DSSIM: a structural similarity index for floating-point data

Allison H. Baker*, Alexander Pinard†, and Dorit M. Hammerling‡
*National Center for Atmospheric Research, Boulder, CO 80305
abaker@ucar.edu
†Colorado School of Mines, Golden, CO 80401
pinarda@mines.edu, hammerling@mines.edu

Abstract—Data visualization is a critical component in terms of interacting with floating-point output data from large model simulation codes. Indeed, postprocessing analysis workflows on simulation data often generate a large number of images from the raw data, many of which are then compared to each other or to specified reference images. In this image-comparison scenario, image quality assessment (IQA) measures are quite useful, and the Structural Similarity Index (SSIM) continues to be a popular choice. However, generating large numbers of images can be costly, and plot-specific (but data independent) choices can affect the SSIM value. A natural question is whether we can apply the SSIM directly to the floating-point simulation data and obtain an indication of whether differences in the data are likely to impact a visual assessment, effectively bypassing the creation of a specific set of images from the data. To this end, we propose an alternative to the popular SSIM that can be applied directly to the floating-point data, which we refer to as the Data SSIM (DSSIM). While we demonstrate the usefulness of the DSSIM in the context of evaluating differences due to lossy compression on large volumes of simulation data from a popular climate model, the DSSIM may prove useful for many other applications involving simulation or image data.

I. 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. The newly generated image may differ from the reference image for a number of reasons that a scientist may wish to explore which are related to differences in the respective datasets: model simulation data versus experimental or observational data; two simulation data sets 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) [1], [2], though a number of such image-comparison techniques have been developed (e.g., [3]–[7]).

Because lossy compression remains a powerful tool in reducing the enormous (and often problematic) 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 results in an irreversible loss of information.) 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 (i.e., details needed to make an accurate diagnosis) is a concern. The SSIM has been advocated as a means of evaluating compressed-medical-image quality in many studies (e.g., see [8]–[12]).

An 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 (e.g., [13]–[16]), 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 [17]. In a follow-up work [18], we proposed a minimum threshold for SSIM values (which indicated when differences could be seen) for evaluating images from lossily-compressed climate simulation data. 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 (created from uncompressed data).

While the SSIM is undoubtedly useful for objectively comparing images, several shortcomings arose in the context of its use in our compression-related research. We address those issues in this manuscript for both climate model data as...
well as floating-point simulation data in general. In particular, because the SSIM calculation is based on two images (pixel values), creating plots from the dataset values to be compared is required before the SSIM can be computed. Note that the SSIM value is 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. A simple example for climate data is the decision as to whether or not to outline continent boundaries in the image. In other words, for a particular dataset, two images that are considered indistinguishable (based on their SSIM value) given one set of plot parameters may no longer be indistinguishable if these settings are changed. An additional concern is that creating many plots to compute the SSIM values, for example, from a long time series of climate data (potentially at a high spatial resolution), can be computationally intensive.

Indeed, the computational cost of generating images made the SSIM a much more expensive measure of lossily compressed data quality than the other measures in our suite (e.g., Kolmogorov–Smirnov statistic, spatial relative error, and correlation coefficient). The cost and dependence of the SSIM value on plotting choices motivates applying the SSIM directly to the floating-point dataset, rather than the pixel values in an image. This choice makes particular sense in our use-case given that we do not necessarily know how the user will create a plot from the data, but would still like to know whether a plot from a dataset is likely to show a difference. Simply applying the standard SSIM formula to floating-point data, rather than pixel values, does not result in behavior that was conceptually similar to the SSIM on images created from floating-point datasets. Therefore, we introduce the Data SSIM (DSSIM) to apply directly to floating-point datasets, such that it is independent of the plotting parameters and settings, obtains comparable behavior to the SSIM, and better discriminates between the differences in datasets.

This paper is organized as follows. In Section II we review the SSIM measurement and its implementation, and we illustrate the effect of certain image choices relevant to climate data on the computed SSIM value. Next, in Section III we address applying the SSIM directly to floating-point data and introduce our new DSSIM approach. In Section IV we discuss the applicability of the DSSIM in the context of evaluating lossy compression on climate data. We provide concluding remarks in Section V.

II. 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 [18] that the SSIM showed good predictive ability to gauge when expert users 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 on the climate data in our study [18]. It is important to note that the plots of most interest to the climate community (e.g., in diagnostic packages) 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 (i.e., not segmented) 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 is quite sensitive to accessories on the plot such as plot grids, labels, etc. (e.g., [19]), which should be avoided. Example of the types of pseudocolor plots of interest are given later in this section (and also in [18]).

A. Method

The SSIM, introduced in [1], 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., [20], [21]), it nevertheless remains very popular in practice (due in large part to its simplicity) [2] and is a useful statistical measure. The SSIM is the product of three factors that are intended to represent luminance, contrast, and structure. Consider comparing two images $X$ and $Y$, each with $M$ pixel values. The SSIM is computed by first calculating so-called per-pixel SSIM values comparing local patches (or windows) of the images. Let $x_i$ and $y_i$ be local image patches (i.e., vectors) taken from the same location in $X$ and $Y$, respectively. Subscript $i$ indicates the pixel index in $X$ and $Y$ that is at the center of the local window ($i \leq M$). And, let $N$ be the number of pixels in the local window. Then, in the local window centered at pixel $i$ with vectors $x_i$ and $y_i$ containing $N$ pixel values, the per-pixel SSIM value is

$$SSIM(x_i, y_i) = \left(l(x_i, y_i)\right)^{\alpha} \cdot \left(c(x_i, y_i)\right)^{\beta} \cdot \left(s(x_i, y_i)\right)^{\gamma},$$

where $l(x_i, y_i)$ is the luminance term, $c(x_i, y_i)$ is the contrast term, and $s(x_i, y_i)$ is the structure term ($\alpha$, $\beta$, and $\gamma$ are parameters for adjusting the relative importance of the three terms). The luminance, contrast, and structure terms for each local patch $i$ use the means ($\mu_{x_i}$, $\mu_{y_i}$), variances ($\sigma_{x_i}$, $\sigma_{y_i}$) and covariance ($\sigma_{x_i,y_i}$) that are computed on the local window (typically with Gaussian weights) with length $N$ vectors $x_i$ and $y_i$:

$$l(x_i, y_i) = \frac{2\mu_{x_i}\mu_{y_i} + C_1}{\mu_{x_i}^2 + \mu_{y_i}^2 + C_1} + C_1,$$

$$c(x_i, y_i) = \frac{2\sigma_{x_i}\sigma_{y_i} + C_2}{\sigma_{x_i}^2 + \sigma_{y_i}^2 + C_2} + C_2,$$

$$s(x_i, y_i) = \frac{\sigma_{x_i,y_i} + C_3}{\sigma_{x_i}\sigma_{y_i} + C_3} + C_3.$$
the following assumptions are suggested in [1]: \( \alpha = \beta = \gamma = 1 \) and \( C_2 = C_2/2 \). This yields a simpler form of the per-pixel equation (1):

\[
SSIM(x_i, y_i) = S_1(x_i, y_i)S_2(x_i, y_i),
\]

where

\[
S_1(x_i, y_j) = \frac{(2\mu_{x_i}\mu_{y_j} + C_1)}{\mu_{x_i}^2 + \mu_{y_j}^2 + C_1}
\]

and

\[
S_2(x_i, y_j) = \frac{(2\sigma_{x_i y_i} + C_2)}{\sigma_{x_i}^2 + \sigma_{y_i}^2 + C_2}.
\]

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

\[
SSIM(X, Y) = \frac{1}{M} \sum_{i=1}^{M} SSIM(x_i, y_i)
\]

The SSIM value has a couple of important properties. First, \( SSIM(X, Y) = 1 \) if and only if \( X = Y \). Also, \(-1 \leq SSIM(X, Y) \leq 1\), and the closer \( SSIM(X, Y) \) is to one, then 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).

**B. Implementation**

For the implementation proposed in [1], the authors suggest the following constant values:

\[
C_1 = (K_1L)^2 \quad C_2 = (K_2L)^2
\]

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 [1]. The constant values are of particular interest when applying 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 [1] is an \( 11 \times 11 \) 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 \( 11 \times 11 \) kernel), the Gaussian filter requires special treatment to handle the missing values (i.e., outside the boundary of the image). However, in the implementation in [1], 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.

As previously mentioned, the SSIM is quite popular, and two widely used implementations are the MATLAB® implementation of SSIM (available from the MATLAB Image Processing Toolkit™) and the Python implementation of SSIM (available in SCIKIT-IMAGE [23]). Both of these SSIM implementations closely follow that of the simplified version given in (5) from [1], using (by default) the suggested parameters for the constants \( (C_1 \) and \( C_2 \) and the Gaussian kernel size \((11 \times 11)\). Both implementations also ignore the border per-pixel SSIM values when computing the overall mean SSIM. The main difference is that in the SCIKIT-IMAGE version, one must specify “gaussian_weights=True” and “use_sample_covariance=False” to match the original implementation in [1]. Finally, we note that the MATLAB SSIM function takes two grayscale images as input, and the SCIKIT-IMAGE version will accept n-dimensional arrays as input.

**C. SSIM value dependencies**

The computed SSIM value for a pair of images depends on a number of plot choices and SSIM parameter choices (that are independent of the floating-point data) values. Here we give a few examples to demonstrate the effect of these choices on two commonly used climate variables: surface temperature (TS) and precipitation rate (PRECT). The application dataset from which we obtained these data is described in detail in Section [IV] but for the moment it suffices to show the two images (one generated from the original data and one from data that has undergone lossy compression) that we are comparing in Figures [1] and [2]. These images are representative of the types of plots that climate scientists typically create. For these two variables, the two images being compared are very similar (SSIM = .99985 and SSIM = .99153 for TS and PRECT, respectively), and the differences cannot be seen for TS at this scale. The images and SSIM values are computed via the LDPCP Python package [24].

\[^1\text{http://https://github.com/NCAR/Ldcpy}\]
which uses the SSIM from scikit-image and generates images using matplotlib and cartopy. Also note that the TS and PRECT data used in this section are included in the ldcpy package in data/cam-fv, from files zfp1e-1.TS.100days.nc, orig.TS.100days.nc, zfp1e-7.PRECT.60days.nc, and orig.PRECT.60days.nc.

In the top half of Table I, we list the results of several modifications to demonstrate how slightly different images created from the same dataset may result in a different SSIM value. (Plots of a subset of these modifications for TS are displayed in Figure 3.) These differences in SSIM give an idea of the effect of the plotting and constant choices for these two sample variables, which have quite different characteristics: variable TS is relatively smooth and has a modest-sized range ≈ 100°K, while PRECT data contain zero and near zero values, change more abruptly, and span several orders of magnitude. Other climate variables may be more or less sensitive to these changes, but that is not the focus of this work. The first row in Table I shows the SSIM values for TS and PRECT with the default plot settings, as shown in Figures 1 and 2. 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 [18]. 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 I 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 I (row 4). The type of data projection onto a map (common for climate data) influences the SSIM as well. The equal-area map projection (row 7) actually has more of an influence for TS (Figure 3c) than for PRECT. On the other hand, the often used equirectangular projection (row 8) has no affect 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 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 [2], [19], [27], when we encode the floating-point data into a colormap, the characteristics of the colormap can influence the SSIM [19]. For example, because the prism colormap is more segmented than the default, this affects the SSIM more 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 Gaussian weights, and how the edge pixels are treated [28]. Many of these factors also affect SSIM performance [2].

Regarding the SSIM constants C1 and C2 as defined in [9], it is customary to use the recommended defaults proposed in [1]. However, contrary to claims in [1], we have found the SSIM calculation to be sensitive to the values for K1 and K2 (which in theory should only contribute to numerical stability). Our experience is supported by the previous work in [29], which analyzes SSIM value sensitivity to K1 and K2. The study in [29] shows that the SSIM is quite sensitive to K2 in particular, which is what we found with our climate data as well. In the bottom half of Table I, we list a few examples demonstrating the sensitivity to K1 and K2 on the surface temperature and precipitation rate images in Figures 1 and 2. Recall that the first row in Table I (labeled “default”) gives the SSIM values for the default values of K1 and K2. For the number of significant digits that we list, only the effects of changing K2 are noticeable.

For our use case of interest (and likely many others), the observation that the SSIM calculation as in [5] is more sensitive to K2 than K1 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, as in (6), if µx1 and µy1 are nearly the same over the small window, then the value of C1 (and therefore K1) is unimportant, and S1 will be nearly one. However, the situation is less clear for the second term in the SSIM calculation (7). The magnitude of the numerator and denominator statistics, σx1,y1 and (σx12 + σy12), respectively, may be less stable than the mean over the 11 x 11 window, in which case the value of C2 (and thus K2) becomes more influential. This consideration is important as we now move to applying the SSIM directly to the floating-point data.
III. APPLICATION TO FLOATING-POINT DATA

Recall that our primary interest in the SSIM is for evaluating the effects of lossy compression on climate simulation data. More generally, though, we want to be able to compare two datasets and evaluate whether a difference between them is likely to be noticeable in images created from the data. We demonstrated in the previous section that plot-specific (but data-independent) decisions (color, scale, axes, text, grid transform, etc.) can result in different SSIM values for images created from the same datasets. Our goal, then, is to determine whether we can apply the SSIM to the raw simulation data and obtain an indication of whether differences in the data are likely to impact a visual assessment, without committing to the creation of a specific set of images from the data with a specific tool.

Another advantage of applying a SSIM-like statistic directly to the floating-point data (rather than the image pixel values) is the reduction in computational cost. In particular, avoiding the expense of creating many plots that are not actually needed saves unnecessary computation time, especially when evaluating large amounts of data. (Cost is not likely to be an issue when working with small or moderate amounts of data."

Finally, in developing a modified SSIM for floating-point data, we were able to rethink the choices for the constants. While the constants are needed to prevent dividing by zero, the constants should not noticeably affect the SSIM values as shown in the previous section. For example, 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 will have an out-sized effect.

A few considerations of note that we do not specifically address are related to the grid that the floating-point data live on. When an image is created from the gridded data, the image may have more pixels than the number of grid points 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. If we have an unstructured grid, we would need to structure it (reorder or interpolate to a structured grid) before computing a SSIM value. Note that this step is already needed for plotting unstructured data (as well as for better compressor efficiency).

A. A straightforward approach

The naive or 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 vectors $x_i$ and $y_i$ 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.

There are a number of considerations when applying the SSIM formula directly to floating-point data instead of to pixel values. For example, the size of constants must be chosen appropriately, and the suggested values for $K_1$ and $K_2$ no longer make sense for every dataset given the wide range of possibilities now that we are not limited to pixel values. Additionally, we need to consider what to do when a NaN (or a fill value or missing value) is encountered in the data - we do not want such values to render the entire local window invalid. Another consideration is that while pixel values are typically nonnegative, floating-point values are often negative (or a mix of positive or negative). Therefore, the situation where the sign of means $\mu_x$ and $\mu_y$ are opposite can occur and cause $S_2(x_i, y_i)$ as given in [7] to be negative.

B. Data SSIM (DSSIM)

Our proposed variant of the SSIM for floating-point data, which we refer to as DSSIM, differs from the straightforward approach in a couple of ways. First, we normalize both sets (the original and that to compare) of floating-point data to the range $[0, 1]$. This choice eliminates the need for the $L$ term in the constants in [9], as we would have $L = 1$. This change also
ensures that the $S_2(x_i, y_i)$ term in (7) will be nonnegative, as is the case with the SSIM, and makes determining the constants more straightforward.

The two constants $C_1$ and $C_2$ should be the same so as not to make one term more influential than the other (we do not need different values for $K_1$ and $K_2$), and the suggested values $K_1 = .01$ and $K_2 = .03$ no longer make sense. We simply need $C_1$ and $C_2$ to be small enough to not disproportionately influence the value of the DSSIM, yet big enough to prevent dividing by zero. Therefore DSSIM uses

\begin{equation}
C_1 = 1e-8, \quad C_2 = 1e-8, \quad (10)
\end{equation}

in all cases. In addition, after normalizing the floating-point data to the range $[0, 1]$, DSSIM quantizes the data into 256 bins. This step allows DSSIM to use a similar precision on the floating-point data as SSIM is using on image data (pixels). (Recall that this work is aimed at predicting SSIM results for pseudocolor plot types.) The importance of this modification is demonstrated in the next section.

We address any NaN values (or fill or missing values, which we hereafter refer to as NaNs) present in the data when the Gaussian kernel is applied locally to each $11 \times 11$ window. If the center point of the window (grid point $i$) is NaN, that window calculation is simply excluded from the mean SSIM calculation given in (8). However, if the value at the center 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 $(x_i, y_i)$ value will be set to NaN). This needed modification for proper treatment of NaN values is available via the convolution and 2D Gaussian kernel functions in the Astropy\footnote{http://www.astropy.org} package \cite{30, 31}. In particular, for the kernel, DSSIM uses Gaussian2DKernel($x_{\text{stddev}}=1.5, \ y_{\text{size}}=11$) and then we convolve with filter $\text{args} = \text{`boundary': 'fill', 'preserve_nan': True}$. The boundary option for the convolve is not important in the current implementation as the DSSIM values whose windows extend past the boundary are ignored when computing the mean SSIM over the grid – as usual with the SSIM. (Note that scipy convolution routines do not properly deal with NaNs at this time.)

Finally, we note that the DSSIM algorithm is available in the previously mentioned LDCPY Python package (which is open source). In the next section, we will use climate simulation data to demonstrate the effectiveness of using the DSSIM instead of the SSIM.

\section{IV. Application to Climate Data}

Now we describe our investigation into whether we can use the DSSIM instead of the SSIM for evaluating the effects of lossy compression on climate data. Our focus here is on data from the Community Earth System Model (CESM)\footnote{http://pangeo.io} \cite{32}, which is a widely used and popular climate model that generates far too much data (e.g., terabytes or petabytes) – hence the interest in lossy compression and its effects. In previous work in \cite{17} and \cite{18}, we applied the SSIM to images created from CESM data with the NCAR Command Language (NCL) \cite{33}. These images were created to be similar to those generated by the Atmospheric 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\footnote{http://pangeo.io} and creating their own images. This change in post-processing analysis provided further motivation for us to use a SSIM-like measurement that was independent of plot choices.

\subsection{A. 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) \cite{53}. 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 from the Community Atmosphere Model (CAM) component, which uses a one-degree latitude-longitude grid corresponding to 192 $\times$ 288 grid points per vertical level (30 vertical levels). Each CAM variable (more than 200 total) 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).

For these experiments, we compress CESM-LENS data with the popular ZFP compressor \cite{36}. 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 (as discussed in \cite{37}). This feature will be available in ZFP 0.5.6 (to be released soon), but is currently available 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 “\texttt{cmake -DZFP_ROUNDING_MODE=ZFP_ROUND_FIRST -DZFP_WITH_TIGHT_ERROR=ON}.”

\subsection{B. Discussion of DSSIM modifications}

We compare the SSIM, the proposed DSSIM, and the straightforward implementation of a floating-point SSIM as described in Section III-A (referred to as SF-DSSIM) to demonstrate the rationale behind the two main modifications that we made for DSSIM, which are as follows. First, recall that after normalizing the data to the range $[0, 1]$, we quantize...
the data into 256 bins. Second, we modify the value of the constants to be small values that do not dominate the terms in the DSSIM. The overarching reason for these modifications is to spread out the range of DSSIM values that we attain (compared to the SSIM and SF-DSSIM) and mitigate sensitivity to the constants.

To demonstrate this spreading effect, we show plots for five popular variables from the CESM-LENS collection described in the previous subsection: TS (surface temperature), PS (surface pressure), LHFLX (surface latent heat flux), FLUT (upwelling long-wave flux at the top of model), and QFLX (surface water flux). As in Section II-C SSIM, DSSIM, and SF-DSSIM values are again computed via the LDCPY Python package with the default settings. (Note that the default plot settings for the SSIM are the same as in Table I.) In particular, we plot the SSIM, DSSIM, and SF-DSSIM values that compare the original TS, PS, LHFLX, FLUT, and QFLX datasets to those datasets compressed with ten different levels of ZFP fixed-precision compression ($p = 6, 8, 10, 12, 14, 16, 18, 20, 22, 24$). Recall that $p = 6$ is the most aggressive compressor option, and therefore should result in the smallest SSIM values for all variables, and $p = 24$ should result in values very close to one (or even equal to one). Results for each of the five variables are given in the subplots in Figure 4. For each variable, the plots on the left contain the three types of SSIM calculations for all ten compression levels, and the rightmost plots are zoomed-in versions of the plots on the left.

Looking at the plots on the left in Figure 4, it is clear that the DSSIM obtains much lower values for more aggressive compressor options (i.e., smaller values of $p$) than the SSIM. The slope of the DSSIM line is much steeper in this aggressive compressor region. The SSIM (and consequently the SF-DSSIM) never get too small as their constants dominate the other terms in the formula, preventing the value from being very small even when the images are quite different. The DSSIM behavior resulting from the smaller constants is desirable to us as intuitively we want compressor choices that are really too aggressive to have a more noticeable drop in the DSSIM value. For example, Figure 5 shows the images created by LDCPY for computing the SSIM value comparing the original data (left) to aggressively compressed data ($p = 8$). The differences between the two plots in this figure are quite obvious. From the data in Figure 5 for $p = 8$, we have SSIM $\approx 0.93$, DSSIM $\approx 0.44$, and SF-DSSIM $\approx 0.85$.

The other characteristic apparent in Figure 4 is that as the compression becomes more conservative (increasing $p$ values), the SF-DSSIM values reach 1.0 sooner than either the SSIM or DSSIM. Recall that for the SSIM, values of 1.0 imply that the images are exactly the same. When applying this calculation directly to the floating-point data, it is reasonable to not want 1.0 when the SSIM value is still below 1.0. The DSSIM, on the other hand, is more sensitive to small differences when the data is quite similar (higher values of $p$), which can be useful for identifying minor differences. This sensitivity at more conservative compression levels is a result of the quantization step in DSSIM (quantization can amplify small differences between neighboring grid points as when creating image pixels). In effect, the quantization step of the raw data helps the DSSIM produce results that are more conceptually similar to SSIM results (applied to images). Finally, it is interesting to note that 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 first). This statement may not be universally true as the SSIM is sensitive to plot choices, but
appears to hold for the LDCPY routines on this climate dataset.

C. A cutoff threshold for DSSIM

The primary reason for developing the DSSIM was as a replacement to the SSIM that would be insensitive to plotting choices and computationally faster for large amounts of data. In our work in [18], we found 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. Note that this threshold is much tighter than the generally accepted SSIM indistinguishably threshold of 0.99 (e.g., [21]) and the 0.98 suggested for medical imaging (e.g., [8], [9]). Because we are using the SSIM with a hard threshold, small effects from plotting choices can be important in regions near the threshold, and thus the need for independence from plotting decisions. However, to use the DSSIM instead of the SSIM for evaluating the CESM data, we need to determine an appropriate cutoff threshold for the DSSIM. Here, by using the results from the previous study in [18] as well as statistical techniques, we briefly support our claim that the DSSIM can be used to evaluate lossy compression artifacts in climate data with an appropriate cutoff threshold. (Note that the full details of this investigation are available in a technical report [38].)

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 find an appropriate DSSIM threshold is to treat the DSSIM as a covariate and use binary regression to model the connection between a dataset’s DSSIM value and SSIM test result. This approach (shown in [38]) gives a general idea of an appropriate threshold, but we found that using classification matrices (commonly used tools to evaluate the results of a classification model) is more useful. In our case, the classification matrix is a $2 \times 2$ matrix where the columns correspond to the true pass or fail status of the data (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 the same 10 parameters for $p$ as in the previous subsection, 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 again computed via the LDCPY Python package 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.

As explained in [38], this analysis for finding a cutoff threshold assumes that the distribution of the data in our analysis should be representative of the distribution of the data in practice. 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...
Fig. 7: The relation between various amounts of lossy compression and the compressed data size (in bytes) and DSSIM values for five 2D CESM-LENS variables: FLNS (monthly net longwave flux at the surface), PS (monthly surface pressure), FLUT (daily upwelling-longwave flux at top of model), PRECT (daily precipitation rate), and Z500 (daily geopotential at 500mbar). The horizontal axis indicates different levels of ZFP compression from most aggressive (left) to least aggressive (right). The dashed lines correspond to the DSSIM values on the right axis and the solid lines correspond to the data set sizes (for 75 years of data) on the left axis. Note that this figure is a modified version of Figure 10 in [39].

in [13] 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).

Figure 7 gives an indication of how much compression can be achieved with ZFP in fixed-precision mode for five different CESM-LENS variables, including how the data reduction corresponds to the DSSIM. We see that the data size increases linearly with the ZFP precision parameter as expected, but 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 compression. Moreover, there is clearly a limit to the amount of compression possible while maintaining a high DSSIM value. The smaller the file size, the lower the DSSIM tends to be, which can easily be seen by noting how the colors of the dashed and solid lines follow nearly the same order from top to bottom (except for the variable PS). The work in [39] provides a more in-depth examination of DSSIM behavior for climate data with varying levels of compression.

D. SSIM vs. DSSIM speedup

Finally, we show that, as expected, the DSSIM is indeed much faster than SSIM as we do not have to create images. We timed the DSSIM and SSIM using the implementations in LDCPY package via a Jupyter notebook. We used three variables from the CESM-LENS data that are included with the LDCPY package in data/cam-fv (files zfple-1.TS.100days.nc, orig.TS.100days.nc, and c.fpzip.cam-fv.T.3months.nc), where T is the 3D temperature field and PRECT is the precipitation rate. Timing results (in seconds) are given in Table II. While the actual times for these calculations will vary depending on the computing platform (these were performed on a laptop), the speedup indicates what type of performance can be gained by using the DSSIM instead of the SSIM. These time savings are important for our use case of comparing compressor results with climate data, particularly as we automate the testing and must calculate the DSSIM on large amounts of data.

TABLE II: Timings for computing the SSIM and DSSIM via the DataCalcs object in LDCPY. Times reported are the fastest of 5 executions in seconds. The first time slice is used for each variable.

| variable | dimension | SSIM (s) | DSSIM (s) | speedup |
|----------|-----------|----------|-----------|---------|
| TS       | 2D        | 7.72     | .0396     | 145x    |
| PRECT    | 2D        | 7.89     | .0373     | 212x    |
| T        | 3D        | 231      | .912      | 253x    |

V. CONCLUDING REMARKS

In this manuscript, we have proposed an alternative to the popular SSIM IQA that can be applied directly to floating-point data. The Data SSIM, or DSSIM is particularly useful for situations in which we need a general idea of whether images created from the data will be similar, but want to
remain independent of plot-specific choices that can affect the SSIM. Furthermore, applying the DSSIM to the floating-point data is much cheaper than generating images and then applying the SSIM. This reduced computational cost is quite important when analysing large volumes of data. Note that the DSSIM is available in the LDCPY (Large Data Comparison for PYthon) package [24].

The DSSIM has been beneficial to us as a means of comparing lossily compressed climate model data to uncompressed data. Prior to its development, the SSIM had become an important measurement in our compression evaluation toolkit. However, in that context, the SSIM was much more expensive to compute than all other measurements on large data volumes because of the image generation requirement. Further, the SSIM’s dependence on specific plot choices and constants was undesirable given that we were using a hard cutoff threshold and did not necessarily need to compare specific images. In practice, we have only been using the DSSIM instead of the SSIM with success in terms of correctly classifying data and saving compute time when evaluating data compression. 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 measurement in other application areas as well—especially those producing large volumes of simulation data.

ACKNOWLEDGMENT

We thank John Clyne (NCAR) for his helpful suggestions. We would also like to 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.

REFERENCES

[1] 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, vol. 13, no. 4, pp. 600–612, 2004.
[2] A. K. Venkataramanan, C. Wu, A. Bovik, I. Katsavounidis, and Z. Shahid, “A hitchhiker’s guide to structural similarity,” IEEE Access, vol. 9, pp. 28872–28896, 2021.
[3] H. R. Sheikh and A. C. Bovik, “Image information and visual quality,” IEEE Transactions on Image Processing, vol. 15, no. 2, pp. 430–444, 2006.
[4] 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, vol. 23, no. 2, pp. 684–695, 2014.
[5] E. Larson and D. Chandler, “Most apparent distortion: Full-reference image quality assessment and the role of strategy,” Journal of Electronic Imaging, vol. 19, 2010.
[6] L. Zhang, L. Zhang, X. Mou, and D. Zhang, “FSIM: A feature similarity index for image quality assessment,” IEEE Transactions on Image Processing, vol. 20, no. 8, pp. 2378–2386, 2011.
[7] 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, vol. 26, no. 1, pp. 98–117, 2009.
[8] V. T. Georgiev, A. Karahalioiu, S. Skiadopoulos, N. Arikidisa, A. Kazantz, G. Panayiotakis, and L. Costaridou, “Quantitative visually lossless compression ratio determination of JPEG2000 in digitized mammograms,” Journal of digital imaging, vol. 26, no. 3, pp. 427–439, 2012.
[9] A. Wegener, “Compression of medical sensor data,” IEEE Signal Processing Magazine, vol. 27, no. 4, pp. 125–130, July 2010.

[10] Y. Gaudeau, J. Lambert, N. Labonne, and J. Moureau, “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. IEEE, 2014, pp. 501–505.
[11] 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. IEEE, 2013, pp. 6–10.
[12] I. Kowalik-Uribaniak, D. Brunet, J. Wang, D. Koff, N. Smolarski-Koff, E. R. Vrcay, B. Wallace, and Z. Wang, “The quest for ‘diagnostically lossy’ 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.
[13] 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), 2013, pp. 343–356.
[14] J. Woodring, S. M. Mniszewski, C. M. Brislaw, D. E. DeMarle, and J. P. Ahrens, “Revisiting wavelet compression for large-scale climate data using JPEG2000 and ensuring data precision,” in IEEE Symposium on Large Data Analysis and Visualization (LDAV), D. Rogers and C. T. Silva, Eds. IEEE, 2011, pp. 31–38.
[15] 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, ser. HPDC ’14, 2014, pp. 203–214.
[16] M. Kuhn, J. Kunkel, and T. Ludwig, “Data compression for climate data,” Supercomputing frontiers and innovations, vol. 3, no. 1, pp. 75–94, 2016.
[17] 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, Springer, 2017, pp. 30–42.
[18] 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, vol. 38, no. 3, pp. 517–528, 2019.
[19] R. Veras and C. Collins, “Discriminability tests for visualization effectiveness and scalability,” IEEE Trans. Vis. Comput. Graph., vol. 26, no. 1, pp. 749–758, 2020. [Online]. Available: https://doi.org/10.1109/TVCG.2019.2934429
[20] R. Dosselmann and X. D. Yang, “A comprehensive assessment of the structural similarity index,” Signal, Image, and Video Processing, vol. 5, no. 1, pp. 81–91, 2011.
[21] J. Nilsson and T. Akenine-Moller, “Understanding ssim,” 2020.
[22] Z. Wang and A. Bovik, “A universal image quality index,” IEEE Signal Processing Letters, vol. 13, no. 3, pp. 212–215, 2006.
[23] 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, vol. 2, e453, 6 2014.
[24] A. Pinard, D. M. Hammerling, and A. H. Baker, “Assessing differences in large spatio-temporal climate datasets with a new python package,” in 2020 IEEE International Conference on Big Data (Big Data), 2020, pp. 2699–2707.
[25] J. D. Hunter, “Matplotlib: A 2d graphics environment,” Computing in Science & Engineering, vol. 9, no. 3, pp. 90–95, 2007.
[26] Met Office, Cartopy: a cartographic python library with a matplotlib interface, Exeter, Devon, 2010 - 2015. [Online]. Available: http://scitools.org.uk/cartopy
[27] M. Hassan and C. Bhagvati, “Structural similarity measure for color images,” International Journal of Computer Applications, vol. 43, pp. 7–12, 2012.
[28] 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, vol. 70, pp. 67–76, 2016.
[29] S. Sawant, “Compressed image quality measurement,” Master’s thesis, National Institute of Technology, Rourkela, India, 2013.
[30] Astropy Collaboration, T. P. Robitaille, E. J. Tollerud, P. Greenfield, M. Droettboom, E. Bray, T. Aldcroft, M. Davis, A. Ginsburg, A. M. Price-Whelan, W. E. Kerzendorf, A. Conley, N. Crighton, K. Barbary, D. M. Bray, W. Gunn, H. Fergusson, F. Grollier, M. M. Parikh, P. H. Nair, H. M. Uнтер, C. Deil, J. Woillez, S. Conseil, R. Kramer, J. E. H. Turner, L. Singer, R. Fox, B. A. Weaver, V. Zabalza, Z. I. Edwards, K. Azalee Bostroem, D. J. Burke, A. R. Casey, S. M. Crawford,
