Predicting and Optimizing Image Compression

Oleksandr Murashko
School of Computer Science
University of St Andrews, UK
om21@st-andrews.ac.uk

John Thomson
School of Computer Science
University of St Andrews, UK
j.thomson@st-andrews.ac.uk

Hugh Leather
School of Informatics
University of Edinburgh, UK
hleather@inf.ed.ac.uk

ABSTRACT

Image compression is a core task for mobile devices, social media and cloud storage backend services. Key evaluation criteria for compression are: the quality of the output, the compression ratio achieved and the computational time (and energy) expended. Predicting the effectiveness of standard compression implementations like libjpeg and WebP on a novel image is challenging, and often leads to non-optimal compression.

This paper presents a machine learning-based technique to accurately model the outcome of image compression for arbitrary new images in terms of quality and compression ratio, without requiring significant additional computational time and energy. Using this model, we can actively adapt the aggressiveness of compression on a per image basis to accurately fit user requirements, leading to a more optimal compression.

Keywords

Image compression; machine learning; JPEG; WebP

1. INTRODUCTION

Photography is one of the key use scenarios for mobile devices. The demands of the modern consumer are ever increasing – a single photo might now be expected to be automatically uploaded to multiple cloud services, re-compressed, edited, and shared with others via social media apps. This combined with the trend for high resolution sensors in mobile cameras means significantly increased pressure on cellular networks and storage requirements. Similarly, social media and cloud storage services frequently serve images at different resolutions, requiring multiple server-side compressions of the uploaded source, often on-the-fly. There is a clear incentive to minimize file size of images, while maintaining a required level of quality, and not increasing energy consumption, which is so critical to both mobile and cloud services.

MM '16, October 15–19, 2016, Amsterdam, The Netherlands.

ISBN 978-1-4503-3603-1/16/10…$15.00

DOI: http://dx.doi.org/10.1145/2964284.2967305

This paper presents a machine learning-based technique to accurately predict the outcome of image compression from an uncompressed source like a camera sensor, or a resized image, in terms of image quality and compression ratio. Using our models, we can actively adapt the aggressiveness of compression on a per image basis to accurately fit user requirements in terms of required image quality (as measured by PSNR or MSSIM [21]) or compression ratio, while maximizing the free objective.

User requirements are essentially arbitrary, but our technique can adapt to any desired quality level or target file size. Taking a common usage scenario where a mobile device compresses 16 MP image into JPEG at *quality factor* (QF) 90, we observe the 5th percentile of quality over a large dataset. Assuming this as a minimum acceptable quality, we can maintain the same minimum quality while reducing the average file size by 38%, resulting in consequent savings in bandwidth, storage and energy of transmission.

We employ a classic machine learning approach to build a statistical model – extracting features from uncompressed images using a fast analysis pass, and mapping these features and compression parameters to compression ratio. By building the model based on a dataset of 50 000 different images, from 0.24 MP to 24 MP, we are able to adequately characterise the space of images, such that new, unseen images can be predicted accurately. We implement our technique for both *libjpeg* [2] and Google's *WebP* [1] compressors, with each requiring its own models.

2. RELATED WORK

The most well known open source JPEG codecs, *libjpeg* at the time of writing, [2] has no explicit means to control target file size or quality. Instead, the user is asked to select a *quality factor* between 0 and 100, which scales the quantization table. Importantly, the user is effectively asked to parametrize an intricacy of an algorithm, rather than specifying their actual needs in a meaningful or measurable way. It is well known that both compression ratio and quality of output are highly dependant on the content of the input image when using JPEG [4]. Some image processing suites attempt to solve this problem by performing an exhaustive or binary search with different compression options, but this is unsatisfactory – such a search increases the compression time linearly with the number of iterations, and while this may have limited effect when compressing a single image, the effect is considerable when batch processing. Additionally, *energy consumption* is a key metric in multimedia from embedded systems to data centers, and in the general case,

execution time and energy consumption are known to be highly correlated [9].

Google's WebP compressor does provide options to more accurately target a particular image quality (specified in PSNR) or target file size, but at the expense of significantly longer compression time. The WebP manual [1] explains this additional time by making 'several passes of partial encoding' to fit the given constraints. By contrast, our work requires no additional recompressions.

Several papers have looked at the problem of predicting the effects of recompressing already compressed images.

Chandra et al. consider JPEG image transcoding [4]. Their method involves generating a statistical table to determine if recompressing a particular image is worthwhile in terms of predefined tradeoff between reduced size and lower quality. To navigate this table, they use parameters of the compressed JPEG image obtained from manipulating DCT coefficients. The idea of using coefficients of image spectral transformations has inspired our feature design.

Coulombe and Pigeon present their own predictor of file size and image complexity for JPEG over a number of papers describing similar techniques [14, 5, 15, 16, 17, 11, 18]. The key idea underpinning these works is to use the compression ratio of an already encoded image as input to nearest neighbor and table-based predictors, under the assumption that file size and quality grow monotonically with increasing quality factor, and neighbor values of quality factor correspond to similar compression ratio and quality level. Our approach differs as it does not require the input image to be compressed and instead works on uncompressed images. Nevertheless, we compare experimentally with their work and show a significantly lower error in our model, despite having less information as input.

There are a number of papers that discuss image feature extraction for a range of purposes. Many of these features are complex and related mostly to computer vision problems like in [12]. However, others consider low-level features – in particular, discrete wavelet transform coefficients are used by Hanghang et al. [19] to indicate the amount of blur in the images and by Viola et al. [20] to detect faces. We do not believe that features designed for vision are likely to be transferable to our problem.

3. METHODOLOGY

In order to predict the image quality and file size of our compressed images, we employ a classical machine learning approach. We gathered a dataset of 500 000 uncompressed images – a set of 100 different image resolutions logarithmically distributed by size in the range $[0.24; 24]$ megapixels. These images were resampled down from larger photographs to the desired size using the Fant method [6], which is a geometrically accurate anti-aliasing technique. The original images of different sizes from 24 MP to approximately 36 MP were collected from a wide selection of image-hosting websites and personal archives. None of the images were duplicates, which we engineered by design, and verified using an external tool – "Awesome Duplicate Photo Finder" [8]. The dataset was equally split into standard training (50 000 images), validation (50 000) and testing (400 000) sets. The experiments were conducted on an Intel Core i7 5820K @ 4.0 GHz, 16 GB RAM running Ubuntu 14.04 64-bit with GCC 4.8 compiler. The software used was libjpeg-turbo-1.4.2 and libwebp-0.4.3.

3.1 Feature description

Descriptive features are the key to any successful machine learning approach. We designed a feature set to capture the complexity and compression characteristics of the image from its pixel data.

The features were devised manually, based on the assumption that highly detailed and noisy images are less compressible – *e.g.* the average magnitude of the image gradient, the amplitude of high-frequency components of different spectral transformations like discrete cosine transform, discrete Haar wavelet transform and the Walsh-Hadamard transform [13, fig. 3]. The assumption behind the selection was that some high-frequency image characteristics correlate with the amount of noise in the image.

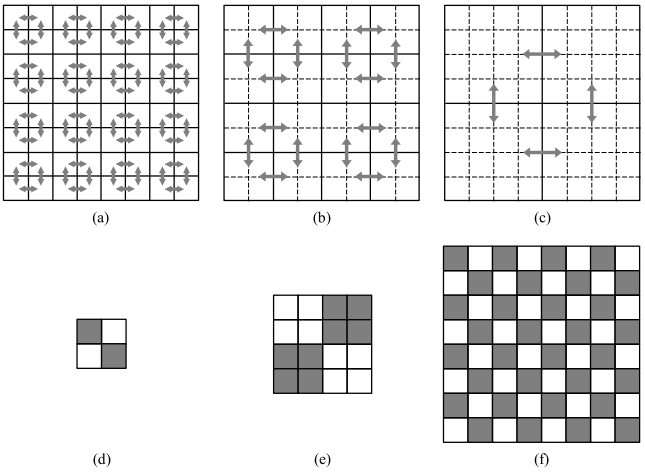

Figure 1: Differences between pixels and blocks for calculating features: (a) f_1 and f_4, (b) f_2 and f_5, (c) f_3 and f_6; and convolutions for features (d) f_7, (e) f_8, (f) f_9

The first step in calculating a set of content features for an image is to divide it onto fragments 8×8 pixels. This facilitates usage of Intel AVX instructions in program implementation. Subsequently, the raw pixel values in RGB format are converted to YC_bC_r color space following ITU-R BT.601 standard [7], but using full range for luminance and chrominance components.

We define feature $\boldsymbol{f_1}$ as a mean absolute neighbor pixel difference. Figure 1a shows selected differences in 8×8 fragment with arrows. More simply, if a and b are brightness levels of adjacent pixels, then this feature is an arithmetical average of $|a - b|$. In our implementation we consider only some of possible differences in the fragment to optimize for specific AVX instructions like horizontal subtraction.

Feature $\boldsymbol{f_2}$ is a mean absolute difference between neighbor blocks of 2×2 pixels. It can be considered as a "lower frequency" modification of the $\boldsymbol{f_1}$. Conceptually $\boldsymbol{f_2}$ is the same as $\boldsymbol{f_1}$, but instead of pixels a and b we use mean values for adjacent blocks 2×2 (figure 1b). Again, not all possible differences are considered due to usage of AVX instructions.

Feature $\boldsymbol{f_3}$ is a mean absolute difference between neighbor blocks of 4×4 pixels. $\boldsymbol{f_3}$ is an analogue of $\boldsymbol{f_1}$ and $\boldsymbol{f_2}$ (figure 1c).

Features $\boldsymbol{f_4}$, $\boldsymbol{f_5}$ and $\boldsymbol{f_6}$ are very similar to the described $\boldsymbol{f_1}$, $\boldsymbol{f_2}$, $\boldsymbol{f_3}$. They only use squared differences instead of

absolute, for example, $\boldsymbol{f_4}$ is a mean squared neighbor pixel difference.

Features $\boldsymbol{f_7}$ and $\boldsymbol{f_9}$ are mean absolute values of "checkboard" convolutions in all non-overlapping blocks of 2×2 and 8×8 pixels respectively (fig. 1d and 1f). By "checkboard" convolution we mean the last high-frequency basis function in 2-dimensional Walsh-Hadamard transform (WHT) of the respective size [13, fig. 3]. Feature $\boldsymbol{f_8}$ is another WHT coefficient of lower frequency calculated in blocks of 4×4 pixels (fig. 1e).

$\boldsymbol{f_{10}}$ is the only feature calculated for chrominance components of the image. $\boldsymbol{f_{10}}$ is exactly the same as $\boldsymbol{f_2}$, but it is calculated and averaged for both color components C_b and C_r instead of luminance Y.

The dynamic range of these features is quite large, and the majority of values are close to zero (but always > 0). Consequently the coverage of the feature space is very irregular. Therefore, we logarithmize all 10 features to make them suitable for regression: $\boldsymbol{f_i} \leftarrow ln(\boldsymbol{f_i} + 1)$. Increment is required to avoid $ln(0)$. Logarithmizing makes feature values distribution close to normal.

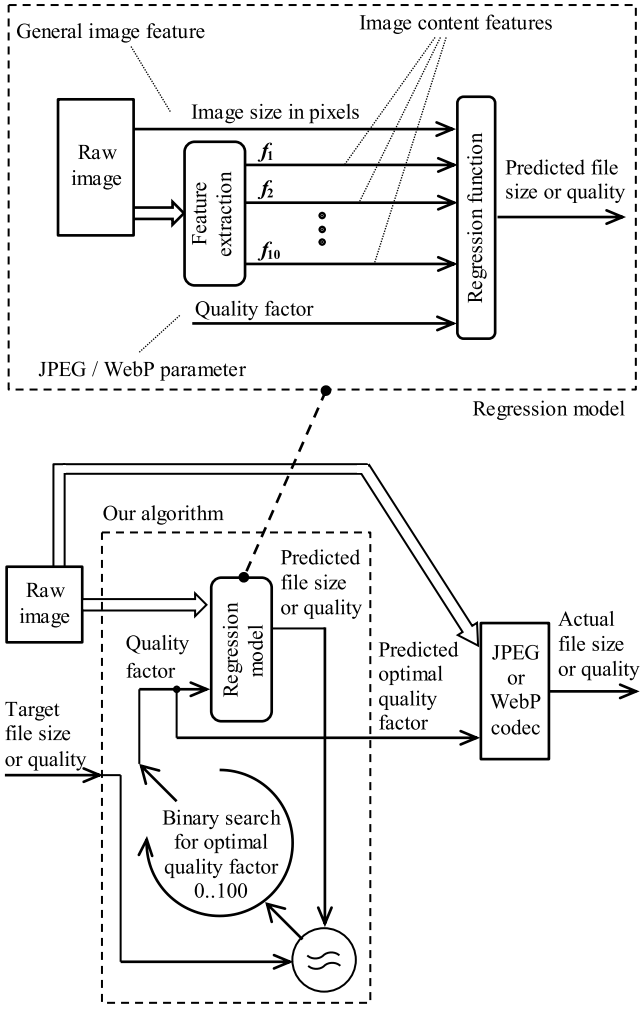

Figure 2: Sketch of our method.

3.2 Modelling

We construct our models empirically from the training data set, modelling the mapping from uncompressed input image and quality factor to either quality or file size. We do this for both *libjpeg* and WebP, meaning we use four models in total. As described in figure 2, the inputs to the model consist of: the image size in pixels, the quality factor used for compression and ten features extracted from the raw image as described above. In addition, all inputs to the regression models should be statistically standardized as we do not know which are the most important, so we assume that they have approximately the same influence on the result during training process. For this purpose we use z-score standardization [10, p. 524]:

$$standardized\_input\_value = \frac{input\_value - mean}{standard\_deviation}.$$

We employ a classic feedforward, back propogation multilayer perceptron with one hidden layer, using gradient descent, as described by Bishop [3]. Initially, we trained a simple linear regression model [3], but the results were poor, justifying the more complex non-linear approach. We also trained both linear regressors and MLPs with just the image size and quality factor as inputs, ignoring the features altogether. This approach was also unsuccessful, illustrating the importance of features in the regression. The best parametrization of the model was selected using the validation set, and the results reported on the testing set as is standard practice.

Once the models are trained, they can be used as a proxy to quickly search the optimization space for specified user requirement, as show in figure 2.

4. RESULTS

We present results in terms of the accuracy of predictions, and in real-world use case scenarios where we use the models to actually find optimal compression parameters. We demonstrate the technique and its universality using both JPEG and WebP formats.

A summary of the prediction error of the core regression models is presented in table 1. These models map input image and quality factor to the objective. The real-world compression results which follow all use these models.

Table 1: Mean errors of core regression functions

Objective	Average test set error
JPEG file size	3.06% (percentage error)
JPEG MSSIM	0.008 (absolute error)
WebP file size	4.75% (percentage error)
WebP PSNR	0.22 dB (absolute error)

4.1 JPEG

We evaluate the accuracy of our system for predicting an optimal JPEG quality factor in two real-world scenarios: targeting a file size and *mean SSIM* (MSSIM) value. We calculate the difference in file size and quality metric of the compressed image with the predicted quality factor from a sensible range of target values across our dataset, and compare average errors in our method with the respective values for the JPEG transcoding predictor.

We compare against a reimplementation of the Coulombe and Pigeon's JPEG transcoding method [5]. This method works only for recompressing already compressed images, so we initially compress the input image with random QF 60–95 as input for their model, requiring an additional compression. Nevertheless, we outperform Coulombe and Pigeon substantially in both prediction of file size, and quality as shown in figure 3.

Using our method, the file size mean absolute percentage error for the entire test set is 3.2%, which is expectedly similar to the error of the respective regression model in table 1. JPEG transcoding performed considerably worse, producing 10.3% error (fig. 3). The mean absolute error of fitting the MSSIM requirement in our method was much smaller than in JPEG transcoding: 0.008 versus 0.023.

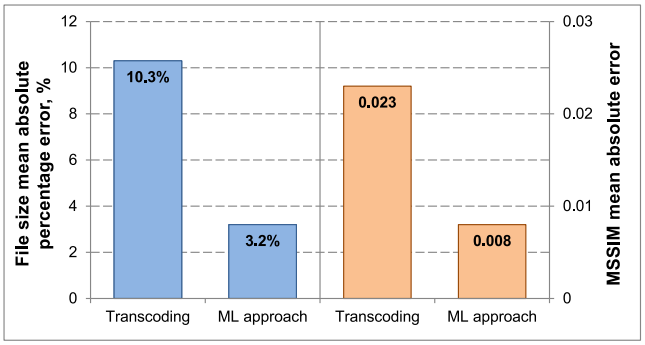

Figure 3: Error in objectives between target and actual JPEG images

The accuracy of the model allows users to confidently ask more from the compressor while maintaining important constraints. A potential use case might be a requirement for the minimum quality of compression not to drop below 0.94 MSSIM in more than 5% of cases. This is roughly the effect of selecting a QF of 90 for *libjpeg* – a commonly used setting. Using our accurate predictor, we can compress more aggressively in general, moving the error distribution, while maintaining the minimum quality constraint.

Figure 4 illustrates this on a dataset of 1000 unseen images of 16 MP – a common camera phone resolution. More aggressive compression gives a mean file size of 1.94 MB per image, compared to 3.12 MB for QF 90.

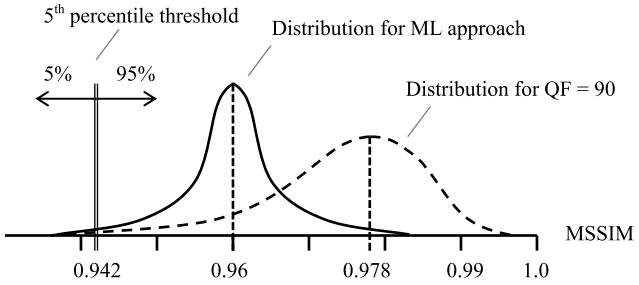

Figure 4: Distributions of quality for 16 MP images

4.2 WebP

Unlike *libjpeg*, WebP encoder already supports targeting quality and file size, allowing us to compare directly with the existing implementation. Here we use PSNR as the image quality metric as that is the metric used by WebP for quality targeting. In order to increase the accuracy of prediction, WebP supports a multiple pass option, which performs partial recompressions to facilitate a search to fit the objective, at the cost of increased compression time.

Figures 5 and 6 show a comparison of our approach with WebP's pass options for compressing 4000 images of 24 MP. Our approach greatly outperforms WebP in terms of accuracy for both quality and file size prediction, with a mean absolute error in PSNR of 0.63 dB compared to 5.36 dB, and a mean absolute percentage error of 5% compared to 109%. It is slightly slower however, due to the feature extraction stage. Allowing WebP four passes closes the gap significantly at the cost of longer compression time, but it is still outperformed by our technique. In an energy sensitive environment, such an increase is likely to be unacceptable.

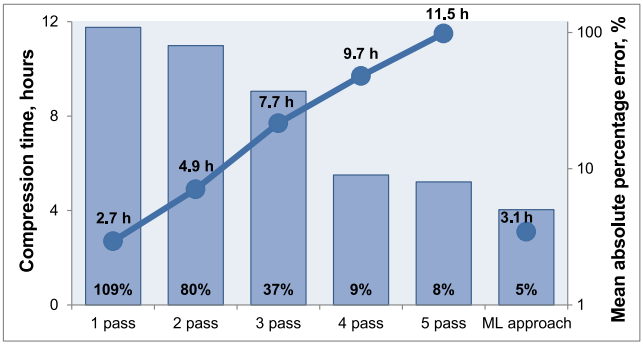

Figure 5: Error in file size between target and actual WebP images against time

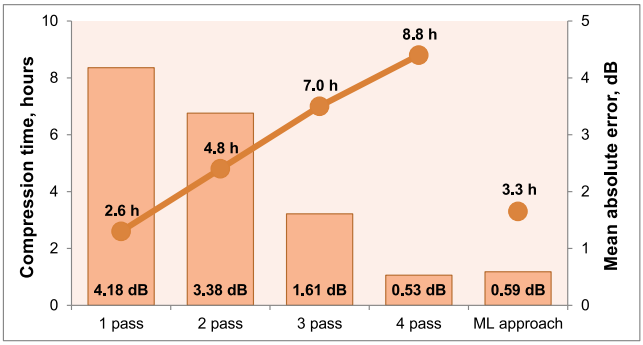

Figure 6: Error in image quality between target and actual WebP images against time

5. CONCLUSION

This paper presents a methodology to predict the results of image compression, based only on an uncompressed source. Building models for two standard compressors, *libjpeg* and WebP, we can automatically control image compression aggression to accurately fulfil user requirements for file size and quality. We significantly improve upon previous approaches for JPEG, even from an uncompressed source, and we are able to obtain higher accuracy of prediction than WebP's implementation while taking significantly less time.

6. REFERENCES

[1] WebP manual, Google Developers. https://developers.google.com/speed/webp/docs/cwebp, December 2015.

[2] Independent JPEG group. http://www.ijg.org/, January 2016.

[3] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA, 1995.

[4] S. Chandra and C. S. Ellis. JPEG compression metric as a quality aware image transcoding. In *2nd USENIX Symposium on Internet Technologies & Systems (USITS'99)*, 1999.

[5] S. Coulombe and S. Pigeon. Low-complexity transcoding of JPEG images with near-optimal quality using a predictive quality factor and scaling parameters. *Image Processing, IEEE Transactions on*, 19(3):712–721, 2010.

[6] K. Fant. A nonaliasing, real-time spatial transform technique. *Computer Graphics and Applications, IEEE*, 6(1):71–80, January 1986.

[7] ITU. Recommendation ITU-R BT.601-7. https://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.601-7-201103-I!!PDF-E.pdf, March 2011.

[8] U. Javaid. Find & remove similar photos instantly. http://www.addictivetips.com/windows-tips/find-remove-similar-photos-instantly, June 2010.

[9] A. Lewis, S. Ghosh, and N.-F. Tzeng. Run-time energy consumption estimation based on workload in server systems. In *Proceedings of the 2008 Conference on Power Aware Computing and Systems*, HotPower'08, pages 4–4, Berkeley, CA, USA, 2008. USENIX Association.

[10] T. Little. *The Oxford Handbook of Quantitative Methods, Vol. 2: Statistical Analysis*. Oxford Library of Psychology. Oxford University Press, 2013.

[11] H. Louafi, S. Coulombe, and U. Chandra. Efficient near-optimal dynamic content adaptation applied to JPEG slides presentations in mobile web conferencing. In *Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on*, pages 724–731, March 2013.

[12] L. Marchesotti, F. Perronnin, D. Larlus, and G. Csurka. Assessing the aesthetic quality of photographs using generic image descriptors. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1784–1791, Nov 2011.

[13] M. Montgomery. Next generation video: Introducing daala part 3. https://people.xiph.org/~xiphmont/demo/daala/demo3.shtml, August 2013.

[14] S. Pigeon and S. Coulombe. Computationally efficient algorithms for predicting the file size of JPEG images subject to changes of quality factor and scaling. In *Communications, 2008 24th Biennial Symposium on*, pages 378–382, June 2008.

[15] S. Pigeon and S. Coulombe. Efficient clustering-based algorithm for predicting file size and structural similarity of transcoded JPEG images. In *Multimedia (ISM), 2011 IEEE International Symposium on*, pages 137–142, Dec 2011.

[16] S. Pigeon and S. Coulombe. Optimal quality-aware predictor-based adaptation of multimedia messages. In *Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), 2011 IEEE 6th International Conference on*, volume 1, pages 496–499, Sept 2011.

[17] S. Pigeon and S. Coulombe. K-means based prediction of transcoded JPEG file size and structural similarity. *International Journal of Multimedia Data Engineering and Management (IJMDEM)*, 3(2):41–57, 2012.

[18] S. Pigeon and S. Coulombe. Quality-aware predictor-based adaptation of still images for the multimedia messaging service. *Multimedia tools and applications*, 72(2):1841–1865, 2014.

[19] H. Tong, M. Li, H. Zhang, and C. Zhang. Blur detection for digital images using wavelet transform. In *Multimedia and Expo, 2004. ICME '04. 2004 IEEE International Conference on*, volume 1, pages 17–20 Vol.1, June 2004.

[20] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511–I–518 vol.1, 2001.

[21] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4):600–612, April 2004.