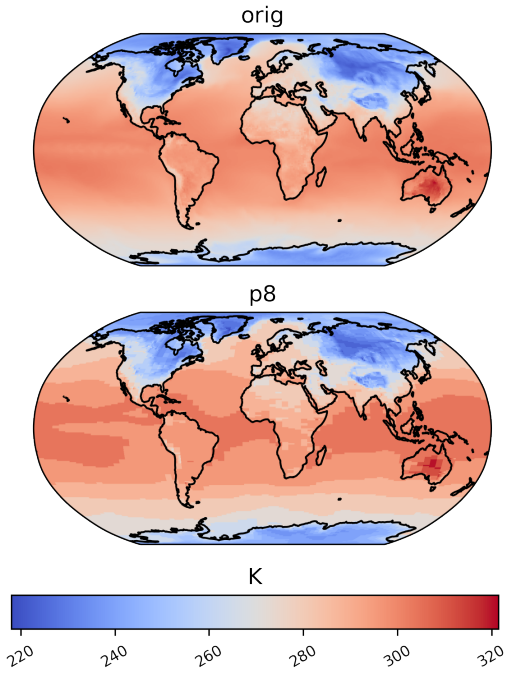


Fig. 1: The plots for surface temperature (TS) data for the original data (top) and the lossily compressed data (bottom) with compressor ZFP and p=8. (Compressor details are discussed in Section 4.1.)

respectively. Subscript i indicates the pixel index in $\mathbf{X}$ and $\mathbf{Y}$ that is at the center of the local window ($i \leq M$). And, let $N = n^2$ be the number of pixels in the local window. Then, in the local window centered at pixel i with $\mathbf{x}_i$ and $\mathbf{y}_i$ containing N pixel values, the per-pixel (i.e., "local") SSIM value is

$$SSIM(\mathbf{x}_i, \mathbf{y}_i) = (l(\mathbf{x}_i, \mathbf{y}_i))^{\alpha} \cdot (c(\mathbf{x}_i, \mathbf{y}_i))^{\beta} \cdot (s(\mathbf{x}_i, \mathbf{y}_i))^{\gamma}, \quad (1)$$

where $l(\mathbf{x}_i, \mathbf{y}_i)$ is the luminance term, $c(\mathbf{x}_i, \mathbf{y}_i)$ is the contrast term, and $s(\mathbf{x}_i, \mathbf{y}_i)$ is the structure term. Parameters α, β, and γ are for adjusting the relative importance of the three terms. The luminance, contrast, and structure terms for each local patch i use the means ($\mu_{\mathbf{x}_i}$, $\mu_{\mathbf{y}_i}$), variances ($\sigma_{\mathbf{x}_i}$, $\sigma_{\mathbf{y}_i}$) and covariance ($\sigma_{\mathbf{x}_i\mathbf{y}_i}$) that are computed on the local window, typically with Gaussian weights, from the arrays $\mathbf{x}_i$ and $\mathbf{y}_i$:

$$l(\mathbf{x}_i, \mathbf{y}_i) = \frac{2\mu_{\mathbf{x}_i}\mu_{\mathbf{y}_i} + C_1}{\mu_{\mathbf{x}_i}^2 + \mu_{\mathbf{y}_i}^2 + C_1}, \quad (2)$$

$$c(\mathbf{x}_i, \mathbf{y}_i) = \frac{2\sigma_{\mathbf{x}_i}\sigma_{\mathbf{y}_i} + C_2}{\sigma_{\mathbf{x}_i}^2 + \sigma_{\mathbf{y}_i}^2 + C_2}, \quad (3)$$

$$s(\mathbf{x}_i, \mathbf{y}_i) = \frac{\sigma_{\mathbf{x}_i\mathbf{y}_i} + C_3}{\sigma_{\mathbf{x}_i}\sigma_{\mathbf{y}_i} + C_3}. \quad (4)$$

Constants C_1, C_2 and C_3 are chosen to provide numerical stability by avoiding a zero denominator. Note that the choice $C_1 = C_2 = C_3 = 0$ is equivalent to the universal quality index [33] or UQI, which is a precursor to the SSIM. To simplify (1), the following assumptions are suggested in [35]: $\alpha = \beta = \gamma = 1$ and $C_3 = C_2/2$. This yields a simpler form of the per-pixel equation (1):

$$SSIM(\mathbf{x}_i, \mathbf{y}_i) = S_1(\mathbf{x}_i, \mathbf{y}_i) S_2(\mathbf{x}_i, \mathbf{y}_i), \quad (5)$$

where

$$S_1(\mathbf{x}_i, \mathbf{y}_j) = \frac{(2\mu_{\mathbf{x}_i}\mu_{\mathbf{y}_i} + C_1)}{(\mu_{\mathbf{x}_i}^2 + \mu_{\mathbf{y}_i}^2 + C_1)} \quad (6)$$

and

$$S_2(\mathbf{x}_i, \mathbf{y}_j) = \frac{(2\sigma_{\mathbf{x}_i\mathbf{y}_i} + C_2)}{(\sigma_{\mathbf{x}_i}^2 + \sigma_{\mathbf{y}_i}^2 + C_2)}. \quad (7)$$

Then the SSIM for the entire image, $SSIM(\mathbf{X}, \mathbf{Y})$, which is sometimes referred to as the mean SSIM, is the average of the per-pixel SSIM values calculated for each local window i:

$$SSIM(\mathbf{X}, \mathbf{Y}) = \frac{1}{M} \sum_{i=1}^{M} SSIM(\mathbf{x}_i, \mathbf{y}_i). \quad (8)$$

The SSIM value has a couple of important properties. First, $SSIM(\mathbf{X}, \mathbf{Y}) = 1$ if and only if $\mathbf{X} = \mathbf{Y}$. Also, $-1 \leq SSIM(\mathbf{X}, \mathbf{Y}) \leq 1$, and the closer $SSIM(\mathbf{X}, \mathbf{Y})$ is to 1, the more similar the images are. In practice, most calculated SSIM values are positive, with a negative value only occurring when the covariance term is negative (assuming nonnegative pixel values).

2.2 Implementation

For the implementation proposed in [35], the authors suggest the following constant values:

$$C_1 = (K_1 L)^2 \qquad C_2 = (K_2 L)^2, \quad (9)$$

where $K_1 = 0.01$, $K_2 = 0.03$, and L is the dynamic range of the pixels (so $L = 255$ for 8-bit images or $L = 1$ if the image range is $[0, 1]$). The authors note that the choice of these constants is "somewhat arbitrary," but claim that the SSIM is "fairly insensitive" to their values [35]. The constant values are of particular interest when applied directly to floating-point simulation data, as discussed in the next section.

Another implementation detail is the local window (or patch) for which the per-pixel SSIM value statistics (mean, variance and covariance) are computed in (8). The recommendation in [35] is an $N = 11 \times 11$ window with a Gaussian filter kernel. Note that for the windows centered on the pixels at the boundaries of the image (i.e., within five pixels of the edge for the 11×11 kernel), the Gaussian filter requires special treatment to handle the missing values (i.e., outside the image boundary). However, in the implementation in [35], these per-pixel SSIM values from the edge regions are simply excluded in the averaged SSIM value in (8), which could lead to reduced emphasis on pixels near the edges of an image.

Because the SSIM is quite popular, many implementations are available. We use the Python implementation of the SSIM that is available via SCIKIT-IMAGE [29]. This version closely follows that of the simplified SSIM from [35], given here in (5), using the default suggested parameters for the constants C_1 and C_2 and the Gaussian kernel size 11×11. This implementation similarly ignores the border per-pixel SSIM values when computing the overall mean SSIM. Note that for this SCIKIT-IMAGE version, one must specify "gaussian_weights=True" and "use_sample_covariance=False" to match the implementation in [35]. Finally, we note that in [35], the SSIM is defined for grayscale images. Therefore, for a multichannel images, implementations may vary depending on whether the image is first converted to grayscale, the channels (e.g., RGB) are given equal weight and averaged, or the 2D RGB image is treated as a 3D volumetric image. We prefer an initial grayscale conversion for using the SSIM.

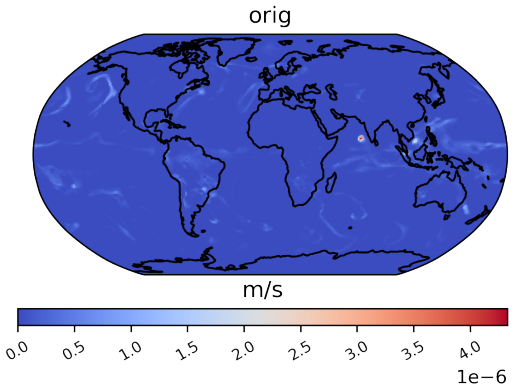


Fig. 2: The default plot of original data for precipitation rate (PRECT) in meters per second (m/s).

2.3 SSIM value dependencies

The computed SSIM value for a pair of images naturally depends on a number of plot and SSIM parameter choices that are independent of the floating-point data. Here we give a few examples to demonstrate the effect of these choices on the computed SSIM for two commonly used climate variables: surface temperature (TS) and precipitation rate (PRECT). Recall that the original TS data is shown in the top panel of Figure 1, and Figure 2 displays the PRECT data. These plots are representative of the types that climate scientists typically create and were generated with the LDCPY[1] Python package, which uses MATPLOTLIB [12] and CARTOPY [20]. While our test data is further detailed in Section 4, note that these two sample variables have quite different characteristics: TS is relatively smooth and has a modest-sized range $\approx$ 100K, while PRECT contains zero and near-zero values, changes more abruptly, and spans several orders of magnitude.

For reference, we briefly familiarize ourselves with the range of SSIM values that result from comparing climate variable images such as these. Table 1 lists the SSIM value in the third column that results from comparing the original TS data, as shown at the top of Figure 1, with several test data cases (columns 4 and 5 are discussed in later sections). For example, the LOSSY case uses the lossily compressed data shown in the bottom of Figure 1, and the visual difference due to the rather aggressive choice of compression is quite obvious, resulting in an SSIM value of 0.93453. The remaining test data cases are generated as follows. We refer to the original TS data as *ts_orig* and its maximum and minimum values as *max* and *min*, respectively. The data for the inverse (INV) case are computed by $inv\_data = max - ts\_orig + min$. Data for the RAND case are set to randomly generated values between *min* and *max*, and data for MEAN are constant values equal to the mean over all data in *ts_orig*. Finally, the MIN data are all equal to *min*, ZERO indicates an array of all zeros, and PERT adds a random perturbation to *ts_orig* from $[1.0\text{e-}7, 0.1]$ Note that when comparing two images created from simulation data with the SSIM, one needs to ensure that both figures use the same colorbar, meaning that the transform from data to image space must be the same. For the tests in Table 1, we use the same colorbar shown in Figure 1 with its extents set to the minimum and maximum values from the two datasets being compared.

1. http://github.com/NCAR/ldcpy

TABLE 1: A comparison of the original TS data to multiple test cases that are modifications of the TS test data.

case	data description	SSIM	SF-DSSIM	DSSIM
LOSSY	ZFP with p=8	.93453	.84785	.44112
INV	inverse values in [min,max]	.84872	.28281	-.70008
RAND	random values in [min,max]	.34278	.01050	-.00061
MEAN	constant mean value	.90384	.73068	.00024
MIN	constant minimum value	.72282	.70886	.00000
ZERO	all zero values	.72804	.00013	.00000
PERT	add perturb. in [1.0e-7,0.1]	.99875	.99995	.98031

The SSIM values for these test cases in column 3 of Table 1 may not be immediately intuitive. For example, comparing to all zeros does not result in a SSIM value of zero. In fact, because the comparison uses the same colorbar for the transformation to image space, the ZERO test case is more visually similar to the original than one would think. In particular, because the colorbar range has been expanded to include zero, the original data visually appears much closer to constant-valued than in the top of Figure 1. Thus, when compared to the constant of zeros values, the SSIM value is surprisingly far from zero (.72804). Interestingly, the MIN and ZERO cases have nearly the same SSIM value, both of which are noticeably lower than that of the MEAN case. Considering that the MEAN is also constant-valued, one might desire the SSIM to be lower than it is (.90384). Indeed, that the SSIM of MEAN is quite higher than that of the MIN is not necessarily intuitive, as approximating the original data by a constant-valued array of the mean or minimum would both make for quite a poor compressor in terms of visual quality. The RAND case is by far the worst according to the SSIM, which is rather expected as all structure is gone. The INV case is interesting as well as structure has been maintained in some sense, but the data are quite different. These test cases are presented not to draw sweeping conclusions about the SSIM, but simply to illustrate what the SSIM value range may look like for a particular variable.

We now explore the effects of plot choices and SSIM constants on the SSIM calculation for both TS and PRECT and list the results in Table 2. Each row in the table compares the same two sets of data for both TS and PRECT. In particular, the TS and PRECT data used in this section are included in the LDCPY package in `data/cam-fv` directory (from NetCDF files `zfp1e-1.TS.100days.nc`, `orig.TS.100days.nc`, `zfp1e-7.PRECT.60days.nc`, and `orig.PRECT.60days.nc`). The SSIM values in subsequent rows differ from the default due to either choices in the way the plots are generated or the SSIM is calculated; the underlying floating-point data is the same for each row. The first row of the table, labeled "default," lists the result of computing the SSIM via LDCPY, which uses the previously mentioned SCIKIT-IMAGE SSIM implementation. Recall that the default images for the original TS and PRECT variable data are given in Figure 1 (top) and Figure 2, respectively. The compressed data that we compare against are not shown as they have been chosen to be similar enough that differences cannot be seen for TS at this scale and are unlikely to be noticed for PRECT either, as indicated by the relatively high SSIM values for TS and PRECT: .99985 and .99153, respectively. We purposely chose a default case with high SSIM values for each variable as values near a potential SSIM cutoff threshold (e.g., that indicates whether differences are noticeable to a human) are of most interest for our

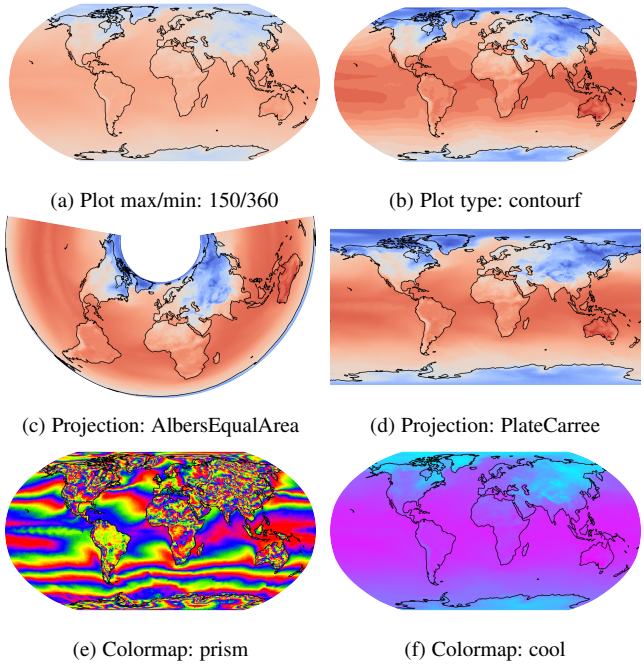
This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3332843

TABLE 2: Examples of modifications to plot choices and constant values that affect the SSIM values. Differences between the new and default SSIM values are given in parentheses (negative values indicate a decrease due to the modification).

Setting description	Default	Modification	SSIM (TS)	SSIM (PRECT)
default	-	-	.99985	.99153
show coastlines	yes (width=0.5)	no	.99983 (-.00002)	.98958 (-.00195)
plot min/max	TS:216.74/315.58 PRECT:-1.86e-8/1.18e-6	TS:150/360 PRECT:-5.0e-8/5.0e-6	.99995 (+.00010)	.99866 (+.00713)
plot type	pcolormesh	contourf (TS:lev=25 PRECT:lev=50)	.99944 (-.00041)	.96864 (-.02289)
figure dpi (dots/inch)	300	100	.99987 (+.00002)	.99323 (+.00170)
projection	Robinson	AlbersEqualArea	.99857 (-.00128)	.99235 (+.00082)
projection	Robinson	PlateCarree	.99985 (.00000)	.99113 (-.00040)
colormap	coolwarm	prism	.99544 (-.00441)	.88386 (-.10767)
colormap	coolwarm	cool	.99994 (+.00009)	.99572 (+.00319)
modified constants	K_1=.01, K_2=.03	K_1=.01, K_2=.01	.99909 (-.00076)	.96125 (-.03028)
modified constants	"	K_1=.01, K_2=1e-8	.99274 (-.00711)	.75449 (-.23704)
modified constants	"	K_1=.03, K_2=.01	.99909 (-.00076)	.96125 (-.03028)
modified constants	"	K_1=1e-8, K_2=.03	.99985 (.00000)	.99153 (.00000)
modified constants	"	K_1=1e-8, K_2=1e-8	.99274 (-.00711)	.75449 (-.23704)
modified constants	"	K_1=1e-3, K_2=1e-3	.99298 (-.00687)	.82219 (-.16934)
modified constants	"	K_1=0, K_2=0	NaN (–)	NaN (–)

application. This concept of a cutoff threshold is discussed in the context of climate data in Section 4.

In the top half of Table 2, a subset of the parameter choices used to create these default plots are listed in the second column. For each row after the "default" case, the parameter that is changed when creating both the original and compressed images is indicated in the third column, labeled "modification". For reference, plots for a subset of these modifications for TS are displayed in Figure 3. The differences in SSIM values give an idea of the effect of the plotting choices for these two sample variables. Other climate variables may be more or less sensitive to these changes, but that is not our focus. Note that while the first two significant digits are the same (.99) in each row for TS, we are interested in five significant digits to match the number of digits in the SSIM threshold from previously mentioned quality measures for climate compression [4]. The second row shows that removing the coastlines does not have much effect on the SSIM for TS, but a bit more for PRECT. Enlarging the extents on the colormap (Figure 3a) as in row 3 of Table 2 moves the SSIM closer to 1.0 for both variables, as would be expected. Figure 3b shows the plot generated using `contourf()` instead of `pcolormesh()`, which also has a small effect for TS but a larger effect for PRECT as seen in Table 2 (row 3). The type of data projection onto a map, which is common for climate data, influences the SSIM as well. The equal-area map projection (row 6) actually has more of an influence for TS (Figure 3c) than for PRECT. On the other hand, the often used equirectangular projection (row 7) has no effect on TS (Figure 3d), but does affect PRECT. The last two rows in the top half of the table illustrate that using a different colormap can definitely affect the SSIM, particularly when it is quite different from the original, such as that shown for TS in Figure 3e. The colormap changes the SSIM less (but still notably for PRECT) when it is similar to the original (Figure 3f). Note that while the SSIM is not influenced by the color or hue of an image [10], [30], [31], when we encode the floating-point data into a colormap, the characteristics of the colormap can influence the SSIM [31]. For example, because the *prism* color map is more segmented than the default, this affects the SSIM more [31] than the *cool* colormap which is more similar to the default *coolwarm* map. Other factors that we have not yet mentioned that have been shown to affect the SSIM include the size of the local window, whether or not to use



(a) Plot max/min: 150/360 (b) Plot type: contourf

(c) Projection: AlbersEqualArea (d) Projection: PlateCarree

(e) Colormap: prism (f) Colormap: cool

Fig. 3: Examples of data-independent plot choices that can affect the SSIM values for surface temperature (TS). Note that the corresponding SSIM values are given in Table 2.

Gaussian weights, and how the edge pixels are treated [14].

The bottom half of Table 2 focuses on the SSIM constants K_1 and K_2 as defined in (9). While it is customary to use the recommended defaults proposed in [35], we have found the SSIM calculation to be sensitive to the values for K_1 and K_2 (which in theory should only contribute to numerical stability) in some cases. We note that the SSIM's sensitivity to constants is shown in [26] as well, particularly the sensitivity to K_2, which is what we found with our climate data. Recall that the first row in Table 2 (labeled "default") gives the SSIM values for the default values of K_1 and K_2. For the number of significant digits that we list, only the effects of changing K_2 are noticeable. Indeed, Table 2 shows that for the PRECT data, changing K_2 can cause quite a large difference, exceeding anything shown in the top half of the

table by orders of magnitude.

For our use case of interest and likely others, the observation that the SSIM calculation as in (5) is more sensitive to K_2 than K_1 meets expectations. In practice, the SSIM is generally used to compare images that are supposed to be similar in some sense (e.g., in our case, differing only in compressor-induced artifacts). Therefore, it is reasonable to assume that the means of the two images will be quite similar. As a result, for the first term in the SSIM, S_1 as given in (6), if μ_{x_i} and μ_{y_i} are nearly the same over the local window, then the value of C_1 (and therefore K_1) is unimportant, and S_1 will be nearly one. However, the situation is less clear for the second term, S_2 as given in (7), in the SSIM calculation. The magnitude of the numerator and denominator statistics, $\sigma_{x_i y_i}$ and $(\sigma_{x_i}^2 + \sigma_{y_i}^2)$, respectively, may be less stable than the mean over the 11×11 window, in which case the value of C_2 (and thus K_2) becomes more influential. This influence is important as we now move to applying the SSIM directly to the floating-point data.

A few considerations that we do not specifically address are related to the grid that the floating-point data live on. When an image is created from gridded data, the image may have either more or fewer pixels than grid points - depending on the chosen image resolution and the data grid size. In addition, we are assuming that we have structured grid data.

3 Application to floating-point data

Recall that our primary interest in the SSIM is as a tool for evaluating the effects of lossy compression on climate simulation data. While we also use other metrics to evaluate compression quality, the SSIM has proven useful for quantifying visual differences in images created from climate data [4]. The primary advantage of and motivation for applying an SSIM-like statistic directly to the floating-point data (rather than the image pixel values) is the reduction in computational cost associated with generating images, particularly when data volumes are large and the images created are not actually needed for any other purpose. A second more minor advantage in operating directly on the floating-point data is that we can avoid plot-specific but data-independent decisions (color map, scale, axes, grid transform, etc.), which may result in different SSIM values for images created from the same datasets. To summarize, then, our goal is to determine whether we can apply a SSIM-like metric to the raw simulation data and obtain a useful indication of the differences in the data. While data differences are likely to impact a visual assessment, we cannot predict this without committing to rendering a specific set of images, which we want to avoid.

Finally, in developing a modified SSIM for floating-point data, we were motivated to rethink the choices for the constants. While the SSIM constants were introduced to prevent dividing by zero [35], we would prefer that they do not noticeably affect the SSIM values, as shown in the previous section for PRECT especially. In particular, if the SSIM statistics in the local window (i.e., mean, covariance, variance) are close to zero for the floating-point data, then the constants may have an out-sized effect. This characteristic in not uncommon when comparing to data with modest compression. Further, it is known that SSIM values tend to saturate toward one, and we see this effect even when the data are quite different, as for TS in Table 1. By making the constants less influential, we can spread the range away from one.

In this section, we first describe applying the SSIM formula, without modification, directly to floating-point data and explain why further modifications were desired. We then discuss the further modifications that collectively result in the DSSIM and how they are useful. We compare the SSIM variants on climate model data with compression in Section 4.

3.1 A straightforward approach (SF-DSSIM)

The straightforward approach to extending the SSIM to floating-point data is simply to use the SSIM equation in (8) to compare the 2D arrays with M grid points, where arrays $\mathbf{x}_i$ and $\mathbf{y}_i$ now contain the floating-point values in the local window of size N, where typically $N = 11 \times 11$, and are centered at grid location i. The constant definitions and defaults in (9) remain the same, but now L is the dynamic range of the floating point data. We refer to this variant as straightforward data SSIM, or SF-DSSIM.

There are a number of considerations when applying the SSIM formula directly to floating-point data instead of to pixel values. First, the suggested SSIM values for K_1 and K_2 may not be appropriate for every dataset. While for the SSIM, the pixel range L is typically L=255 or L=1, for floating-point numbers, the range may be much larger. If the range is quite large, then the suggested values for K_1 and K_2 may result in constants that are too big compared to the data values at some locations. This situation is of concern as the constants are only meant to prevent division by zero. Another scenario is that in which the dynamic range for a set of data could be L=1, with all values in $[0,1]$, including many very small near-zero values (e.g., of order 1e-20) as happens for the previously mentioned PRECT variable. In other words, if the data range is small but the range of exponents is quite large, then the constants will dominate the SF-DSSIM computation. This size mismatch results in SF-DSSIM values of one or near one, even when the two datasets are quite different. (Recall that the SSIM value is 1.0 only when the two images are identical.)

Two more minor considerations when dealing with floating-point simulation data include that a NaN (or fill value or missing value) can be encountered in the data, which is common for climate data. In this case, we do not want such values to propagate to the entire local window when the Gaussian filter is applied, and the code must handle this situation. Another point is that while pixel values are typically nonnegative, floating-point values are often negative (or a mix of positive and negative). Therefore, the situation where the sign of means $\mu_{\mathbf{x_i}}$ and $\mu_{\mathbf{y_i}}$ are opposite can occur and cause $S_1(\mathbf{x_i}, \mathbf{y_i})$ as given in (6) to be negative.

In Table 1, the second column from the right lists the SF-DSSIM values for the modified TS datasets. Note that the MIN and ZERO cases have nearly the same SSIM values, but quite different SF-DSSIM values, which may arguably make more intuitive sense in terms of a comparison to all zeros resulting in a nearly zero SSIM value. For the PERT case, the SF-DSSIM is actually closer to 1.0 than the SSIM value. As will be discussed further in the context of climate data, this behavior is largely the result of the quantization step in rendering the images for the SSIM calculations. This particular perturbation results in quantization bin changes (which can increase the difference) for some of the values perturbed at the high end of the interval. If the perturbation values were all small enough, e.g. in $[1.0\text{e-}7, 1.0\text{e-}5]$, then the SSIM and SF-SSIM would both be 1.0 (to five significant digits).

3.2 Data SSIM (DSSIM)

We now explain the modifications to the SF-DSSIM approach that collectively result in our proposed variant of the SSIM for floating-point data, which we refer to as DSSIM. To begin, we normalize both sets (the original and that to compare) of floating-point data to the range $[0,1]$. We normalize the data for a couple of reasons. First, normalizing to this range makes determining appropriate constants, which we discuss shortly, much easier. This step both eliminates the need for the L term in the constants in (9), as $L = 1$, and ensures that $S_1(\mathbf{x_i}, \mathbf{y_i})$ is nonnegative, as is typically the case with the SSIM (meaning that a negative value can only result from a negative covariance). Also, when visualizing floating-point data, the first step is to transform the data to the color bar range. We assume a linear transform and note that pixel values are often normalized so that each pixel value has a value between 0 and 1.

In determining the choice of constants for the DSSIM, recall that the two constants C_1 and C_2 are intended to provide numerical stability. We simply want C_1 and C_2 to be small enough to not disproportionately influence the value of the DSSIM, yet big enough to prevent dividing by zero. We set them to equal values largely for convenience: $C_1 = C_2$ and $K_1 = K_2$. Therefore, because $L = 1$ for the DSSIM, we have $C_1 = K_1^2$, and we find that

$$C_1 = K_1^2 = \text{1e-8}, \tag{10}$$

is a reasonable choice for the DSSIM. We verified this choice for our application data by examining the influence of changing the constant on the DSSIM calculation for a number of different variables and compressor levels and finding the largest constant value that no longer influences the DSSIM value. For example, in Figure 4, we show the effect of changing the constant values for the SSIM, SF-DSSIM, and DSSIM when comparing the original TS data with the lossy compressed version shown in Figure 1 and listed in the first row of Table 1. The dashed lines indicate the default value of the three SSIM approaches, which is at approximately $K_1 = K_2 = .01$ on the plot for SSIM and SF-DSSIM ("approximately" because $K_2 = .03$ in these methods, which results in a subtle difference). The plot shows that the DSSIM value no longer decreases with decreasing constant at $K_1 = .0001$, and this behavior is representative of what we observed for test data from other variables. While this choice is likely reasonable for data from other application areas as well, verifying that the computed DSSIM values are not sensitive to the constants is certainly easy to check.

Another difference for the DSSIM is that after normalizing the floating-point data to the range $[0,1]$, the DSSIM linearly quantizes the data into 256 bins. The quantization mimics a linear color map transformation by allowing the DSSIM to use a similar precision on the floating-point data as the SSIM is using on image pixel data. In our experience, this step is particularly helpful when comparing data that is quite visually similar, meaning that the SSIM value is quite close to one. We demonstrate and further explain the effects of this modification for climate variables in the next section, noting that one could choose to forgo this step or use a different type of quantization.

Finally, we address any NaN values (or fill or missing values, which we hereafter also refer to as NaNs) present in the data when the Gaussian kernel is applied locally to each 11×11 window. If the center point of the window (grid point i) is NaN, that window calculation is simply excluded from the final mean SSIM calculation given in (8). However, if the value at the center

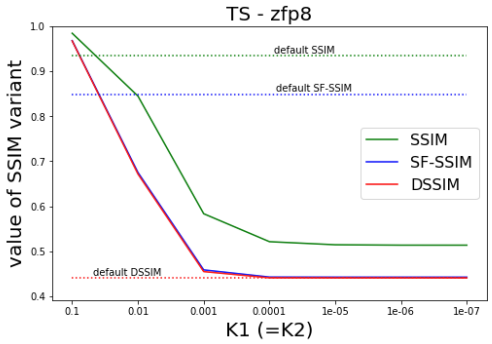


Fig. 4: This plot shows the effect of modifying the value of $K_1 (= K2)$ when computing the SSIM, SF-DSSIM, and DSSIM for comparing the original surface temperature (TS) data to the lossily compressed data with compressor ZFP and p=8.)

of the window is not NaN but any of the other local values in the window are NaN, the filter must be modified (otherwise the $DSSIM(\mathbf{x_i}, \mathbf{y_i})$ value will be set to NaN). In the convolution and 2D Gaussian kernel functions in the ASTROPY[2] package [2] [1], NaNs are replaced by interpolating from neighboring data points within a given kernel. In particular, for the kernel, DSSIM uses `Gaussian2DKernel(x_stddev=1.5, x_size=11, y_size=11)` and then we convolve with `filter_args = 'boundary': 'fill', 'preserve_nan': True`. The boundary option for convolve is not important in the current implementation as the DSSIM values whose windows extend past the boundary are ignored when computing the mean DSSIM over the grid – as with the SSIM. Note that SCIPY convolution routines do not properly deal with NaNs at this time, but our DSSIM implementation, available in the previously mentioned LDCPY package, does. In addition, we note that we evaluate only 2D data sets in this manuscript. To summarize the DSSIM algorithm, pseudocode is provided in Algorithm 1 for comparing two 2D datasets $\mathbf{X}$ and $\mathbf{Y}$, each of dimension $m_x \times m_y$ with $M = m_x m_y$ floating-point values.

Algorithm 1 DSSIM($\mathbf{X}$, $\mathbf{Y}$)

1: Normalize: 2D arrays $\mathbf{X}$ and $\mathbf{Y}$ to [0,1]
2: Constants: $C_1 = C_2 =$ 1e-8
3: Quantization: $\mathbf{X}$, $\mathbf{Y}$ (default: linearly to 256 bins)
4: M = number of grid points in $\mathbf{X}$ and $\mathbf{Y}$
5: **for** $i = 1 : M$ **do**
6: **Compute "local" DSSIM** in 11×11 window centered at i
7: $\mathbf{x}_i \leftarrow 11 \times 11$ values centered at i from $\mathbf{X}$
8: $\mathbf{y}_i \leftarrow 11 \times 11$ values centered at i from $\mathbf{Y}$
9: Apply a Gaussian kernel to $\mathbf{x}_i$ and $\mathbf{y}_i$
10: $\mu_{\mathbf{x_i}}$, $\mu_{\mathbf{y_i}} \leftarrow$ means of $\mathbf{x}_i$ and $\mathbf{y}_i$
11: $\sigma_{\mathbf{x_i}}$, $\sigma_{\mathbf{y_i}} \leftarrow$ variances of $\mathbf{x}_i$ and $\mathbf{y}_i$
12: $\sigma_{\mathbf{x_i y_i}} \leftarrow$ covariance of $\mathbf{x}_i$ and $\mathbf{y}_i$
13: $S_1(\mathbf{x}_i, \mathbf{y}_j) \leftarrow \frac{(2\mu_{\mathbf{x}_i}\mu_{\mathbf{y}_j} + C_1)}{(\mu_{\mathbf{x}_i}^2 + \mu_{\mathbf{y}_i}^2 + C_1)}$
14: $S_2(\mathbf{x}_i, \mathbf{y}_j) \leftarrow \frac{(2\sigma_{\mathbf{x}_i\mathbf{y}_j} + C_2)}{(\sigma_{\mathbf{x}_i}^2 + \sigma_{\mathbf{y}_i}^2 + C_2)}$
15: $DSSIM(\mathbf{x}_i, \mathbf{y}_i) \leftarrow S_1(\mathbf{x}_i, \mathbf{y}_i) S_2(\mathbf{x}_i, \mathbf{y}_i)$
16: **end for**
17: $DSSIM(\mathbf{X}, \mathbf{Y}) = \frac{1}{M} \sum_{i=1}^{M} DSSIM(\mathbf{x}_i, \mathbf{y}_i)$

To extend the DSSIM to 3D, multiple options are possible, depending on the use case. For example, for Earth System Model data, LDCPY reports the minimum DSSIM value over the 2D

2. http://www.astropy.org

horizontal slices for 3D data, as our interest is in the worst-case scenario for a vertical level. (Note that the vertical spacing is typically much larger than the horizontal spacing). One could also compute the 3D DSSIM by averaging the DSSIM value over all 2D slices. Or, one could extend the DSSIM metric into a third dimension by computing the relevant quantities (mean, variance and covariance of each window into the data) on 11x11x11 cubes of the data, and the calculations listed in equations (1) -(8) would remain unchanged. We have not yet explored this latter option, though it may be relevant in other application areas.

Referring back to Table 1, the rightmost column lists the DSSIM values. Note that the negative value for the INV case helpfully reflects the negative correlation in the data. The three constant-valued sets MEAN, MIN, and ZERO are now all similar and very close to zero. RAND is now also near zero. From our user base's point of view, the DSSIM values are more intuitively in line with the type of changes represented by the test cases. In the next section, we will use climate simulation data to demonstrate the benefit of using the DSSIM.

4 Application to Climate Data

We now describe our investigation into whether we can use the DSSIM instead of the SSIM for evaluating the effects of lossy compression on climate data in order to avoid the computational cost incurred by generating images. We use data from the Community Earth System Model (CESM) [13], which is a popular climate model that generates far too much data (e.g., terabytes or petabytes) – hence the interest in lossy compression. In previous work in [5] and [4], we applied the SSIM to images created from CESM data with the NCAR Command Language (NCL) [28], which are similar to those generated by the Atmosphere Working Group Diagnostics Package (AMWG-DP). While AMWG-DP-type images are familiar to scientists in the Earth science community because of its historical widespread use, Python has been quickly replacing NCL as the analysis tool of choice in recent years. In fact, many scientists are doing their own analyses in Python with the help of communities such as Pangeo[3] [9] and creating their own images. This change in post-processing analysis provided further motivation for us to use an SSIM-like measurement that is independent of plot choices.

4.1 Experimental data details

The experiments in this paper use a subset of data from the popular and publicly available CESM Large Ensemble Community Project (CESM-LENS) [15]. In particular, we use the CESM-LENS data corresponding to the RCP8.5 forcing period, which begins in January 2006, and ensemble member 31. We focus on the atmospheric model output, which uses a one-degree latitude-longitude grid corresponding to 192×288 grid points per vertical level and 30 vertical levels. Each of the more than 200 atmospheric variables is stored in a NetCDF-formatted time-series file according to its output frequency: monthly, daily, or 6-hourly. And while CESM performs computations in double precision (64-bit), it writes data to file in single precision (32-bit). In Table 3, we list a subset of the atmospheric variables for which we show results. These variables are among the most frequently downloaded and analyzed from the CESM-LENS dataset and have differing characteristics. For example, in terms of compression, TS is considered an "easy" variable to compress due to its relatively narrow range and smoothness, while variables such as PRECT, which has a large range of values, including some very small values close to zero, are typically challenging for lossy compressors.

For these experiments, we compress CESM-LENS data with the popular ZFP compressor [19]. ZFP is a high-speed lossy compressor designed for compressing logically regular and spatially correlated arrays of floating-point numbers, compressing data based on various accuracy or size constraints. We use ZFP 0.5.5 in fixed-precision mode, meaning that the precision encoded for the transform coefficients is fixed. The fixed-precision mode parameter (p) specifies how many uncompressed bits per value to store (related to the relative error), so the smaller the value of p, the more aggressive the compression. To improve compression quality, we also use a newer ZFP feature that addresses biased error and is available in ZFP 0.5.5 by checking out the "feature/unbiased-error" branch from the ZFP Github page. In particular, we enable the pre-rounding mode by configuring ZFP with "`cmake -DZFP_ROUNDING_MODE=ZFP_ROUND_FIRST -DZFP_WITH_TIGHT_ERROR=ON`".

4.2 SSIM variants and compressed climate data

To better illustrate their differences, we compare the SSIM, SF-SSIM, and DSSIM on the atmospheric variables in Table 3, each of which is compressed by varying amounts. The SSIM quantities are again computed via LDCPY with default settings as in Section 2.3, and results for each variable are given in Figure 5. For each, we plot the SSIM, DSSIM, and SF-DSSIM values that compare the original dataset to its compressed version. We show results for eight different levels of ZFP fixed-precision compression ($p = 6, 8, 10, 12, 14, 16, 18, 20$). As p increases, the compressor is increasingly conservative. In particular, $p = 6$ is the most aggressive compressor option shown and should result in the smallest SSIM values for all variables, and $p = 20$ is fairly conservative and should result in values close or even equal to one. We do not show $p > 20$ as they are indistinguishable from each other and 1.0 in the plots. A fourth SSIM variant appears in these figures as well: "DSSIM (no quant)". This approach is equivalent to DSSIM without the quantization step and will be discussed shortly.

Recall that the primary motivation for using the DSSIM instead of the SSIM is to avoid the cost associated with computing the SSIM by generating plots that are otherwise not needed, as is the case here for evaluating lossy compressor artifacts. In applying the SSIM formula directly to the floating-point data, we took the opportunity to make a few beneficial modifications, collectively referred to as the DSSIM, whose effects can be seen in the plots in Figure 5. First, note that by normalizing the data to $[0, 1]$ and then choosing constants that do not have an out-sized influence on the computed SSIM values (e.g., as discussed for Figure 4), the DSSIM obtains much lower values for more aggressive compressor options, i.e., smaller values of p, than the SSIM and SF-DSSIM.

Another trend in Figure 5 is that as the compression becomes more conservative with increasing p values, the SF-DSSIM values reach 1.0 sooner than the SSIM for all of the variables. Recall that for the SSIM, values of 1.0 are obtained only when the images are exactly the same. When applying this calculation directly to the floating-point data, it is reasonable to not want a value of 1.0 when

3. http://pangeo.io

TABLE 3: Sample atmospheric variables with their descriptions and selected characteristics (minimum, absolute nonzero minimum, and maximum values) for the first time slice. Note that the absolute nonzero minimum is not listed when it is equivalent to the minimum. All listed variables are 2D. (CLOUD is normally a 3D variable, but here we use vertical level 20 only.)

variable name	description	units	output frequency	mean	min	abs. min (nonzero)	max
TS	surface temperature	K	daily	284.64	218.12	–	321.47
PRECT	precipitation rate	m/s	daily	2.74e-8	-1.16e-20	8.60e-32	4.33e-6
PS	surface pressure	Pa	monthly	9.85e4	5.24e4	–	1.04e5
FLUT	upwelling long-wave flux (top of model)	W/m^2	daily	233.72	80.529	–	338.08
QFLX	surface water flux	$kg/m^2/s$	monthly	3.53e-5	-2.26e-6	1.57e-11	1.44e-4
CLOUD	cloud fraction	fraction	monthly	0.14	0.0	4.86e-12	0.74

TABLE 4: A list of the smallest ZFP compression parameters (p) for which the SSIM variants equal 1.0. Parameters in bold indicate that the SSIM variant (SF-DSSIM, DSSIM, or DSSIM (no quant.)) reached 1.0 for a more aggressive compression parameter (lower p) than the original SSIM value (second column) did, which is undesirable.

variable	SSIM	SF-DSSIM	DSSIM	DSSIM (no quant.)
TS	24	**18**	24	**20**
PRECT	16	**10**	18	**14**
PS	22	**18**	24	22
FLUT	20	**16**	22	**16**
QFLX	20	**14**	22	**14**
CLOUD	18	**14**	18	**14**

the SSIM value is still below 1.0, indicating that the images are not equivalent. Instead, the DSSIM's sensitivity to small differences when the data is quite similar (higher values of p) can be beneficial for identifying minor differences. Because identifying when each SSIM variant reaches 1.0 (to five significant digits) is difficult to discern in the Figure 5, we list the corresponding ZFP parameter p at which this occurs in Table 4. In our experience, the DSSIM value is always less than the SSIM value, whereas the SF-DSSIM lines typically intersect the SSIM line at some point, because for smaller p, the SF-DSSIM is usually smaller than the SSIM (but then levels off to 1.0 more quickly). This statement may not be universally true as the SSIM is sensitive to plot choices, but holds for all the variables that we have examined in this representative climate dataset. As mentioned previously, recall that we assume that the floating-point data is smoothly mapped to RGB values, as with pseudocolor plots that use a smooth colormap.

Figure 5 and Table 4 also include results from running the DSSIM without the quantization step, as it is more difficult to intuit the usefulness of this modification. In fact, for TS and FLUT variables, the quantization has virtually no noticeable effect in the subplots in Figure 5, and for the remainder of the variables, it is difficult to see what is happening when the SSIM value is approaching one. Again, Table 4 better illustrates the situation by providing the DSSIM results without quantization in the rightmost column. As with the SF-DSSIM, DSSIM (no quant.) reached 1.0 before (i.e., at a smaller/more aggressive value of p) the SSIM for 5 of the 6 variables. For TS and CLOUD, the SSIM and DSSIM reach 1.0 at the same compressed dataset parameter p, and the DSSIM reaches 1.0 under more conservative compression for the remainder. This behavior is acceptable as we prefer to err on the conservative side, and these SSIM and DSSIM values near 1.0 are important when determining a similarity cutoff threshold, which we address in the next subsection. And, as previously discussed, we do not want a value of 1.0 from a similarity measure on the floating-point data when the SSIM on the images is less than 1.0, implying that there is a visual difference. Note that the data compression here with ZFP is lossy for all tested values of p, meaning that the two floating-point datasets being compared are not equivalent. However, we do *not* have the requirement that a DSSIM of 1.0 comparing datasets X and Y, implies that $X = Y$.

In effect, the DSSIM quantization step is useful in terms of better differentiating changes in quality of the data in this region of interest where the data compression is quite conservative, meaning that differences between the two datasets are quite small and the SSIM and SSIM-like variants should be nearly 1.0. Consider the situation when the two datasets are nearly equivalent and the data is varying smoothly in a local patch, then quantizing the data values will amplify small differences in data that spans quantization bin boundaries and remove small differences within a quantization bin. Even though most of the differences will fall within a bin (and thus be zero after quantization), when the differences between bin boundaries are an order of magnitude or so larger than differences between the datasets (as happens when the DSSIM is nearly 1.0), then in our experience, the quantization step allows the DSSIM to highlight the differences in the data in a similar manner to transforming data to an image with 256 colors as done with the SSIM. While the exact DSSIM value depends on the number of quantization bins, of course, we find that the DSSIM with sensible choices for numbers of bins (e.g., 128, 256, 512, 1024) better differentiates differences near 1.0 than the DSSIM without quantization.

As we have shown, this quantization has been useful for our application. More generally, because the algorithmic choices made for the DSSIM are in some sense "close enough" to the rendering process that we originally used to create an image from the 2D floating-point data (from which the SSIM is computed), we are able to use the DSSIM as a substitute for a visual quality metric. To be clear, the DSSIM itself is not a visual quality metric. Indeed, there are too many dependencies on the image generation choices. To use the DSSIM as a visual quality metric (or have it return an equivalent value to the SSIM, for example), one would need to customize the algorithm to incorporate the specific quantization and mapping for the desired image rendering. And depending on the relative dataset size and image resolution chosen, this mapping would include resizing from the data grid to the pixel grid. While possible, this is not our focus here as we find the proposed DSSIM to be quite useful in practical situations as a less costly substitute when there is no need to generate or retain the actual images. We further comment on its potential role in other use cases in Section 5.

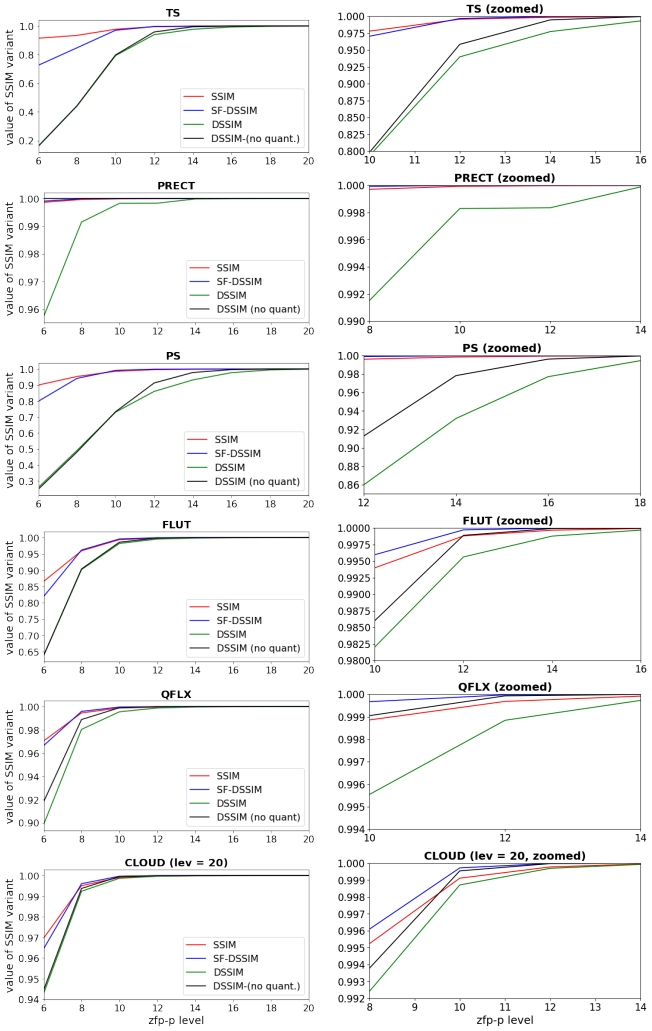


Fig. 5: A similarity comparison between the original data and the ZFP-compressed data for different compression parameter (p-level on the x-axis). The range on the y-axis differs for each plot, and the plots in the right column are zoomed-in versions of the plots in the left column.

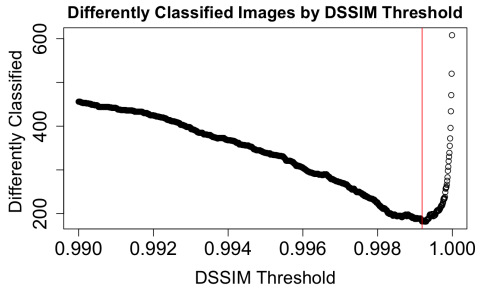
4.3 A cutoff threshold for DSSIM

The primary reason for developing the DSSIM as a replacement to the SSIM is to reduce computational costs when evaluating compression effects on large amounts of data. When using the SSIM to quantify the similarity of an image generated with the compressed climate data to that with the original, we use a so-called cutoff threshold, above which the quality of similarity is deemed acceptable. More specifically, in [4] we determined that the SSIM with a cutoff threshold of 0.99995 indicated whether climate scientists would be able to detect a difference in CESM diagnostic images after compression. This threshold, based on a large user study, is much tighter than the generally accepted SSIM indistinguishability threshold of 0.99 (e.g., [21]) and the 0.98 suggested for medical imaging (e.g., [8], [36]). Therefore, note that because we are using the SSIM with a hard threshold, small effects from plotting choices can be important in regions near the threshold. Thus, the DSSIM's independence from plotting decisions is a desirable addition to the cost-savings gained by not generating specific images. However, to use the DSSIM instead of the SSIM for evaluating the climate data, we need to determine an appropriate cutoff threshold for the DSSIM, and conducting another large user study is not feasible at this time. Instead, we use statistical techniques and the previous study results to show that the DSSIM can be used to evaluate lossy compression artifacts in climate data with an appropriate cutoff threshold.

Ideally, we want to find a DSSIM threshold such that datasets that pass the SSIM threshold test also pass the corresponding DSSIM threshold test, and datasets that fail the SSIM threshold test also fail the corresponding DSSIM threshold test. We define a compressed dataset to be a true "pass" if the SSIM value meets or exceeds the 0.99995 threshold, otherwise we consider that dataset to be a true "fail". One method to assess an appropriate DSSIM threshold is with classification matrices, which are commonly used tools to evaluate the results of a classification model. In our case, the classification matrix is a 2×2 matrix where the columns correspond to the true pass or fail status of the data as determined by the SSIM. The rows correspond to whether our model (i.e., the DSSIM) passes or fails the dataset, which is based on whether the DSSIM is above (pass) or below (fail) the DSSIM threshold being tested. This setup means that the diagonal entries of the matrix correspond to the number of instances where there is agreement or consistency between the DSSIM and the SSIM, and the off-diagonal elements correspond to the number of instances where the DSSIM and SSIM disagree (an inconsistency) in their classification decision. For example, if we take the first 3 time slices from the 79 2D monthly variables in the CESM-LENS dataset and apply the ZFP compressor using 10 parameters for p ($p = 6, 8, 10, 12, 14, 16, 18, 20, 22, 24$), then we obtain 2370 SSIM values obtained by plotting and comparing the original dataset and the compressed dataset. As before, SSIM and DSSIM values are computed via LDCPY with the default settings. In Figure 6, the top plot shows the number of images for which the DSSIM result was classified differently than the SSIM result (i.e., "inconsistent") for a range of DSSIM thresholds. The bottom plot is the classification matrix corresponding to a DSSIM threshold of 0.99919, which minimizes the inconsistent results (the sum of the off-diagonal entries in orange). Note that alternatively one could choose to minimize either the number of inconsistent fails or inconsistent passes, depending on the use case.

This analysis for finding a cutoff threshold assumes that the data distribution in our analysis is representative of the data distribution in practice. Because the CESM variables have quite different characteristics, an appropriate DSSIM threshold based on only a single variable may be slightly different (lower or higher). However, because the SSIM threshold determined in [4] is quite conservative, we find that the corresponding DSSIM threshold of 0.99919 is conservative enough to use on all CESM variables. In practice, we often reduce this threshold further to allow for more aggressive data compression (e.g., to 0.995 or 0.99).

Figure 7 gives an indication of how much compression can be achieved with ZFP in fixed-precision mode for the variables in Table 3, including how the data reduction corresponds to the DSSIM. The compression ratio (CR) is the size of the losslessly compressed data divided by the lossily compressed data with ZFP. We see that the CR decreases as the ZFP precision parameter p increases, as expected. CR values near 1.0 mean that the lossy compression is so conservative that there is little reduction beyond what lossless compression has achieved. The DSSIMs increase in a nonlinear fashion, which is consistent with the idea that we get diminishing returns in the data fidelity as we approach lossless



(a)

Pass **consistent** with SSIM 1128 (47.6%)	Pass **inconsistent** with SSIM 71 (3.0%)
Fail **inconsistent** with SSIM 111 (4.7%)	Fail **consistent** with SSIM 1060 (44.7%)

(b)

Fig. 6: The top plot shows the number of differently classified (i.e., inconsistent) datasets by DSSIM threshold, which is minimized when the DSSIM threshold is 0.99919. The bottom plot contains the classification matrix for a DSSIM threshold of 0.99919.

compression. Moreover, there is clearly a limit to the amount of compression possible while maintaining a high DSSIM value. The larger the CR, the lower the DSSIM tends to be, which can easily be seen by noting how the colors of the dashed and solid lines are nearly in reverse order from top to bottom. Further examinations of the DSSIM in the context of data compression for climate model data can be found in technical reports [23] and [22].

4.4 SSIM vs. DSSIM cost

The algorithmic costs of the DSSIM and SSIM are quite comparable, given the similarity of the algorithms. Therefore, differences in computational cost are largely due to the cost of rendering the two images from the floating-point data that are needed to calculate the SSIM. This difference assumes that the images are only being generated for the SSIM calculation and that in-situ rendering is not an option, both of which are true in our use case. At present, the CESM data are first output from the model, then post-processed (which minimally includes a conversion to time series), and then evaluated for compression.

The computational cost of rendering the two images for the SSIM calculation will, of course, vary depending on the computing platform. Therefore, whether this associated expense is significant enough to motivate the user to use the DSSIM will be specific to their particular hardware, software stack, data volume, and workflow. For our application that compares compressor results with climate data, the cost savings with the DSSIM have been significant, particularly as we automate testing and must evaluate the data quality and similarity on large amounts of data.

As an example, for the CESM-LENS data described in Section 4.1, a single vertical level has 55,296 grid points that we use to compute the SSIM and DSSIM (via the `DataCalcs` object in LDCPY). Because the parallelism in our workflow is data-level, meaning that different tasks operate on different variables or different time periods of the same variable, the DSSIM and SSIM calculations are serial for a particular variable and time slice. If we time the two calculations via a Jupyter notebook on a modern laptop (again using the TS and PRECT variables from CESM-LENS data that are included with LDCPY), we find the speedup of the DSSIM over the SSIM to be a factor of at least 200x. Running a compute node on the Cheyenne supercomputer at NCAR (not using Jupyter), yields a speedup of more than 300x. The large speedup occurs because the DSSIM calculations are on the order of hundredths of a second, while the SSIM calculations (which includes the rendering of two images) are on the order of seconds. Again, the particular benefit will depend on the computing platform, software, and amount of data (e.g., high-resolution climate data will be more expensive, but optimized hardware will reduce the cost).

Finally, we note that in previous work, we found the SSIM to be a better indication of image similarity than other measurements for our application, but its relative expense made its use hard to justify. The cost of the DSSIM, on the other hand, is more reasonable for our large-scale evaluations and like the SSIM, has proven useful in practice. For example, a current project involves generating labeled data for a supervised learning project for climate data compression. The labeled data consists of roughly 250 variables on a typical one-degree-resolution grid, and, of course, we want to use as many time slices as possible. We are able to compute DSSIMs for two years of data for each daily variable in minutes, whereas this would take days using the SSIM. In fact, the DSSIM is competitive from a computational standpoint to other metrics for data similarity applied to the floating-point data, like the mean-squared error (MSE) or the peak signal-to-noise ratio (PSNR). However, we prefer the DSSIM to the MSE and PSNR metrics. While the MSE and PSNR can be applied to either floating-point or image data, when applied to floating-point data, it is well known they are not particularly indicative of visual similarity. (Though they may be useful for evaluating visual similarity in some cases when applied to image data.)

5 Concluding Remarks

In this manuscript, we have proposed the DSSIM, a data quality metric analogous to the popular SSIM that can be applied directly to floating-point data. While conceptually simple, the DSSIM has been tremendously beneficial to us for comparing lossily compressed to uncompressed climate model data. Prior to our development of the DSSIM, the SSIM had become an important measurement in our compression evaluation toolkit, largely due to its intuitiveness for the user and ability to represent visual similarity well. However, the SSIM was prohibitively more expensive to compute than all other measurements in the toolkit on large data volumes because of the image generation requirement, motivating us to propose the DSSIM. In practice, we now have been using the DSSIM instead of the SSIM with success in terms of identifying compression artifacts and saving compute time when evaluating data compression. Of course, the DSSIM cannot predict with certainty what the effects of data differences (e.g., due to compression) will be on all images rendered from the data without the specifics of the plotting choices. However, we have demonstrated that for certain situations in which we would like a general idea of whether spatial images created from the data will likely be similar (and do not need specific images), such as our described use case, the DSSIM is both appropriate and beneficial.

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3332843

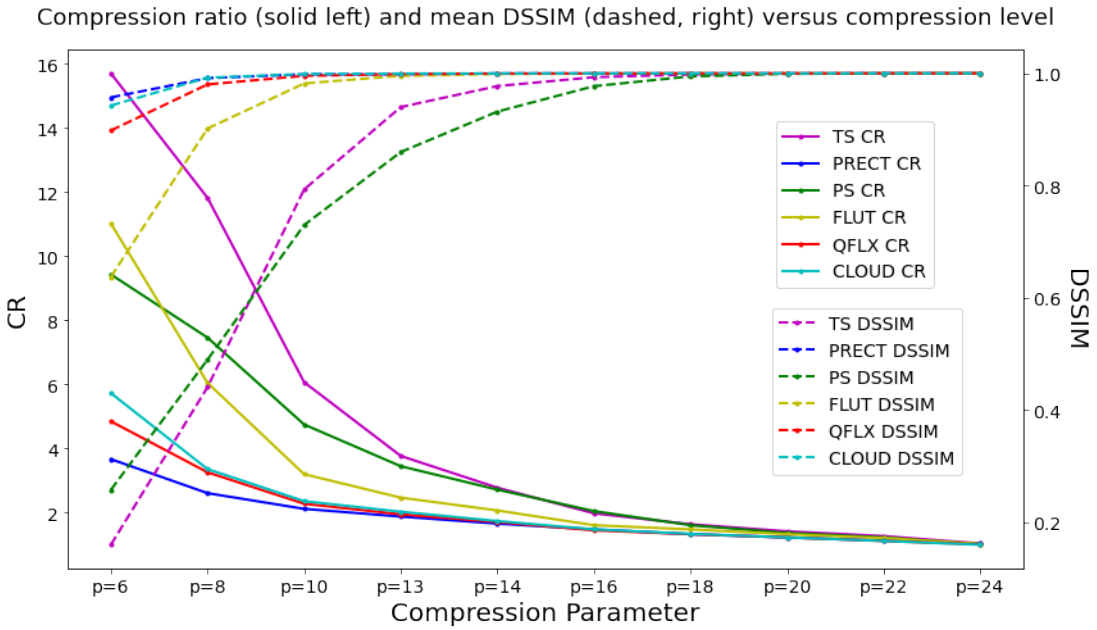


Fig. 7: The relation between various amounts of lossy compression and the compression ratio (CR) and DSSIM values for the variables in Table 3. The horizontal axis indicates the different fixed precision parameters for ZFP compression from most (left) to least aggressive (right). The dashed lines correspond to the DSSIM values (right axis) and the solid lines correspond to the CR (left axis). The compression ratio is the losslessly compressed file size divided by the lossy compressed file size.

In other words, although the DSSIM as presented is not a visual quality metric, we are able to use the DSSIM instead of the SSIM to assess differences in visual quality of our datasets.

While we have only evaluated the DSSIM in the context of comparing climate model simulation data, we are optimistic that it could be a useful assessment tool in other application areas as well - especially those producing large volumes of simulation data. In particular, if images are being created solely for the purpose of evaluating data quality via the SSIM, then the cost of image generation can be avoided by using the DSSIM. Avoiding this additional cost per image, regardless of how optimized the image generation process may be, is desirable for large volumes of data. In addition, because the DSSIM is necessarily independent of plot-specific choices that can affect the SSIM, the DSSIM measurement can be more intuitive in terms of measuring differences in datasets. It is important to note, however, that if one is interested in assessing the visual quality of specific plots with specific plotting parameters (or one already has generated particular plots of interest), then the SSIM is the appropriate choice. On a related note, though, is the idea of creating a more generic DSSIM in which the user would essentially provide the rendering algorithm. That is, if one doesn't need to actually generate specific plots for any other reason other than to assess visual quality, but does have specific mappings in mind, a more general version of the DSSIM could be useful. Instead of defaulting to the quantization or mapping choices implied by the current DSSIM algorithm, one could specify, for example, the mapping or color transfer function. We did not explore this more customised approach as it was not needed for our application, though it could be interesting to explore in future work or other use cases.

Acknowledgment

We acknowledge high-performance computing support from Cheyenne (doi:10.5065/D6RX99HX) provided by NCAR's Computational and Information Systems Laboratory, sponsored by the National Science Foundation. We thank the anonymous reviewers for their helpful comments.

References

[1] Astropy Collaboration, A. M. Price-Whelan, et al. The Astropy Project: Building an Open-science Project and Status of the v2.0 Core Package. *The Astronomical Journal*, 156(3):123, 2018. doi: 10.3847/1538-3881/aabc4f

[2] Astropy Collaboration, T. P. Robitaille, et al. Astropy: A community Python package for astronomy. *Astronomy and Astrophysics*, 558:A33, 2013. doi: 10.1051/0004-6361/201322068

[3] A. Baker, H. Xu, J. Dennis, M. Levy, D. Nychka, S. Mickelson, J. Edwards, M. Vertenstein, and A. Wegener. A methodology for evaluating the impact of data compression on climate simulation data. In *Proceedings of the 23rd International Symposium on High-performance Parallel and Distributed Computing*, HPDC '14, pp. 203–214, 2014. doi: 10.1145/2600212.2600217

[4] A. H. Baker, D. M. Hammerling, and T. L. Turton. Evaluating image quality measures to assess the impact of lossy data compression applied to climate simulation data. *Computer Graphics Forum*, 38(3):517–528, 2019. doi: 10.1111/cgf.13707

[5] A. H. Baker, H. Xu, D. M. Hammerling, S. Li, and J. P. Clyne. Toward a multi-method approach: Lossy data compression for climate simulation data. In *International Conference on High Performance Computing*, pp. 30–42. Springer, 2017. doi: 10.1007/978-3-319-67630-2_3

[6] R. Dosselmann and X. D. Yang. A comprehensive assessment of the structural similarity index. *Signal, Image, and Video Processing*, 5(1):81–91, 2011. doi: 10.1007/s11760-009-0144-1

[7] Y. Gaudeau, J. Lambert, N. Labonne, and J. Moureaux. Compressed image quality assessment: Application to an interactive upper limb radiology atlas. In *2014 IEEE International Conference on Image Processing, ICIP 2014, Paris, France, October 27-30, 2014*, pp. 501–505. IEEE, 2014. doi: 0.1109/ICIP.2014.7025100

[8] V. T. Georgiev, A. Karahaliou, S. Skiadopoulos, N. Arikidisa, A. Kazantzi, G. Panayiotakis, and L. Costaridou. Quantitative visually lossless compression ratio determination of JPEG2000 in digitized mammograms. *Journal of Digital Imaging*, 26(3):427–439, 2012. doi: 10.1007/s10278-012-9538-7

[9] J. Hamman, M. Rocklin, and R. Abernathy. Pangeo: A Big-data Ecosystem for Scalable Earth System Science. In *EGU General Assembly Conference Abstracts*, EGU General Assembly Conference Abstracts, p. 12146, 2018.

[10] M. Hassan and C. Bhagvati. Structural similarity measure for color images. *International Journal of Computer Applications*, 43:7–12, 2012. doi: 10.5120/6169-8590

[11] N. Hübbe, A. Wegener, J. M. Kunkel, Y. Ling, and T. Ludwig. Evaluating lossy compression on climate data. In *Proceedings of the International Supercomputing Conference (ISC '13)*, pp. 343–356, 2013. doi: 10.1007/978-3-642-38750-0_26

[12] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55

[13] J. Hurrell, M. Holland, et al. The Community Earth System Model: A framework for collaborative research. *Bulletin of the American Meteorological Society*, 94:1339–1360, 2013. doi: 10.1175/BAMS-D-12-00121.1

[14] E. L. Jones, L. Rendell, E. Pirotta, and J. A. Long. Novel application of a quantitative spatial comparison tool to species distribution data. *Ecological Indicators*, 70:67–76, 2016. doi: 10.1016/j.ecolind.2016.05.051

[15] J. E. Kay, C. Deser, et al. The Community Earth System Model (CESM) Large Ensemble Project: A Community Resource for Studying Climate Change in the Presence of Internal Climate Variability. *Bulletin of the American Meteorological Society*, 96(8):1333–1349, 2015. doi: 10.1175/BAMS-D-13-00255.1

[16] I. Kowalik-Urbaniak, D. Brunet, J. Wang, D. Koff, N. Smolarski-Koff, E. R. Vrscay, B. Wallace, and Z. Wang. The quest for 'diagnostically lossless' medical image compression: A comparative study of objective quality metrics for compressed medical images. In *Progress in Biomedical Optics and Imaging - Proceedings of SPIE*, vol. 9037, 02 2014. doi: 10.1117/12.2043196

[17] M. Kuhn, J. Kunkel, and T. Ludwig. Data compression for climate data. *Supercomputing Frontiers and Innovations*, 3(1):75–94, 2016. doi: 10.14529/jsfi160105

[18] E. Larson and D. Chandler. Most apparent distortion: Full-reference image quality assessment and the role of strategy. *Journal of Electronic Imaging*, 19, 2010. doi: 10.1117/1.3267105

[19] P. Lindstrom. Fixed-rate compressed floating-point arrays. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2674–2683, 2014. doi: 10.1109/TVCG.2014.2346458

[20] Met Office. *Cartopy: a cartographic Python library with a matplotlib interface*. Exeter, Devon, 2010 - 2015.

[21] J. Nilsson and T. Akenine-Möller. Understanding SSIM, 2020. doi: 10.48550/arXiv.2006.13846

[22] A. Pinard, A. H. Baker, and D. M. Hammerling. Examining variations in the optimal compression level of spatiotemporal datasets determined using the data structural similarity index measure (DSSIM). Technical Report NCAR/TN-570+STR, National Center for Atmospheric Research, 2021. doi: 10.5065/4q49-t141

[23] A. Pinard, A. H. Baker, and D. M. Hammerling. A statistical approach to obtaining a data structural similarity index cutoff threshold. Technical Report NCAR/TN-568+STR, National Center for Atmospheric Research, 2021. doi: 10.5065/jc2r-5289

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

[25] M. Razaak and M. G. Martini. Medical image and video quality assessment in e-health applications and services. In *IEEE 15th International Conference on e-Health Networking, Applications and Services, Healthcom 2013, Lisbon, Portugal, October 9-12, 2013*, pp. 6–10. IEEE, 2013. doi: 10.1109/HealthCom.2013.6720628

[26] S. Sawant. Compressed image quality measurement. Master's thesis, National Institute of Technology, Rourkela, India, 2013.

[27] H. R. Sheikh and A. C. Bovik. Image information and visual quality. *IEEE Transactions on Image Processing*, 15(2):430–444, 2006. doi: 10.1109/TIP.2005.859378

[28] UCAR/NCAR/CISL/TDD. The NCAR Command Language [Software], 2017. Version 6.4.0.

[29] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014. doi: 10.7717/peerj.453

[30] A. K. Venkataramanan, C. Wu, A. Bovik, I. Katsavounidis, and Z. Shahid. A hitchhiker's guide to structural similarity. *IEEE Access*, 9:28872–28896, 2021. doi: 10.1109/ACCESS.2021.3056504

[31] R. Veras and C. Collins. Discriminability tests for visualization effectiveness and scalability. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):749–758, 2020. doi: 10.1109/TVCG.2019.2934432

[32] Z. Wang. Applications of objective image quality assessment methods [Applications Corner]. *IEEE Signal Processing Magazine*, 28(6):137–142, 2011. doi: 10.1109/MSP.2011.942295

[33] Z. Wang and A. Bovik. A universal image quality index. *IEEE Signal Processing Letters*, 9(3):81–84, 2002. doi: 10.1109/97.995823

[34] Z. Wang and A. C. Bovik. Mean squared error: Love it or leave it? A new look at signal fidelity measures. *IEEE Signal Processing Magazine*, 26(1):98–117, 2009. doi: 10.1109/MSP.2008.930649

[35] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi: 0.1109/TIP.2003.819861

[36] A. Wegener. Compression of medical sensor data. *IEEE Signal Processing Magazine*, 27(4):125–130, July 2010. doi: 10.1109/MSP.2010.936781

[37] J. Woodring, S. M. Mniszewski, C. M. Brislawn, D. E. DeMarle, and J. P. Ahrens. Revisiting wavelet compression for large-scale climate data using JPEG2000 and ensuring data precision. In D. Rogers and C. T. Silva, eds., *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, pp. 31–38. IEEE, 2011. doi: 10.1109/LDAV.2011.6092314

[38] W. Xue, L. Zhang, X. Mou, and A. C. Bovik. Gradient magnitude similarity deviation: A highly efficient perceptual image quality index. *IEEE Transactions on Image Processing*, 23(2):684–695, 2014. doi: 10.1109/TIP.2013.2293423

[39] L. Zhang, L. Zhang, X. Mou, and D. Zhang. FSIM: A feature similarity index for image quality assessment. *IEEE Transactions on Image Processing*, 20(8):2378–2386, 2011.

[40] L. Zhang, L. Zhang, X. Mou, and D. Zhang. A comprehensive evaluation of full reference image quality assessment algorithms. In *2012 19th IEEE International Conference on Image Processing*, pp. 1477–1480, 2012. doi: 10.1109/ICIP.2012.6467150

6 Biography Section

Allison Baker received her BS in Mechanical Engineering from Rice University and her PhD in Applied Mathematics from the University of Colorado at Boulder. She is currently a Project Scientist III in the Computational and Information Systems Laboratory, National Center for Atmospheric Research, Boulder, CO, USA.

Alex Pinard received his BS in Electrical and Computer Engineering from Oregon State University. He is currently a graduate student in the Department of Applied Mathematics and Statistics, Colorado School of Mines, Golden, CO, USA.

Dorit Hammerling received her MA in Statistics from the University of Michigan, her MS in Civil Engineering from Michigan Technological University, and her PhD in Environmental Engineering from the University of Michigan. She is currently an Associate Professor in the Department of Applied Mathematics and Statistics, Colorado School of Mines, Golden, CO, USA.