# END-TO-END OPTIMIZED IMAGE COMPRESSION

**Johannes Ballé***
Center for Neural Science
New York University
New York, NY 10003, USA
johannes.balle@nyu.edu

**Valero Laparra**
Image Processing Laboratory
Universitat de València
46980 Paterna, Spain
valero.laparra@uv.es

**Eero P. Simoncelli***
Center for Neural Science and Courant Institute of Mathematical Sciences
New York University
New York, NY 10003, USA
eero.simoncelli@nyu.edu

## ABSTRACT

We describe an image compression method, consisting of a nonlinear analysis transformation, a uniform quantizer, and a nonlinear synthesis transformation. The transforms are constructed in three successive stages of convolutional linear filters and nonlinear activation functions. Unlike most convolutional neural networks, the joint nonlinearity is chosen to implement a form of local gain control, inspired by those used to model biological neurons. Using a variant of stochastic gradient descent, we jointly optimize the entire model for rate–distortion performance over a database of training images, introducing a continuous proxy for the discontinuous loss function arising from the quantizer. Under certain conditions, the relaxed loss function may be interpreted as the log likelihood of a generative model, as implemented by a variational autoencoder. Unlike these models, however, the compression model must operate at any given point along the rate–distortion curve, as specified by a trade-off parameter. Across an independent set of test images, we find that the optimized method generally exhibits better rate–distortion performance than the standard JPEG and JPEG 2000 compression methods. More importantly, we observe a dramatic improvement in visual quality for all images at all bit rates, which is supported by objective quality estimates using MS-SSIM.

## 1 INTRODUCTION

Data compression is a fundamental and well-studied problem in engineering, and is commonly formulated with the goal of designing codes for a given discrete data ensemble with minimal entropy (Shannon, 1948). The solution relies heavily on knowledge of the probabilistic structure of the data, and thus the problem is closely related to probabilistic source modeling. However, since all practical codes must have finite entropy, continuous-valued data (such as vectors of image pixel intensities) must be quantized to a finite set of discrete values, which introduces error. In this context, known as the *lossy compression problem*, one must trade off two competing costs: the entropy of the discretized representation (*rate*) and the error arising from the quantization (*distortion*). Different compression applications, such as data storage or transmission over limited-capacity channels, demand different rate–distortion trade-offs.

Joint optimization of rate and distortion is difficult. Without further constraints, the general problem of optimal quantization in high-dimensional spaces is intractable (Gersho and Gray, 1992). For this reason, most existing image compression methods operate by linearly transforming the data vector into a suitable continuous-valued representation, quantizing its elements independently, and then encoding the resulting discrete representation using a lossless *entropy code* (Wintz, 1972; Netravali and Limb, 1980). This scheme is called *transform coding* due to the central role of the transforma-

*JB and EPS are supported by the Howard Hughes Medical Institute.

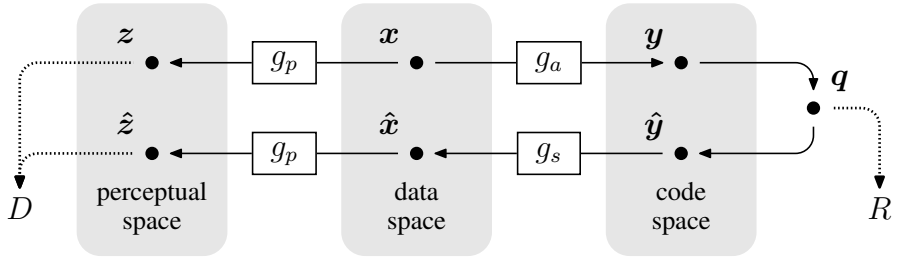



Figure 1: General nonlinear transform coding framework (Ballé, Laparra, and Simoncelli, 2016). A vector of image intensities $\boldsymbol{x} \in \mathbb{R}^N$ is mapped to a latent *code space* via a parametric *analysis* transform, $\boldsymbol{y} = g_a(\boldsymbol{x}; \boldsymbol{\phi})$. This representation is quantized, yielding a discrete-valued vector $\boldsymbol{q} \in \mathbb{Z}^M$ which is then compressed. The *rate* of this discrete code, $R$, is lower-bounded by the entropy of the discrete probability distribution of the quantized vector, $H[P_{\boldsymbol{q}}]$. To reconstruct the compressed image, the discrete elements of $\boldsymbol{q}$ are reinterpreted as a continuous-valued vector $\hat{\boldsymbol{y}}$, which is transformed back to the data space using a parametric *synthesis* transform $\hat{\boldsymbol{x}} = g_s(\hat{\boldsymbol{y}}; \boldsymbol{\theta})$. Distortion is assessed by transforming to a *perceptual space* using a (fixed) transform, $\hat{\boldsymbol{z}} = g_p(\hat{\boldsymbol{x}})$, and evaluating a metric $d(\boldsymbol{z}, \hat{\boldsymbol{z}})$. We optimize the parameter vectors $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ for a weighted sum of the rate and distortion measures, $R + \lambda D$, over a set of images.

tion. For example, JPEG uses a discrete cosine transform on blocks of pixels, and JPEG 2000 uses a multi-scale orthogonal wavelet decomposition. Typically, the three components of transform coding methods – transform, quantizer, and entropy code – are separately optimized (often through manual parameter adjustment).

We have developed a framework for end-to-end optimization of an image compression model based on *nonlinear* transforms (figure 1). Previously, we demonstrated that a model consisting of linear–nonlinear block transformations, optimized for a measure of perceptual distortion, exhibited visually superior performance compared to a model optimized for mean squared error (MSE) (Ballé, Laparra, and Simoncelli, 2016). Here, we optimize for MSE, but use a more flexible transforms built from cascades of linear convolutions and nonlinearities. Specifically, we use a generalized divisive normalization (GDN) joint nonlinearity that is inspired by models of neurons in biological visual systems, and has proven effective in Gaussianizing image densities (Ballé, Laparra, and Simoncelli, 2015). This cascaded transformation is followed by uniform scalar quantization (i.e., each element is rounded to the nearest integer), which effectively implements a parametric form of vector quantization on the original image space. The compressed image is reconstructed from these quantized values using an approximate parametric nonlinear inverse transform.

For any desired point along the rate–distortion curve, the parameters of both analysis and synthesis transforms are jointly optimized using stochastic gradient descent. To achieve this in the presence of quantization (which produces zero gradients almost everywhere), we use a proxy loss function based on a continuous relaxation of the probability model, replacing the quantization step with additive uniform noise. The relaxed rate–distortion optimization problem bears some resemblance to those used to fit generative image models, and in particular variational autoencoders (Kingma and Welling, 2014; Rezende, Mohamed, and Wierstra, 2014), but differs in the constraints we impose to ensure that it approximates the discrete problem all along the rate–distortion curve. Finally, rather than reporting differential or discrete entropy estimates, we implement an entropy code and report performance using actual bit rates, thus demonstrating the feasibility of our solution as a complete lossy compression method.

## 2 Choice of forward, inverse, and perceptual transforms

Most compression methods are based on orthogonal linear transforms, chosen to reduce correlations in the data, and thus to simplify entropy coding. But the joint statistics of linear filter responses exhibit strong higher order dependencies. These may be significantly reduced through the use of joint local nonlinear gain control operations (Schwartz and Simoncelli, 2001; Lyu, 2010; Sinz and Bethge, 2013), inspired by models of visual neurons (Heeger, 1992; Carandini and Heeger, 2012). Cascaded versions of such models have been used to capture multiple stages of visual transformation

(Simoncelli and Heeger, 1998; Mante, Bonin, and Carandini, 2008). Some earlier results suggest that incorporating local normalization in linear block transform coding methods can improve coding performance (Malo et al., 2006), and can improve object recognition performance of cascaded convolutional neural networks (Jarrett et al., 2009). However, the normalization parameters in these cases were not optimized for the task. Here, we make use of a generalized divisive normalization (GDN) transform with optimized parameters, that we have previously shown to be highly efficient in Gaussianizing the local joint statistics of natural images, much more so than cascades of linear transforms followed by pointwise nonlinearities (Ballé, Laparra, and Simoncelli, 2015).

Note that some training algorithms for deep convolutional networks incorporate "batch normalization", rescaling the responses of linear filters in the network so as to keep it in a reasonable operating range (Ioffe and Szegedy, 2015). This type of normalization is different from local gain control in that the rescaling factor is identical across all spatial locations. Moreover, once the training is completed, the scaling parameters are typically fixed, which turns the normalization into an affine transformation with respect to the data – unlike GDN, which is spatially adaptive and can be highly nonlinear.

Specifically, our analysis transform $g_a$ consists of three stages of convolution, subsampling, and divisive normalization. We represent the $i$th input channel of the $k$th stage at spatial location $(m, n)$ as $u_i^{(k)}(m, n)$. The input image vector $\boldsymbol{x}$ corresponds to $u_i^{(0)}(m, n)$, and the output vector $\boldsymbol{y}$ is $u_i^{(3)}(m, n)$. Each stage then begins with an affine convolution:

$$v_i^{(k)}(m, n) = \sum_j \big(h_{k,ij} * u_j^{(k)}\big)(m, n) + c_{k,i}, \tag{1}$$

where $*$ denotes 2D convolution. This is followed by downsampling:

$$w_i^{(k)}(m, n) = v_i^{(k)}(s_k m, s_k n), \tag{2}$$

where $s_k$ is the downsampling factor for stage $k$. Each stage concludes with a GDN operation:

$$u_i^{(k+1)}(m, n) = \frac{w_i^{(k)}(m, n)}{\Big(\beta_{k,i} + \sum_j \gamma_{k,ij} \big(w_j^{(k)}(m, n)\big)^2\Big)^{\frac{1}{2}}}. \tag{3}$$

The full set of $h$, $c$, $\beta$, and $\gamma$ parameters (across all three stages) constitute the parameter vector $\boldsymbol{\phi}$ to be optimized.

Analogously, the synthesis transform $g_s$ consists of three stages, with the order of operations reversed within each stage, downsampling replaced by upsampling, and GDN replaced by an approximate inverse we call IGDN (more details in the appendix). We define $\hat{u}_i^{(k)}(m, n)$ as the input to the $k$th synthesis stage, such that $\hat{\boldsymbol{y}}$ corresponds to $\hat{u}_i^{(0)}(m, n)$, and $\hat{\boldsymbol{x}}$ to $\hat{u}_i^{(3)}(m, n)$. Each stage then consists of the IGDN operation:

$$\hat{w}_i^{(k)}(m, n) = \hat{u}_i^{(k)}(m, n) \cdot \Big(\hat{\beta}_{k,i} + \sum_j \hat{\gamma}_{k,ij} \big(\hat{u}_j^{(k)}(m, n)\big)^2\Big)^{\frac{1}{2}}, \tag{4}$$

which is followed by upsampling:

$$\hat{v}_i^{(k)}(m, n) = \begin{cases} \hat{w}_i^{(k)}(m/\hat{s}_k, n/\hat{s}_k) & \text{if } m/\hat{s}_k \text{ and } n/\hat{s}_k \text{ are integers,} \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where $\hat{s}_k$ is the upsampling factor for stage $k$. Finally, this is followed by an affine convolution:

$$\hat{u}_i^{(k+1)}(m, n) = \sum_j \big(\hat{h}_{k,ij} * \hat{v}_j^{(k)}\big)(m, n) + \hat{c}_{k,i}. \tag{6}$$

Analogously, the set of $\hat{h}$, $\hat{c}$, $\hat{\beta}$, and $\hat{\gamma}$ make up the parameter vector $\boldsymbol{\theta}$. Note that the down-/upsampling operations can be implemented jointly with their adjacent convolution, improving computational efficiency.

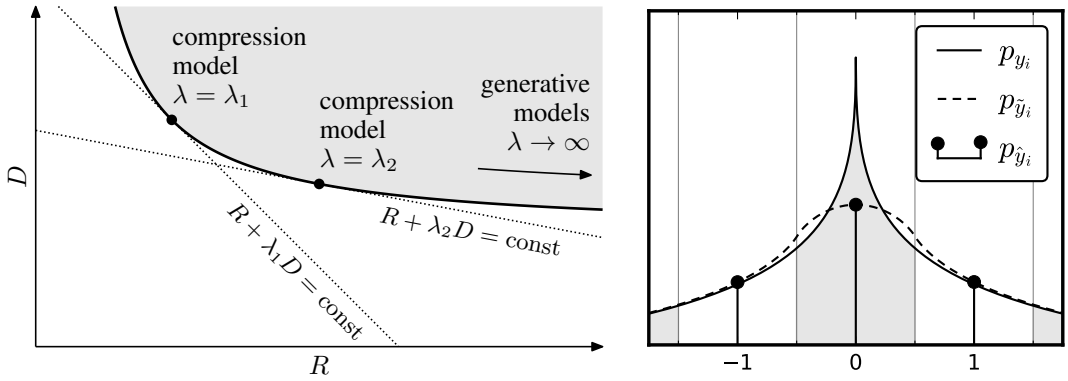



Figure 2: Left: The rate–distortion trade-off. The gray region represents the set of all rate–distortion values that can be achieved (over all possible parameter settings). Optimal performance for a given choice of $\lambda$ corresponds to a point on the convex hull of this set with slope $-1/\lambda$. Right: One-dimensional illustration of relationship between densities of $y_i$ (elements of code space), $\hat{y}_i$ (quantized elements), and $\tilde{y}_i$ (elements perturbed by uniform noise). Each discrete probability in $p_{\hat{y}_i}$ equals the probability mass of the density $p_{y_i}$ within the corresponding quantization bin (indicated by shading). The density $p_{\tilde{y}_i}$ provides a continuous function that interpolates the discrete probability values $p_{\hat{y}_i}$ at integer positions.

In previous work, we used a perceptual transform $g_p$, separately optimized to mimic human judgements of grayscale image distortions (Laparra et al., 2016), and showed that a set of one-stage transforms optimized for this distortion measure led to visually improved results (Ballé, Laparra, and Simoncelli, 2016). Here, we set the perceptual transform $g_p$ to the identity, and use mean squared error (MSE) as the metric (i.e., $d(\boldsymbol{z}, \hat{\boldsymbol{z}}) = \|\boldsymbol{z} - \hat{\boldsymbol{z}}\|_2^2$). This allows a more interpretable comparison to existing methods, which are generally optimized for MSE, and also allows optimization for color images, for which we do not currently have a reliable perceptual metric.

## 3 Optimization of nonlinear transform coding model

Our objective is to minimize a weighted sum of the rate and distortion, $R + \lambda D$, over the parameters of the analysis and synthesis transforms and the entropy code, where $\lambda$ governs the trade-off between the two terms (figure 2, left panel). Rather than attempting optimal quantization directly in the image space, which is intractable due to the high dimensionality, we instead assume a fixed uniform scalar quantizer in the code space, and aim to have the nonlinear transformations warp the space in an appropriate way, effectively implementing a parametric form of vector quantization (figure 1). The actual rates achieved by a properly designed entropy code are only slightly larger than the entropy (Rissanen and Langdon, 1981), and thus we define the objective functional directly in terms of entropy:

$$L[g_a, g_s, P_{\boldsymbol{q}}] = -\,\mathbb{E}\big[\log_2 P_{\boldsymbol{q}}\big] + \lambda\,\mathbb{E}\big[d(\boldsymbol{z}, \hat{\boldsymbol{z}})\big], \tag{7}$$

where both expectations will be approximated by averages over a training set of images. Given a powerful enough set of transformations, we can assume without loss of generality that the quantization bin size is always one and the representing values are at the centers of the bins. That is,

$$\hat{y}_i = q_i = \text{round}(y_i), \tag{8}$$

where index $i$ runs over all elements of the vectors, including channels and spatial locations. The marginal density of $\hat{y}_i$ is then given by a train of discrete probability masses (Dirac delta functions, figure 2, right panel) with weights equal to the probability mass function of $q_i$:

$$P_{q_i}(n) = \int_{n-\frac{1}{2}}^{n+\frac{1}{2}} p_{y_i}(t)\,\mathrm{d}t, \quad \text{for all } n \in \mathbb{Z}\,. \tag{9}$$

Note that both terms in (7) depend on the quantized values, and the derivatives of the quantization function (8) are zero almost everywhere, rendering gradient descent ineffective. To allow optimization via stochastic gradient descent, we replace the quantizer with an additive i.i.d. uniform noise

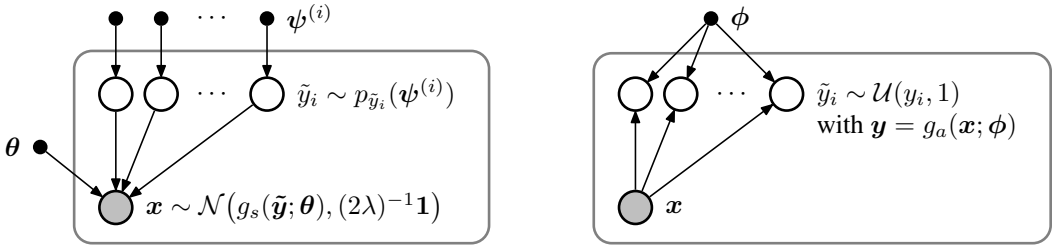



Figure 3: Representation of the relaxed rate–distortion optimization problem as the encoder and decoder graphs of a variational autoencoder. Nodes represent random variables, and gray shading indicates observed data; small filled nodes represent parameters; arrows indicate dependency; and nodes within boxes are per-image.

source $\Delta\boldsymbol{y}$, which has the same width as the quantization bins (one). This relaxed formulation has two desirable properties. First, the density function of $\tilde{\boldsymbol{y}} = \boldsymbol{y} + \Delta\boldsymbol{y}$ is a continuous relaxation of the probability mass function of $\boldsymbol{q}$ (figure 2, right panel):

$$p_{\tilde{\boldsymbol{y}}}(\boldsymbol{n}) = P_{\boldsymbol{q}}(\boldsymbol{n}), \quad \text{for all } \boldsymbol{n} \in \mathbb{Z}^M, \tag{10}$$

which implies that the differential entropy of $\tilde{\boldsymbol{y}}$ can be used as an approximation of the entropy of $\boldsymbol{q}$. Second, independent uniform noise approximates quantization error in terms of its marginal moments, and is frequently used as a model of quantization error (Gray and Neuhoff, 1998). We can thus use the same approximation for our measure of distortion. We examine the empirical quality of these rate and distortion approximations in section 4.

We assume independent marginals in the code space for both the relaxed probability model of $\tilde{\boldsymbol{y}}$ and the entropy code, and model the marginals $p_{\tilde{y}_i}$ non-parametrically to reduce model error. Specifically, we use finely sampled piecewise linear functions which we update similarly to one-dimensional histograms (see appendix). Since $p_{\tilde{y}_i} = p_{y_i} * \mathcal{U}(0, 1)$ is effectively smoothed by a box-car filter – the uniform density on the unit interval, $\mathcal{U}(0, 1)$ – the model error can be made arbitrarily small by decreasing the sampling interval.

Given this continuous approximation of the quantized coefficient distribution, the loss function for parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ can be written as:

$$L(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{\boldsymbol{x}, \Delta\boldsymbol{y}}\Bigg[-\sum_i \log_2 p_{\tilde{y}_i}(g_a(\boldsymbol{x}; \boldsymbol{\phi}) + \Delta\boldsymbol{y}; \boldsymbol{\psi}^{(i)}) \\ + \lambda\, d\Big(g_p\big(g_s(g_a(\boldsymbol{x}; \boldsymbol{\phi}) + \Delta\boldsymbol{y}; \boldsymbol{\theta})\big), g_p(\boldsymbol{x})\Big)\Bigg]. \tag{11}$$

where vector $\boldsymbol{\psi}^{(i)}$ parameterizes the piecewise linear approximation of $p_{\tilde{y}_i}$ (trained jointly with $\boldsymbol{\theta}$ and $\boldsymbol{\varphi}$). This is continuous and differentiable, and thus well-suited for stochastic optimization.

### 3.1 Relationship to variational generative image models

We derived our formulation directly from the classical rate–distortion optimization problem. However, once the transition to a continuous loss function is made, the optimization problem resembles those encountered in fitting generative models of images, and can more specifically be cast in the context of variational autoencoders (Kingma and Welling, 2014; Rezende, Mohamed, and Wierstra, 2014). In Bayesian variational inference, we are given an ensemble of observations of a random variable $x$ along with a generative model $p_{x|y}(x|y)$. We seek to find a posterior $p_{y|x}(y|x)$, which generally cannot be expressed in closed form. The approach followed by Kingma and Welling (2014) consists of approximating this posterior with a density $q(y|x)$, by minimizing the Kullback–Leibler divergence between the two:

$$\begin{aligned} D_{\mathrm{KL}}\big[q \| p_{y|x}\big] &= \mathbb{E}_{y\sim q} \log q(y|x) - \mathbb{E}_{y\sim q} \log p_{y|x}(y|x) \\ &= \mathbb{E}_{y\sim q} \log q(y|x) - \mathbb{E}_{y\sim q} \log p_{x|y}(x|y) - \mathbb{E}_{y\sim q} \log p_y(y) + \text{const.} \end{aligned} \tag{12}$$

This objective function is equivalent to our relaxed rate–distortion optimization problem, with distortion measured as MSE, if we define the generative model as follows (figure 3):

$$p_{\boldsymbol{x}|\tilde{\boldsymbol{y}}}(\boldsymbol{x}|\tilde{\boldsymbol{y}};\lambda,\boldsymbol{\theta}) = \mathcal{N}\big(\boldsymbol{x};\, g_s(\tilde{\boldsymbol{y}};\boldsymbol{\theta}),\, (2\lambda)^{-1}\mathbf{1}\big), \tag{13}$$

$$p_{\tilde{\boldsymbol{y}}}(\tilde{\boldsymbol{y}};\boldsymbol{\psi}^{(0)},\boldsymbol{\psi}^{(1)},\dots) = \prod_i p_{\tilde{y}_i}(\tilde{y}_i;\boldsymbol{\psi}^{(i)}), \tag{14}$$

and the approximate posterior as follows:

$$q(\tilde{\boldsymbol{y}}|\boldsymbol{x};\boldsymbol{\phi}) = \prod_i \mathcal{U}(\tilde{y}_i; y_i, 1) \quad \text{with } \boldsymbol{y} = g_a(\boldsymbol{x};\boldsymbol{\phi}), \tag{15}$$

where $\mathcal{U}(\tilde{y}_i; y_i, 1)$ is the uniform density on the unit interval centered on $y_i$. With this, the first term in the Kullback–Leibler divergence is constant; the second term corresponds to the distortion, and the third term corresponds to the rate (both up to additive constants). Note that if a perceptual transform $g_p$ is used, or the metric $d$ is not Euclidean, $p_{\boldsymbol{x}|\tilde{\boldsymbol{y}}}$ is no longer Gaussian, and equivalence to variational autoencoders cannot be guaranteed, since the distortion term may not correspond to a normalizable density. For any affine and invertible perceptual transform and any translation-invariant metric, it can be shown to correspond to the density

$$p_{\boldsymbol{x}|\tilde{\boldsymbol{y}}}(\boldsymbol{x}|\tilde{\boldsymbol{y}};\lambda,\boldsymbol{\theta}) = \frac{1}{Z(\lambda)} \exp\left(-\lambda\, d\Big(g_p\big(g_s(\tilde{\boldsymbol{y}};\boldsymbol{\theta})\big), g_p(\boldsymbol{x})\Big)\right), \tag{16}$$

where $Z(\lambda)$ normalizes the density (but need not be computed to fit the model).

Despite the similarity between our nonlinear transform coding framework and that of variational autoencoders, it is worth noting several fundamental differences. First, variational autoencoders are continuous-valued, and digital compression operates in the discrete domain. Comparing differential entropy with (discrete) entropy, or entropy with an actual bit rate, can potentially lead to misleading results. In this paper, we use the continous domain strictly for optimization, and perform the evaluation on actual bit rates, which allows comparison to existing image coding methods. We assess the quality of the rate and distortion approximations empirically.

Second, generative models aim to minimize differential entropy of the data ensemble under the model, i.e., explaining fluctuations in the data. This often means minimizing the variance of a "slack" term like (13), which in turn *maximizes* $\lambda$. Transform coding methods, on the other hand, are optimized to achieve the best trade-off between having the model explain the data (which increases rate and decreases distortion), and having the slack term explain the data (which decreases rate and increases distortion). The overall performance of a compression model is determined by the shape of the convex hull of attainable model distortions and rates, over all possible values of the model parameters. Finding this convex hull is equivalent to optimizing the model for *particular* values of $\lambda$ (see figure 2). In contrast, generative models operate in a regime where $\lambda$ is inferred and ideally approaches infinity for noiseless data, which corresponds to the regime of lossless compression. Even so, lossless compression methods still need to operate in a discretized space, typically directly on quantized luminance values. For generative models, the discretization of luminance values is usually considered a nuisance (Theis, van den Oord, and Bethge, 2015), although there are examples of generative models that operate on quantized pixel values (van den Oord, Kalchbrenner, and Kavukcuoglu, 2016).

Finally, although correspondence between the typical slack term (13) of a generative model (figure 3, left panel) and the distortion metric in rate–distortion optimization holds for simple metrics (e.g., Euclidean distance), a more general perceptual measure would be considered a peculiar choice from a generative modeling perspective, if it corresponds to a density at all.

## 4 Experimental results

We jointly optimized the full set of parameters $\boldsymbol{\phi}$, $\boldsymbol{\theta}$, and all $\boldsymbol{\psi}$ over a subset of the ImageNet database (Deng et al., 2009) consisting of 6507 images using stochastic descent. This optimization was performed separately for each $\lambda$, yielding separate transforms and marginal probability models for each value.

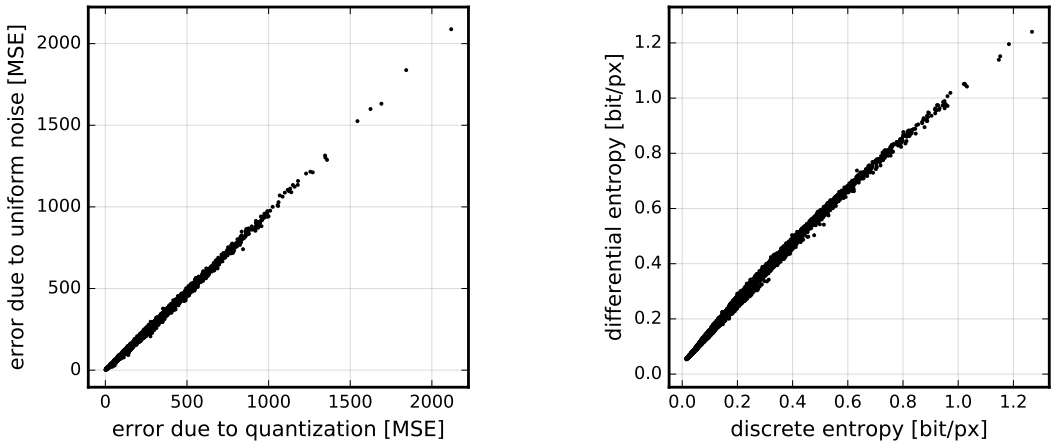



Figure 4: Scatter plots comparing discrete vs. continuously-relaxed values of the two terms of the objective function, evaluated for the optimized GDN model. Points correspond to different values of $\lambda$ between 32 and 2048 (inclusive), for images drawn from a random subset of 2169 images (one third) from the training set. Left: distortion term, evaluated for $g_s(\hat{\boldsymbol{y}})$ vs. $g_s(\tilde{\boldsymbol{y}})$. Right: rate term, $H[P_{q_i}]$ vs. $h[p_{\tilde{y}_i}]$ (summed over $i$).

For the grayscale analysis transform, we used 128 filters (size $9 \times 9$) in the first stage, each subsampled by a factor of 4 vertically and horizontally. The remaining two stages retain the number of channels, but use filters operating across all input channels ($5 \times 5 \times 128$), with outputs subsampled by a factor of 2 in each dimension. The net output thus has half the dimensionality of the input. The synthesis transform is structured analogously. For RGB images, we trained a separate set of models, with the first stage augmented to operate across three (color) input channels. For the two largest values of $\lambda$, and for RGB models, we increased the network capacity by increasing the number of channels in each stage to 256 and 192, respectively. Further details about the parameterization of the transforms and their training can be found in the appendix.

We first verified that the continuously-relaxed loss function given in section 3 provides a good approximation to the actual rate–distortion values obtained with quantization (figure 4). The relaxed distortion term appears to be mostly unbiased, and exhibits a relatively small variance. The relaxed (differential) entropy provides a somewhat positively biased estimate of the discrete entropy for the coarser quantization regime, but the bias disappears for finer quantization, as expected. Note that since the values of $\lambda$ do not have any intrinsic meaning, but serve only to map out the convex hull of optimal points in the rate–distortion plane (figure 2, left panel), a constant bias in either of the terms would simply alter the effective value of $\lambda$, with no effect on the compression performance.

We compare the rate–distortion performance of our method to two standard methods: JPEG and JPEG 2000. For our method, all images were compressed using uniform quantization (the continuous relaxation using additive noise was used only for training purposes). To make the comparisons more fair, we implemented a simple entropy code based on the context-based adaptive binary arithmetic coding framework (CABAC; Marpe, Schwarz, and Wiegand, 2003). All sideband information needed by the decoder (size of images, value of $\lambda$, etc.) was included in the bit stream (see appendix). Note that although the computational costs for training our models are quite high, encoding or decoding an image with the trained models is efficient, requiring only execution of the optimized analysis transformation and quantizer, or the synthesis transformation, respectively. Evaluations were performed on the Kodak image dataset[1], an uncompressed set of images commonly used to evaluate image compression methods. We also examined a set of relatively standard (if outdated) images used by the compression community (known by the names "Lena", "Barbara", "Peppers", and "Mandrill") as well as a set of our own digital photographs. None of these test images was included in the training set. All test images, compressed at a variety of bit rates using all three methods, along with their associated rate–distortion curves, are available online at `http://www.cns.nyu.edu/~lcv/iclr2017`.

[1]Downloaded from `http://www.cipr.rpi.edu/resource/stills/kodak.html`

**JPEG**, 4283 bytes (0.121 bit/px), PSNR: luma 24.85 dB/chroma 29.23 dB, MS-SSIM: 0.8079

**Proposed method**, 3986 bytes (0.113 bit/px), PSNR: luma 27.01 dB/chroma 34.16 dB, MS-SSIM: 0.9039

**JPEG 2000**, 4004 bytes (0.113 bit/px), PSNR: luma 26.61 dB/chroma 33.88 dB, MS-SSIM: 0.8860

Figure 5: A heavily compressed example image, $752 \times 376$ pixels. Note the appearance of artifacts, especially near edges, in both the JPEG and JPEG2000 images.

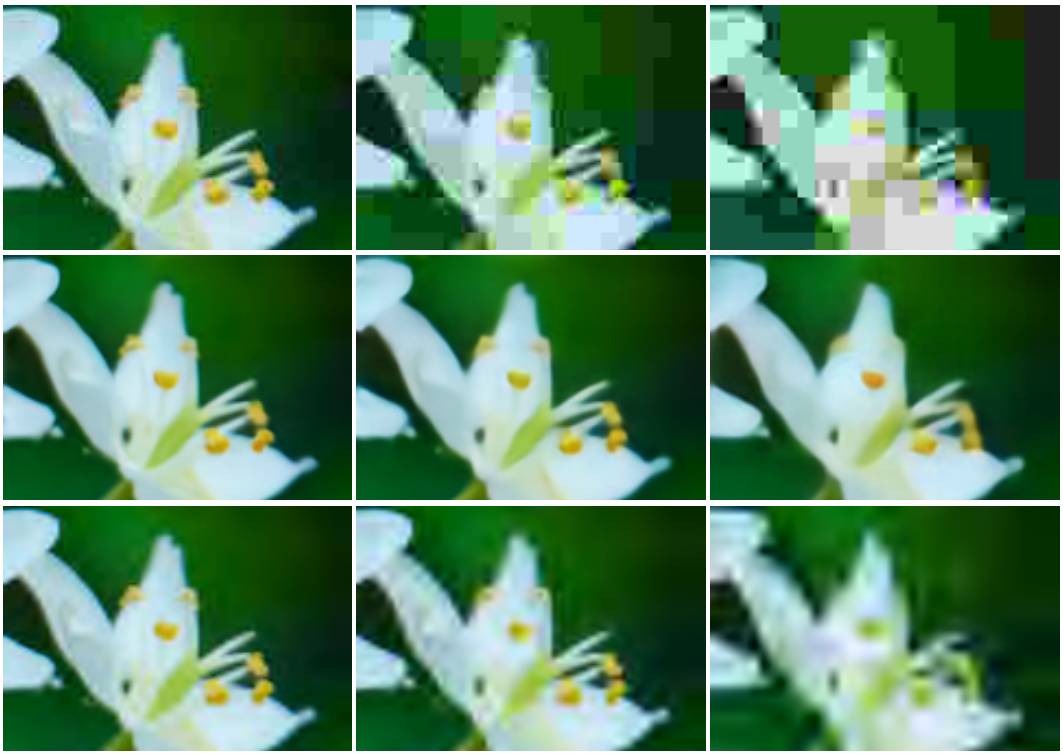


Figure 6: Cropped portion of an image compressed at three different bit rates. Middle row: the proposed method, at three different settings of $\lambda$. Top row: JPEG, with three different quality settings. Bottom row: JPEG 2000, with three different rate settings. Bit rates within each column are matched.

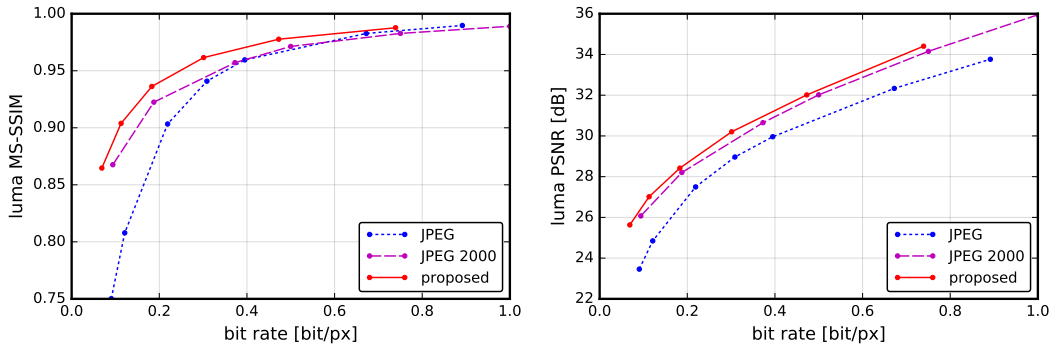



Figure 7: Rate–distortion curves for the luma component of image shown in figure 5. Left: perceptual quality, measured with multi-scale structural similarity (MS-SSIM; Wang, Simoncelli, and Bovik (2003)). Right: peak signal-to-noise ratio ($10 \log_{10}(255^2/\mathrm{MSE})$).

Although we used MSE as a distortion metric for training, the appearance of compressed images is both qualitatively different and substantially improved, compared to JPEG and JPEG 2000. As an example, figure 5 shows an image compressed using our method optimized for a low value of $\lambda$ (and thus, a low bit rate), compared to JPEG/JPEG 2000 images compressed at equal or greater bit rates. The image compressed with our method has less detail than the original (not shown, but available online), with fine texture and other patterns often eliminated altogether, but this is accomplished in a way that preserves the smoothness of contours and sharpness of many of the edges, giving them a natural appearance. By comparison, the JPEG and JPEG 2000 images exhibit artifacts that are common to all linear transform coding methods: since local features (edges, contours, texture elements,

etc.) are represented using particular combinations of localized linear basis functions, independent scalar quantization of the transform coefficients causes imbalances in these combinations, and leads to visually disturbing blocking, aliasing, and ringing artifacts that reflect the underlying basis functions.

Remarkably, we find that the perceptual advantages of our method hold for *all* images tested, and at all bit rates. The progression from high to low bit rates is shown for an example image in figure 6 (additional examples provided in appendix and online). As bit rate is reduced, JPEG and JPEG 2000 degrade their approximation of the original image by coarsening the precision of the coefficients of linear basis functions, thus exposing the visual appearance of those basis functions. On the other hand, our method appears to progressively simplify contours and other image features, effectively concealing the underlying quantization of the representation. Consistent with the appearance of these example images, we find that distortion measured with a perceptual metric (MS-SSIM; Wang, Simoncelli, and Bovik, 2003), indicates substantial improvements across all tested images and bit rates (figure 7; additional examples provided in the appendix and online). Finally, when quantified with PSNR, we find that our method exhibits better rate–distortion performance than both JPEG and JPEG 2000 for most (but not all) test images, especially at the lower bit rates.

## 5 Discussion

We have presented a complete image compression method based on *nonlinear transform coding*, and a framework to optimize it end-to-end for rate–distortion performance. Our compression method offers improvements in rate–distortion performance over JPEG and JPEG 2000 for most images and bit rates. More remarkably, although the method was optimized using mean squared error as a distortion metric, the compressed images are much more natural in appearance than those compressed with JPEG or JPEG 2000, both of which suffer from the severe artifacts commonly seen in linear transform coding methods. Consistent with this, perceptual quality (as estimated with the MS-SSIM index) exhibits substantial improvement across all test images and bit rates. We believe this visual improvement arises because the cascade of biologically-inspired nonlinear transformations in the model have been optimized to capture the features and attributes of images that are represented in the statistics of the data, parallel to the processes of evolution and development that are believed to have shaped visual representations within the human brain (Simoncelli and Olshausen, 2001). Nevertheless, additional visual improvements might be possible if the method were optimized using a perceptual metric in place of MSE (Ballé, Laparra, and Simoncelli, 2016).

For comparison to linear transform coding methods, we can interpret our analysis transform as a single-stage linear transform followed by a complex vector quantizer. As in many other optimized representations – e.g., sparse coding (Lewicki and Olshausen, 1998) – as well as many engineered representations – e.g., the steerable pyramid (Simoncelli, Freeman, et al., 1992), curvelets (Candès and Donoho, 2002), and dual-tree complex wavelets (Selesnick, Baraniuk, and Kingsbury, 2005) – the filters in this first stage are localized and oriented and the representation is *overcomplete*. Whereas most transform coding methods use complete (often orthogonal) linear transforms with spatially separable filters, the overcompleteness and orientation tuning of our initial transform may explain the ability of the model to better represent features and contours with continuously varying orientation, position and scale (Simoncelli, Freeman, et al., 1992).

Our work is related to two previous publications that optimize image representations with the goal of image compression. Gregor, Besse, et al. (2016) introduce an interesting hierarchical representation of images, in which degradations are more natural looking than those of linear representations. However, rather than optimizing directly for rate–distortion performance, their modeling is generative. Due to the differences between these approaches (as outlined in section 3.1), their procedure of obtaining coding representations from the generative model (scalar quantization, and elimination of hierarchical levels of the representation) is less systematic than our approach and unlikely to be optimal. Further, no entropy code is provided, and the authors therefore resort to comparing entropy estimates to bit rates of established compression methods, which can be unreliable. The model developed by Toderici et al. (2016) is optimized to provide various rate–distortion trade-offs and directly output a binary representation, making it more easily comparable to other image compression methods. Moreover, their formulation has the advantage over ours that a single representation is sought for all rate points. However, it is not clear whether their formulation necessarily leads to rate–distortion optimality (and their empirical results suggest that this is not the case).

We are currently testing models that use simpler rectified-linear or sigmoidal nonlinearities, to determine how much of the performance and visual quality of our results is due to use of biologically-inspired joint nonlinearities. Preliminary results indicate that qualitatively similar results are achievable with other activation functions we tested, but that rectified linear units generally require a substantially larger number of model parameters/stages to achieve the same rate–distortion performance as the GDN/IGDN nonlinearities. This suggests that GDN/IGDN transforms are more efficient for compression, producing better models with fewer stages of processing (as we previously found for density estimation; Ballé, Laparra, and Simoncelli, 2015), which might be an advantage for deployment of our method, say, in embedded systems. However, such conclusions are based on a somewhat limited set of experiments and should at this point be considered provisional. More generally, GDN represents a multivariate generalization of a particular type of sigmoidal function. As such, the observed efficiency advantage relative to pointwise nonlinearities is expected, and a variant of a universal function approximation theorem (e.g., Leshno et al., 1993) should hold.

The rate–distortion objective can be seen as a particular instantiation of the general unsupervised learning or density estimation problems. Since the transformation to a discrete representation may be viewed as a form of classification, it is worth considering whether our framework offers any insights that might be transferred to more specific supervised learning problems, such as object recognition. For example, the additive noise used in the objective function as a relaxation of quantization might also serve the purpose of making supervised classification networks more robust to small perturbations, and thus allow them to avoid catastrophic "adversarial" failures that have been demonstrated in previous work (Szegedy et al., 2013). In any case, our results provide a strong example of the power of end-to-end optimization in achieving a new solution to a classical problem.

### ACKNOWLEDGMENTS

We thank Olivier Hénaff and Matthias Bethge for fruitful discussions.

## REFERENCES

Ballé, Johannes, Valero Laparra, and Eero P. Simoncelli (2015). "Density Modeling of Images Using a Generalized Normalization Transformation". In: *arXiv e-prints*. Presented at the 4th Int. Conf. for Learning Representations, 2016. arXiv: 1511.06281.

– (2016). "End-to-end optimization of nonlinear transform codes for perceptual quality". In: *arXiv e-prints*. Presented at 2016 Picture Coding Symposium. arXiv: 1607.05006.

Candès, Emmanuel J. and David L. Donoho (2002). "New Tight Frames of Curvelets and Optimal Representations of Objects with $C^2$ Singularities". In: *Comm. Pure Appl. Math.* 57, pp. 219–266.

Carandini, Matteo and David J. Heeger (2012). "Normalization as a canonical neural computation". In: *Nature Reviews Neuroscience* 13. DOI: 10.1038/nrn3136.

Deng, J. et al. (2009). "ImageNet: A Large-Scale Hierarchical Image Database". In: *IEEE Conf. on Computer Vision and Pattern Recognition*. DOI: 10.1109/CVPR.2009.5206848.

Gersho, Allen and Robert M. Gray (1992). *Vector Quantization and Signal Compression*. Kluwer. ISBN: 978-0-7923-9181-4.

Gray, Robert M. and David L. Neuhoff (1998). "Quantization". In: *IEEE Transactions on Information Theory* 44.6. DOI: 10.1109/18.720541.

Gregor, Karol, Frederic Besse, et al. (2016). "Towards Conceptual Compression". In: *arXiv e-prints*. arXiv: 1604.08772.

Gregor, Karol and Yann LeCun (2010). "Learning Fast Approximations of Sparse Coding". In: *Proceedings of the 27th International Conference on Machine Learning*.

Heeger, David J. (1992). "Normalization of cell responses in cat striate cortex". In: *Visual Neuroscience* 9.2. DOI: 10.1017/S0952523800009640.

Ioffe, Sergey and Christian Szegedy (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariance Shift". In: *arXiv e-prints*. arXiv: 1502.03167.

Jarrett, Kevin et al. (2009). "What is the Best Multi-Stage Architecture for Object Recognition?" In: *2009 IEEE 12th International Conference on Computer Vision*. DOI: 10.1109/ICCV.2009.5459469.

Kingma, Diederik P. and Jimmy Lei Ba (2014). "Adam: A Method for Stochastic Optimization". In: *arXiv e-prints*. Presented at the 3rd Int. Conf. for Learning Representations, 2015. arXiv: 1412.6980.

Kingma, Diederik P. and Max Welling (2014). "Auto-Encoding Variational Bayes". In: *arXiv e-prints*. arXiv: 1312.6114.

Laparra, Valero et al. (2016). "Perceptual image quality assessment using a normalized Laplacian pyramid". In: *Proceedings of SPIE, Human Vision and Electronic Imaging XXI*.

Leshno, Moshe et al. (1993). "Multilayer Feedforward Networks With a Nonpolynomial Activation Function Can Approximate Any Function". In: *Neural Networks* 6.6. DOI: 10.1016/S0893-6080(05)80131-5.

Lewicki, Michael S. and Bruno Olshausen (1998). "Inferring sparse, overcomplete image codes using an efficient coding framework". In: *Advances in Neural Information Processing Systems 10*, pp. 815–821.

Lyu, Siwei (2010). "Divisive Normalization: Justification and Effectiveness as Efficient Coding Transform". In: *Advances in Neural Information Processing Systems 23*, pp. 1522–1530.

Malo, Jesús et al. (2006). "Non-linear image representation for efficient perceptual coding". In: *IEEE Transactions on Image Processing* 15.1. DOI: 10.1109/TIP.2005.860325.

Mante, Valerio, Vincent Bonin, and Matteo Carandini (2008). "Functional Mechanisms Shaping Lateral Geniculate Responses to Artificial and Natural Stimuli". In: *Neuron* 58.4. DOI: 10.1016/j.neuron.2008.03.011.

Marpe, Detlev, Heiko Schwarz, and Thomas Wiegand (2003). "Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard". In: *IEEE Transactions on Circuits and Systems for Video Technology* 13.7. DOI: 10.1109/TCSVT.2003.815173.

Netravali, A. N. and J. O. Limb (1980). "Picture Coding: A Review". In: *Proceedings of the IEEE* 68.3. DOI: 10.1109/PROC.1980.11647.

Oord, Aäron van den, Nal Kalchbrenner, and Koray Kavukcuoglu (2016). "Pixel Recurrent Neural Networks". In: *arXiv e-prints*. arXiv: 1601.06759.

Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (2014). "Stochastic Backpropagation and Approximate Inference in Deep Generative Models". In: *arXiv e-prints*. arXiv: 1401.4082.

Rippel, Oren, Jasper Snoek, and Ryan P. Adams (2015). "Spectral Representations for Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 28*, pp. 2449–2457.

Rissanen, Jorma and Glen G. Langdon Jr. (1981). "Universal modeling and coding". In: *IEEE Transactions on Information Theory* 27.1. DOI: 10.1109/TIT.1981.1056282.

Schwartz, Odelia and Eero P. Simoncelli (2001). "Natural signal statistics and sensory gain control". In: *Nature Neuroscience* 4.8. DOI: 10.1038/90526.

Selesnick, Ivan W., Richard G. Baraniuk, and Nick C. Kingsbury (2005). "The Dual-Tree Complex Wavelet Transform". In: *IEEE Signal Processing Magazine* 22.6. DOI: 10.1109/MSP.2005.1550194.

Shannon, Claude E. (1948). "A Mathematical Theory of Communication". In: *The Bell System Technical Journal* 27.3. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

Simoncelli, Eero P., William T. Freeman, et al. (1992). "Shiftable Multiscale Transforms". In: *IEEE Transactions on Information Theory* 38.2. DOI: 10.1109/18.119725.

Simoncelli, Eero P. and David J. Heeger (1998). "A model of neuronal responses in visual area MT". In: *Vision Research* 38.5. DOI: 10.1016/S0042-6989(97)00183-1.

Simoncelli, Eero P. and Bruno Olshausen (2001). "Natural image statistics and neural representation". In: *Annual Review of Neuroscience* 24. DOI: 10.1146/annurev.neuro.24.1.1193.

Sinz, Fabian and Matthias Bethge (2013). "What Is the Limit of Redundancy Reduction with Divisive Normalization?" In: *Neural Computation* 25.11. DOI: 10.1162/NECO_a_00505.

Szegedy, Christian et al. (2013). "Intriguing properties of neural networks". In: *arXiv e-prints*. arXiv: 1312.6199.

Theis, Lucas, Aäron van den Oord, and Matthias Bethge (2015). "A note on the evaluation of generative models". In: *arXiv e-prints*. Presented at the 4th Int. Conf. for Learning Representations. arXiv: 1511.01844.

Toderici, George et al. (2016). "Full Resolution Image Compression with Recurrent Neural Networks". In: *arXiv e-prints*. arXiv: 1608.05148.

Wang, Zhou, Eero P. Simoncelli, and Alan Conrad Bovik (2003). "Multi-Scale Structural Similarity for Image Quality Assessment". In: *Conf. Rec. of the 37th Asilomar Conf. on Signals, Systems and Computers, 2004*. DOI: 10.1109/ACSSC.2003.1292216.

Wintz, Paul A. (1972). "Transform Picture Coding". In: *Proceedings of the IEEE* 60.7. DOI: 10.1109/PROC.1972.8780.

# 6 APPENDIX

## 6.1 NETWORK ARCHITECTURE AND OPTIMIZATION

As described in the main text, our analysis transform consists of three stages of convolution, downsampling, and GDN. The number and size of filters, downsampling factors, and connectivity from layer to layer are provided in figure 8 for the grayscale transforms. The transforms for RGB images and for high bit rates differ slightly in that they have an increased number of channels in each stage. These choices are somewhat ad-hoc, and a more thorough exploration of alternative architectures could potentially lead to significant performance improvements.

We have previously shown that GDN is highly efficient in Gaussianizing the local joint statistics of natural images (Ballé, Laparra, and Simoncelli, 2015). Even though Gaussianization is a quite different optimization problem than the rate–distortion objective with the set of constraints defined above, it is similar in that a marginally independent latent model is assumed in both cases. When optimizing for Gaussianization, the exponents in the parametric form of GDN control the tail behavior of the Gaussianized densities. Since tail behavior is less important here, we chose to simplify the functional form, fixing the exponents as well as forcing the weight matrix to be symmetric (i.e., $\gamma_{k,ij} = \gamma_{k,ji}$).

The synthesis transform is meant to function as an approximate inverse transformation, so we construct it by applying a principle known from the LISTA algorithm (Gregor and LeCun, 2010) to the fixed point iteration previously used to invert the GDN transform (Ballé, Laparra, and Simoncelli, 2015). The approximate inverse consists of one iteration, but with a separate set of parameters from the forward transform, which are constrained in the same way, but trained separately. We refer to this nonlinear transform as "inverse GDN" (IGDN).

The full model (analysis and synthesis filters, GDN and IGDN parameters) were optimized, for each $\lambda$, over a subset of the ImageNet database (Deng et al., 2009) consisting of 6507 images. We applied a number of preprocessing steps to the images in order to reduce artifacts and other unwanted contaminations: first, we eliminated images with excessive saturation. We added a small amount of uniform noise, corresponding to the quantization of pixel values, to the remaining images. Finally, we downsampled and cropped the images to a size of $256 \times 256$ pixels each, where the amount of downsampling and cropping was randomized, but depended on the size of the original image. In order to reduce high-frequency noise and compression artifacts, we only allowed resampling factors less than 0.75, discarding images that were too small to satisfy this constraint.

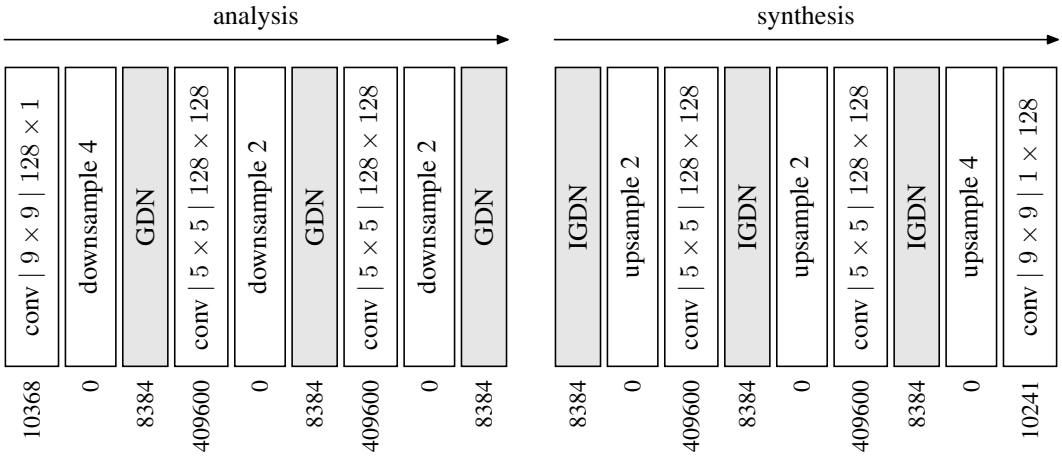



Figure 8: Parameterization of analysis ($g_a$) and synthesis ($g_s$) transforms for grayscale images. *conv*: affine convolution (1)/(6), with filter support ($x \times y$) and number of channels (output×input). *down-/upsample*: regular down-/upsampling (2)/(5) by given factor (implemented jointly with the adjacent convolution). *GDN/IGDN*: generalized divisive normalization across channels (3), and its approximate inverse (4); see text. Number of parameters for each layer given at the bottom.

To ensure efficient and stable optimization, we used the following techniques:

- We used the Adam optimization algorithm (Kingma and Ba, 2014) to obtain values for the parameters $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$, starting with $\alpha = 10^{-4}$, and subsequently lowering it by a factor of 10 whenever the improvement of both rate and distortion stagnated, until $\alpha = 10^{-7}$.
- Linear filters were parameterized using their discrete cosine transform (DCT) coefficients. We found this to be slightly more effective in speeding up the convergence than discrete Fourier transform (DFT) parameterization (Rippel, Snoek, and Adams, 2015).
- We parameterized the GDN parameters in terms of the elementwise relationship

  $$\beta_{k,i} = (\beta'_{k,i})^2 - 2^{-10}.$$

  The squaring ensures that gradients are smaller around parameter values close to 0, a regime in which the optimization can otherwise become unstable. To obtain an unambiguous mapping, we projected each $\beta'_{k,i}$ onto the interval $[2^{-5}, \infty)$ after each gradient step. We applied the same treatment to $\gamma_{k,ij}$, and additionally averaged $\gamma'_{k,ij}$ with its transpose after each step in order to make it symmetric as explained above. The IGDN parameters were treated in the same way.
- To remove the scaling ambiguity between the each linear transform and its following nonlinearity (or preceding nonlinearity, in the case of the synthesis transform), we renormalized the linear filters after each gradient step, dividing each filter by the square root of the sum of its squared coefficients. For the analysis transform, the sum runs over space and all input channels, and for the synthesis transform, over space and all output channels.

We represented each of the marginals $p_{\tilde{y}_i}$ as a piecewise linear function (i.e., a linear spline), using 10 sampling points per unit interval. The parameter vector $\boldsymbol{\psi}^{(i)}$ consists of the value of $p_{\tilde{y}_i}$ at these sampling points. We did not use Adam to update $\boldsymbol{\psi}^{(i)}$; rather, we used ordinary stochastic gradient descent to minimize the negative expected likelihood:

$$L_\psi(\boldsymbol{\psi}^{(0)}, \boldsymbol{\psi}^{(1)}, \dots) = -\,\mathbb{E}_{\tilde{\boldsymbol{y}}} \sum_i p_{\tilde{y}_i}(\tilde{y}_i; \boldsymbol{\psi}^{(i)}). \tag{17}$$

and renormalized the marginal densities after each step. After every $10^6$ gradient steps, we used a heuristic to adapt the range of the spline approximation to cover the range of values of $\tilde{y}_i$ obtained on the training set.

### 6.2 Entropy code

We implemented an entropy code based on the context-adaptive binary arithmetic coding (CABAC) framework defined by Marpe, Schwarz, and Wiegand (2003). Arithmetic entropy codes are designed to compress discrete-valued data to bit rates closely approaching the entropy of the representation, assuming that the probability model used to design the code describes the data well. The following information was encoded into the bitstream:

- the size of the image (two 16-bit integers, bypassing arithmetic coding),
- whether the image is grayscale or RGB (one bit, bypassing arithmetic coding),
- the value of $\lambda$ (one 16-bit integer, bypassing arithmetic coding), which provides an index for the parameters of the analysis and synthesis transforms as well as the initial probability models for the entropy codes (these are fixed after optimization, and assumed to be available to encoder and decoder).
- the value of each element of $\boldsymbol{q}$, iterating over channels, and over space in raster-scan order, using the arithmetic coding engine.

Since CABAC operates on binary values, the quantized values in $\boldsymbol{q}$ need to be converted to binary decisions. We follow a simple scheme inspired by the encoding of H.264/AVC transform coefficients as detailed by Marpe, Schwarz, and Wiegand (2003). For each $q_i$, we start by testing if the encoded value is equal to the mode of the distribution. If this is the case, the encoding of $q_i$ is completed. If not, another binary decision determines whether it is smaller or larger than the mode. Following

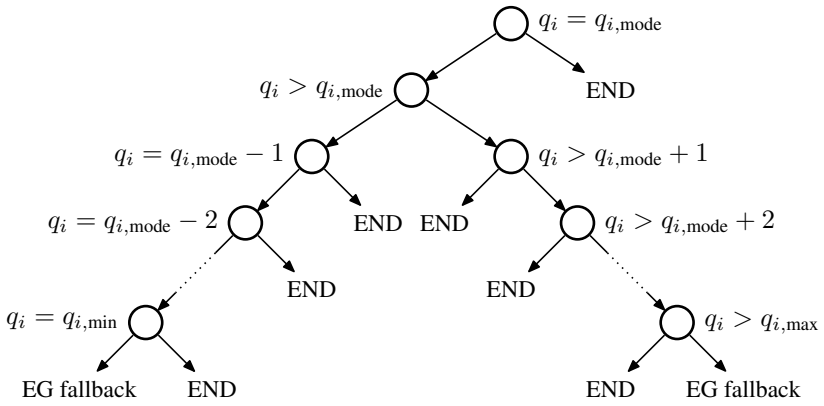



Figure 9: Binarization of a quantized value for binary arithmetic coding. Each circle represents a binary decision encoded with its own CABAC context. Arrows pointing left represent "false", arrows pointing right "true". On reaching END, the encoding of the quantized value is completed. On reaching EG fallback, the magnitude of $q_i$ which falls outside of the range $[q_{i,\text{min}}, q_{i,\text{max}}]$ is encoded using an exponential Golomb code, bypassing the arithmetic coding engine.

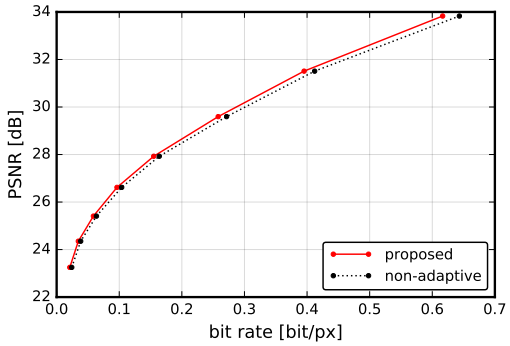



Figure 10: Rate–distortion comparison of adaptive vs. non-adaptive entropy coding, averaged (for each value of $\lambda$) over the 24 images in the Kodak test set. The non-adaptive entropy code is simulated by computing the entropy of $\boldsymbol{q}$ assuming the probability model determined during optimization (which is also used to initialize the adaptive code).

that, each possible integer value is tested in turn, which yields a bifurcated chain of decisions as illustrated in figure 9. This process is carried out until either one of the binary decisions determines $q_i$, or some minimum ($q_{i,\text{min}}$) or maximum ($q_{i,\text{max}}$) value is reached. In case $q_i$ is outside of that range, the difference between it and the range bound is encoded using an exponential Golomb code, bypassing the arithmetic coding engine.

Adaptive codes, such as CABAC, can potentially further improve bit rates, and to some extent correct model error, by adapting the probability model on-line to the statistics of the data. In our code, this is achieved by sharing the marginal probability model $P_{q_i}$ of each element in $\boldsymbol{q}$ across space within each channel. We derived the initial probability models by subsampling the continuous densities $p_{\tilde{y}_i}$ determined during optimization, as in (10). However, note that due to the simple raster-scan ordering, the coding scheme presented above only crudely exploits spatial adaptation of the probability model compared to existing coding methods such as JPEG 2000 and H.264/AVC. Thus, the performance gains compared to a well-designed non-adaptive entropy code are relatively small (figure 10), and likely smaller than those achieved by the entropy code in JPEG 2000, to which we compare.

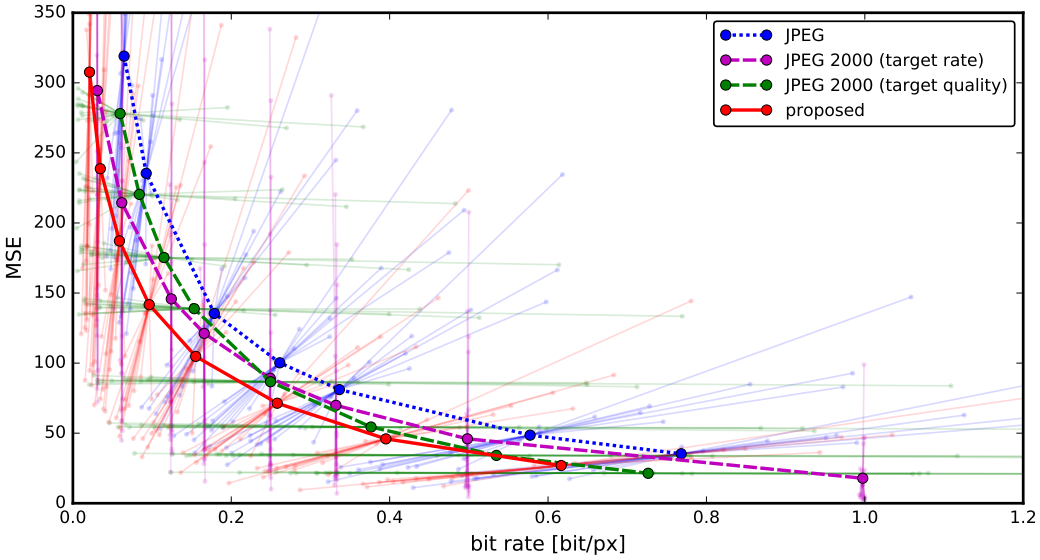



Figure 11: Summary rate–distortion curves, computed by averaging results over the 24 images in the Kodak test set. Each point is connected by translucent lines to the set of 24 points corresponding to the individual image R-D values from which it was derived. JPEG results are averaged over images compressed with identical quality settings. Results of the proposed method are averaged over images compressed with identical $\lambda$ values (and thus, computed with exactly the same forward and inverse transforms). The two JPEG 2000 curves are computed with the same implementation, by averaging over images compressed with the same target rate or the same target quality. Note that these two methods of selecting points to be averaged lead to significantly different average results.

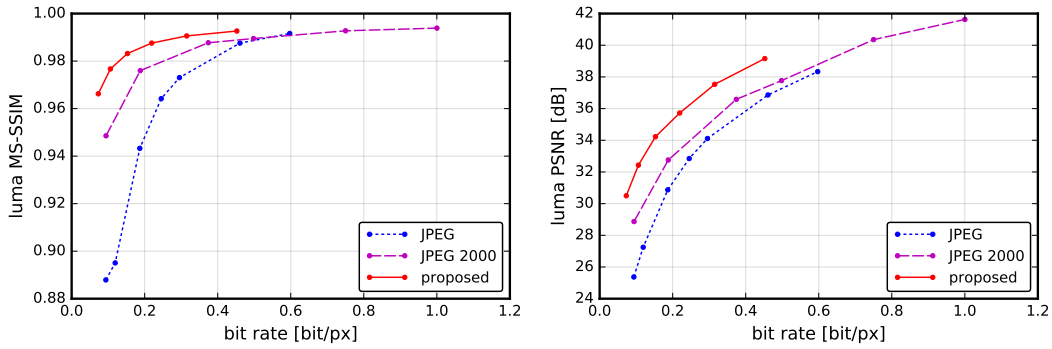
### 6.3 Evaluation details and additional example images

Although it is desirable to summarize and compare the rate–distortion behavior of JPEG, JPEG 2000, and our method across an image set, it is difficult to do this in a way that is fair and interpretable. First, rate–distortion behavior varies substantially across bit rates for different images. For example, for the image in figure 12, our method achieves the same MSE with roughly 50% of the bits needed by JPEG 2000 for low rates, and about 30% for high rates. For the image in figure 17, the gains are more modest, although still significant through the range. But for the image in figure 15, our method only slightly outperforms JPEG 2000 at low rates, and under-performs it at high rates. Note that these behaviors are again different for MS-SSIM, which shows a significant improvement for all images and bit rates (consistent with their visual appearance).

Second, there is no obvious or agreed-upon method for combining rate–distortion curves across images. More specifically, one must decide which points in the curves to combine. For our method, it is natural to average the MSE and entropy values across images compressed using the same choice of $\lambda$, since these are all coded and decoded using exactly the same representation and quantization scheme. For JPEG, it seems natural to average over images coded at the same "quality" setting, which appear to be coded using the same quantization choices. The OpenJPEG implementation of JPEG 2000 we use allows selection of points on the rate–distortion curve *either* through specification of a target bit rate, *or* a target quality. This choice has no effect on rate–distortion plots for individual images (verified, but not shown), but has a substantial effect when averaging over images, since the two choices lead one to average over a different set of R-D points. This is illustrated in figure 11. Even if points were selected in exactly the same fashion for each of the methods (say, matched to a given set of target rates), summary plots can still over- or underemphasize high rate vs. low rate performance.

We conclude that summaries of rate–distortion are of limited use. Instead, we encourage the reader to browse our extensive collection of test images, with individual rate–distortion plots for each image, available at `http://www.cns.nyu.edu/~lcv/iclr2017` in both grayscale and RGB.

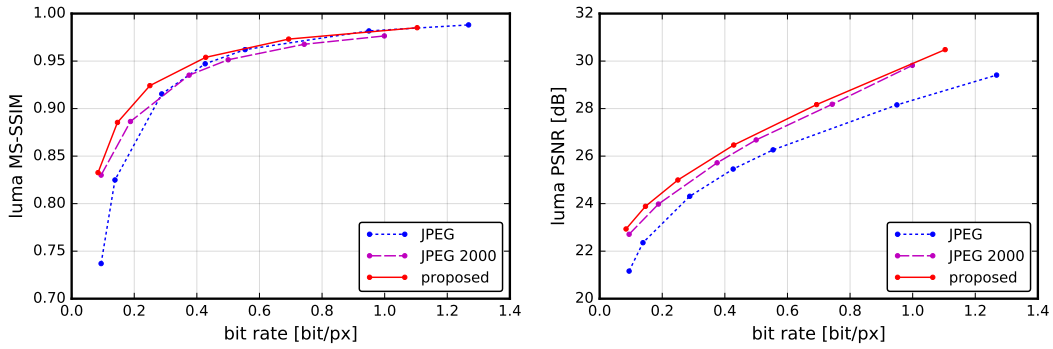
In the following pages, we show additional example images, compressed at relatively low bit rates, in order to visualize the qualitative nature of compression artifacts. On each page, the JPEG 2000 image is selected to have the lowest possible bit rate that is equal or greater than the bit rate of the proposed method. In all experiments, we compare to JPEG with 4:2:0 chroma subsampling, and the OpenJPEG implementation of JPEG 2000 with the default "multiple component transform". For evaluating PSNR, we use the JPEG-defined conversion matrix to convert between RGB and Y'CbCr. For evaluating MS-SSIM (Wang, Simoncelli, and Bovik, 2003), we used only the resulting luma component. Original images are not shown, but are available online, along with compressed images at a variety of other bit rates, at `http://www.cns.nyu.edu/~lcv/iclr2017`.




**Proposed method**, 3749 bytes (0.106 bit/px), PSNR: luma 32.43 dB/chroma 34.00 dB, MS-SSIM: 0.9767

**JPEG 2000**, 3769 bytes (0.107 bit/px), PSNR: luma 29.49 dB/chroma 32.99 dB, MS-SSIM: 0.9520

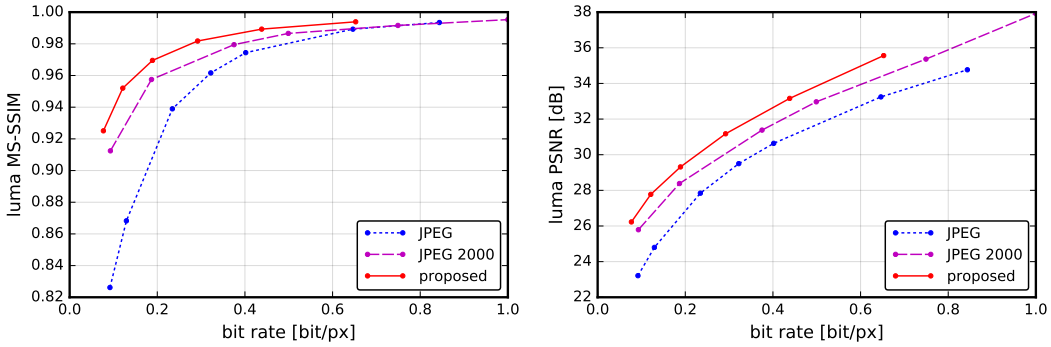
Figure 12: RGB example, from our personal collection, downsampled and cropped to $752 \times 376$ pixels.




**Proposed method**, 2978 bytes (0.084 bit/px), PSNR: luma 22.93 dB/chroma 31.45 dB, MS-SSIM: 0.8326

**JPEG 2000**, 2980 bytes (0.084 bit/px), PSNR: luma 22.53 dB/chroma 31.09 dB, MS-SSIM: 0.8225

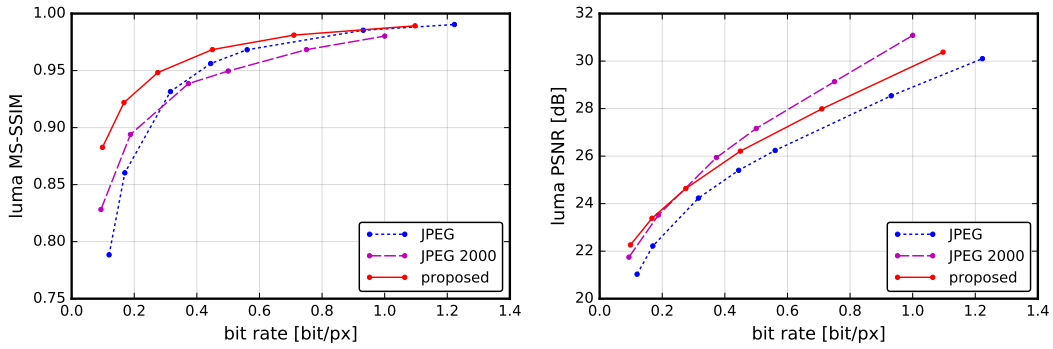
Figure 13: RGB example, from our personal collection, downsampled and cropped to $752 \times 376$ pixels.




**Proposed method**, 6680 bytes (0.189 bit/px), PSNR: luma 29.31 dB/chroma 36.17 dB, MS-SSIM: 0.9695

**JPEG 2000**, 6691 bytes (0.189 bit/px), PSNR: luma 28.45 dB/chroma 35.32 dB, MS-SSIM: 0.9586

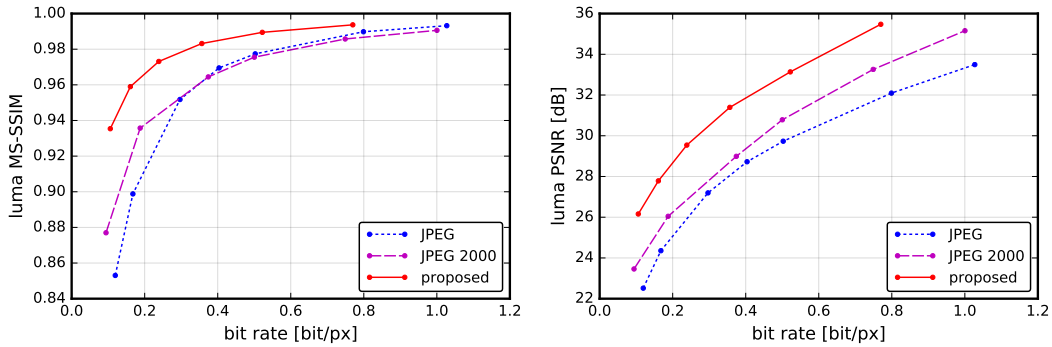
Figure 14: RGB example, from our personal collection, downsampled and cropped to $752 \times 376$ pixels.




**Proposed method**, 5908 bytes (0.167 bit/px), PSNR: luma 23.38 dB/chroma 31.86 dB, MS-SSIM: 0.9219

**JPEG 2000**, 5908 bytes (0.167 bit/px), PSNR: luma 23.24 dB/chroma 31.04 dB, MS-SSIM: 0.8803

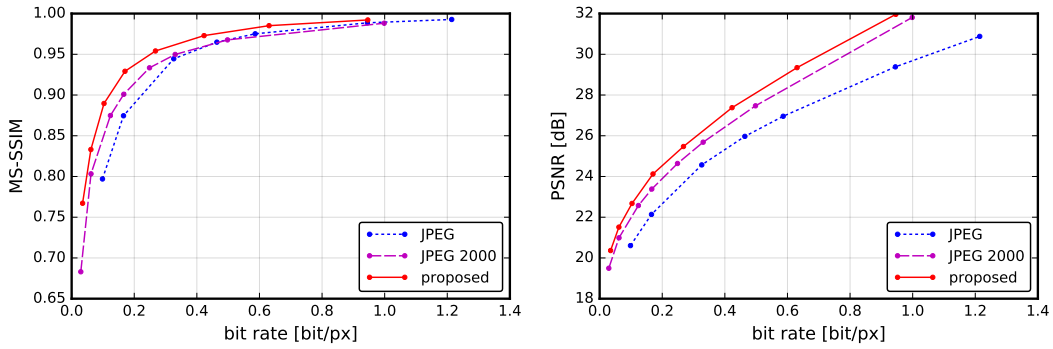
Figure 15: RGB example, from our personal collection, downsampled and cropped to $752 \times 376$ pixels.




**Proposed method**, 5683 bytes (0.161 bit/px), PSNR: luma 27.78 dB/chroma 32.60 dB, MS-SSIM: 0.9590

**JPEG 2000**, 5724 bytes (0.162 bit/px), PSNR: luma 25.36 dB/chroma 31.20 dB, MS-SSIM: 0.9202

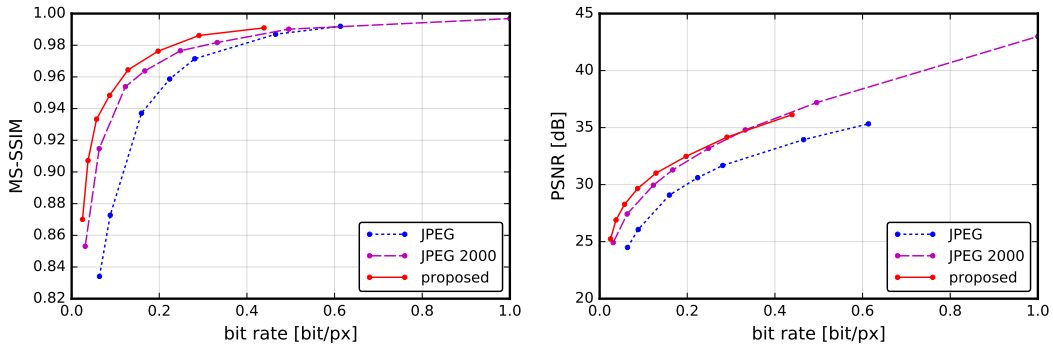
Figure 16: RGB example, from our personal collection, downsampled and cropped to $752 \times 376$ pixels.




**Proposed method**, 6021 bytes (0.170 bit/px), PSNR: 24.12 dB, MS-SSIM: 0.9292

**JPEG 2000**, 6037 bytes (0.171 bit/px), PSNR: 23.47 dB, MS-SSIM: 0.9036

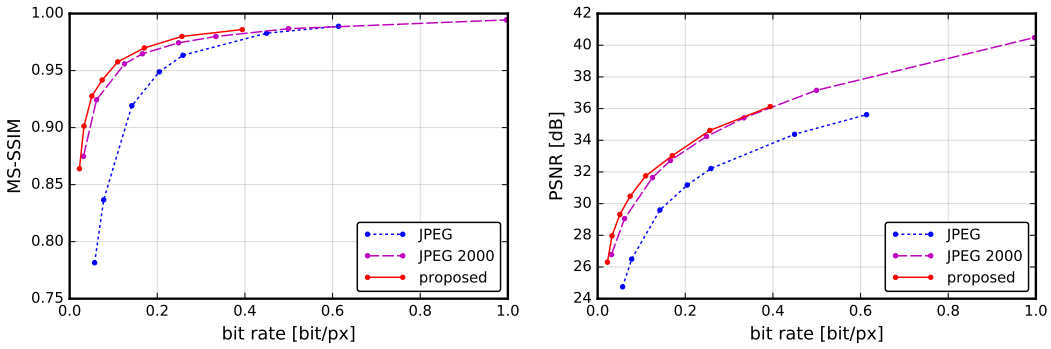
Figure 17: Grayscale example, from our personal collection, downsampled and cropped to $752 \times 376$ pixels.

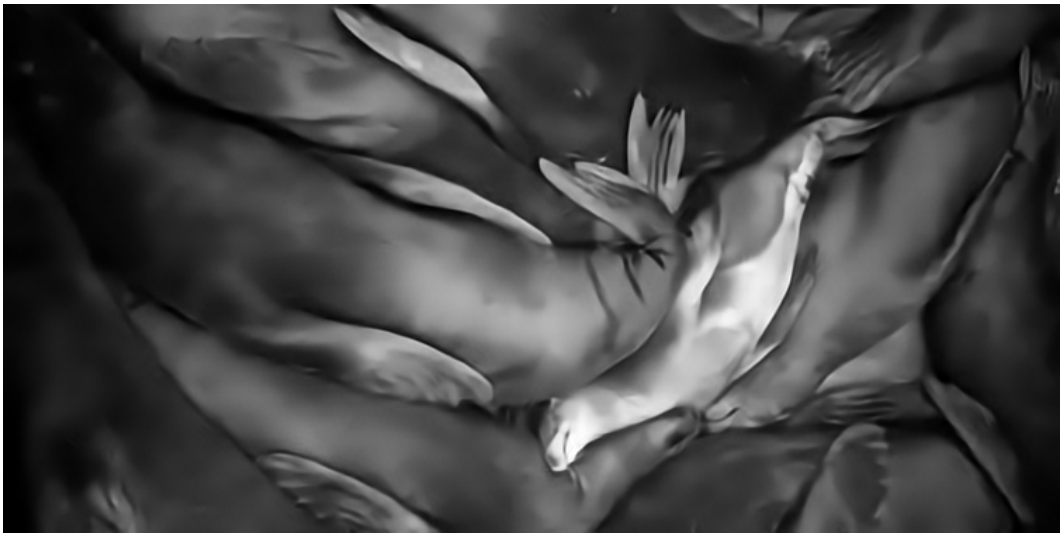



**Proposed method**, 4544 bytes (0.129 bit/px), PSNR: 31.01 dB, MS-SSIM: 0.9644

**JPEG 2000**, 4554 bytes (0.129 bit/px), PSNR: 30.17 dB, MS-SSIM: 0.9546

Figure 18: Grayscale example, from our personal collection, downsampled and cropped to $752 \times 376$ pixels.

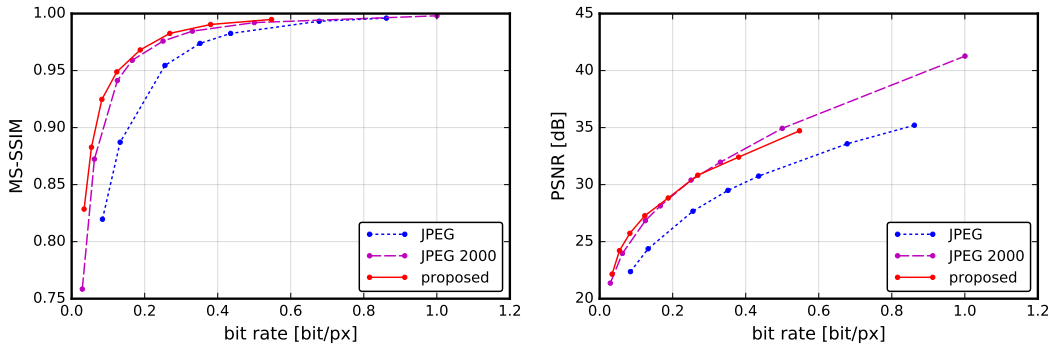


**Proposed method**, 3875 bytes (0.110 bit/px), PSNR: 31.75 dB, MS-SSIM: 0.9577

**JPEG 2000**, 3877 bytes (0.110 bit/px), PSNR: 31.24 dB, MS-SSIM: 0.9511

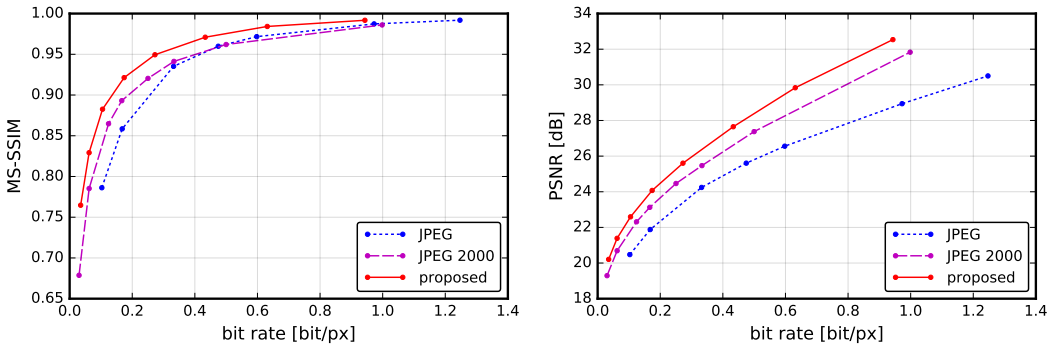
Figure 19: Grayscale example, from our personal collection, downsampled and cropped to $752 \times 376$ pixels.




**Proposed method**, 6633 bytes (0.188 bit/px), PSNR: 28.83 dB, MS-SSIM: 0.9681

**JPEG 2000**, 6691 bytes (0.189 bit/px), PSNR: 28.83 dB, MS-SSIM: 0.9651

Figure 20: Grayscale example, from our personal collection, downsampled and cropped to $752 \times 376$ pixels.




**Proposed method**, 10130 bytes (0.287 bit/px), PSNR: 25.27 dB, MS-SSIM: 0.9537

**JPEG 2000**, 10197 bytes (0.289 bit/px), PSNR: 24.41 dB, MS-SSIM: 0.9320

Figure 21: Grayscale example, from the Kodak test set, downsampled and cropped to $752 \times 376$ pixels.