Digital Object Identifier

A Hitchhiker's Guide to SSIM

ABHINAU K. VENKATARAMANAN[1], CHENGYANG WU[1], ALAN C. BOVIK[1](FELLOW, IEEE), IOANNIS KATSAVOUNIDIS[2], AND ZAFAR SHAHID[2]
[1]Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX 78705 USA
[2]Facebook, Menlo Park, CA 94025 USA

Corresponding author: Abhinau K. Venkataramanan (e-mail: abhinaukumar@utexas.edu).

This research was supported by funding from Facebook Video Infrastructure

ABSTRACT The Structural Similarity (SSIM) Index is a very widely used image/video quality model that continues to play an important role in the perceptual evaluation of compression algorithms, encoding recipes and numerous other image/video processing algorithms. Several public implementations of the SSIM and Multiscale-SSIM (MS-SSIM) algorithms have been developed, which differ in efficiency and performance. This "bendable ruler" makes the process of quality assessment of encoding algorithms unreliable. To address this situation, we studied and compared the functions and performances of popular and widely used implementations of SSIM, and we also considered a variety of design choices. Based on our studies and experiments, we have arrived at a collection of recommendations on how to use SSIM most effectively, including ways to reduce its computational burden.

INDEX TERMS Image/Video Quality Assessment, Structural Similarity Index

I. INTRODUCTION

With the explosion of social media platforms and online streaming services, video has become the most widely consumed form of content on the internet, accounting for 60% of global internet traffic in 2019 [1]. Social media platforms have also led to an explosion in the amount of image data being shared and stored online. Handling such large volumes of image and video data is inconceivable without the use of compression algorithms such as JPEG [2] [3], H.264 [4] [5], HEVC [6], VP9 [7] and AV1 [8].

The goal of these algorithms is to perform lossy compression of images and videos to significantly reduce file sizes and bandwidth consumption, while incurring little or acceptable reduction of visual quality. In addition to compression-distorted streaming videos, a large fraction of the images and videos that are shared on social media are User Generated Content (UGC) [9] [10], i.e., not professionally created. As a result, even without any additional processing, these images and videos can have impaired quality because they were captured by uncertain hands. In all these circumstances, it is imperative to have available automatic perceptual quality models and algorithms which can accurately, reliably, and consistently predict the subjective quality of images/videos over this wide range of applications.

One way that perceptual quality models can provide significant gains in compression is by conducting perceptual Rate-Distortion Optimization RDO [11], where quantization parameters, encoding "recipes", and mode decisions are evaluated by balancing the resulting bitrates against the perceptual quality of the decoded videos. Typically, a set of viable encodes is arrived at by constructing a perceptually-guided, Pareto-optimal bitrate ladder.

To understand Pareto-optimality, consider two encodes of a video $E_1 = (R_1, D_1)$ and $E_2 = (R_2, D_2)$, where R and D denote the bitrate and the (perceptual) distortion associated with each encode. If $R_1 \leq R_2$ and $D_1 \leq D_2$, we say that E_1 "dominates" E_2, since we obtain better performance (lower distortion) at a lower cost (bitrate). So, we can prune any set of encodes $S = \{E_i\}$, by removing all those dominated by any other encode. The pruned set, say S', has the property that for any two encodes E_1 and E_2 such that $R_1 < R_2$, $D_1 > D_2$. That is, we obtain a set of encodes such that an increase in bitrate corresponds to a decrease distortion. Such a set is said to be Pareto-optimal. In general, we can define Pareto-optimality for any setting where a cost is incurred (bitrate, running time, etc.) to achieve better performance (accuracy, distortion, etc.)

The first distortion/quality metric used to measure image/video quality was the humble Peak Signal to Noise Ratio (PSNR), log-reciprocal Mean Squared Error (MSE) between a reference video and possibly distorted versions of it, e.g., by compression. However, while the PSNR metric is simple and easy to calculate, it does not correlate very well with subjective opinion scores of picture or video quality [12].

This is because PSNR is a pixel-wise fidelity metric which does not account for spatial or temporal perceptual processes.

An important breakthrough on the perceptual quality problem emerged in the form of the Universal Quality Index (UQI) [13], the first form of the Structural Similarity Index (SSIM). Given a pair of images (reference and distorted), UQI creates a local quality map, by measuring luminance, contrast and structural similarity over local neighborhoods, then pooling (averaging) values of this spatial quality map yielding a single quality prediction (per picture or video frame). SSIM was later refined to better account for the interplay between adaptive gain control of the visual signal (the basis for masking effects) and saturation at low signal levels [14].

The SSIM concept reached its highest performance in the form of Multi-Scale SSIM (MS-SSIM) [15], which applies SSIM at five spatial resolutions obtained by successive dyadic sampling. Contrast and structure similarities are computed at each scale, while luminance similarity is only calculated at the coarsest scale. These scores are then combined using exponential weighting. SSIM and MS-SSIM are widely used by the streaming and social media industries to perceptually control the encodes of many billions of picture and video contents.

While SSIM and MS-SSIM are most commonly deployed on a frame-by-frame basis, temporal extensions of SSIM have also been developed. In [16], the authors compute frame-wise quality scores, weighted by the amount of motion in each frame. In [17], explicit motion fields are used to compute SSIM along motion trajectories, an idea that was elaborated on in the successful MOVIE index [18].

Following the success of SSIM, a great variety of picture and video quality models have been proposed. Among these, the most successful have relied on perceptually relevant “natural scene statistics" (NSS) models, which accurately and reliably describe bandpass and nonlinearly normalized visual signals [19] [20]. Distortions alter these statistics reliably, making it possible to create highly competitive picture and video quality predictors like the Full-Reference (FR) Visual Information Fidelity (VIF) index [21], the Spatio-Temporal Reduced Reference Entropic Differences (ST-RRED) model [22], the efficient SpEED-QA [23], and the Video Multi-method Assessment Fusion (VMAF) [24] model, which uses a simple learning model to fuse quality features derived from NSS models to obtain high performance and widespread industry adoption. Despite these advances, SSIM remains the most widely used perceptual quality algorithm because of its high performance, natural definition, and compute simplicity. Moreover, the success of SSIM can also be explained by NSS, at least in part [25].

In many situations, reference information is not available as a “gold standard" against which the quality of a test picture or video can be evaluated. No-Reference (NR) quality models have been developed that can accurately predict picture or video quality without a reference, by measuring NSS deviations. Notable NR quality models include BLIINDS [26], DIIVINE [27], BRISQUE [28], and NIQE [29]. The latter two, which have attained significant industry penetration, are similar to SSIM since they are defined by simple bandpass operations over multiple scales, normalized by local spatial energy. For encoding applications where the source video to be encoded is already impaired by some distortion(s), e.g. UGC, as is often found on sites like YouTube, Facebook, and Instagram, SSIM and NIQE can be combined via a 2-step assessment process to produce significantly improved encode quality predictions [30] [31].

Evaluating picture and video encodes at scale remains the most high-volume application of quality assessment, and SSIM continues to play a dominant role in this space. Nevertheless, many widely used versions of SSIM exist having different characteristics. Understanding and unifying these various implementations would be greatly useful to industry. Moreover, there remain questions regarding the use of SSIM across different display sizes, devices and viewing distances, as well as how to handle color, and how to combine (pool) SSIM scores. Our objective here is to answer these questions, at least in part.

II. BACKGROUND

The basic SSIM index is a FR picture quality model defined between two luminance images of size $M \times N$, $I_1(i,j)$ and $I_2(i,j)$ as a multiplicative combination of three terms - luminance similarity $l(i,j)$, contrast similarity $c(i,j)$ and structure similarity $s(i,j)$. Color may be considered, but we will do so later.

These three terms are defined in terms of the local means $\mu_1(i,j), \mu_2(i,j)$, standard deviations $\sigma_1(i,j), \sigma_2(i,j)$, and correlations $\sigma_{12}(i,j)$ of luminance, as follows. Let $\mathcal{W}_{ij}$ denote a windowed region of size $k \times k$ spanning the indices $\{i, \dots, i+k-1\} \times \{j, \dots, j+k-1\}$ and let $w(m,n)$ denote weights assigned to each index (m,n) of this window. In practice, these weighting functions sum to unity, and have a finite-extent Gaussian or rectangular shape.

The local statistics are then calculated on (and between) the two images as

$$\mu_1(i,j) = \sum_{m,n \in W_ij} w(m,n) I_1(m,n), \tag{1}$$

$$\mu_2(i,j) = \sum_{m,n \in W_ij} w(m,n) I_2(m,n), \tag{2}$$

$$\sigma_1^2(i,j) = \sum_{m,n \in W_ij} w(m,n) I_1^2(m,n) - \mu_1^2(i,j), \tag{3}$$

$$\sigma_2^2(i,j) = \sum_{m,n \in W_ij} w(m,n) I_2^2(m,n) - \mu_2^2(i,j), \tag{4}$$

$$\sigma_{12}(i,j) = \sum_{m,n \in W_ij} w(m,n) I_1(m,n) I_2(m,n) - \mu_1(i,j)\mu_2(i,j). \tag{5}$$

Using these local statistics, the luminance, contrast and structural similarity terms are respectively defined as

$$l(i,j) = \frac{2\mu_1(i,j)\mu_2(i,j) + C_1}{\mu_1^2(i,j) + \mu_2^2(i,j) + C_1}, \tag{6}$$

$$c(i,j) = \frac{2\sigma_1(i,j)\sigma_2(i,j) + C_2}{\sigma_1^2(i,j) + \sigma_2^2(i,j) + C_2}, \tag{7}$$

$$s(i,j) = \frac{\sigma_{12}(i,j) + C_3}{\sigma_1(i,j)\sigma_2(i,j) + C_3}, \tag{8}$$

where C_1, C_2 and C_3 are saturation constants that contribute to numerical stability. Local quality scores are then defined as

$$Q(i,j) = l(i,j) \cdot c(i,j) \cdot s(i,j). \tag{9}$$

Adopting the common choice of $C_3 = C_2/2$, the contrast and structure terms combine:

$$cs(i,j) = c(i,j)s(i,j) = \frac{2\sigma_{12}(i,j) + C_2}{\sigma_1^2(i,j) + \sigma_2^2(i,j) + C_2} \tag{10}$$

In this way, a SSIM quality map $Q(i,j)$ is defined, which can be used to visually localize distortions. Since a single picture quality score is usually desired, the average value of the quality map can be reported as the Mean SSIM (MSSIM) score between the two images:

$$SSIM(I_1, I_2) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} Q(i,j). \tag{11}$$

SSIM obeys the following desirable properties.

1) Symmetry: $SSIM(I_1, I_2) = SSIM(I_2, I_1)$
2) Boundedness: $|SSIM(I_1, I_2)| \leq 1$[1]
3) Unique Maximum: $SSIM(I_1, I_2) = 1 \iff I_1 = I_2$

An important property of SSIM is that it accounts for the perceptual phenomenon of Weber's Law, whereby a Just Noticeable Difference (JND) is proportional to the local neighborhood property Q. This is the basis for perceptual masking of distortions, whereby the visibility of a distortion ΔQ is mediated by the relative perturbation $\Delta Q/Q$.

To illustrate the connection between SSIM and Weber masking, consider an error $\Delta\mu$ of local luminance μ_1 in a test image:

$$\mu_2 = \mu_1 + \Delta\mu = \mu_1(1 + \Lambda), \tag{12}$$

where $\Lambda = \Delta\mu/\mu_1$ is the relative change in luminance. Then, the luminance similarity term (6) becomes (dropping spatial indices)

$$l = \frac{2\mu_1\mu_2 + C_1}{\mu_1^2 + \mu_2^2 + C_1} \tag{13}$$

$$= \frac{2\mu_1^2(1+\Lambda) + C_1}{\mu_1^2\left(1 + (1+\Lambda)^2\right) + C_1} \tag{14}$$

$$= \frac{2(1+\Lambda) + C_1/\mu_1^2}{1 + (1+\Lambda)^2 + C_1/\mu_1^2}. \tag{15}$$

[1]Very rarely, distortion can cause a negative correlation between reference and test image patches.

Since it is usually true that $C_1 \ll \mu_1^2$, the luminance term l is approximately only a function of the relative luminance change, reflecting luminance masking.

Similarly, a locally perturbed contrast σ_1 in a test image may be expressed as $\sigma_2 = (1+\Sigma)\sigma_1$, where $\Sigma = \Delta\sigma/\sigma_1$ is the relative change in contrast from distortion. Similar to the above, we can express the contrast term (7) as

$$c = \frac{2(1+\Sigma) + C_2/\sigma_1^2}{1 + (1+\Sigma)^2 + C_2/\sigma_1^2}. \tag{16}$$

Since generally $C_2 \ll \sigma_1^2$, the contrast term c is approximately a function of the relative, rather than absolute, change in contrast, thereby accounting the perceptual contrast masking.

Given an 8-bit luminance image, assume the nominal dynamic range $[0, L]$, where $L = 255$. Most commonly, the saturation constants are chosen relative to the dynamic range as $C_1 = (K_1L)^2$ and $C_2 = (K_2L)^2$, where K_1 and K_2 are small constants.

SSIM is quite flexible and allows room for design choices. The recommended implementation of SSIM in [14] is

- If $\min(M,N) > 256$, resize images such that $\min(M,N) = 256$.
- Use a Gaussian weighting window in (1) - (5) having $k = 11$ and $\sigma = 1.5$.
- Choose regularization constants $K_1 = 0.01, K_2 = 0.03$.

One of our goals here is to compare and test different, commonly used implementations of SSIM and MS-SSIM, which make different design choices. We conduct performance evaluations on existing, well-regarded image and video quality databases. We study the effects of several design choices and make recommendations on best practices when utilizing SSIM.

III. DATABASES

One of our main goals is to help "standardize" the way SSIM is defined and used. Since many versions of SSIM exist, and implementing SSIM involves design choices, reliable and accurate subjective test beds that capture the breadth of theoretical and practical distortions are the indispensable tools for our analysis. Among these, we selected two picture quality databases and two video quality databases that are widely used.

A. LIVE IMAGE QUALITY ASSESSMENT DATABASE

The LIVE IQA database [32] [33] contains 29 reference pictures, each subjected to the following five distortions (each at four levels of severity).

- JPEG compression
- JPEG2000 compression
- Gaussian blur
- White noise
- Bit errors in JPEG2000 bit streams

LIVE IQA contains 982 pictures with nearly 30,000 corresponding Difference Mean Opinion (DMOS) human subject scores.

B. TAMPERE IMAGE DATABASE 2013

The Tampere Image Database 2013 (TID2013) [34] contains 3000 distorted pictures subjected to 24 impairments at 5 distortion levels synthetically applied to 25 pristine images. The 24 distortions are

- Additive Gaussian noise
- Additive noise in color components is more intensive than additive noise in the luminance component
- Spatially correlated noise
- Masked noise
- High frequency noise
- Impulse noise
- Quantization noise
- Gaussian blur
- Image denoising
- JPEG compression
- JPEG2000 compression
- JPEG transmission errors
- JPEG2000 transmission errors
- Non eccentricity pattern noise
- Local block-wise distortions of different intensity
- Mean shift (intensity shift)
- Contrast change
- Change of color saturation
- Multiplicative Gaussian noise
- Comfort noise
- Lossy compression of noisy images
- Image color quantization with dither
- Chromatic aberrations
- Sparse sampling and reconstruction

The 3000 pictures in TID 2013 also have human subject scores along with over 500,000 human subjective MOS.

C. LIVE VIDEO QUALITY ASSESSMENT DATABASE

The LIVE VQA database [35] contains 10 reference videos, each subjected to the following four distortions (each applied at four levels of severity).

- MPEG-2 compression
- H.264 compression
- Error-prone IP networks
- Error-prone wireless networks

A total of 150 distorted videos are obtained, on which 4350 subjective DMOS were obtained.

D. NETFLIX PUBLIC DATABASE

The Netflix Public Database, obtained from the VMAF [24] Github repository, contains 9 reference videos, each distorted by spatial scaling and compression, yielding 70 distorted videos. We selected this database because of its high relevance and commonly observed distortions characteristic of SSIM streaming video deployments at the largest scales.

IV. VERSIONS OF SSIM

Next, we take a deep dive into publicly available and commonly used implementations of SSIM and MS-SSIM. We compare various aspects of their performance, explain the differences between them, and provide recommendations for best practices when using SSIM. This is especially important because, as we will see, subtle differences in design choices can lead to significant changes in performance and efficiency.

A. PUBLIC SSIM AND MS-SSIM IMPLEMENTATIONS

We considered the following ten SSIM implementations when carrying out our experiments:

1) FFMPEG [36]
2) LIBVMAF [24]
3) VideoClarity ClearView Player (ClearView)
4) Avisynth HDRTools plugin (HDRTools) [37]
5) Daala [38]
6) Scikit-Image in Python (Block) [39]
7) Scikit-Image in Python (Gaussian) [39]
8) Scikit-Video in Python (Block) [40]
9) Scikit-Video in Python (Gaussian) [40]
10) Tensorflow in Python [41]
11) MATLAB
12) MATLAB (Fast)

"Block" refers to using a rectangular (constant weight) window function to calculate local statistics, while "Gaussian" refers to using a Gaussian-shaped window function to calculate local statistics, as in [14]. Only the Python and MATLAB implementations allow the user to set parameters such as the SSIM window size. Hence, we tested the other implementations using the default parameters. "Fast" in item 12 refers to an accelerated implementation of SSIM in MATLAB.

In addition, the following eight MS-SSIM implementations were tested:

1) LIBVMAF
2) ClearView
3) HDRTools
4) Daala
5) Daala (Fast)
6) Scikit-Video in Python (Sum)
7) Scikit-Video in Python (Product)
8) Tensorflow in Python

"Sum" and "Product" in 6 and 7 refer to different ways of aggregating SSIM scores across scales. "Product" refers to the method proposed in [15], where MS-SSIM is computed as an exponentially-weighted product of SSIM scores from each scale. In "Sum", the MS-SSIM score is instead a weighted average of SSIM scores across scales. "Fast" refers to an accelerated implementation of MS-SSIM in Daala.

B. SALIENT FEATURES OF SSIM AND MS-SSIM IMPLEMENTATIONS

The salient features of various SSIM implementations are listed below. In some cases, we point out errors that we have found in the implementations, and the effect of correcting these errors.

1) FFMPEG
   a) An 8x8 averaging window with a stride of 4 is used to calculate local statistics.
   b) Input images are processed at their native resolutions.
   c) The first regularization constant C_1 is calculated incorrectly. Fixing this resulted in a negligible change (sometimes a small decrease of the order of 10^{-3}) in correlations against human subjective scores.
2) LIBVMAF
   a) An 11x11 Gaussian window was used to calculate local statistics.
   b) Computes SSIM only on the luma channel.
   c) Images are down-sampled by a factor of $\max(1, [\min(W,H)/256)]$, where $[\cdot]$ denotes the rounding operation. This is achieved by low-pass filtering the image using an averaging (rectangular window) filter.
   d) Only valid convolution outputs are considered when calculating local statistics, to avoid border effects.
3) ClearView
   a) A Gaussian window is used to calculate local statistics.
   b) Features both a graphical interface for content viewing and command line tools for batch execution.
   c) Functionalities are restricted to several fixed screen sizes: 4K, 2K, 1080p, 720p, etc. Contents of arbitrary resolutions are not supported.
   d) Uses thresholds to filter out frame scores. These thresholds can be modified.
4) HDRTools
   a) Uses 8x8 Gaussian windows by default, but the size of the window can be configured.
   b) A Tensorflow implementation of SSIM is integrated.
5) Daala
   a) The Gaussian window is chosen such that the value of the window at the point of truncation is less than 0.5. It can be shown that the length corresponding to this is $k = 2 * (\sigma\sqrt{-2\log(\sigma\sqrt{\frac{\pi}{2}})) + 1$.
   b) The standard deviation of the Gaussian is $\sigma = 1.5 * h/256$, instead of down-sampling the images to have height 256 and using $\sigma = 1.5$.
   c) Only valid convolution outputs are considered when calculating local statistics, to avoid border effects.
6) Scikit-Image
   a) Allows choosing either a rectangular or a Gaussian window of specified size and value of σ to calculate local statistics.
   b) Input images are processed at their native resolutions.
   c) When using a rectangular window to compute local statistics, the input is extended by reflecting the picture about its edges to obtain an output the same size as the input.
7) Scikit-Video
   a) Any window function can be used to calculate local statistics.
   b) Images are down-sampled by a factor of $\max(1, [\min(W,H)/256])$ by low-pass filtering the image using an averaging filter and down-sampling.
   c) The algorithm uses zero-padding and always crops 5 pixels on all sides to avoid border artifacts when calculating local statistics, irrespective of the size of the filter.
8) Tensorflow
   a) A Gaussian window of any size can be used to calculate the local statistics.
   b) Input images are processed at their native resolutions.
   c) Only valid convolution outputs are considered when calculating local statistics, to avoid border effects.
9) Custom
   a) A custom python implementation that we created to have more fine-grained control over design choices. This is publicly available here.
   b) A rectangular window is used to calculate local statistics, since it produces more accurate predictions.
   c) The rectangular window is implemented using integral images.
10) MATLAB
   a) A Gaussian window of any size can be used to calculate the local statistics.
   b) The window size is inferred from the standard deviation σ as $2 \times \lceil 3\sigma \rceil + 1$.
11) MATLAB (Fast)
   a) Uses block averaging and integer arithmetic to accelerate MATLAB SSIM.

The salient features of various MS-SSIM implementations have been listed below.

1) LIBVMAF
   a) Dyadic down-sampling is performed using a 9/7 biorthogonal wavelet filter.
2) Daala
   a) Uses $\sigma = 1.5$, leading to a Gaussian window of size 11.

TABLE 1: Off-the-shelf performance of SSIM implementations

(a) Performance of SSIM implementations

Implementation	LIVE IQA			TID 2013			LIVE VQA		
	PCC	SROCC	1-RMSE	PCC	SROCC	1-RMSE	PCC	SROCC	1-RMSE
FFMPEG	**0.942**	**0.931**	**0.908**	0.707	0.658	0.874	0.603	0.599	0.826
LIBVMAF	**0.946**	**0.946**	**0.912**	**0.802**	**0.755**	**0.894**	**0.700**	**0.695**	**0.844**
Daala SSIM	0.940	0.929	0.907	0.701	0.657	0.873	0.621	0.618	0.829
ClearView	0.791	0.789	0.883	**0.718**	**0.683**	**0.876**	0.455	0.376	0.805
HDRTools	0.845	0.831	0.898	0.667	0.605	0.868	0.471	0.452	0.807
Scikit-Image (Block)	0.942	0.930	0.908	0.692	0.639	0.872	**0.665**	**0.668**	**0.837**
Scikit-Image (Gauss)	0.939	0.925	0.906	0.677	0.628	0.869	0.563	0.549	0.819
Scikit-Video (Block)	0.942	0.930	0.908	0.692	0.640	0.872	**0.665**	**0.668**	**0.837**
Scikit-Video (Gauss)	0.939	0.925	0.906	0.677	0.628	0.869	0.562	0.549	0.819
Custom	**0.946**	**0.934**	**0.911**	**0.711**	**0.661**	**0.875**	0.656	0.661	0.835
MATLAB (Fast)	0.864	0.851	0.904	0.687	0.627	0.871	0.577	0.531	0.821
MATLAB	0.862	0.848	0.903	0.686	0.624	0.871	0.563	0.521	0.819
Tensorflow	0.939	0.925	0.906	0.677	0.628	0.869	0.562	0.549	0.819

(b) Performance of MS-SSIM Implementations

Implementation	LIVE IQA			TID 2013			LIVE VQA		
	PCC	SROCC	1-RMSE	PCC	SROCC	1-RMSE	PCC	SROCC	1-RMSE
LIBVMAF	0.943	0.946	0.909	**0.837**	**0.788**	**0.903**	0.737	0.732	0.852
Daala	**0.946**	0.949	0.911	0.832	**0.792**	0.901	0.739	0.734	0.853
Daala Fast-SSIM	0.942	0.940	0.909	0.782	0.728	0.889	0.654	0.631	0.835
ClearView	0.871	0.870	0.906	0.749	0.699	0.882	0.654	0.627	0.835
HDRTools	0.904	0.906	**0.919**	**0.834**	**0.789**	**0.902**	0.733	0.726	0.851
Scikit-Video (Product)	0.944	**0.954**	0.910	0.813	0.763	0.896	**0.756**	**0.748**	**0.857**
Scikit-Video (Sum)	0.943	**0.954**	0.909	0.810	0.762	0.896	**0.760**	**0.748**	**0.858**
Custom	**0.947**	**0.951**	**0.912**	0.825	0.778	0.899	**0.760**	**0.751**	**0.858**
MATLAB	0.902	0.905	0.918	0.829	0.782	0.901	0.737	0.729	0.852
Tensorflow	**0.947**	0.951	0.912	**0.833**	0.786	**0.902**	0.745	0.743	0.854

b) Dyadic down-sampling is performed by 2x2 average pooling.

3) Daala (Fast)
   a) Multiscale processing is performed at 4 levels. The first four exponents used in the standard MS-SSIM formulation are renormalized to sum to 1.
   b) An integer approximation to Gaussian is used.
   c) Dyadic down-sampling is performed by 2x2 average pooling.
   d) When image dimensions were not multiples of 16, we found that this implementation suffered from memory leaks, which led to a considerable decrease in accuracy. A simple fix restored performance to expected levels.
4) Scikit-Video
   a) Dyadic down-sampling is performed by low pass filtering using a 2x2 average filter, followed by down-sampling.
   b) Allows aggregating across scales by summation instead of product. For summation, the exponents β_i are normalized to sum to 1, leading to a convex combination.
   c) At the coarsest scale, the algorithm uses $\alpha_M = \beta_M = \gamma_M = 1$.
5) Tensorflow
   a) Dyadic down-sampling is carried out by average pooling 2x2 neighborhoods.

C. OFF-THE-SHELF PERFORMANCE USING DEFAULT PARAMETERS

In this section, we evaluate the off-the-shelf performance of the implementations discussed above. We first normalized the subjective scores of pictures/videos each database to the range $[0, 1]$ by scaling and shifting. In all of the experiments in this section, unless mentioned otherwise, we computed SSIM scores only on the luma channel.

It is well known that the relationship between SSIM (or any other quality metric) and subjective scores is non-linear. To account for this, we fit the five-parameter logistic (5PL) function [42] shown in (17) from SSIM values to subjective scores:

$$Q(x) = \beta_1\left(\frac{1}{2} - \frac{1}{1+\exp(\beta_2(x-\beta_3))}\right) + \beta_4 x + \beta_5. \quad (17)$$

After linearizing the SSIM values in this manner, we reported the Pearson Correlation Coefficient (PCC) which is a measure of the linear correlation between the predicted and true quality, the Spearman Rank Order Correlation Coefficient (SROCC) which is a measure of the rank correlation (monotonicity), and the Root Mean Square Error (RMSE) which is a measure of the error in predicting subjective quality.

Table 1 shows the results of the experiments, and the best three results in each column have been highlighted. Among SSIM implementations, LIBVMAF generally outperformed all other algorithms. The Custom Python implementation is also generally among the top-three implementations.

TABLE 2: Off-the-shelf performance of SSIM and MS-SSIM implementations on compression

(a) Performance of SSIM Implementations

Implementation	LIVE IQA (Comp)			TID 2013 (Comp)			LIVE VQA (Comp)			Netflix Public		
	PCC	SROCC	1-RMSE	PCC	SROCC	1-RMSE	PCC	SROCC	1-RMSE	PCC	SROCC	1-RMSE
FFMPEG	0.970	0.967	0.930	0.938	0.926	0.918	0.652	0.652	0.845	0.696	0.657	0.797
LIBVMAF	0.941	0.954	0.902	**0.961**	**0.947**	**0.935**	**0.689**	**0.684**	**0.852**	**0.768**	**0.765**	**0.819**
Daala SSIM	0.970	0.968	0.930	0.938	0.926	0.919	0.657	0.658	0.846	**0.707**	**0.680**	**0.800**
ClearView	0.856	0.851	0.901	0.863	0.854	0.881	0.532	0.352	0.827	0.590	0.538	0.772
HDRTools	0.912	0.901	0.922	0.895	0.903	0.895	0.579	0.532	0.833	0.589	0.558	0.772
Scikit-Image (Block)	**0.971**	**0.969**	**0.932**	0.944	0.928	**0.923**	**0.686**	**0.691**	**0.851**	0.681	0.656	0.793
Scikit-Image (Gauss)	0.969	0.965	0.929	0.924	0.917	0.910	0.637	0.622	0.842	0.642	0.621	0.783
Scikit-Video (Block)	**0.971**	**0.969**	**0.931**	**0.944**	**0.928**	**0.923**	**0.680**	**0.690**	**0.851**	0.681	0.657	0.793
Scikit-Video (Gauss)	0.969	0.965	0.929	0.924	0.917	0.910	0.636	0.624	0.842	0.642	0.621	0.783
Custom	**0.972**	**0.969**	**0.930**	**0.946**	**0.930**	0.906	0.670	0.680	0.836	**0.787**	**0.778**	**0.826**
MATLAB (Fast)	0.930	0.917	0.930	0.924	0.917	0.910	0.642	0.609	0.843	0.645	0.596	0.784
MATLAB	0.928	0.915	0.929	0.918	0.913	0.907	0.630	0.610	0.842	0.626	0.592	0.780
Tensorflow	0.969	0.965	0.929	0.924	0.917	0.910	0.636	0.624	0.842	0.642	0.621	0.783

(b) Performance of MS-SSIM Implementations

Implementation	LIVE IQA (Comp)			TID 2013 (Comp)			LIVE VQA (Comp)			Netflix Public		
	PCC	SROCC	1-RMSE	PCC	SROCC	1-RMSE	PCC	SROCC	1-RMSE	PCC	SROCC	1-RMSE
LIBVMAF	0.931	0.956	0.895	0.967	0.943	0.940	0.693	0.694	0.852	0.754	0.739	0.814
Daala	0.936	**0.962**	0.898	**0.968**	**0.952**	**0.941**	0.692	0.696	0.852	0.755	0.741	0.815
Daala Fast-SSIM	**0.949**	0.947	0.909	0.926	0.900	0.911	0.595	0.565	0.835	**0.769**	0.749	**0.819**
ClearView	0.945	0.927	**0.938**	0.945	0.929	0.923	**0.714**	**0.696**	**0.857**	**0.812**	**0.794**	**0.835**
HDRTools	**0.946**	0.927	**0.938**	**0.974**	**0.957**	**0.947**	**0.698**	0.683	**0.854**	0.755	0.738	0.815
Scikit-Video (Product)	0.930	0.962	0.894	**0.967**	**0.950**	**0.940**	0.692	0.686	0.852	0.755	0.752	0.815
Scikit-Video (Sum)	**0.961**	**0.967**	0.920	0.967	0.949	0.940	0.697	0.688	0.853	0.754	**0.751**	0.814
Custom	0.928	0.959	0.888	0.955	0.943	0.914	0.696	**0.696**	0.841	**0.774**	**0.762**	**0.821**
MATLAB	0.942	0.925	**0.936**	0.966	0.949	0.939	**0.701**	0.685	**0.854**	0.753	0.739	0.814
Tensorflow	0.942	**0.962**	0.904	0.966	0.949	0.939	0.698	**0.701**	0.853	0.754	0.746	0.814

Among the MS-SSIM implementations, there was no consistent "winners." Python implementations like Scikit-Video and the Custom implementation were often among the top-three implementations. Tensorflow's MS-SSIM implementation also performs well, lending strong empirical support to the use of MS-SSIM as a training objective for deep networks implemented in Tensorflow.

Since compression forms an important class of distortions encountered in practice, we also report the off-the-shelf performance of SSIM and MS-SSIM implementations on compression distorted data in Table 2. When restricting the comparisons to compression, the LIBVMAF and Custom implementations still generally outperforms other SSIM implementations, while HDRTools and ClearView generally outperform other MS-SSIM implementations.

D. PERFORMANCE-EFFICIENCY TRADEOFFS

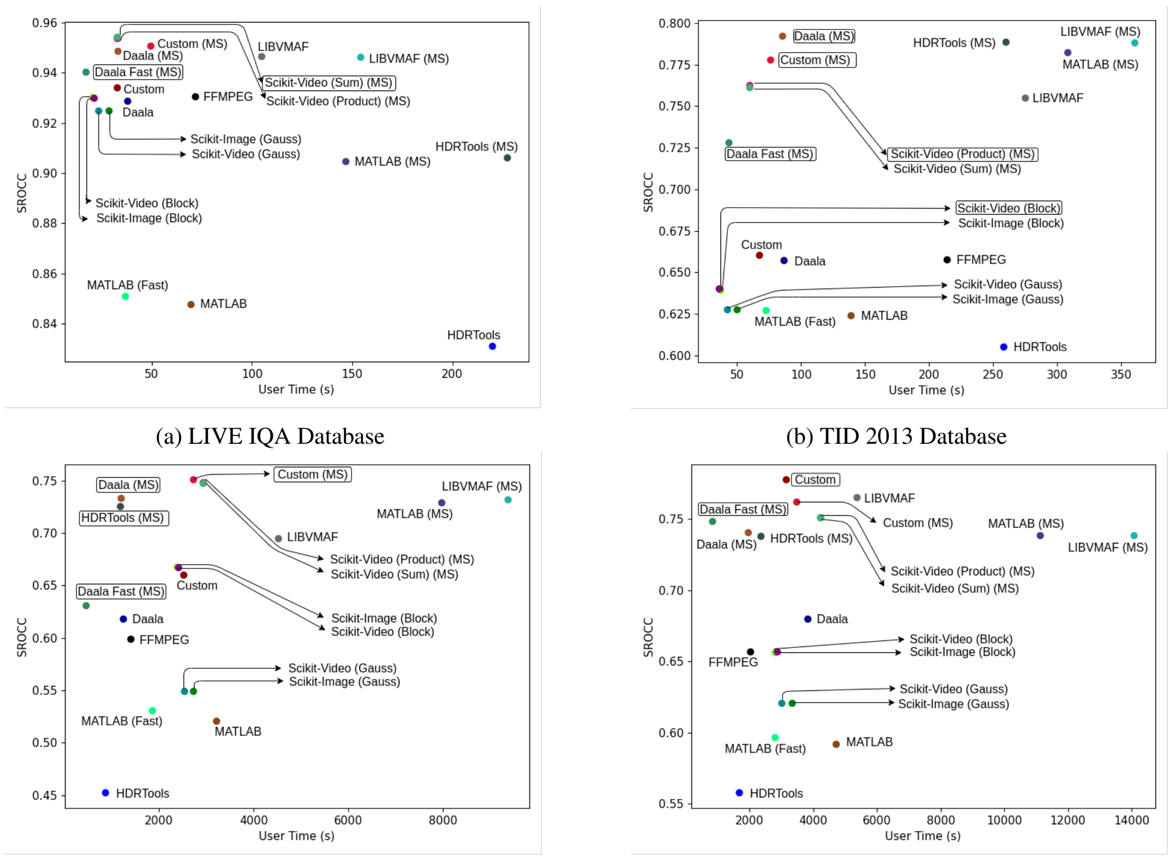
In addition to performance (i.e., correlation with subjective scores), it is important to consider the compute efficiency of these implementations. Algorithms that employ sophisticated techniques for down-sampling, calculation of local statistics and multi-scale processing may provide improvements in performance, but often incur the cost of additional computational complexity. When deployed at scale, these additional costs can be significant.

To evaluate the compared algorithms in the context of this performance-efficiency tradeoff, we plotted the SROCC achieved by each algorithm against their execution time. Since some methods leverage multithreading/multiprocessing, we report the user time instead of the wall time of the processes.

As with any run-time experiments, we expected to observe slight variations in execution times between runs due to varying system conditions. To account for this, we ran every SSIM implementation on each database five times and recorded the total execution time of each run. We then reported the median run-time over the five runs.

We omitted Tensorflow implementations from these experiments because they run prohibitively slowly on CPUs, and we cannot compare their GPU run-times while all other implementations are run on the CPU. we also omit ClearView implementations because we had to run them on custom hardware. The results on each database are shown in Fig. 1, where the Pareto-optimal implementations have been circled.

From these plots, it may be observed that among the implementations that we tested, Daala's Fast MS-SSIM implementation is Pareto-optimal most often, followed the Custom MS-SSIM implementation. Among the SSIM implementations, Daala, LIBVMAF and the Custom implementations were often Pareto-optimal across databases. Note that while the concept of Pareto-optimality is often used in the context of "optimizing" an encode in a rate-distortion sense by varying a parameter, no parameters were optimized during our experiments. In our setting, an implementation is Pareto-optimal among the set of considered implementations if there is no implementation that both achieves a higher SROCC and



(a) LIVE IQA Database

(b) TID 2013 Database

(c) LIVE VQA Database

(d) Netflix Public Database

FIGURE 1: Correlation vs execution time

runs in lesser time.

The nominal computational complexity of SSIM is $O(MNk^2)$. We propose a method to improve the efficiency of SSIM if the weighting function is rectangular, i.e., $w(i,j) = 1/k^2$, by using integral images, also known as summed-area tables [43] [44].

This can be done by forming five integral images as follows:

$$I_1^{(1)}(i,j) = \begin{cases} \sum\limits_{m\le i}\sum\limits_{n\le j} I_1(m,n) & i,j>0 \\ 0 & \text{otherwise} \end{cases}, \tag{18}$$

$$I_2^{(1)}(i,j) = \begin{cases} \sum\limits_{m\le i}\sum\limits_{n\le j} I_2(m,n) & i,j>0 \\ 0 & \text{otherwise} \end{cases}, \tag{19}$$

$$I_1^{(2)}(i,j) = \begin{cases} \sum\limits_{m\le i}\sum\limits_{n\le j} I_1^2(m,n) & i,j>0 \\ 0 & \text{otherwise} \end{cases}, \tag{20}$$

$$I_2^{(2)}(i,j) = \begin{cases} \sum\limits_{m\le i}\sum\limits_{n\le j} I_2^2(m,n) & i,j>0 \\ 0 & \text{otherwise} \end{cases}, \tag{21}$$

$$I_{12}(i,j) = \begin{cases} \sum\limits_{m\le i}\sum\limits_{n\le j} I_1(m,n)I_2(m,n) & i,j>0 \\ 0 & \text{otherwise} \end{cases}, \tag{22}$$

Given the integral image $I_1^{(1)}$, calculate the sum in any $k\times k$ window $\mathcal{W}_{ij}$ in constant time via

$$\begin{aligned} S_1^{(1)}(i,j) = {} & I_1^{(1)}(i+k-1,j+k-1) + I_1^{(1)}(i-1,j-1) \\ & - I_1^{(1)}(i+k-1,j-1) - I_1^{(1)}(i-1,j+k-1). \end{aligned} \tag{23}$$

This operation would require $O(k^2)$ time without the use of an integral image. Similarly, calculate local sums using the other integral images, and denote them $S_2^{(1)}(i,j)$, $S_1^{(2)}(i,j)$, $S_2^{(2)}(i,j)$, and $S_{12}(i,j)$, respectively. Then calculate the necessary local statistics as

$$\mu_1(i,j) = S_1^{(1)}(i,j)/k^2, \tag{24}$$

$$\mu_2(i,j) = S_2^{(1)}(i,j)/k^2, \tag{25}$$

$$\sigma_1^2(i,j) = S_1^{(2)}(i,j)/k^2 - \mu_1^2(i,j), \tag{26}$$

$$\sigma_2^2(i,j) = S_2^{(2)}(i,j)/k^2 - \mu_2^2(i,j), \tag{27}$$

$$\sigma_{12}(i,j) = S_{12}(i,j)/k^2 - \mu_1(i,j)\mu_2(i,j). \tag{28}$$

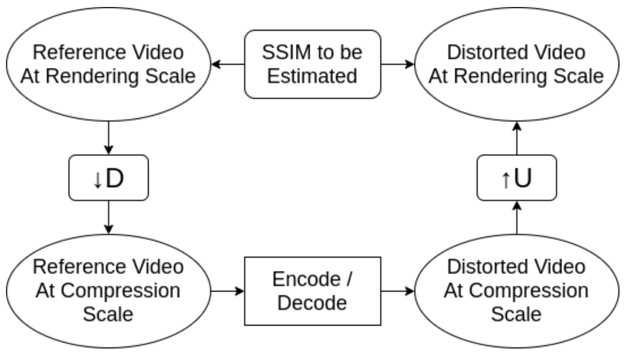
In this new way of computing rapid SSIM index, which is applicable when using a rectangular SSIM window, the compute complexity of SSIM is reduced to $O(MN)$.

V. SCALED SSIM

Arguably the most widespread use of SSIM (and other picture/video quality models) is in evaluating the quality of compression encodes. On streaming and social media platforms, pictures and videos are commonly encoded at lower resolutions for transmission. This is done either because the source has low-complexity content and can be down-sampled with relatively little additional loss (or if the available bandwidth requires it) or to decrease the decoding load at the user's end. Perceptual distortion models have become common tools for determining the quality of encodes for Rate-Distortion Optimization (RDO) [11]. Advances in video hardware have enabled the accelerated encoding and decoding of videos, making the distortion estimation step the bottleneck when optimizing encoding "recipes." From Section IV-D, we know that the computational complexity of SSIM in terms of image dimensions is $O(MN)$. Including a scale factor α by which we resize the image, the computational complexity is $O(\alpha^2 MN)$. Due to this quadratic growth, the computational load of distortion estimation is an increasingly relevant issue given the prevalence of high-resolution videos.

Therefore, it is of great interest to be able to accurately predict the quality of high-resolution videos that are distorted in two steps - scaling followed by compression. For example, consider High Definition (HD) videos that are first resized to a lower resolution, which we call the compression resolution, then encoded and decoded using, for example, H.264 at this compression resolution. The videos are then up-sampled to the original resolution before they are rendered for display. We will refer to this higher resolution as the rendering resolution.

We aim to reduce the computational burden of perceptually-driven RDO by circumventing the computation of SSIM at the rendering resolution, i.e. between the HD source and rendered videos. We propose a suite of algorithms, called Scaled SSIM in [45], which predict SSIM by only using SSIM values computed at the lower compression resolution during runtime. The video compression pipeline in which we solve the Scaled SSIM problem is illustrated in Fig. 2.


Reference Video At Rendering Scale
SSIM to be Estimated
Distorted Video At Rendering Scale
↓D
↑U
Reference Video At Compression Scale
Encode / Decode
Distorted Video At Compression Scale
FIGURE 2: Video compression pipeline

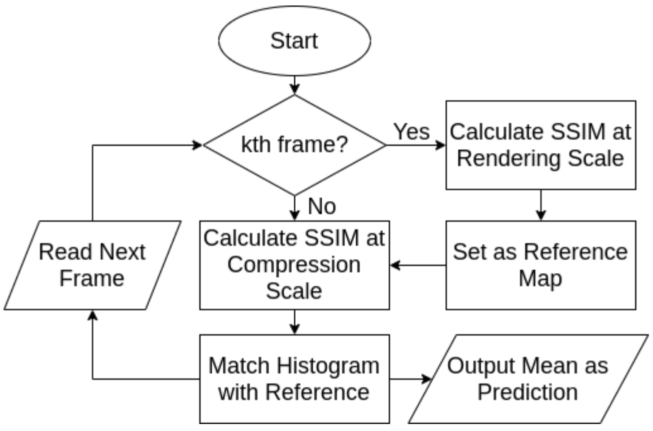
We achieve this using two classes of models that efficiently predict Scaled SSIM, which we refer to as

- Histogram Matching
- Feature-based models

All of the proposed models operate on a per-frame basis.

We first trained and tested the performance of these models on an in-house video corpus of 60 pristine videos. We compressed these videos at 6 compression scales - 144p, 240p, 360p, 480p, 540p, 720p using FFMPEG's H.264 (libx264) encoder at 11 choices of the Quantization Parameter (QP) - $1, 6, 11, \ldots, 51$. In this manner, we obtained a total of 3960 videos having almost 1.75M frames.

On this corpus, we can evaluate the accuracy of predicting SSIM scores, i.e., the correlation between predicted and true SSIM, which was computed using the "ssim" filter in FFMPEG. However, the end goal is to predict subjective scores, which are not available for this corpus. So, we instead evaluated the performance of our models against subjective scores on the Netflix Public Database.


Start
kth frame?
Yes
No
Calculate SSIM at Rendering Scale
Read Next Frame
Calculate SSIM at Compression Scale
Set as Reference Map
Match Histogram with Reference
Output Mean as Prediction
FIGURE 3: Histogram Matching Solution

A. HISTOGRAM MATCHING

We observe a non-linear relationship between SSIM values across encoding resolutions. Because framewise SSIM scores are calculated by averaging the local quality map obtained from SSIM, we can estimate SSIM scores by matching the histograms of these quality maps. However, to match histograms, we require the true histogram at the rendering resolution, which is what we wish to *avoid* estimating.

So, we instead calculate the "true" quality map just once every k frames, and assume that the shape of the true histogram does not change significantly over a short period of $k-1$ frames. This allows us to reuse this "reference map" for the next $k-1$ frames as a heuristic model against which we match the shapes of the next $k-1$ histograms at the compression scale. This histogram matching algorithm is illustrated in Fig. 3.

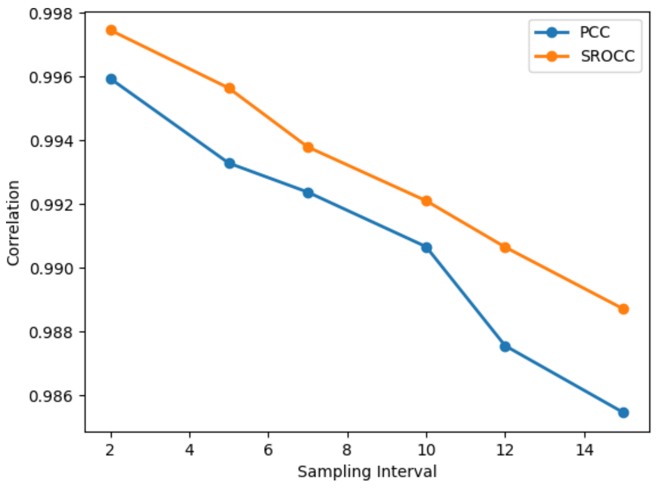


FIGURE 4: Correlation vs Sampling interval for Histogram Matching

Let $\alpha \in (0,1)$ be the factor by which we down-sampled the source video. Then, the ratio of required computation using our proposed approach to SSIM computation directly at the rendered scale is approximately

$$\left(1-\frac{1}{k}\right)\alpha^2\left(1+\beta+\gamma\right)+\frac{1}{k}(1+\beta). \tag{29}$$

The factors β and γ account for computing and matching the histograms respectively, which are both $O(MN)$ operations. This ratio is a decreasing function of k, and approaches $\alpha^2(1+\beta+\gamma)$ as $k \to \infty$.

By comparison, if the rendered SSIM map were not sampled, the ratio would be approximately α^2. In practice, we have observed that the time taken to compute and match histograms is comparable to the time taken to compute the SSIM map at the compression scale. So, the computational burden of the matching step is small, albeit not negligible.

This reduction in computational complexity as k increases is accompanied by a reduction in performance (accurate prediction of true SSIM), as shown in Fig. 4. We chose $k = 5$ in all the experiments unless otherwise mentioned.

One drawback of this method is that it requires "guiding information" in the form of periodically updated reference quality maps. However, this issue is not a factor in the second class of models.

B. FEATURE-BASED MODELS

As we observed earlier, the net quality degradation occurs in two steps - scaling and compression. So, we calculate the contribution of each operation and use these as features to calculate the net distortion.

Let X be a source video and $S_\alpha(X)$ denote the video obtained by scaling X by a factor α. Then, the result of up-sampling the down-sampled video back to the original resolution may be denoted by $S_{\frac{1}{\alpha}}(S_\alpha(X))$. The SSIM value between X and $S_{\frac{1}{\alpha}}(S_\alpha(X))$ is a measure of the loss in quality from down-sampling the video. Since this SSIM is independent of the choice of codec and compression parameters, this can be pre-computed.

The second source of quality degradation is compression. Let $C(X;q)$ be the decoded video obtained by encoding the source video X using a Quantization Parameter (QP) q. Then, the SSIM value between $S_\alpha(X)$ and $C(S_\alpha(X);q)$ measures the loss of quality resulting from compression of the video.

We use these two SSIM scores as features to predict the true SSIM and refer to these models as Two-Feature Models. In addition, we can also use the scaling factor α and the QP q as features. We call such models four-feature models.

In both cases, we train three regressors to predict the SSIM value at the rendering scale on each frame. The three regressors considered are

- Linear Support Vector Regressor (Linear SVR)
- Gaussian Radial Basis Function SVR (Gaussian SVR)
- Fully Connected Neural Network (NN)

The Neural Network is a small fully connected network having a single hidden layer with twice the number of neurons as input features. We compared these models to a simple learning-free model, which is used as a baseline. The output of the baseline model is the simple product of the two SSIM features. This is similar to the 2stepQA picture quality model proposed in [31] [30], for two-stage distorted pictures. We call this the Product model.

C. RESULTS

The correlations between predicted SSIM and true SSIM achieved by the various models on our in-house corpus is shown in Table 3, where "2" and "4" denote the number of features input to each learning-based model. Among the feature-based models, four-feature NN performed best. This is to be expected, given the great learning capacity of NNs.

TABLE 3: Correlation with True SSIM on corpus test data

Model	PCC	SROCC
NN 2	0.9461	0.9834
NN 4	**0.9845**	**0.9869**
Linear SVR 2	0.9529	0.9759
Linear SVR 4	0.9215	0.9201
Gaussian SVR 2	0.8571	0.9591
Gaussian SVR 4	0.9598	0.9628
Product (Baseline)	0.9662	0.9829
Histogram Matching	**0.9933**	**0.9956**

TABLE 4: Correlation with DMOS on Netflix Public Database

Model	PCC	SROCC
True SSIM	0.6962	0.6567
NN 2	**0.6759**	**0.6425**
Linear SVR 2	0.6746	0.6196
Gaussian SVR 2	0.6756	0.6373
Product (Baseline)	0.6715	0.6215
Histogram Matching	**0.6848**	**0.6616**

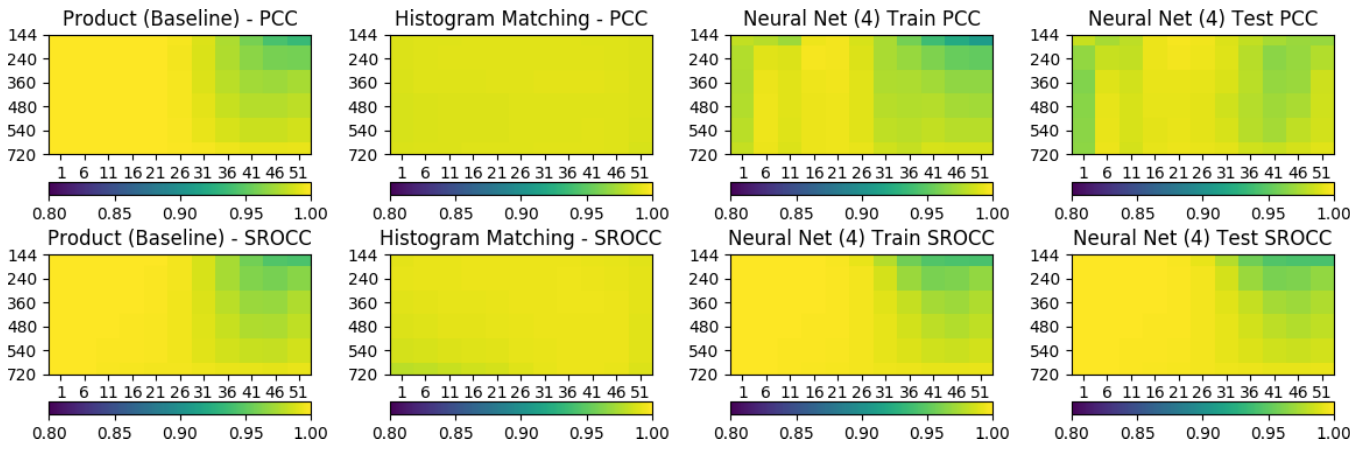


FIGURE 5: Variation in performance with choice of Encoding Scale and QP

It is interesting to note, however, that the learning-free product model yielded comparable or better performance at a negligible computational cost. Finally, the Histogram Matching model provided near-perfect predictions, outperforming all other models. The cost of this performance is the additional periodic calculation of reference quality maps/histograms.

Because our corpus contains videos generated at various compression scales and QPs, we were able to evaluate the sensitivity of our best models' performance to these choices of encoding parameters. We illustrate this in Fig. 5.

We observe that histogram matching performed consistently well across all encoding parameters with only a slight decrease in parameters at high-quality regions, i.e., high compression scale and low QP. We attribute this to the fact that most quality values at low QPs are very close to 1. As a result, the histogram of quality scores at the compression scale is concentrated close to 1, making histogram matching difficult.

On the other hand, the feature-based models performed poorly in low-quality regions, i.e., low compression scale and high QP. However, videos are seldom compressed at such low qualities, so this does not affect performance in most practical use cases.

Table 4 compares models based on the correlation they achieved against subjective opinion scores on the Netflix Public Database. Because our goal is to predict SSIM efficiently, we hold the performance of "true" SSIM as the gold standard against which we evaluated the performance of the Scaled SSIM models. Because videos in this database were generated by setting bitrates instead of QPs, we only tested our two-feature and histogram matching models. It is important to note that these models were **not** retrained on the Netflix database.

From the table, it may be observed that SSIM estimated by Histogram Matching matches the performance of true SSIM. We also observe that the feature-based models approach true SSIM's performance, with the Product Model offering an effective low complexity alternative to the learning-based models.

VI. DEVICE DEPENDENCE

The Quality of Experience of an individual user varies to a great extent, depending on not only the visual quality of the content, but also various other factors. These include, but may not be limited to [46]:

1. Context of media contents.
2. Users' viewing preferences.
3. Condition of terminal and application used for displaying contents.
4. Network bandwidth and latency.
5. The environment where users view content. (Background lighting conditions, viewing distances, audio devices, etc.)

From the perspective of compression and quality assessment algorithms, the factors to be considered are network fluctuations and terminal screens. The rest of this section restricts the scope to only the size of screens and pixel resolutions, excluding the influence of other variables.

A. IMPACT OF SCREEN SIZE

The issue of screen size on viewing quality originally arose in the context of television [47]. Large wall-sized, theatre(s) and IMAX displays provide better experiences, with subjects reporting increased feelings of 'reality,' i.e. the illusion that viewers are present at the scene. Studies have also shown a possible relation between screen size and viewer's intensity of response to contents.

B. DISPLAYED CONTENT AND PERCEIVED PROJECTION

The authors of [46] compared the experiences of users of devices of different screen sizes, both on web browsing and video viewing. Mean Opinion Scores were found to vary significantly, with an approximate difference of $\Delta = 1MOS$ between high-end devices and low-end devices of that time. It was also discovered that viewing of videos on tablets (iPad, etc.) benefited more from displaying contents of higher resolutions (more significant impact on MOS) as compared to mobile phones. User experiences are a combined function of screen size, content resolution, and viewing distance. This is quantified by the contrast sensitivity (CSF) of the HVS [48] which is broadly band pass so that increases in resolutions beyond a certain limit are not perceivable, hence pose little impact on user experience. The pass band of the human spatial CSF peaks between 1-10 cycles/degree (cpd) (depending on illumination and temporal factors), falling off rapidly beyond. Naturally, it is desirable that picture and video quality assessment algorithms be able to adapt to screen size against assumed viewing distance, i.e., by characterizing the projection of the screen on the retina [49]. Given a screen height H and a viewing distance D, full-screen viewing angle is

$$\alpha = 2\arctan\left(\frac{H}{2D}\right) \tag{30}$$

then, if the viewing screen contains L pixels on each (vertical or horizontal) line, the spatial frequency of the pixel spacing is defined as

$$f_{max} = \frac{L}{2\alpha} \tag{31}$$

in cpd.

C. TRANSFORMING ACROSS VARIOUS SCALES

While primitive specifications of viewing distances and pixel resolutions have been provided by the ITU [50], existing subjective picture and video quality databases are mainly defined by their content and their testing environments. Likewise, nearly all quality assessment models that operate over scales use the down-sampling transform

$$Z_\alpha = \max\left(1, \left\lceil \frac{H_I}{256} \right\rceil\right) \tag{32}$$

which creates discontinuities as the height is varied (e.g., $H_1 = 510$ and $H_2 = 520$) and does not account for viewing distance D. However, the Self-Adaptive Scale Transform (SAST) [51] seeks to remediate these weaknesses. SAST uses both the viewing distance and the user's angle of view (including the distance)

$$\begin{aligned} Z_s &= \sqrt{\frac{H_I \cdot W_I}{H \cdot I}} \\ &= \sqrt{\frac{1}{4tan\left(\frac{\theta_H}{2}\right) tan\left(\frac{\theta_W}{2}\right)} \cdot \frac{H_I}{D} \cdot \frac{W_I}{D}} \end{aligned}$$

where D denotes the viewing distance, H_I and W_I are the height and width of the display, and the corresponding projection on the retina is H by W. Commonly assumed, horizontal and vertical viewing angles are $\theta_H = 40°$ and $\theta_W = 50°$.

Because of the band pass nature of the visual system, a picture or video may be preprocessed using Adaptive High-frequency Clipping (AHC) [52]. In this method, instead of losing high-frequency information during scaling, high frequencies are selectively assigned smaller weights in a wavelet domain.

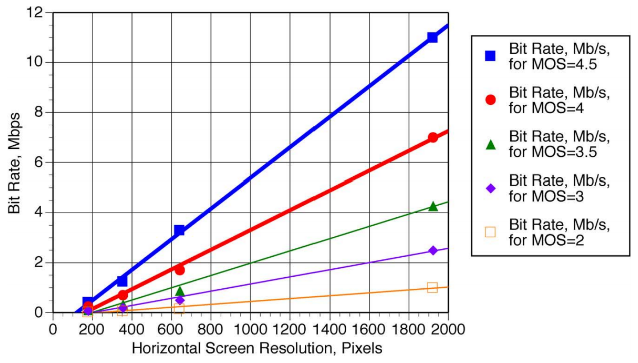


FIGURE 6: Required bit rates for different screen sizes

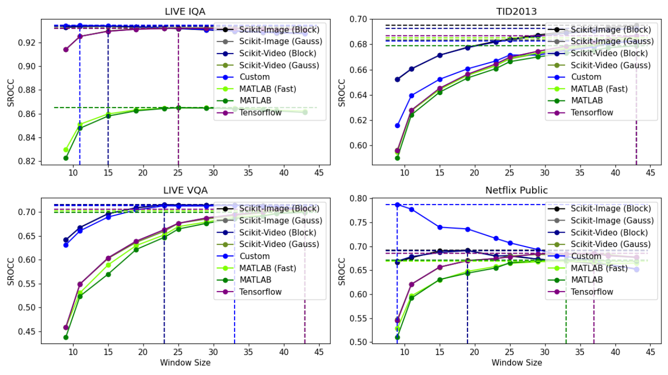
A dedicated database called VDID2014 [53] was created to record human visual responses under varying viewing distances and pixel resolutions. The authors derived an optimal scale selection (OSS) model based on the SAST and AHC model. Instead of directly comparing reference and distorted contents, all frames of the picture or video are first preprocessed according to an assumed or derived viewing distance and pixel resolution, before applying quality assessment algorithms such as SSIM. OSS model significantly boosted performance of both legacy and modern IQA/VQA models while not significantly increasing computational complexity.

D. TRADEOFFS

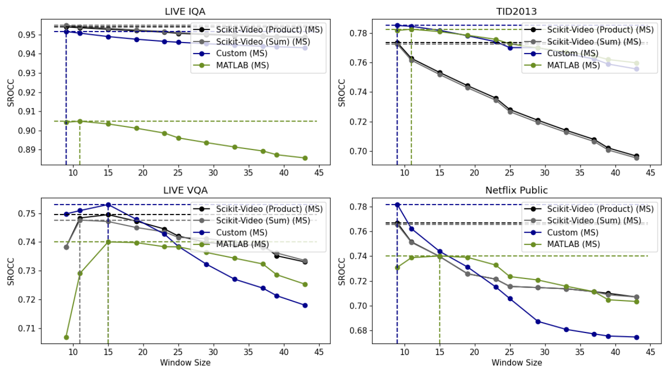
While modern viewers maintain high expectations of visual quality, regardless of the device and picture or video applications being used, efforts to achieve consistent quality have been inconsistent across screens of various sizes and resolutions. A study of H.264 streams without packet loss [54] demonstrated that the required bit rate grows linearly with the horizontal screen size, while the expected level of subjective quality was kept fixed, as illustrated by Fig. 6. However, the required bit rates grew much faster with increased expectations of perceived quality, with saturation of MOS at high bit rates, and little quality improvement with increased bit rate. Notwithstanding future improvements in picture/video representation and compression technologies, the curves in Fig. 6 supply approximate upper bounds of MOS against content resolution and bit rate.

E. MOBILE DEVICES

Mobile devices have advanced rapidly in recent years, featuring larger screens and higher resolutions. Most legacy databases that focused on explaining the impact of screen sizes and pixel resolutions were constructed using Personal Digital Assistants (PDA) or older cell phones with displays smaller than 4.5 inches. Popular resolutions of the time of these studies were 320p or 480p, which rarely appear on contemporary mobile devices. In an effort to investigate the



(a) Variation of SSIM performance with window size



(b) Variation of MS-SSIM performance with window size

FIGURE 7: Effect of size and choice of window function on performance

same issue on screens larger than 5.7 inches and resolutions of 1440p (2K) or 2160p (4K), a recent subjective study [55] focused on more recent mobile devices having larger resolutions and screen sizes. The contents viewed by the subjects were re-scaled to 4K.

The results from a one-way analysis of variance (ANOVA) suggested no significant relevance between screen size and perceived quality on screens ranging from 4 inches to 5.7 inches. However, a considerable MOS improvement of 0.15 was achieved by 1080p content over 720p content, but no further improvement was observed by increasing the content resolutions to 1440p, suggesting a saturation of perceived quality with resolution. Of course, MOS tends to remain constant across content resolutions higher than that of the display, suggesting that service providers restrict spatial resolutions whenever device specifications are available.

VII. EFFECT OF WINDOW FUNCTIONS ON SSIM

At the heart of SSIM lies the computation of the local statistics - means, variances, and covariances - comprising the luminance, contrast and structure terms. As described in Section II, the computational complexity of SSIM is $O(MNk^2)$.

A. EFFECT OF WINDOW SIZE

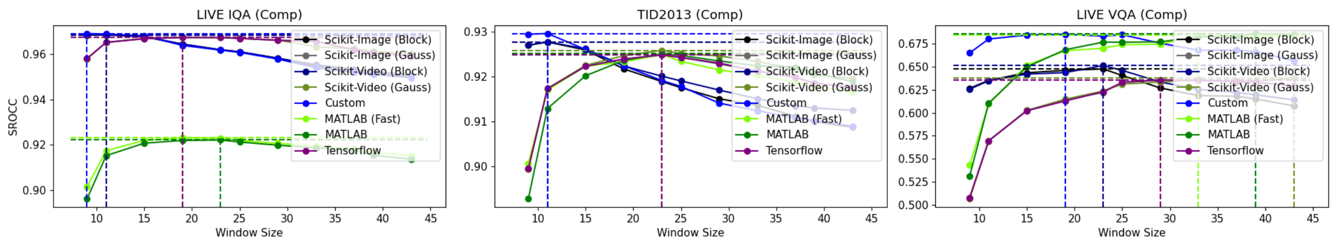
While computational complexity increases quadratically with window size, using larger windows does not guarantee better performance. Indeed, since picture and video frames are non-stationary, computing local statistics is highly advantageous for local quality prediction. While small values of k can lead to noisier estimates due to lack of samples, choosing large values of k risks losing the relevance of local luminance, contrast, structure, and distortion.

As mentioned earlier, the two most common choices of SSIM windows are Gaussian-shaped and rectangular-shaped. Traditionally, the use of rectangular windows is not recommended in image processing, due to frequency side-lobes and resulting "noise-leakage." Because the frequency response of a rectangular window is a sinc function, undesired high-frequencies can leak into the output. To mitigate this effect, Gaussian filtering is usually preferred, especially for denoising applications or if noise may be present. For these reasons, Gaussian-shaped windows were recommended by the authors of SSIM when calculating local statistics.

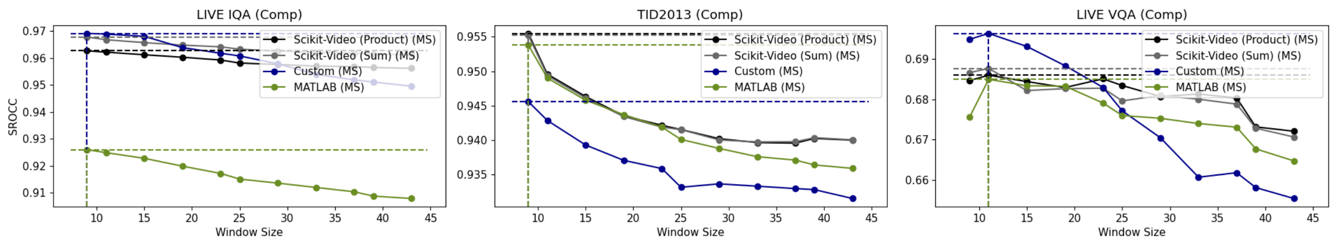
To investigate the effect of the choice of window and window size, we considered a set of scaling constant values $\sigma = 1.0, 1.5, \dots 6.0$ of the Gaussian window, while truncating them at a width of about 7σ and forcing the width to be odd. Since only Python implementations allowed setting σ, we restricted our experiments to these implementations. All these experiments were conducted using a stride of 1.

Given a Gaussian window of standard deviation σ, one can construct analogous rectangular windows in three ways - having the same physical size (i.e., width and height), having the same variance (considering the rectangular window as a sampled uniform distribution), or having the same (3dB) bandwidth. To specify a rectangular window of size $2K+1$, we only need to specify K. Equating the variance of the Gaussian to that of a uniform distribution yields $K = \lceil \sigma\sqrt{3} \rceil$, where $\lceil \cdot \rceil$ denotes the ceiling operation. Similarly, equating the 3dB bandwidths of the two filters requires $K = \lceil 1.602\sigma \rceil$.

The variation of SSIM performance against window choice is shown in Fig. 7. For simplicity, we only show the performance of rectangular windows having the same physical size. We observed similar results for rectangular windows having the same variance and 3dB bandwidth. Surprisingly, the figure indicates that rectangular windows are an objectively better design choice than equivalent Gaussian windows! In particular, rectangular windows outperform Gaussian windows for smaller window sizes, achieving slightly higher peak performance. Our experiments suggest that using windows of linear dimension in the range 15 to 20 offers a good tradeoff between performance and computation on both picture databases. Using rectangular windows also offers the possibility of a significant computational advantage, because they can be implemented efficiently using integral images, as discussed in Section II.



(a) Variation of SSIM performance with window size



(b) Variation of MS-SSIM performance with window size

FIGURE 8: Effect of size and choice of window function on performance on compression data

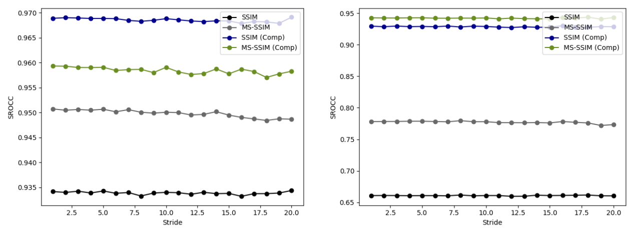
We report the variation of performance against window size on compression distorted data in Fig. 8. From these plots, it may be seen that when tested on compression distortions, SROCC is maximized for smaller window sizes, in the range 7 to 15.

B. EFFECT OF STRIDE

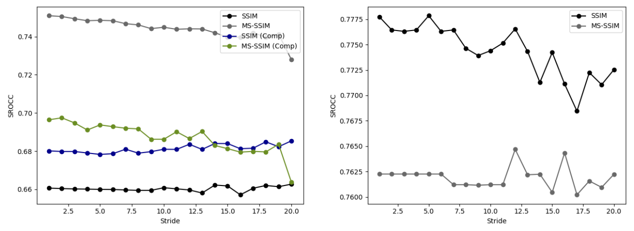
It is also possible to compute SSIM in a subsampled way, by including a stride s, where s is the distance between adjacent windows on which SSIM is computed. Then, the computational complexity of SSIM is $O(MNk^2/s^2)$.

We tested the effect of stride on performance using our custom python implementation, because none of the existing implementations allow varying the stride. In Fig. 9, we report the variation of SROCC with stride, where lines labelled "(Comp)" denote the performance on compression distorted data.

From the figure, it may be seen that the SROCC is largely unaffected by stride for $s \leq 5$. This means that by choosing a stride of $s = 5$, we can obtain a significant improvement in efficiency (25x speedup), with little change in prediction performance.



(a) LIVE IQA Database (b) TID 2013 Database



(c) LIVE VQA Database (d) Netflix Public Database

FIGURE 9: Variation of performance with stride

VIII. MAPPING SCORES TO SUBJECTIVE QUALITY

Because of the different parameter configuration and approximations they use, the many available implementations of SSIM tend to disagree with each other, producing (usually slightly) different scores on the same contents. Of course, any inconsistencies between deployed SSIM models is undesirable, since otherwise in a given application (such as controlling an encoder), changing the SSIM implementation may lead to unpredictable results.

One way to address this issue is by applying a predetermined function to map the obtained SSIM results to subjective scores. Among a collection of both nonlinear and piecewise linear mappings, the 5PL function in (17) is particularly useful.

A. IMPROVEMENT DUE TO MAPPING

Fig. 10 shows a typical example of fitting raw results (scatter plot of MOS vs. MS-SSIM) to a five-parameter logistic function. Both the MOS and objective scores cover fairly wide ranges while the mapping function lies approximately in the middle these.

It can be easily observed that utilizing the fitted curve yields a considerable improvement in PCC and RMSE, while the SROCC remains the same due to the monotonic nature

TABLE 5: Improvement in performance due to linearization

(a) LIVE IQA Database

Metric	PCC		RMSE	
	Raw	Fitted	Raw	Fitted
ClearView	0.5928	0.7929	0.4265	0.1161
HDRTools	0.7002	0.8445	0.3110	0.1021
MATLAB	0.7311	0.8610	0.3047	0.0969
ClearView (MS)	0.7677	0.8766	0.3720	0.0917
HDRTools (MS)	0.6673	0.9124	0.4576	0.0780

(b) TID 2013 Database

Metric	PCC		RMSE	
	Raw	Fitted	Raw	Fitted
ClearView	0.6612	0.7175	0.3492	0.1239
HDRTools	0.6205	0.6653	0.2704	0.1327
MATLAB	0.652	0.686	0.263	0.129
ClearView (MS)	0.7308	0.7508	0.3008	0.1174
HDRTools (MS)	0.7870	0.8384	0.3738	0.0969

(c) LIVE VQA Database

Metric	PCC		RMSE	
	Raw	Fitted	Raw	Fitted
ClearView	0.4277	0.4625	0.4385	0.1938
HDRTools	0.4469	0.4789	0.3780	0.1919
MATLAB	0.4767	0.5595	0.3798	0.1812
ClearView (MS)	0.5491	0.6569	0.4210	0.1648
HDRTools (MS)	0.6701	0.7394	0.4571	0.1472

(d) Netflix Public Database

Metric	PCC		RMSE	
	Raw	Fitted	Raw	Fitted
ClearView	0.5856	0.5896	0.4567	0.2283
HDRTools	0.5800	0.5800	0.3820	0.2302
MATLAB	0.6150	0.6150	0.3758	0.2228
ClearView (MS)	0.7862	0.8119	0.4359	0.1650
HDRTools (MS)	0.7171	0.7552	0.4632	0.1852

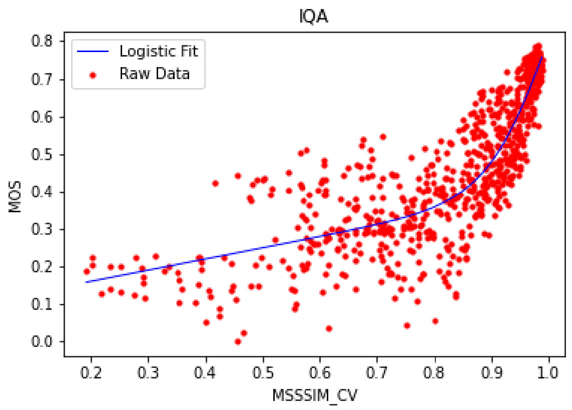


FIGURE 10: Example of fitting 5PL curve to a scatter plot of MOS vs. MS-SSIM

of the function. Table 5 shows the improvement obtained in PCC and RMSE after utilizing the fitted curve. For most of the evaluated models there is a considerable performance enhancement.

B. GENERALIZING TO OTHER DATABASES

While better performance against subjective scores is obtained after mapping the raw data using logistic functions, this only works on databases where subjective scores are available to help optimize the model parameters. In real life, however, social media and streaming service providers lack subjective opinions of their shared of streamed content. This means that it is uncertain whether a set of fitted parameters will apply well to unknown data. In order to study the generalizability provided by logistic mapping, we optimized the function parameters on each individual database, mapped the raw data in the other three databases using the obtained model, as a way of assessing performance on unknown content. If the correlation metrics were to remain similar, it would demonstrate that the logistic function can be used to provided steady performance on unseen data.

The results of the cross-database generalization experiments are shown in Table 6. The result in each cell of the table is the RMSE obtained by training a 5PL function on a "source" database (SD), then testing it on each "target" database (TD). From these experiments, we observed that when using MATLAB SSIM, the 5PL function generalized well between the TID 2013 and LIVE IQA Databases.

However, we observed poor generalization when fitting 5PL on the LIVE VQA database, then testing on the LIVE IQA database. While it may be too much to expect strong performance on video distortions after training on pictures and picture distortions, the lesson learned is still that a user or service provider either select the most relevant database to train on, or conduct a user study directed to their use case, on which a SSIM mapping may be optimized.

TABLE 6: Generalizability of 5PL SSIM mappings across databases

SD \ TD	Netflix Public	LIVE VQA	TID 2013	LIVE IQA
Netflix Public	0.228	0.197	0.307	0.718
LIVE VQA	0.232	0.194	0.243	0.464
TID 2013	0.250	0.209	0.124	0.147
LIVE IQA	0.245	0.208	0.152	0.116

IX. COLOR SSIM

Of course, the vast majority of shared and streamed pictures and videos are in color. Hence it is naturally of interest to understand whether SSIM can be optimized to also account for color distortions. however, most available SSIM implementations only operate on the luminance channel. Distortions of the color components may certainly exert considerable influence on subjective quality. The most common approach to incorporating color information into SSIM is to calculate it on each color channel, whether RGB, YUV, or other color space, then combine the channel results.

More sophisticated approaches have been taken to incorporate color channel information into quality models. For example, CMSSIM [56] utilized CIELAB color space [57] distances to better distinguish color distortions and noises. This approach evolved, based on a later subjective study, into CSSIM [58], which generalizes the calculations of SSIM.

Another approach, called SHSIM [59] defines hue similarity (HSIM) much like structural similarity, then combines uses SSIM scores together with the HSIM scores. The combination of two was found to better predict subjective quality than when only using luminance or color.

The method called Quaternion SSIM (QSSIM) [60] combines multi-valued RGB (or any other tristimulus) color space vectors from picture or video pixels into a quaternion representation, providing a formal way to assess luminance and chrominance signals and their degradations together.

Although different in their formulations, these algorithms express individual frames in a tristimulus color space, whether RGB, YUV, or CIELAB depending on the application. In the following, we will define and assess each of these approaches.

A. QUATERNION SSIM

The quaternion SSIM algorithm uses quaternions [60] to represent color pixels with a vector of complex-like components (quaternions are often described as extensions of complex or phasor representations):

$$q(m,n) = r(m,n)\cdot i + g(m,n)\cdot j + b(m,n)\cdot k. \tag{33}$$

The quaternion picture or video frame can then be decomposed into constituent "DC" and "AC" components via

$$dc \triangleq \mu_q = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N} q(m,n), \tag{34}$$

and

$$ac_q \triangleq q(m,n) - \mu_q. \tag{35}$$

The quaternion contrast is then defined as

$$\sigma_q = \sqrt{\frac{1}{(M-1)(N-1)}\sum_{m=1}^{M}\sum_{n=1}^{N} \|ac_q\|_2}, \tag{36}$$

which, when computed on both reference and test signals, is used to form a correlation factor

$$\sigma_{q_{ref,dis}} = \frac{1}{(M-1)(N-1)}\sum_{m=1}^{M}\sum_{n=1}^{N} ac_{q_{ref}}\cdot \overline{ac}_{q_{dis}}, \tag{37}$$

yielding a quaternion formulation similar to the legacy grayscale SSIM:

$$QSSIM = \left|\left(\frac{2\mu_{q_{ref}}\cdot\mu_{q_{dis}}}{\mu_{q_{ref}}{}^2 + \mu_{q_{dis}}{}^2}\right)\left(\frac{\sigma_{q_{ref,dis}}}{\mu_{q_{ref}}{}^2 + \mu_{q_{dis}}{}^2}\right)\right|. \tag{38}$$

B. CMSSIM

The CMSSIM algorithm first transforms the input picture or video signal into the CIE XYZ tristimulus color space. These XYZ pixels are then transformed into luminance, red-green, and blue-yellow planes [58] as

$$\begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \end{bmatrix} = \begin{bmatrix} 0.279 & 0.72 & -0.107 \\ -0.449 & 0.29 & -0.077 \\ 0.086 & -0.59 & 0.501 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{39}$$

The resulting chromatic channels are then smoothed using Gaussian kernels, then transformed back into XYZ tristimulus color space via

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.6204 & -1.8704 & -0.1553 \\ 1.3661 & 0.9316 & 0.4339 \\ 1.5013 & 1.4176 & 2.5331 \end{bmatrix}\begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \end{bmatrix} \tag{40}$$

and then into the CIELAB L*, a*, and b*.

The dissimilarities between the reference and test chromatic components is then found as

$$\Delta E(x,y) = ((L_1^*(x,y) - L_2^*(x,y))^2 \tag{41}$$
$$+ (a_1^*(x,y) - a_2^*(x,y))^2 \tag{42}$$
$$+ (b_1^*(x,y) - b_2^*(x,y))^2)^{\frac{1}{2}} \tag{43}$$

which are the used to weight the values of the final SSIM map:

$$CSSIM = l(x,y)\cdot c(x,y)\cdot s(x,y)\cdot\left(1 - \frac{\Delta E(x,y)}{45}\right). \tag{44}$$

C. HSSIM

The HSSIM index is calculated by first transforming pictures or frames into an HSV color space. The color quality is then predicted using a weighted average of SSIM and hue similarity:

$$HSSIM\,(x,y) = \frac{SSIM\,(x,y) + 0.2\times H\,(x,y)}{1.2}, \tag{45}$$

where $H\,(x,y)$ is of the same form as SSIM but operates on hue channel instead of grayscale.

Next we discuss straight-forward channel-wise SSIM as applied in YUV space. This model is also tested on the four databases.

D. CHANNEL-WISE SSIM

While image sensors normally capture RGB data in accordance with photopic (daylight) retinal sampling, most of the structural information is present in the luminance signal. This implies the existence of a lower information (reduced bandwidth) color representation. In fact, both the retinal representation and modern opponent (chromatic) color spaces exploit this property of visual signals. Modern social media and streaming platforms ordinarily process RGB into a chromatic

TABLE 7: Comparison of color SSIM models

(a) LIVE IQA Database

Method	PCC	SRCC	RMSE
Baseline SSIM	0.8594	0.8449	0.0975
Quaternion (RGB)	0.8845	0.8748	0.0889
Quaternion (YUV)	0.8766	0.8657	0.0917
Quaternion (LAB)	0.5983	0.5946	0.1527
CMSSIM	0.7873	0.7806	0.1175
HSSIM	0.7873	0.7809	0.1175
FFMPEG	0.8650	0.8507	0.0956
Daala (SSIM)	0.7547	0.7163	0.1250
Daala (MSSSIM)	0.8193	0.8113	0.1093
Daala (FASTSSIM)	0.7489	0.7500	0.1263
CSSIM	**0.8810**	**0.8790**	**0.0902**

(b) TID 2013 Database

Method	PCC	SRCC	RMSE
Baseline SSIM	0.6902	0.6337	0.1287
Quaternion (RGB)	0.7456	0.7155	0.1185
Quaternion (YUV)	**0.7735**	**0.7564**	**0.1127**
Quaternion (LAB)	0.3537	0.3015	0.1663
CMSSIM	0.6216	0.6124	0.1393
HSSIM	0.6217	0.6125	0.1393
FFMPEG	0.7168	0.6781	0.1240
Daala (SSIM)	0.4691	0.4497	0.1570
Daala (MSSSIM)	0.7414	0.7472	0.1193
Daala (FASTSSIM)	0.6348	0.6037	0.1374
CSSIM	0.6961	0.6411	0.1277

(c) LIVE VQA Database

Method	PCC	SRCC	RMSE
Baseline SSIM	0.5429	0.5089	0.1836
Quaternion (RGB)	**0.6709**	**0.6622**	**0.1621**
Quaternion (YUV)	0.6139	0.5985	0.1726
Quaternion (LAB)	0.2914	0.2651	0.2091
CMSSIM	0.4233	0.3826	0.1980
HSSIM	0.4194	0.3842	0.1984
FFMPEG	0.4651	0.4415	0.1935
Daala (SSIM)	0.4702	0.4443	0.1929
Daala (MS-SSIM)	0.6488	0.6351	0.1663
Daala (FastSSIM)	0.5526	0.5208	0.1822
CSSIM	0.6249	0.5742	0.1707

(d) Netflix Public Database

Method	PCC	SRCC	RMSE
Baseline SSIM	0.6335	0.5904	0.2187
Quaternion (RGB)	0.7621	0.7557	0.1830
Quaternion (YUV)	**0.7816**	**0.7763**	**0.1763**
Quaternion (LAB)	0.4508	0.3690	0.2523
CMSSIM	0.5417	0.4379	0.2375
HSSIM	0.5501	0.4444	0.2360
FFMPEG	0.6695	0.6267	0.2099
Daala (SSIM)	0.6766	0.6475	0.2081
Daala (MS-SSIM)	0.7585	0.7398	0.1842
Daala (FastSSIM)	0.7695	0.7385	0.1805
CSSIM	0.6643	0.6056	0.2112

space such as YUV or YCrCb prior to compression. Likewise, IQA/VQA may be defined on color frames represented by luminance and chrominance.

Since chromatic representations reduce the correlation between color planes, the chromatic components with reduced entropies may be down-sampled prior to compression. YCbCr values can be obtained directly from RGB via a linear transformation, typically

$$Y = 0.213 \times R + 0.715 \times G + 0.072 \times B$$
$$Cb = 0.539 \times (B - Y)$$
$$Cr = 0.635 \times (R - Y)$$

assuming ITU-R BT.709 conversion. However the YCbCr components are defined, a chromatic SSIM model may be defined based on a weighted average of the objective qualities of the individual YCbCr channels

$$F(ref, dis) = (f(Y_{ref}, Y_{dis}) \times 1.0 + f(Cb_{ref}, Cb_{dis}) \times \alpha + f(Cr_{ref}, Cr_{dis}) \times \beta)/(1.0 + \alpha + \beta)$$

where $f(.,.)$ denotes similarity between reference and distorted frames.

In our case, the baseline SSIM is used as the base QA measure, i.e. $f(.,.) = SSIM(\text{ref}, \text{dis})$. This method is used in popular image and video processing tools like FFMPEG and Daala, where a Color SSIM is calculated as

$$SSIM_{ij} = 0.8 \cdot SSIM_{ij}^{Y} + 0.1 \cdot SSIM_{ij}^{Cb} + 0.1 \cdot SSIM_{ij}^{Cr}$$

Instead of testing all possible combinations of the hyperparameters during our experiments, we fixed $\alpha = \beta$. We found $\alpha = \beta = -0.3$ to yield optimal results in our experiments, with the obtained performances of this optimized Color SSIM (CSSIM) on the four databases included in Table 7.

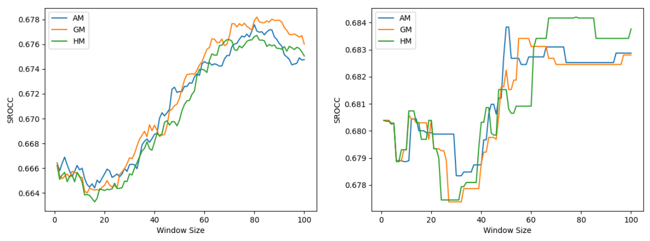
E. RESULTS

We compared performances of the above four Color SSIM models on the same selected databases, with the results tabulated in Table 7. As may be observed, using color information can noticeably boost SSIM's quality prediction power, with QSSIM RGB and YUV yielding the largest gains.

X. SPATIO-TEMPORAL AGGREGATION OF QUALITY SCORES

In its native form, SSIM is defined on a pair of image regions and returns a local quality score. When applied to a pair of images, a quality map is obtained of (approximately) the same size as the image. The most common method of aggregating these local quality values is to calculate Mean SSIM (MSSIM) to obtain a SSIM score on the entire image. This method of aggregating quality scores is also usually applied when applying SSIM to videos. Frame-wise MSSIM scores are calculated between pairs of corresponding frames, and the average value of MSSIM (over time) is reported as the single SSIM score of the entire video.

In the context of HTTP streaming, the authors of [61] evaluated various ways of temporal pooling SSIM scores, and found that over longer durations, the simple temporal mean performed about as well as other more sophisticated



(a) LIVE VQA Database (b) Netflix Public Database

FIGURE 11: Windowed-Moment-Pooling SROCC vs Window Size

TABLE 8: Performance of Windowed-Moment-Pooling

Database	Method	PCC	SROCC	RMSE
LIVE VQA	Baseline SSIM	0.6645	0.6664	0.1633
	Windowed-AM	0.6778	0.6776	0.1607
	Windowed-GM	**0.6788**	**0.6782**	**0.1605**
	Windowed-HM	**0.6788**	0.6767	**0.1605**
	Windowed-CoV	0.4151	0.6044	0.1988
Netflix Public	Baseline SSIM	0.7034	0.6804	0.2009
	Windowed-AM	0.7222	0.6838	0.1955
	Windowed-GM	**0.7233**	**0.6834**	**0.1951**
	Windowed-HM	0.7166	**0.6842**	0.1971
	Windowed-CoV	0.5825	0.5997	0.2560

pooling strategies. Here, we summarize and expand this work by simultaneously testing various spatial and temporal aggregation methods on the two video databases.

We begin by discussing various spatial and temporal pooling strategies that can be used to pool SSIM. Some of these methods require the tuning of hyperparameters. To optimize these hyperparameters, we use the baseline sample mean as the other pooling method, by comparing the SROCC achieved by each choice of hyperparameters. That is, when optimizing a spatial pooling method, we use temporal mean pooling, and vice versa.

As we will discuss below, many methods have been proposed which leverage either side information, such as visual attention maps, or computationally intensive procedures like optical flow estimation. While these methods offer principled ways to improve the SSIM model, we omit them from our comparisons because they have a high cost, which is often unsuitable for practical deployments of SSIM at large scales.

In subsequent sections, all of the SSIM quality maps were generated using the Scikit-Image implementation of SSIM, with rectangular windows and the default parameters. While the exact results of the experiments may vary slightly with the choice of "base implementation," we expect these trends to hold across implementations.

A. MOMENT-BASED POOLING

A straightforward extension of the averaging operation used in SSIM is to replace it by one of the other two Pythagorean means - the Geometric Mean (GM) and the Harmonic Mean (HM). Since local SSIM scores can be negative, we can only use GM and HM pooling on the structural dissimilarity (DSSIM) scores, i.e. $1 - SSIM$. However, this means that if the SSIM at any location is close to 1, the pooled score is dramatically decreased. So, we do not recommend using GM or HM for spatial pooling. However, framewise SSIM scores are nearly always positive, so we investigate the use of the Pythagorean means for temporal pooling. We also consider the sample median, since it is a robust measure of central tendency (MCT), unlike the mean.

Another method of pooling quality scores can be found in the MOVIE index [18]. It was found that the coefficient of variation (CoV) of quality values correlated well with subjective scores. Let $\mathbf{x} = (i, j)$ denote spatial indices. Given a quality map $Q(\mathbf{x}, t)$ having mean value $\mu_Q(t)$ and standard deviation $\sigma_Q(t)$, the CoV-pooled score is defined as

$$S_{CoV}(t) = \sigma_Q(t)/\mu_Q(t). \tag{46}$$

The same method can be used to pool frame-wise quality scores. One can also adapt the CoV method of temporal pooling using windowing, where the CoV is computed over short temporal windows before being averaged. That is, given a sequence of "local" temporal CoV values $\rho_Q(t)$, define the windowed-CoV-pooled scores

$$S_{W-CoV} = \frac{1}{T}\sum_t \rho_Q(t). \tag{47}$$

In the same vein, windowed versions of the three Pythagorean means can be used for temporal pooling. Using framewise SSIM scores obtained using the Scikit-Image implementation with a rectangular window of size 11, we tested the performance of the three windowed means (W-AM, W-GM, W-HM) and windowed-CoV (W-CoV) pooling. We analyzed the variation of performance of the windowed means against window size, for window sizes $w = 1, 2, \dots 100$. The results of these experiments are shown in Fig. 11. We omitted the W-CoV method from this plot because it gave significantly inferior performance, as shown in Table 8, which lists the best performance of each windowed method.

These plots reveal similar trends. There is an initial decrease in performance with increased window size, but for large enough windows, there is improvement in performance over the baseline. While windowed-CoV performed very poorly, the difference between the three Pythagorean means is small, with windowed-GM being a good choice. However, to observe a reliable improvement in performance, a large window size is needed, of $k \approx 80$ on the LIVE VQA database and $k \approx 50$ on the Netflix Public database. However, such large values of k could lead to significant delays in real-time applications, which may not be a reasonable cost considering the small increase in performance.

B. FIVE-NUMBER SUMMARY POOLING

The five-number summary (FNS) [62] method was proposed as a better way of summarizing the histogram of a spatial quality map, as compared to the simple mean. Given a spatial quality map $Q(\mathbf{x}, t)$ at time t, let $Q_{min}(t)$ denote the minimum value, $Q_1(t)$ denote the 25th percentile, (lower quartile), $Q_{med}(t)$ denote the median value, $Q_3(t)$ denote the 75th percentile (upper quartile) and $Q_{max}(t)$ denote the maximum value. The five number summary is then defined as

$$S_{FNS}(t) = \frac{Q_{min}(t) + Q_1(t) + Q_{med}(t) + Q_3(t) + Q_{max}(t)}{5} \tag{48}$$

Of course, FNS may likewise be applied to the framewise quality scores as a way of temporal pooling.

C. MEAN-DEVIATION POOLING

In [63], the authors proposed a SSIM-like quality index, which is then pooled using a "mean-deviation" operation. The deviation is defined as the power o of the Minkowski distance of order p between the quality values and its mean. More concretely, given a spatial quality map $Q(\mathbf{x}, t)$ at time t having mean value $\mu_Q(t)$, the pooled mean-deviation quality score is given by

$$S_{MD}^{(p,o)}(t) = \left(\left(\frac{1}{MN}\sum_{\mathbf{x}}(Q(\mathbf{x},t) - \mu_Q(t))^p\right)^{1/p}\right)^o. \tag{49}$$

In our experiments, the most common optimal choice was $p = 2$, corresponding to the standard deviation. When applying MD pooling to temporal scores, the final exponent o does not affect the SROCC, since exponentiation is a monotonic function. So, while we select p using the SROCC as discussed above, we choose o for temporal pooling by comparing PCC values.

D. LUMINANCE-WEIGHTED POOLING

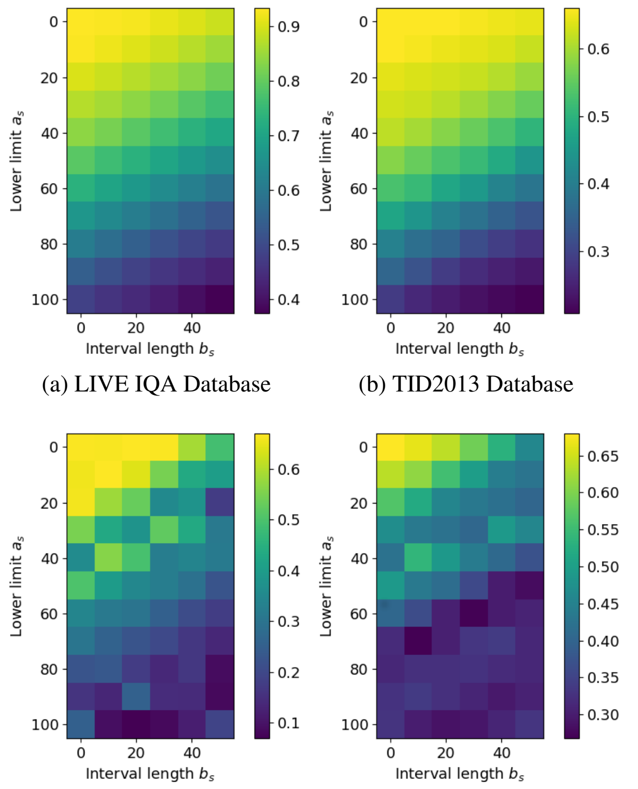
In [16], the authors proposed a method of spatial pooling which assigns weights to regions of an image based on the local luminance (brightness), which we call Luminance-Weighted (LW) pooling. These weights are used to account for the fact that the HVS is less sensitive to distortions in dark regions. Following our earlier convention, the local mean $\mu_1(\mathbf{x})$ is a measure of the local luminance in the reference image. Given a lower limit a_s and an interval length b_s, the weighting function is defined as

$$w_{LW}(\mathbf{x}) = \begin{cases} 0 & \mu_1(\mathbf{x}) < a_s \\ (\mu_1(\mathbf{x}) - a_s)/b_s & a_s \le \mu_1(\mathbf{x}) < a_s + b_s \\ 1 & \mu_1(\mathbf{x}) \ge a_s + b_s \end{cases} \tag{50}$$

Then, the spatially-weighted SSIM score is given by

$$S_{LW}(t) = \frac{1}{MN}\sum_{\mathbf{x}} w_{LW}(\mathbf{x})Q(\mathbf{x},t) \tag{51}$$

We tested the performance of LW-pooling on all four databases for values of $a_s = 0, 10, \ldots, 100$ and $b_s = 0, 10, \ldots 50$. Note that choosing $a_s = b_s = 0$ corresponds to the standard baseline SSIM. The experimental variation of performance (SROCC) against choices of a_s and b_s is shown in Fig. 12.



(a) LIVE IQA Database (b) TID2013 Database

(c) LIVE VQA Database (d) Netflix Public Database

FIGURE 12: LW-Pooling SROCC vs parameters a_s and b_s

On the LIVE IQA and Netflix Public databases, we observed that the best performance was achieved by the baseline $a_s = b_s = 0$. On the other two databases, the improvement in performance was insignificant, with an elevation of SROCC of less than 0.002. So, we do not recommend using LW-pooling.

E. DISTORTION-WEIGHTED POOLING

Distortion-Weighted (DW) pooling is a method that assigns different weights to low and high-quality regions. We consider the common method of distortion weighting, where the weight assigned to a quality score is proportional to a power of the quality score. Concretely, given an exponent p, the spatial DW-pooled score of a spatial quality map $Q(\mathbf{x}, t)$ as time t is given by

$$S_{DW}^{(p)}(t) = \frac{\sum_{\mathbf{x}}(1 - Q(\mathbf{x},t))^p\, Q(\mathbf{x},t)}{\sum_{\mathbf{x}}(1 - Q(\mathbf{x},t))^p}. \tag{52}$$

Likewise, DW-pooling may be applied to the times series of framewise quality scores to perform temporal DW-pooling. We tested DW-pooling using values of $p = 1/8, 1/4, \ldots 8$ on all four databases, for both spatial and temporal pooling. While DW-pooling can lead to a considerable increase in performance, we also found that the optimal value of p varied significantly between databases. So, in the absence of a dataset that the user can use to select p reliably, we do not recommend using DW-pooling off the shelf. If a user does have a relevant dataset, then DW-pooling may be profitably applied. We refer the reader to Table 13 for detailed results.

F. MINKOWSKI POOLING

The Minkowski Pooling (Mink) method is a generalization of the arithmetic mean, which provides another way to provide additional weight to low quality scores. Because local quality scores can be negative, we pool the DSSIM scores. Given an exponent p, define the spatial Minkowski-pooled score as

$$S^p_{Mink}(t) = \frac{1}{MN}\sum_{\mathbf{x}} (1 - Q(\mathbf{x}, t))^p\,. \tag{53}$$

Once again, we tested values of $p = 1/8, 1/4, \ldots, 8$. Note that we omitted $p = 1$ since it is identical to the baseline mean pooling. Spatial Minkowski pooling provided improvement in performance on the video databases, with $p = 4$ being a good choice of p across databases. However, as with DW-pooling, the optimal choice of p for temporal pooling varied significantly between databases and any improvement in performance was modest. So, we do not recommend using temporal Minkowski pooling, unless a specific application-relevant dataset is available.

G. PERCENTILE POOLING

More sophisticated techniques have been proposed to spatially pool of SSIM scores. In [64], the authors propose pooling SSIM scores by visual importance. The visual importance of distortions was measured in two ways: visual attention maps using the Gaze-Attentive Fixation Finding Engine (GAFFE) [65], and percentile pooling (PP) of quality scores. Because this guide is tailored towards practical application of SSIM, we omit the additional computation of running GAFFE and focus only on PP.

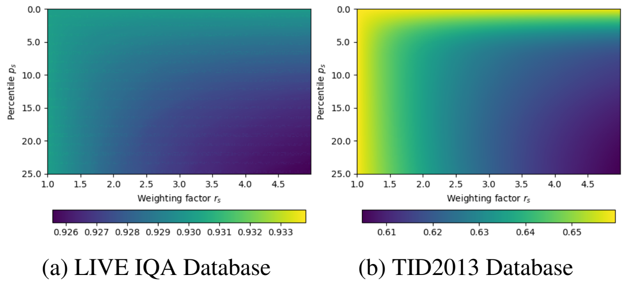
The spatial PP method is specified by two parameters: p_s, the percentile of lower values to be modified, and r_s, the factor by which the lower p_s percentile is weighted. The idea of PP is to heavily weight the worst quality regions, which are likely to heavily bias the perception of quality. Define the lowest p_s percentile of values of the quality map $Q(x, t)$ by $perc(Q, p_s)$. The quality values are then re-weighted as

$$\tilde{Q}^{(r_s,p_s)}(\mathbf{x}, t) = \begin{cases} Q(\mathbf{x}, t)/r_s, & Q(\mathbf{x}, t) \in perc(Q, p_s) \\ Q(\mathbf{x}, t), & otherwise \end{cases}. \tag{54}$$

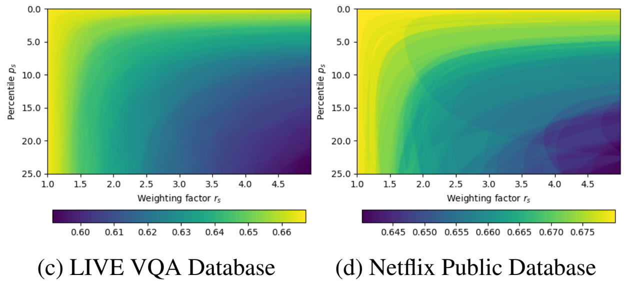
The PP quality score is then defined as the average of the re-weighted quality values:

$$S^{(r_s,p_s)}_{PP}(t) = \frac{1}{MN}\sum_{\mathbf{x}} \tilde{Q}^{(r_s,p_s)}(\mathbf{x}, t). \tag{55}$$

Larger values of p_s penalize more low-quality values, which may dilute the distortion severity, while larger values of r_s weight the low quality regions more heavily. While the authors of [64] were circumspect regarding the value of percentile pooling, they recommended choosing $p_s = 6$ and

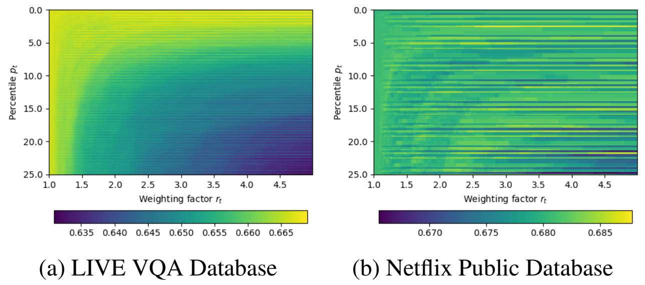


(a) LIVE IQA Database (b) TID2013 Database



(c) LIVE VQA Database (d) Netflix Public Database

FIGURE 13: Spatial PP SROCC vs parameters p_t and r_t



(a) LIVE VQA Database (b) Netflix Public Database

FIGURE 14: Temporal PP SROCC vs parameters p_t and r_t

$r_s = 4000$. However, on all four databases, we found that all choices of parameters p_s and r_s performed worse than the baseline. This behaviour is illustrated in Fig. 13 for values of p_s in the range $[0, 25]$ (%) and r_s in the range $[1, 5]$. While the discrepancy between our results and those of [64] can be attributed to variations in the implementations, we found that the performance of percentile pooling was inferior across all the tested databases. So, we do not recommend spatial percentile pooling.

We also studied temporal PP of aggregated framewise quality scores by penalizing low-quality frames. Once again, to find the optimal choice of the temporal PP parameters p_t and r_t, we computed framewise SSIM scores, one which we tested values of p_t in the range $[0, 25]$ (%) and r_t in the range $[1, 5]$. Figures 14a and 14b plot the variation of performance (SROCC) with choices of p_t and r_t on the LIVE VQA and Netflix Public databases, respectively. The performances of the optimal PP algorithm and baseline SSIM are compared in Table 9.

From Table 9, it may be observed that temporal PP gave only a minor improvement in performance over baseline SSIM. From the accompanying plots, while there were gen-

TABLE 9: Performance of Temporal PP SSIM

Database	Method	PCC	SROCC	RMSE
LIVE VQA	Baseline SSIM	0.5999	0.5971	0.1749
	Temporal PP	**0.6163**	**0.5986**	**0.1721**
Netflix Public	Baseline SSIM	0.6815	0.6574	0.2068
	Temporal PP	**0.6868**	**0.6601**	**0.2054**

TABLE 10: Performance of 3D SSIM/MS-SSIM

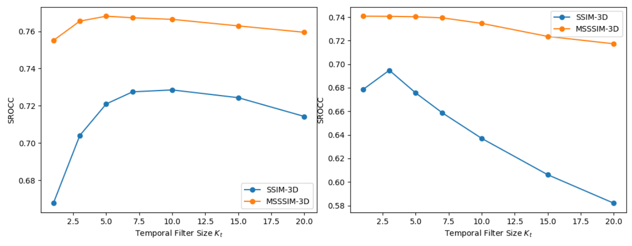
Database	Method	PCC	SROCC	RMSE
LIVE VQA	Framewise SSIM	0.6650	0.6677	0.1632
	SSIM 3D	**0.7300**	**0.7285**	**0.1494**
	Framewise MS-SSIM	0.7631	0.7551	0.1412
	MS-SSIM 3D	**0.7779**	**0.7681**	**0.1374**
Netflix Public	Framewise SSIM	0.7022	0.6784	0.2012
	SSIM 3D	**0.7086**	**0.6948**	**0.1994**
	Framewise MS-SSIM	0.7454	0.7408	0.1884
	MS-SSIM 3D	**0.7512**	**0.7408**	**0.1865**

eral trends in performance with variations of each parameter, the prediction performance of the pooled models was sensitive to small perturbations of the parameters. Coupled with the fact that the observed increases in performance were small, it is difficult to reliably identify good choices of the parameters p_t and r_t. So, we do not recommend using percentile pooling for temporal aggregation either.

H. SPATIO-TEMPORAL SSIM

Efforts have also been made to create spatio-temporal versions of SSIM. In [66], the authors proposed a 3D spatio-temporal SSIM, and its motion-tuned extension. To avoid additional computation related to motion estimation, we consider only the SSIM-3D model and replace the motion-tuned weighting function by a rectangular window. Thus, SSIM-3D was defined as identical to SSIM, other than that local statistics - mean, standard deviation and correlation - were computed on 3D spatio-temporal neighborhoods instead of 2D spatial neighborhoods. Similarly, define MS-SSIM-3D as 3D SSIM computed over multiple spatial scales. Because we choose rectangular windows, we used integral images to efficiently compute the local statistics.

We tested these 3-D variants of SSIM and MS-SSIM on the LIVE VQA and Netflix Public video databases. We used block rectangular filters of size $11 \times 11 \times K_t$, and investigated the variation of the algorithm's performance with K_t. The



(a) LIVE VQA Database (b) Netflix Public Database

FIGURE 15: Variation of SSIM and MS-SSIM 3D performance with Temporal Window Size K_t

performances of the baseline frame-wise SSIM/MS-SSIM ($K_t = 1$) models and the best SSIM/MS-SSIM 3D are listed in Table 10. The variation in performance of SSIM-3D and MS-SSIM-3D with K_t is plotted in Fig. 15.

From these figures, performance increases by both SSIM-3D and MS-SSIM-3D relative to the 2D frame-based versions may be observed on the LIVE VQA database, with the improvement being much more pronounced in the case of single scale SSIM. When tested on the Netflix-Public database, the improvement was much lower. Again, the improvement was lower for MS-SSIM-3D than SSIM-3D. From the plots, choosing K_t from the interval $[3, 10]$ offers solid improvement in performance. Another advantage of this approach is that using small rectangular temporal windows, performance increases can be obtained without any increase in computational complexity, by maintaining rolling sums of the last K_t frames. This can be achieved as below, using a buffer of K_t frames, leading to an $O(MNK_t)$ memory complexity.

As in equations (24) - (28), we can calculate the local statistics from the analogously defined sums over 3D neighborhoods $S_1^{(1)}(i,j,k)$, $S_1^{(2)}(i,j,k)$, $S_2^{(1)}(i,j,k)$, $S_2^{(2)}(i,j,k)$, and $S_{12}(i,j,k)$. As an illustrative example, consider

$$S_1^{(1)}(i,j,k) = \sum_{m=i-l+1}^{i} \sum_{n=j-l+1}^{j} \left(\sum_{o=k-K_t+1}^{k} I_1(m,n,o) \right). \tag{56}$$

Defining the temporal sum

$$T_1^{(1)}(i,j,k) = \left(\sum_{o=k-K_t+1}^{k} I_1(i,j,o) \right), \tag{57}$$

we can rewrite (56) as

$$S_1^{(1)}(i,j,k) = \sum_{m=i-l+1}^{i} \sum_{n=j-l+1}^{j} T_1^{(1)}(m,n,k). \tag{58}$$

Knowing $T_1^{(1)}(m,n,k)$, this sum can be computed efficiently using integral images, using the equations (18) - (23). The temporal sum $T_1^{(1)}(m,n,k)$ itself can be updated efficiently with each new frame, by observing that

$$T_1^{(1)}(i,j,k) = T_1^{(1)}(i,j,k-1) - I_1(i,j,k-Kt) + I_1(i,j,k). \tag{59}$$

In the same manner, we can also compute $S_1^{(2)}(i,j,k)$, $S_2^{(1)}(i,j,k)$, $S_2^{(2)}(i,j,k)$, and $S_{12}(i,j,k)$ efficiently. Combining these two methods, we can compute SSIM-3D in $O(MN)$ time at each frame, irrespective of the temporal size of the window K_t.

Motion vectors were also used to incorporate temporal information in [67], which proposed a Motion Compensated SSIM (MC-SSIM) algorithm. Motion vectors were used to find matching blocks in the reference and test video sequences at each temporal index. The SSIM scores between these matched blocks were then used to calculate a temporal quality score. The clear drawback of this method is the

TABLE 11: Comparing the Performance of Spatial Pooling Methods on Image Databases

(a) LIVE IQA Database

Method	PCC	SROCC	RMSE
AM	0.944	0.934	0.091
CoV	0.940	0.931	0.093
MD$^{(2,1)}$	0.906	0.887	0.116
FNS	0.939	0.929	0.0941
DW$^{(1/4)}$	0.944	0.934	0.090
Mink$^{(2)}$	**0.944**	**0.934**	**0.090**

(b) TID 2013 Database

Method	PCC	SROCC	RMSE
AM	0.711	0.659	0.125
CoV	0.741	0.718	0.119
MD$^{(2,1)}$	0.687	0.705	0.129
FNS	0.692	0.583	0.128
DW$^{(1/8)}$	0.732	0.715	0.121
Mink$^{(4)}$	**0.755**	**0.747**	**0.117**

TABLE 13: Comparing the Performance of Pairs of Spatial and Temporal Pooling Methods on Video Databases

(a) LIVE VQA Database - PCC

SP TP	AM	CoV	MD$^{(2,3)}$	FNS	DW$^{(1)}$	Mink$^{(4)}$
AM	0.664	0.766	0.708	0.580	0.770	0.781
GM	0.665	0.738	0.650	0.522	0.777	0.757
HM	0.661	0.709	0.435	0.518	0.781	0.736
CoV	0.521	0.221	0.153	0.341	0.578	0.161
MD$^{(2,3)}$	0.587	0.668	0.695	0.530	0.622	0.580
FNS	0.647	0.726	0.641	0.510	0.775	**0.784**
W-AM$^{(80)}$	0.677	0.770	0.699	0.540	0.773	**0.782**
W-GM$^{(81)}$	0.678	0.746	0.651	0.541	0.780	0.761
W-HM$^{(81)}$	0.643	0.725	0.615	0.541	**0.784**	0.743
DW$^{(2)}$	0.694	0.762	0.709	0.611	0.767	0.753
Mink$^{(2)}$	0.631	0.770	**0.788**	0.523	**0.783**	0.780

(b) Netflix Public Database - PCC

SP TP	AM	CoV	MD$^{(4,1)}$	FNS	DW$^{(8)}$	Mink$^{(8)}$
AM	0.703	0.802	0.887	0.677	**0.924**	0.885
GM	0.699	0.814	0.892	0.679	0.816	0.890
HM	0.702	0.820	0.897	0.679	0.810	0.894
CoV	0.526	0.111	0.486	0.529	0.532	0.457
MD$^{(2,1)}$	0.570	0.495	0.277	0.511	0.292	0.323
FNS	0.689	0.786	0.885	0.671	**0.917**	0.881
W-AM$^{(50)}$	0.718	0.799	0.880	0.682	**0.921**	0.882
W-GM$^{(55)}$	0.719	0.804	0.882	0.684	0.866	0.884
W-HM$^{(78)}$	0.704	0.809	0.884	0.684	0.842	0.889
DW$^{(1/4)}$	0.698	0.804	0.888	0.680	**0.923**	0.887
Mink$^{(1/4)}$	0.710	0.801	0.886	0.668	**0.901**	0.875

(c) LIVE VQA Database - SROCC

SP TP	AM	CoV	MD$^{(2,3)}$	FNS	DW$^{(1)}$	Mink$^{(4)}$
AM	0.667	0.762	**0.788**	0.584	0.749	0.762
GM	0.666	0.727	0.726	0.579	0.758	0.730
HM	0.660	0.695	0.552	0.574	0.764	0.706
CoV	0.599	0.039	0.119	0.532	0.656	0.153
MD$^{(2,3)}$	0.572	0.669	0.728	0.540	0.610	0.557
FNS	0.640	0.752	0.741	0.556	0.763	0.778
W-AM$^{(80)}$	0.678	0.764	**0.791**	0.592	0.754	0.764
W-GM$^{(81)}$	0.678	0.731	0.731	0.594	0.763	0.736
W-HM$^{(81)}$	0.677	0.699	0.672	0.593	0.766	0.708
DW$^{(2)}$	0.693	0.743	**0.790**	0.564	0.766	0.726
Mink$^{(2)}$	0.663	0.754	**0.789**	0.581	0.775	0.754

(d) Netflix Public Database - SROCC

SP TP	AM	CoV	MD$^{(4,1)}$	FNS	DW$^{(8)}$	Mink$^{(8)}$
AM	0.680	0.768	0.871	0.633	**0.911**	0.872
GM	0.682	0.770	0.872	0.634	0.837	0.874
HM	0.681	0.773	0.875	0.635	0.832	0.877
CoV	0.479	0.143	0.437	0.497	0.610	0.409
MD$^{(2,1)}$	0.445	0.437	0.104	0.463	0.246	0.228
FNS	0.667	0.760	0.868	0.629	**0.886**	0.866
W-AM$^{(50)}$	0.684	0.765	0.863	0.632	**0.896**	0.868
W-GM$^{(55)}$	0.683	0.764	0.865	0.633	0.863	0.869
W-HM$^{(78)}$	0.684	0.765	0.868	0.633	0.839	0.872
DW$^{(8)}$	0.683	0.768	0.870	0.634	**0.910**	0.873
Mink$^{(8)}$	0.682	0.769	0.871	0.633	**0.913**	0.869

(e) LIVE VQA Database - RMSE

SP TP	AM	CoV	MD$^{(2,3)}$	FNS	DW$^{(1)}$	Mink$^{(4)}$
AM	0.163	0.140	0.154	0.178	0.140	0.137
GM	0.163	0.148	0.166	0.186	0.138	0.143
HM	0.164	0.154	0.197	0.187	0.137	0.148
CoV	0.187	0.213	0.216	0.205	0.178	0.216
MD$^{(2,3)}$	0.177	0.163	0.157	0.185	0.171	0.178
FNS	0.167	0.150	0.168	0.188	0.138	**0.136**
W-AM$^{(80)}$	0.161	0.139	0.156	0.184	0.139	**0.136**
W-GM$^{(81)}$	0.161	0.146	0.166	0.184	0.137	0.142
W-HM$^{(81)}$	0.167	0.151	0.172	0.184	**0.136**	0.146
DW$^{(2)}$	0.157	0.142	0.154	0.173	0.140	0.144
Mink$^{(2)}$	0.170	0.139	**0.135**	0.186	**0.136**	0.137

(f) Netflix Public Database - RMSEs

SP TP	AM	CoV	MD$^{(4,1)}$	FNS	DW$^{(8)}$	Mink$^{(8)}$
AM	0.201	0.169	0.130	0.208	**0.108**	0.132
GM	0.202	0.164	0.128	0.208	0.163	0.129
HM	0.201	0.162	0.125	0.208	0.166	0.127
CoV	0.240	0.281	0.247	0.240	0.239	0.251
MD$^{(2,1)}$	0.232	0.246	0.272	0.243	0.270	0.268
FNS	0.205	0.175	0.131	0.210	**0.113**	0.134
W-AM$^{(50)}$	0.197	0.170	0.134	0.206	**0.110**	0.133
W-GM$^{(55)}$	0.196	0.168	0.133	0.206	0.141	0.132
W-HM$^{(78)}$	0.200	0.166	0.132	0.206	0.152	0.130
DW$^{(1/4)}$	0.202	0.168	0.130	0.207	**0.109**	0.130
Mink$^{(1/4)}$	0.199	0.169	0.131	0.210	**0.123**	0.137

computation of motion vectors, which are expensive and may not be readily available.

This issue was addressed in [68], which proposed a spatio-temporal SSIM which did not use motion information. Instead, the authors visualize the video as a 3D volume in $x, y,$ and t, where the frames lie in the $x-y$ planes. The $x-t$ and $y-t$ planes contain both spatial and temporal information, and can also be compared using SSIM. The spatio-temporal SSIM (ST-SSIM) model is then defined as the average of the three SSIM values. Note that neighborhoods in the $x-y$, $x-t$

TABLE 14: Comparing the Performance of Spatial Pooling Methods on Compressed Images

(a) LIVE IQA (Comp) Database

Method	PCC	SROCC	RMSE
AM	0.9701	0.9683	0.0699
CoV	0.8391	0.9675	0.1564
MD$^{(2,1)}$	0.9076	0.9646	0.1209
FNS	0.9675	0.9656	0.0728
DW$^{(1/4)}$	**0.9715**	**0.9690**	**0.0682**
Mink$^{(2)}$	0.8807	0.9693	0.1364

(b) TID2013 (Comp) Database

Method	PCC	SROCC	RMSE
AM	0.9438	0.9283	0.0777
CoV	0.9342	0.9505	0.0839
MD$^{(2,1)}$	0.9621	0.9499	0.0641
FNS	0.9295	0.9109	0.0868
DW$^{(1/4)}$	0.9649	0.9519	0.0618
Mink$^{(2)}$	**0.9703**	**0.955**	**0.0568**

TABLE 16: Comparing the SROCC Achieved by Pooling Methods on Compressed LIVE VQA Videos

TP \ SP	AM	CoV	MD$^{(2,3)}$	FNS	DW$^{(1)}$	Mink$^{(4)}$
AM	0.692	0.704	**0.785**	0.675	0.694	0.708
GM	0.694	0.692	0.691	0.674	0.696	0.694
HM	0.692	0.680	0.266	0.616	0.695	0.688
CoV	0.663	0.139	0.058	0.650	0.680	0.004
MD$^{(2,1)}$	0.649	0.726	**0.760**	0.636	0.662	0.678
FNS	0.669	0.677	0.669	0.655	0.669	0.677
W-AM$^{(99)}$	0.697	0.705	**0.781**	0.684	0.694	0.708
W-GM$^{(89)}$	0.698	0.694	0.693	0.685	0.699	0.696
W-HM$^{(80)}$	0.699	0.686	0.660	0.689	0.699	0.686
DW$^{(2)}$	0.748	0.698	**0.785**	0.723	**0.762**	0.693
Mink$^{(1/8)}$	0.694	0.692	0.702	0.674	0.696	0.694

and $y-t$ directions are special cases of 3-D neighborhoods used in SSIM-3D (obtained by setting the size along one dimension to 1). So, while we discuss ST-SSIM for completeness, we did not include it in our experiments.

I. FINAL RESULTS

Here, we provide comprehensive results of our experiments with the various spatial and temporal pooling algorithms described above. In all cases, we refer to each pooling method by the abbreviations listed above. To include information about the choice of optimal hyperparameters, we added superscripts to the abbreviated algorithm names. So, we denote Mean-Deviation Pooling by MD$^{(p,o)}$, Distortion-Weighted Pooling by DW$^{(p)}$, Minkowski Pooling by Mink$^{(p)}$ and the Windowed AM, GM and HM algorithms by W-AM$^{(k)}$, W-GM$^{(k)}$ and W-HM$^{(k)}$ respectively, where k denotes the window size. Once again, we linearized pooled SSIM scores by fitting them with the five-parameter logistic function in (17), and report performance in terms of the PCC, SROCC and RMSE values.

The performances of the various spatial pooling methods on the two image databases is tabulated in Table 11. On the video databases, we tested all pairs of spatial and temporal pooling methods, and these results are given in Table 13. In these tables, the columns represent the choice of spatial pooling (SP) method, while the rows represent the choice of temporal pooling (TP) methods.

In Table 11, the best performing spatial pooling method is boldfaced. We found that CoV pooling performed best on the challenging TID 2013 database, while the baseline Mean SSIM method performed best on the LIVE IQA database, with CoV pooling a close second.

In Table 13, we boldfaced the five best results in each sub-table. It may be observed that Mean Deviation Pooling and CoV pooling performed best among the spatial pooling methods, while using large windowed means performed well among the temporal pooling methods, significantly outperforming the spatio-temporal SSIM-3D and MS-SSIM-3D algorithms. However, as discussed earlier, using windowed means requires large windows ($k \approx 50, 80$) while providing only a minor performance improvement over the baseline. In addition, because CoV pooling performed consistently well across all databases and does not have any hyperparameters to tune, we recommend using spatial CoV pooling on picture or video frame quality maps, and the standard arithmetic mean pooling of temporal frame SSIM scores. This quality aggregation method is identical to the one used in the MOVIE index.

As in previous sections, we repeated the experiments on compression distorted data, and reported the results in Tables 14 and 16. Even when we restricted the distortion types, we did not obtain concordant values of hyperparameters across the video databases.

Based on our recommendations, we propose a variant of SSIM, called "Enhanced SSIM", which we are making publicly available to the community as a command line tool. The specifications of Enhanced SSIM are described below.

1) Operates only on the luma channel.
2) Uses rectangular windows to calculate local statistics, with a default size of 11x11.
3) Rectangular windows are implemented using integral images, by default.
4) Local quality scores are computed with a stride of 5.
5) The input image is down-sampled by a factor corre-

sponding to provided values of D/H, using a default ratio of 3.0, corresponding to a typical D/H ratio for TV viewing.

6) Coefficient of Variation pooling is used to spatially aggregate the local quality scores.

We compare the performance of our implementation with LIBVMAF in Table 17, since LIBVMAF was the best performing implementation in Section IV. This table highlights the computational and performance benefits of using our recommendations. Once again, "(Comp)" refers to experiments conducted on compression data from each database.

XI. CONCLUSION

In this guide, we detailed the results of a series of experiments we conducted towards determining optimal design choices when deploying SSIM. We first evaluated the off-the-shelf performance and efficiency of several public implementations of SSIM and MS-SSIM, using which we identified a set of Pareto-optimal implementations. Using these results, we also described a method to improve the computational efficiency of SSIM using integral images. We then reviewed a method, called Scaled SSIM, to improve the efficiency of computing SSIM across resolutions, when conducting RDO in encoding pipelines. Following this, we reviewed the dependence of SSIM performance on the viewing device, where we discussed improvements to SSIM which account for viewing distance and screen size. We then investigated the dependence of SSIM on the choice of window, where we conducted extensive experiments to identify good choices for the size and type of window function (rectangular windows of size 15-20), thereby validating some observations we made when testing public SSIM implementations.

Due to the non-linear nature of SSIM, it is crucial to develop a good mapping function from SSIM scores to subjective scores. We tested a popular choice of such a mapping, the five parameter logistic function, and demonstrated its generalizability. Further, while the baseline SSIM model is defined on two luminance images, most practical applications involve media having color. To account for this, we reviewed several Color SSIM models and compared their performance, finding that Quaternion SSIM was a consistently good choice. Finally, we performed a comprehensive evaluation of spatial and temporal aggregation methods used to deploy SSIM on videos. Based on these results, we recommended using spatial CoV pooling and temporal arithmetic mean pooling of framewise SSIM scores.

Database	LIBVMAF	Enhanced
LIVE IQA	**0.9464**	0.9377
TID 2013	0.7558	**0.7617**
LIVE VQA	0.6954	**0.7756**
Netflix Public	0.7652	**0.8360**
LIVE IQA (Comp)	0.9543	**0.9660**
TID 2013 (Comp)	0.9467	**0.9482**
LIVE VQA (Comp)	0.6839	**0.6936**

TABLE 17: Performance of Enhanced SSIM

In all, we have conducted a comprehensive study of many design choices involved when implementing SSIM, and made recommendations on the best practices. In addition, we have incorporated these recommendations into a variant of SSIM which we call Enhanced SSIM, for which we provide an openware command line tool for use by video quality engineers in academia and industry here.

REFERENCES

[1] Global internet phenomena report 2019. [Online]. Available: https://www.sandvine.com/global-internet-phenomena-report-2019

[2] I. DIS, "10918-1. digital compression and coding of continuous-tone still images (jpeg)," *CCITT Recommendation T*, vol. 81, p. 6, 1991.

[3] W. B. Pennebaker and J. L. Mitchell, *JPEG: Still Image Data Compression Standard*. Springer Science & Business Media, 1992.

[4] T. Wiegand, "Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T Rec. H. 264| ISO/IEC 14496-10 AVC)," *JVT-G050*, 2003.

[5] I. E. Richardson, *The H. 264 Advanced Video Compression Standard*. John Wiley & Sons, 2011.

[6] G. J. Sullivan, J. Ohm, W. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[7] D. Mukherjee, J. Bankoski, A. Grange, J. Han, J. Koleszar, P. Wilkins, Y. Xu, and R. Bultje, "The latest open-source video codec VP9 - An overview and preliminary results," in *Picture Coding Symposium (PCS)*, 2013, pp. 390–393.

[8] Y. Chen, D. Murherjee, J. Han, A. Grange, Y. Xu, Z. Liu, S. Parker, C. Chen, H. Su, U. Joshi, C. Chiang, Y. Wang, P. Wilkins, J. Bankoski, L. Trudeau, N. Egge, J. Valin, T. Davies, S. Midtskogen, A. Norkin, and P. de Rivaz, "An overview of core coding tools in the AV1 Video Codec," in *Picture Coding Symposium (PCS)*, 2018, pp. 41–45.

[9] D. Ghadiyaram and A. C. Bovik, "Massive online crowdsourced study of subjective and objective picture quality," *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 372–387, 2016.

[10] Z. Tu, Y. Wang, N. Birkbeck, B. Adsumilli, and A. C. Bovik, "UGC-VQA: Benchmarking blind video quality assessment for user generated content," *ArXiv*, vol. abs/2005.14354, 2020.

[11] I. Katsavounidis. Dynamic optimizer - a perceptual video encoding optimization framework. [Online]. Available: https://netflixtechblog.com/dynamic-optimizer-a-perceptual-video-encoding-optimization-framework-e19f1e3a277f

[12] 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.

[13] Zhou Wang and A. C. Bovik, "A universal image quality index," *IEEE Signal Processing Letters*, vol. 9, no. 3, pp. 81–84, 2002.

[14] 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.

[15] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *Asilomar Conference on Signals, Systems Computers*, 2003, pp. 1398–1402 Vol.2.

[16] Z. Wang, L. Lu, and A. C. Bovik, "Video quality assessment based on structural distortion measurement," *Signal Processing: Image Communication*, vol. 19, no. 2, pp. 121 – 132, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0923596503000766

[17] K. Seshadrinathan and A. C. Bovik, "A structural similarity metric for video based on motion models," in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, vol. 1, 2007, pp. I–869–I–872.

[18] K. Seshadrinathan and A. C. Bovik, "Motion tuned spatio-temporal quality assessment of natural videos," *IEEE Transactions on Image Processing*, vol. 19, no. 2, pp. 335–350, 2010.

[19] M. J. Wainwright and E. P. Simoncelli, "Scale mixtures of gaussians and the statistics of natural images," in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, ser. NIPS'99. Cambridge, MA, USA: MIT Press, 1999, p. 855–861.

[20] Z. Wang and A. C. Bovik, "Reduced- and no-reference image quality assessment," *IEEE Signal Processing Magazine*, vol. 28, no. 6, pp. 29–40, 2011.

[21] 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.

[22] R. Soundararajan and A. C. Bovik, "Video quality assessment by reduced reference spatio-temporal entropic differencing," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 4, pp. 684–694, 2013.

[23] C. G. Bampis, P. Gupta, R. Soundararajan, and A. C. Bovik, "SpEED-QA: Spatial efficient entropic differencing for image and video quality," *IEEE Signal Processing Letters*, vol. 24, no. 9, pp. 1333–1337, 2017.

[24] Z. Li, A. Aaron, I. Katsavounidis, A. Moorthy, and M. Manohara, "Toward a practical perceptual video quality metric," *The Netflix Tech Blog*, vol. 6, p. 2, 2016.

[25] K. Seshadrinathan and A. C. Bovik, "Unifying analysis of full reference image quality assessment," in *2008 15th IEEE International Conference on Image Processing*, 2008, pp. 1200–1203.

[26] M. A. Saad, A. C. Bovik, and C. Charrier, "Blind image quality assessment: A natural scene statistics approach in the DCT domain," *IEEE Transactions on Image Processing*, vol. 21, no. 8, pp. 3339–3352, 2012.

[27] A. K. Moorthy and A. C. Bovik, "Blind image quality assessment: From natural scene statistics to perceptual quality," *IEEE Transactions on Image Processing*, vol. 20, no. 12, pp. 3350–3364, 2011.

[28] A. Mittal, A. K. Moorthy, and A. C. Bovik, "No-reference image quality assessment in the spatial domain," *IEEE Transactions on Image Processing*, vol. 21, no. 12, pp. 4695–4708, 2012.

[29] A. Mittal, R. Soundararajan, and A. C. Bovik, "Making a "completely blind" image quality analyzer," *IEEE Signal Processing Letters*, vol. 20, no. 3, pp. 209–212, 2013.

[30] A. Bovik, "Assessing quality of images or videos using a two-stage quality assessment," Jan. 7 2020, US Patent 10,529,066.

[31] X. Yu, C. G. Bampis, P. Gupta, and A. C. Bovik, "Predicting the quality of images compressed after distortion in two steps," *IEEE Transactions on Image Processing*, vol. 28, no. 12, pp. 5757–5770, 2019.

[32] H. R. Sheikh, "Image and video quality assessment research at live," *http://live. ece. utexas. edu/research/quality*, 2003.

[33] H. R. Sheikh, M. F. Sabir, and A. C. Bovik, "A statistical evaluation of recent full reference image quality assessment algorithms," *IEEE Transactions on Image Processing*, vol. 15, no. 11, pp. 3440–3451, 2006.

[34] N. Ponomarenko, L. Jin, O. Ieremeiev, V. Lukin, K. Egiazarian, J. Astola, B. Vozel, K. Chehdi, M. Carli, F. Battisti, and C.-C. J. Kuo], "Image database TID2013: Peculiarities, results and perspectives," *Signal Processing: Image Communication*, vol. 30, pp. 57 – 77, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0923596514001490

[35] K. Seshadrinathan, R. Soundararajan, A. C. Bovik, and L. K. Cormack, "Study of subjective and objective quality assessment of video," *IEEE Transactions on Image Processing*, vol. 19, no. 6, pp. 1427–1441, 2010.

[36] Ffmpeg. [Online]. Available: https://ffmpeg.org/

[37] A. Tourapis and D. Singer, "HDRTools: A software package for video processing and analysis," in *ISO/IEC JTC1/SC29IWG11 MPEG2014/m35156*, 2014.

[38] T. J. Daede, N. E. Egge, J. Valin, G. Martres, and T. B. Terriberry, "Daala: A perceptually-driven next generation video codec," *CoRR*, vol. abs/1603.03129, 2016. [Online]. Available: http://arxiv.org/abs/1603.03129

[39] S. Van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu, "Scikit-image: image processing in Python," *PeerJ*, vol. 2, p. e453, 2014.

[40] Scikit-video. [Online]. Available: http://www.scikit-video.org/

[41] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[42] P. G. Gottschalk and J. R. Dunn, "The five-parameter logistic: a characterization and comparison with the four-parameter logistic," *Analytical Biochemistry*, vol. 343, no. 1, pp. 54–65, 2005.

[43] F. C. Crow, "Summed-area tables for texture mapping," *SIGGRAPH Comput. Graph.*, vol. 18, no. 3, p. 207–212, Jan. 1984. [Online]. Available: https://doi.org/10.1145/964965.808600

[44] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[45] A. K. Venkataramanan, C. Wu, and A. C. Bovik, "Optimizing video quality estimation across resolutions," *IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP) (Accepted)*, 2020.

[46] R. Schatz and S. Egger, "On the impact of terminal performance and screen size on qoe," in *Proc. ETSI Workshop Sel. Items Telecommun. Qual. Matters*, 2012, pp. 1–26.

[47] M. E. Grabe, M. Lombard, R. D. Reich, C. C. Bracken, and T. B. Ditton, "The role of screen size in viewer experiences of media content," *Visual Communication Quarterly*, vol. 6, no. 2, pp. 4–9, 1999.

[48] F. W. Campbell and J. G. Robson, "Application of fourier analysis to the visibility of gratings," *The Journal of physiology*, vol. 197, no. 3, p. 551, 1968.

[49] S. Winkler, "Issues in vision modeling for perceptual video quality assessment," *Signal Processing*, vol. 78, no. 2, pp. 231–252, 1999.

[50] I. BT, "Recommendation ITU-R BT.500-14 (10/2019): Methodologies for the subjective assessment of the quality of television images," *ITU, Geneva*, 2020.

[51] K. Gu, G. Zhai, X. Yang, and W. Zhang, "Self-adaptive scale transform for iqa metric," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*. IEEE, 2013, pp. 2365–2368.

[52] K. Gu, G. Zhai, M. Liu, Q. Xu, X. Yang, J. Zhou, and W. Zhang, "Adaptive high-frequency clipping for improved image quality assessment," in *Visual Communications and Image Processing (VCIP)*, 2013, pp. 1–5.

[53] K. Gu, M. Liu, G. Zhai, X. Yang, and W. Zhang, "Quality assessment considering viewing distance and image resolution," *IEEE Transactions on Broadcasting*, vol. 61, no. 3, pp. 520–531, 2015.

[54] G. Cermak, M. Pinson, and S. Wolf, "The relationship among video quality, screen resolution, and bit rate," *IEEE Transactions on Broadcasting*, vol. 57, no. 2, pp. 258–262, 2011.

[55] W. Zou, J. Song, and F. Yang, "Perceived image quality on mobile phones with different screen resolution," *Mobile Information Systems*, vol. 2016, 2016.

[56] M. Hassan and C. Bhagvati, "Structural similarity measure for color images," *International Journal of Computer Applications*, vol. 43, no. 14, pp. 7–12, 2012.

[57] C. I. De L'Eclairage, "Recommendations on uniform color spaces, color-difference equations, psychometric color terms," *Paris: CIE*, 1978.

[58] M. A. Hassan and M. S. Bashraheel, "Color-based structural similarity image quality assessment," in *2017 8th International Conference on Information Technology (ICIT)*. IEEE, 2017, pp. 691–696.

[59] Y. Shi, Y. Ding, R. Zhang, and J. Li, "Structure and hue similarity for color image quality assessment," in *2009 International Conference on Electronic Computer Technology*. IEEE, 2009, pp. 329–333.

[60] A. Kolaman and O. Yadid-Pecht, "Quaternion structural similarity: a new quality index for color images," *IEEE Transactions on Image Processing*, vol. 21, no. 4, pp. 1526–1536, 2011.

[61] M. Seufert, M. Slanina, S. Egger, and M. Kottkamp, ""to pool or not to pool": A comparison of temporal pooling methods for http adaptive video streaming," in *2013 Fifth International Workshop on Quality of Multimedia Experience (QoMEX)*, 2013, pp. 52–57.

[62] C. G. Zewdie, M. Pedersen, and Z. Wang, "A new pooling strategy for image quality metrics: Five number summary," in *2014 5th European Workshop on Visual Information Processing (EUVIP)*, 2014, pp. 1–6.

[63] H. Ziaei Nafchi, A. Shahkolaei, R. Hedjam, and M. Cheriet, "Mean deviation similarity index: Efficient and reliable full-reference image quality evaluator," *IEEE Access*, vol. 4, pp. 5579–5590, 2016.

[64] A. K. Moorthy and A. C. Bovik, "Visual importance pooling for image quality assessment," *IEEE Journal of Selected Topics in Signal Processing*, vol. 3, no. 2, pp. 193–201, 2009.

[65] U. Rajashekar, I. van der Linde, A. C. Bovik, and L. K. Cormack, "Gaffe: A gaze-attentive fixation finding engine," *IEEE Transactions on Image Processing*, vol. 17, no. 4, pp. 564–573, 2008.

[66] A. K. Moorthy and A. C. Bovik, "Efficient motion weighted spatio-temporal video SSIM index," in *Human Vision and Electronic Imaging XV*, B. E. Rogowitz and T. N. Pappas, Eds., vol. 7527, International Society for Optics and Photonics. SPIE, 2010, pp. 440 – 448. [Online]. Available: https://doi.org/10.1117/12.844198

[67] A. K. Moorthy and A. C. Bovik, "Efficient video quality assessment along temporal trajectories," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 11, pp. 1653–1658, 2010.

[68] Y. Wang, T. Jiang, S. Ma, and W. Gao, "Spatio-temporal ssim index for video quality assessment," in *2012 Visual Communications and Image Processing*, 2012, pp. 1–6.