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 system, consisting of a nonlinear encoding transformation, a uniform quantizer, and a nonlinear decoding transformation. Like many deep neural network architectures, the transforms consist of layers of convolutional linear filters and nonlinear activation functions, but we use a joint nonlinearity that implements 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 system for rate–distortion performance over a database of training images, introducing a continuous proxy for the discontinuous loss function arising from the quantizer. The relaxed optimization problem resembles that of variational autoencoders, except that it must operate at any point along the rate–distortion curve, whereas the optimization of generative models aims only to minimize entropy of the data under the model. Across an independent database of test images, we find that the optimized coder exhibits significantly better rate–distortion performance than the standard JPEG and JPEG 2000 compression systems, as well as a dramatic improvement in visual quality of compressed images.

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 specific channels, generally call for different rate–distortion trade-offs.

Joint optimization of rate and distortion is difficult. Without further constraints, the general problem of optimal vector quantization is intractable in high dimensions (Gersho and Gray, 1992). For this reason, most existing image compression systems 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). 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 systems – the transform, the (scalar) quantizer, and the entropy coder – are separately optimized (often through manual adjustment).

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

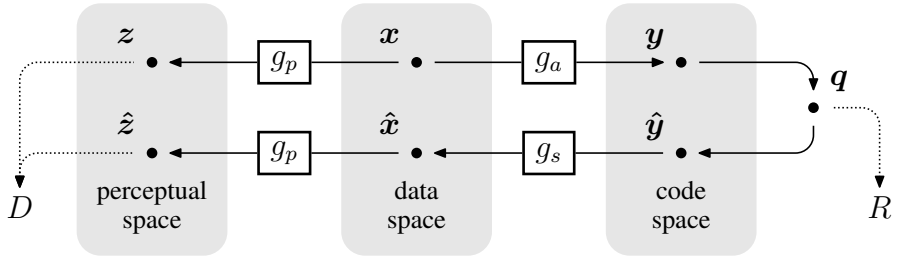


Figure 1: General framework for rate–distortion optimization of nonlinear transform coding (Ballé, Laparra, and Simoncelli, 2016). A vector of image intensities $\boldsymbol{x} \in \mathbb{R}^N$ is mapped to a latent *code space* via a parametric multivariate *analysis* transform, $\boldsymbol{y} = g_a(\boldsymbol{x}; \boldsymbol{\phi})$. 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 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 norm $D(\boldsymbol{z}, \hat{\boldsymbol{z}})$. The parameter vectors $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are optimized for a weighted sum of the rate and distortion measures over a set of images.

We have developed a framework for end-to-end optimization of an image compression system based on nonlinear transforms (figure 1). Previously, we demonstrated that a system consisting of linear–nonlinear block transformations, optimized for a measure of perceptual distortion, exhibited visually superior performance compared to a system optimized for mean squared error (Ballé, Laparra, and Simoncelli, 2016). Here, we use the simpler disortion measure of squared error, and focus on the use of more flexible transforms built from cascades of linear convolutions and nonlinearities, analogous to deep neural networks. Specifically, we make use a generalized divisive normalization (GDN) joint nonlinearity, that is inspired by models of neurons in biological visual systems, and has proven effective in modeling 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 each desired choice of the rate–distortion trade-off, the parameters of both forward and inverse 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 of the quantized representation. The relaxed rate–distortion optimization problem bears some resemblance to those used to learn generative image models, and in particular variational autoencoders, but differs in the constraints we impose to ensure that it approximates the discrete problem across different rate–distortion trade-offs. Finally, rather than reporting discrete entropy estimates, we implement an entropy coder and report performance values using the actual bit rate, thus demonstrate the feasibility of our solution as a complete lossy compression system.

We find that the system, when optimized over a set of photographic images, yields rate–distortion performance significantly better than JPEG or JPEG 2000. More importantly, although we only use mean squared error as a distortion metric in the training, the decoded images look remarkably good, exhibiting almost none of the visually disturbing blocking or "aliasing" artifacts that are a hallmark of essentially all existing compression systems.

2 Optimization of nonlinear transform coding

Our objective is to minimize a weighted sum of the rate and distortion, $R + \lambda D$, over analysis and synthesis transforms and the entropy code, where parameter λ governs the trade-off between rate and distortion (figure 2, left panel). The actual rates achieved by a properly designed entropy code are only slightly larger than the discrete entropy (Rissanen and Langdon, 1981), and thus we define

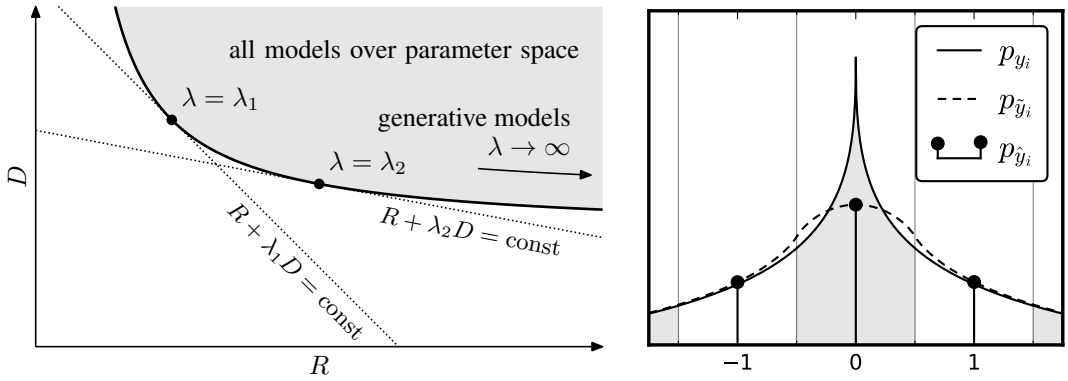


Figure 2: Left: The rate–distortion trade-off. The gray region represents the set of all rate–distortion values that can be achieved with a given model (over all possible parameter settings). Right: One-dimensional illustration of relationship between densities of y_i (elements of latent representation), $\hat{y}_i$ (quantized elements), and $\tilde{y}_i$ (elements perturbed by uniform noise). Each discrete probability value in $p_{\hat{y}_i}$ corresponds to the portion of the density p_{y_i} within the corresponding quantization bin (indicated by shading). Density $p_{\tilde{y}_i}$ provides a continuous interpolated function that is consistent with the discrete probability $p_{\hat{y}_i}$ at integer values.

an objective functional directly in terms of entropy:

$$L[g_a, g_s] = -\,\mathbb{E}\left[\log_2 P_{\boldsymbol{q}}\right] + \lambda\,\mathbb{E}\left\|\boldsymbol{z} - \hat{\boldsymbol{z}}\right\|^2, \tag{1}$$

where both expectations can be approximated by averages over a training set of images. Since optimal quantization is difficult in high dimensional spaces, we instead assume a fixed uniform quantizer, and aim to have the nonlinear transformations warp the code space in an appropriate way, effectively implementing a parametric form of vector quantization. Given a powerful enough set of transformations, we can assume without loss of generality that the quantization bin size is always 1 and the representing values are at the center of the bins. That is,

$$\hat{y}_i = q_i = \text{round}(y_i). \tag{2}$$

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:

$$P_{q_i}(n) = \int_{n-\frac{1}{2}}^{n+\frac{1}{2}} p_{y_i}(t)\,\mathrm{d}t, \text{ for all } n \in \mathbb{Z}, \tag{3}$$

Note that both terms in the functional (1) depend on the quantized values, and the derivatives of the quantization function (2) 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 source $\Delta\boldsymbol{y}$, which has the same width as the quantization bins (one). This solution has two desirable properties: First, the density function of $\tilde{\boldsymbol{y}}$ is a continuous relaxation of the probability mass function of $\boldsymbol{q}$ (figure 2, right panel):

$$p_{\tilde{\boldsymbol{y}}}(\boldsymbol{n}) = P_{\boldsymbol{q}}(\boldsymbol{n}) \text{ for all } \boldsymbol{n} \in \mathbb{Z}^M, \tag{4}$$

which implies that the differential entropy of $\tilde{\boldsymbol{y}} = \boldsymbol{y} + \Delta\boldsymbol{y}$ can be used as an approximation of the discrete 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). Similarly, we can thus use the same approximation for our measure of distortion.

It is critical to ensure that the differential entropy estimate of $\tilde{\boldsymbol{y}}$ computed during the optimization closely tracks the discrete entropy of $\boldsymbol{q}$. Since we cannot give theoretical guarantees, we empirically evaluate the approximation in section 3. Because model error can have an effect on the approximation, we use unbiased, frequentist model estimates of $p_{\tilde{\boldsymbol{y}}}$ rather than Bayesian estimates (details in the appendix). Further, we assume independent marginals in the code space for both the relaxed probability model and the entropy code, and model the marginals $p_{\tilde{y}_i}$ non-parametrically. Specifically, we use finely sampled piecewise linear functions. Since $p_{\tilde{y}_i} = p_{y_i} * \mathcal{U}(0, 1)$ is effectively

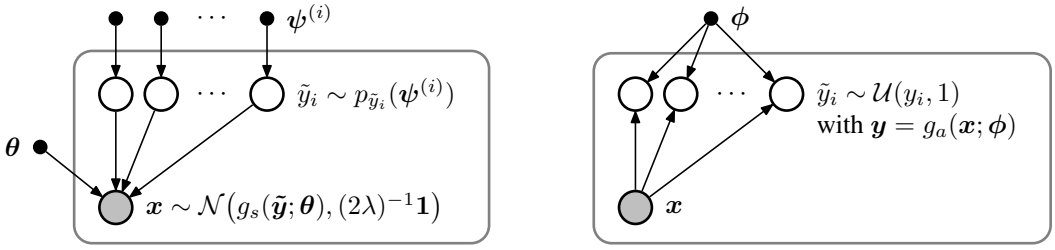


Figure 3: Probabilistic graphical model representation of the relaxed rate–distortion optimization problem, which is analogous to that used for variational autoencoders.

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.

The final loss function we use for optimizing $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ can be written as:

$$\begin{aligned} L(\boldsymbol{\theta},\boldsymbol{\phi}) = &- \mathbb{E}_{\boldsymbol{x},\Delta\boldsymbol{y}}\left[\sum_i \log_2 p_{\tilde{y}_i}(g_a(\boldsymbol{x};\boldsymbol{\phi}) + \Delta\boldsymbol{y};\boldsymbol{\psi}^{(i)})\right] \\ &+ \lambda\, \mathbb{E}_{\boldsymbol{x},\Delta\boldsymbol{y}}\Big\|g_p\big(g_s(g_a(\boldsymbol{x};\boldsymbol{\phi}) + \Delta\boldsymbol{y};\boldsymbol{\theta})\big) - g_p(\boldsymbol{x})\Big\|^2. \end{aligned} \tag{5}$$

where vector $\boldsymbol{\psi}^{(i)}$ parameterizes the piecewise linear approximation of $p_{\tilde{y}_i}$ (details provided in the appendix). This is continuous and differentiable with respect to the parameters, and thus suited for stochastic optimization.

2.1 Relationship to 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:

$$\begin{aligned} D_{\mathrm{KL}}\big[q\|p_{y|x}\big] &= \mathbb{E}_{y\sim q}\log q(y|x) - \mathbb{E}_{y\sim q}\log p_{y|x}(y|x) \\ &= \mathbb{E}_{y\sim q}\log q(y|x) - \mathbb{E}_{y\sim q}\log p_{x|y}(x|y) - \mathbb{E}_{y\sim q}\log p_y(y) + \text{const} \end{aligned} \tag{6}$$

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

$$p_{\boldsymbol{x}|\tilde{\boldsymbol{y}}}(\boldsymbol{x}|\tilde{\boldsymbol{y}};\lambda,\boldsymbol{\theta}) = \mathcal{N}\big(\boldsymbol{x}; g_s(\tilde{\boldsymbol{y}};\boldsymbol{\theta}), (2\lambda)^{-1}\mathbf{1}\big), \tag{7}$$

$$p_{\tilde{\boldsymbol{y}}}(\tilde{\boldsymbol{y}};\boldsymbol{\psi}^{(0)},\boldsymbol{\psi}^{(1)},\dots) = \prod_i p_{\tilde{y}_i}(\tilde{y}_i;\boldsymbol{\psi}^{(i)}), \tag{8}$$

and the approximate posterior as (right panel):

$$q(\tilde{\boldsymbol{y}}|\boldsymbol{x};\boldsymbol{\phi}) = \prod_i \mathcal{U}(\tilde{y}_i; y_i, 1) \quad \text{with } \boldsymbol{y} = g_a(\boldsymbol{x};\boldsymbol{\phi}), \tag{9}$$

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 K–L 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 norm in (5) is not Euclidean, $p_{\boldsymbol{x}|\tilde{\boldsymbol{y}}}$ is no longer Gaussian, but corresponds to a density proportional to an exponential of the distortion metric:

$$p_{\boldsymbol{x}|\tilde{\boldsymbol{y}}}(\boldsymbol{x}|\tilde{\boldsymbol{y}};\lambda,\boldsymbol{\theta}) = \frac{1}{Z(\lambda)} \exp\Big(-\lambda \big\| g_p\big(g_s(\tilde{\boldsymbol{y}};\boldsymbol{\theta})\big) - g_p(\boldsymbol{x}) \big\|\Big), \tag{10}$$

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

Despite these similarities between our compression framework and that of variational autoencoders, it is worth noting that the role of discretization in lossy compression systems leads to fundamental differences:

First, naively discretizing a continuous representation may severely compromise its effectiveness for compression, however well it may perform as a generative model. For example, performing scalar quantization on multivariate data is not even guaranteed to be optimal if the data is marginally independent and the error metric is separable (Gersho and Gray, 1992). For the purpose of evaluation, certain precautions must be taken to be able to compare differential entropy with (discrete) entropy, and discrete entropy can only be reasonably equated to the actual rate of a discrete code if an accurate discrete probability model is available to design the code.

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 (7), which in turn *maximizes* λ. Lossy compression methods, on the other hand, are optimized to achieve the best trade-off between having the actual model explain the data (which increases rate and decreases distortion), and 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 λ (see figure 2). In contrast, generative models operate in a regime where λ is inferred and ideally approaches infinity for noiseless data. This is 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 work on quantized pixel values, such as presented by van den Oord, Kalchbrenner, and Kavukcuoglu (2016).

Finally, correspondence between the typical slack term (7) of a generative model (figure 3, left panel) and the distortion metric in rate–distortion optimization only holds as long as simple metrics, such as the Euclidean distance, are used. A more general noise term corresponding to a perceptual measure, as given in (10), would be considered an odd choice from a generative modeling perspective.

3 Experimental results

We applied our optimization framework on analysis/synthesis transforms constructed from linear–nonlinear cascades. Most compression systems are based on orthogonal linear transforms, usually chosen to reduce correlations in the data, which substantially simplifies entropy coding. But the joint statistics of linear filter responses exhibit strong higher-order dependencies. These may be significantly reduced through the use of a joint 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). Here, we make use of a generalized divisive normalization (GDN) transform that we've 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 marginal nonlinearities (Ballé, Laparra, and Simoncelli, 2015). Specifically, our analysis transform g_a, consists of three stages of convolution, subsampling, and GDN, with the latter defined as:

$$v_i = \frac{u_i}{\big(\beta_i + \sum_j \gamma_{ij} u_j^2\big)^{\frac{1}{2}}}, \tag{11}$$

where the vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ hold the linear and normalized activations at one spatial location across feature maps, respectively, and the vector $\boldsymbol{\beta}$ and the symmetric matrix $\boldsymbol{\gamma}$ are parameters to be optimized. The synthesis transform g_s is formed as an approximate inverse (specifically, it uses one

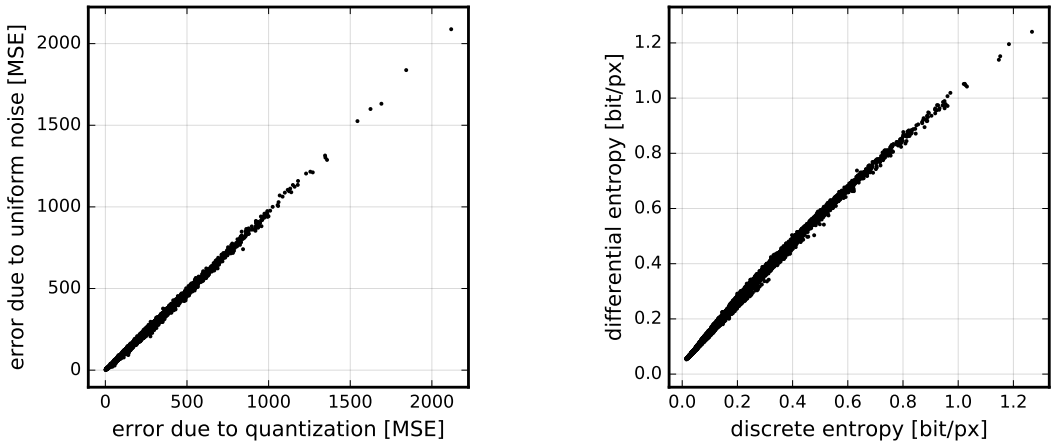


Figure 4: Scatter plots of discrete vs. continuously-relaxed terms of the objective function, evaluated for the GDN model, over the training set, and across values of λ between 32 and 2048 (inclusive). Left: distortion term, evaluated for $\hat{\boldsymbol{y}}$ vs. $\tilde{\boldsymbol{y}}$. Right: rate term, $H[P_{q_i}]$ vs. $h[p_{\tilde{y}_i}]$ (summed over i).

iteration of the fixed point iteration that can be used to invert the GDN transform):

$$v_i = u_i \cdot \left(\beta_i + \sum_j \gamma_{ij} u_j^2\right)^{\frac{1}{2}}, \tag{12}$$

with all parameters defined in the same way as the GDN parameters, but separately optimized.

For the convolutional structure of our transform, we use 128 filters in the first stage, each sampled by a factor of 4 vertically and horizontally, with the remaining two stages using 128 filters operating over the input features form previous stages, and subsampled by factors of 2 in each direction. In previous work, we used a perceptual transform g_p separately optimized to mimic human judgements of 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 squared error as the metric. For training, the full set of parameters (all linear filters, and parameters of the nonlinearities used in the analysis and synthesis transforms) are jointly optimized over a subset of the ImageNet database (Deng et al., 2009) consisting of 6507 images (see Appendix for details).

We first verified that the continuously-relaxed loss function given in section 2 provides a good approximation to the actual rate–distortion values, by comparing both terms of the objective function against their relaxed counterparts (figure 4). Each point corresponds to one image out of a random selection of 2169 images (one third) of the training set. The relaxed distortion term appears to be mostly unbiased, and exhibits a relatively small variance (left panel). The differential entropy provides a somewhat positively biased estimate of the discrete entropy for the coarser quantization regime (right panel), but the bias disappears for finer quantization, as expected. Note that since the values of λ 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 bias in either of the terms would simply alter the effective value of λ, with no effect on the compression performance.

We compare the rate–distortion performance of our system to two standard compression methods, JPEG and JPEG 2000. To make the comparison fair, we implemented a simple entropy code based on the context-based adaptive binary arithmetic coding framework (CABAC; Marpe, Schwarz, and Wiegand, 2003). Evaluations were performed on the Kodak image dataset[1], an uncompressed set of images commonly used to evaluate image compression systems. We report the average rate–distortion performance in figure 5, separately for luminance (left) and chrominance (right). Note that although the computational costs for training our system are quite high, encoding or decoding an image with the trained system is quite efficient, requiring only execution of the optimized analysis transformation and quantizer, or the synthesis transformation, respectively.

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

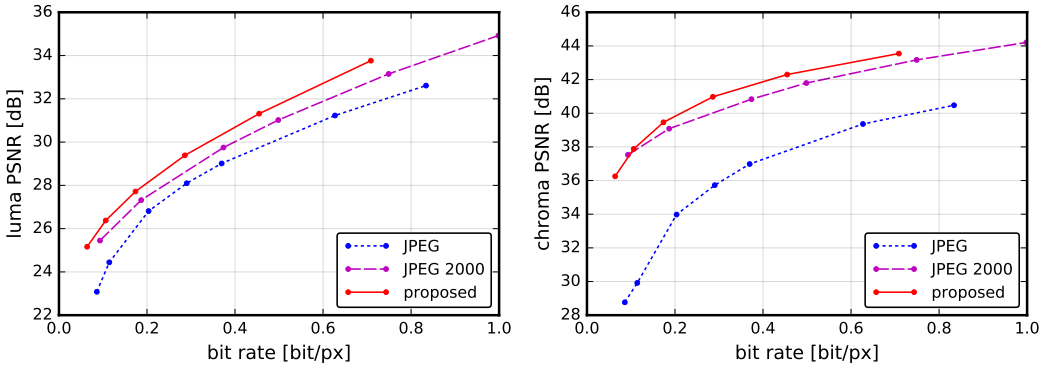


Figure 5: Rate–distortion values averaged over the Kodak set, for luminance (left) and color components (right). Distortion is expressed as peak signal-to-noise ratios, $10 \log_{10}(255^2/\mathrm{MSE})$.

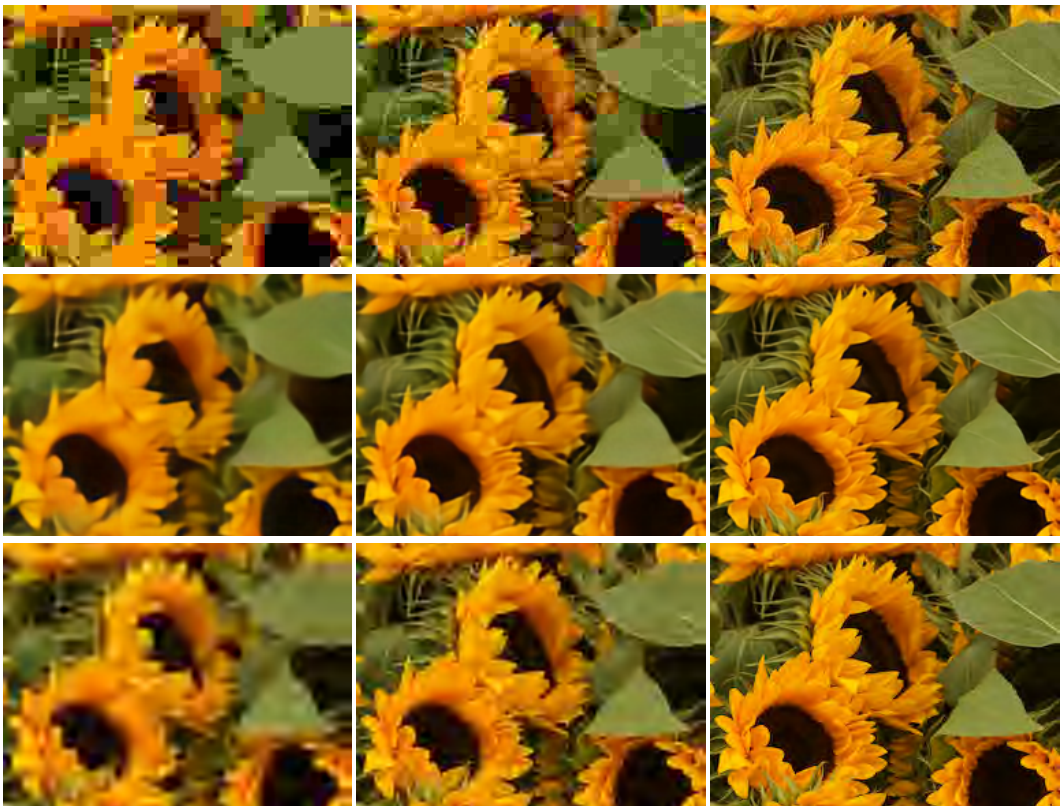


Figure 6: Progressive degradation of a compressed image across various values of λ (middle row) and different quality settings for JPEG (top row) and JPEG 2000 (bottom row), with comparable bit rates across columns.

Our system exhibits better rate–distortion performance than both JPEG and JPEG 2000 for most images and most bitrates, and is significantly better on average (figure 5), offering a reduction in bitrate of approximately 20% relative to JPEG 2000, and over 30% relative to JPEG. More importantly, we find the visual appearance of the compressed images is qualitatively different, exhibiting much less noticeable artifacts. Examples are shown in figures 6 and 7, and in the appendix. In each figure, we show an image compressed using our system for a selected λ value, and compare to JPEG/JPEG 2000 compressed versions of the same image at bit rates equal to or greater than that for our system. The JPEG and JPEG 2000 images exhibit artifacts that are common to all linear transform coders: local features (edges, contours, texture elements, etc.) are represented using particular combinations of localized linear basis functions, and independent scalar quantization of the

JPEG, 6006 bytes (0.170 bit/px), RMSE: 19.75/10.40, PSNR: 22.22 dB/27.79 dB

Proposed method, 5910 bytes (0.167 bit/px), RMSE: 17.27/6.51, PSNR: 23.38 dB/31.86 dB

JPEG 2000, 5918 bytes (0.167 bit/px), RMSE: 17.54/7.15, PSNR: 23.25 dB/31.04 dB

Figure 7: Compressed example image, 752×376 pixels. Note the appearance of artifacts around edges, especially those at oblique orientations (e.g., car wheels, trunk, and fenders).

transform coefficients leads to visually disturbing blocking or "aliasing" artifacts. For our proposed system, we see that although compressed images are less detailed than the original, with fine texture and other patterns often eliminated altogether, this is accomplished in a way that preserves the smoothness and sharpness of many contours and edges, giving them a more "natural" appearance.

4 DISCUSSION

We've presented a complete image compression system based on nonlinear transform coding, and a methodology to optimize it end-to-end for rate–distortion performance. We find that the system offers significant improvements in average rate–distortion performance over the JPEG and JPEG 2000 image compression standards. More remarkably, we found that although the system was optimized using mean squared error as a distortion metric, the compressed images are visually 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. We think this is generally because these cascaded nonlinear transformations have been optimized 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 B. Olshausen, 2001).

For comparison to standard linear transform coders, we can also interpret our analysis transform as single-stage linear transform followed by a complex vector quantizer. As in many other learned representations (e.g., sparse coding (Lewicki and B. A. 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* (128 filters, each subsampled by a factor of 16). Whereas most transform coders use complete (often orthogonal) linear transforms, the overcompleteness and orientation tuning of our initial transform may explain the ability of the system 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 learn image representations for use in compression. Gregor, Besse, et al. (2016) introduce an interesting hierarchical representation of images, in which degradations present in a "natural" way. They derive discrete entropy estimates from the model by quantizing the learnt continuous representation, and obtain images at different entropy levels by eliminating parts of the representation. This is not likely to be optimal for the reasons outlined in section 2.1 (which the authors do acknowledge, stating that the discretization needs further "tuning"). Our end-to-end optimization methodology offers a more systematic way to handle the rate–distortion trade-off. Toderici et al. (2016) acknowledge the need to optimize for various operating points in the rate–distortion sense, and their formulation has the advantage over ours that a single representation is sought for all rate points (this is a topic we are currently exploring). However, it is not clear whether their formulation necessarily leads to rate–distortion optimality, and their empirical results seem to suggest that this is not the case.

We are currently testing systems 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 all activation functions we tested, but that rectified linear units generally require a substantially larger number of model parameters to achieve the same rate–distortion performance as the GDN/IGDN nonlinearities, which might be an advantage for real-world implementation of our method (say, in embedded systems). The results may indicate that GDN/IGDN are more efficient for this particular task (producing better models with fewer stages of processing, as we found for density estimation (Ballé, Laparra, and Simoncelli, 2015)). On the other hand, this type of joint nonlinearity may generally be more powerful, as it represents a multivariate generalization of certain types of sigmoidal functions. We expect that some variant of a universal function approximation theorem should hold for them. However, such conclusions are based on a limited set of experiments and should at this point be considered provisional.

The rate–distortion objective may 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 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 for 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*. Accepted for presentation at the 32nd 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.

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*.

Lewicki, Michael S. and Bruno A. 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.

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

5 APPENDIX

5.1 NETWORK ARCHITECTURE AND OPTIMIZATION

As described in the main text, our analysis transform consists of three stages of convolution, downsampling, and GDN – details of the architectural choices are provided in figure 8. We note that these choices are somewhat ad-hoc and a more careful exploration of alternative architectures could potentially lead to significant performance improvements.

We've 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 we are assuming a marginally independent latent model 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 $\alpha = 2$ and $\varepsilon = 1/2$, and forcing the weight matrix to be symmetric (i.e., $\gamma_{ij} = \gamma_{ji}$).

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). we refer to this nonlinear transform as "inverse GDN" (IGDN). The full synthesis transform consists of three stages of IGDN, upsampling, and convolution, and we untied the optimization of the parameters of these transformations from those used in the analysis stages.

The full model (analysis and synthesis filters, GDN and IGDN parameters) were optimized 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 from the database in order to reduce artifacts and other unwanted contaminations: first, we eliminated images with excessive saturation. Then, 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 256×256 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, the resampling factor was never permitted to be greater than 0.75; we discarded the images that were not large enough.

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

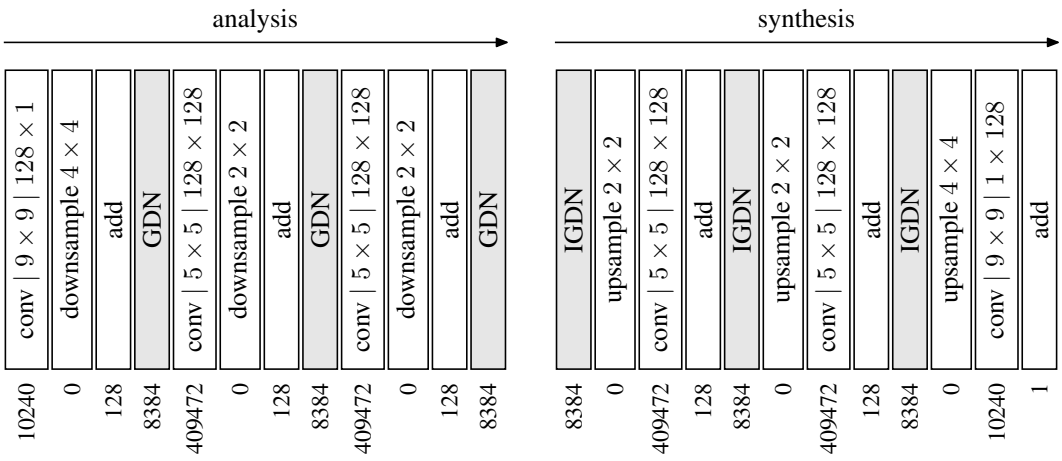


Figure 8: Architecture of synthesis (g_s) and analysis (g_a) transforms for grayscale images. *conv*: convolutional layer, with given filter support and number of output/input feature maps. *down-/upsample*: regular down-/upsampling (implemented jointly with the adjacent convolution). *add*: elementwise addition of a bias parameter per feature map. *GDN/IGDN*: generalized divisive normalization across feature maps (but not across space), and its approximate inverse; see text. Number of parameters for each layer given on the bottom.

- 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 $\alpha = 10^{-7}$.
- We parameterized the linear filters using their discrete cosine transform (DCT) coefficients. We found this to be slightly more effective in speeding up the convergence than DFT parameterization (Rippel, Snoek, and Adams, 2015).
- We parameterized the GDN/IGDN $\boldsymbol{\beta}$ parameters in terms of the elementwise relation

  $$\beta_i = (\beta_i')^2 - 2^{-10}.$$

  This ensures that gradients are smaller around parameter values close to 0, a regime in which the optimization can otherwise become unstable. To ensure an unambiguous mapping, we projected each β_i' onto the interval $[2^{-5}, \infty)$ after each gradient step. We applied the same treatment to the elements of $\boldsymbol{\gamma}$, and additionally averaged $\boldsymbol{\gamma}'$ with its transpose after each step in order to make it symmetric.
- 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 feature maps, and for the synthesis transform, over space and all output feature maps.

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 $\boldsymbol{\psi}^{(i)}$ thus simply 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 loss function

$$L_\psi(\boldsymbol{\psi}^{(0)}, \boldsymbol{\psi}^{(1)}, \dots) = -\,\mathbb{E}_{\tilde{\boldsymbol{y}}} \sum_i p_{\tilde{y}_i}(\tilde{y}_i; \boldsymbol{\psi}^{(i)}) \tag{13}$$

and renormalized the marginal densities after each step. This yields unbiased running estimates of the marginal densities, unlike minimizing the loss function given in (5) over $\boldsymbol{\psi}^{(i)}$. After every 10^6 gradient steps, we used a heuristic to adapt the range of the spline approximation to the values of $\tilde{y}_i$ that had been actually observed.

5.2 Entropy code

We implemented an arithmetic entropy coder based on the 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, given that the probability model used to design the code describes the data well. Adaptive codes can potentially further improve bit rates, and to some extent correct model error, by adapting the probability model on-line to the statistics of each compressed vector. This is possible when several elements of the code vector $\boldsymbol{q}$ are modeled by the same distribution (in our case, each feature map in $\boldsymbol{q}$ is modeled by a marginal distribution, P_{q_i}, which all elements in that feature map share across space). Here, we derived the initial probability model for each feature map element q_i by subsampling the continuous density $p_{\tilde{y}_i}$ learned during training, as in (4).

The following information was encoded into the bitstream:

- the size of the image (two 16-bit integers),
- whether the image is grayscale or RGB (one bit),
- the value of λ (one 16-bit integer),
- the value of each element of $\boldsymbol{q}$, iterating across feature maps, and across space in raster-scan order.

The parameters of the analysis and synthesis transforms as well as the initial probability models for the discrete codes, for each λ, were assumed to be fixed and available to encoder and decoder.

5.3 EVALUATION DETAILS AND ADDITIONAL EXAMPLE IMAGES

To generate the plots in figure 5, we did the following for each value of λ (or quality setting of JPEG/JPEG 2000): we computed the mean squared error per pixel between each reconstructed image and the corresponding original (after converting them to luma and chroma components), averaged across all 23 images, and then converted the mean squared error into PSNR; we divided the bit stream size in bits by the number of pixels in each image and averaged across images to yield the reported bit rate. Full per-image results for grayscale and RGB models are available at `http://www.cns.nyu.edu/~balle/iclr2017-kodak-gray` and `http://www.cns.nyu.edu/~balle/iclr2017-kodak-rgb`, respectively.

In all experiments, we compare to JPEG with 4:2:0 chroma subsampling, and 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.

In what follows, we show additional example images not from the standard set, focusing on lower bit rates, in order to visualize the qualitative nature of compression artifacts. On each page, the JPEG and JPEG 2000 images are selected to have the lowest possible bit rate that is equal or greater than the bit rate of the proposed method.

JPEG, 4215 bytes (0.119 bit/px), RMSE: 11.07/10.60, PSNR: 27.25 dB/27.63 dB

Proposed method, 3751 bytes (0.106 bit/px), RMSE: 6.10/5.09, PSNR: 32.43 dB/34.00 dB

JPEG 2000, 3769 bytes (0.107 bit/px), RMSE: 8.56/5.71, PSNR: 29.49 dB/32.99 dB

Figure 9: RGB example, 752×376 pixels.

JPEG, 16265 bytes (0.460 bit/px), RMSE: 13.24/6.91, PSNR: 25.69 dB/31.34 dB

Proposed method, 15140 bytes (0.428 bit/px), RMSE: 12.11/4.99, PSNR: 26.47 dB/34.16 dB

JPEG 2000, 15161 bytes (0.429 bit/px), RMSE: 12.63/5.50, PSNR: 26.10 dB/33.32 dB

Figure 10: RGB example, 752×376 pixels.

JPEG, 14068 bytes (0.398 bit/px), RMSE: 9.36/8.98, PSNR: 28.70 dB/29.07 dB

Proposed method, 13314 bytes (0.377 bit/px), RMSE: 8.05/5.85, PSNR: 30.02 dB/32.79 dB

JPEG 2000, 13318 bytes (0.377 bit/px), RMSE: 9.28/6.16, PSNR: 28.78 dB/32.34 dB

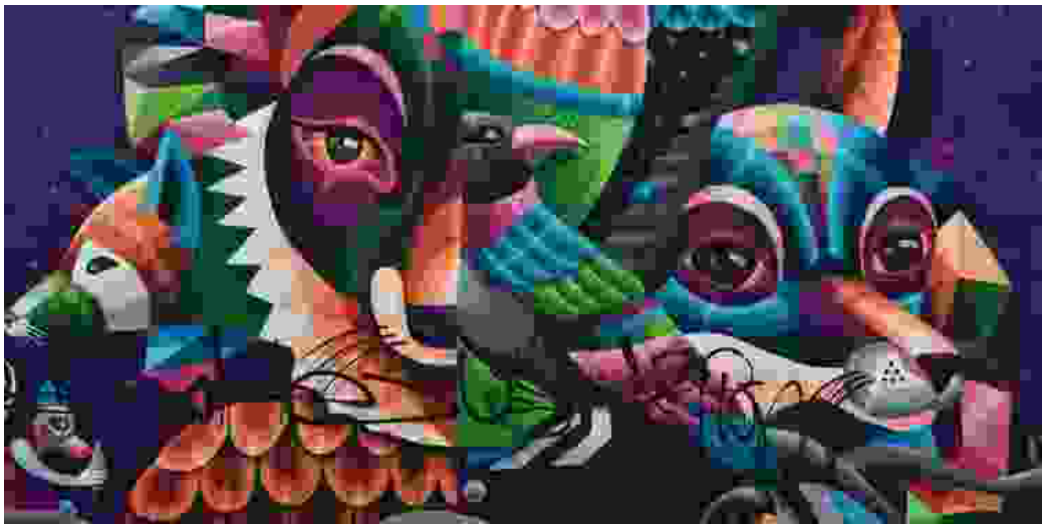
Figure 11: RGB example, 752×376 pixels.

JPEG, 10911 bytes (0.309 bit/px), RMSE: 9.08/5.89, PSNR: 28.96 dB/32.73 dB

Proposed method, 10647 bytes (0.301 bit/px), RMSE: 7.88/3.73, PSNR: 30.20 dB/36.69 dB

JPEG 2000, 10679 bytes (0.302 bit/px), RMSE: 8.21/3.98, PSNR: 29.84 dB/36.13 dB

Figure 12: RGB example, 752×376 pixels.



JPEG, 5928 bytes (0.168 bit/px), RMSE: 15.44/12.40, PSNR: 24.36 dB/26.26 dB

Proposed method, 5685 bytes (0.161 bit/px), RMSE: 10.41/5.98, PSNR: 27.78 dB/32.60 dB

JPEG 2000, 5724 bytes (0.162 bit/px), RMSE: 13.75/7.02, PSNR: 25.36 dB/31.20 dB

Figure 13: RGB example, 752×376 pixels.

JPEG, 11020 bytes (0.312 bit/px), RMSE: 16.90, PSNR: 23.57 dB

Proposed method, 10108 bytes (0.286 bit/px), RMSE: 13.87, PSNR: 25.29 dB

JPEG 2000, 10158 bytes (0.287 bit/px), RMSE: 15.28, PSNR: 24.45 dB

Figure 14: Grayscale example, 752×376 pixels.

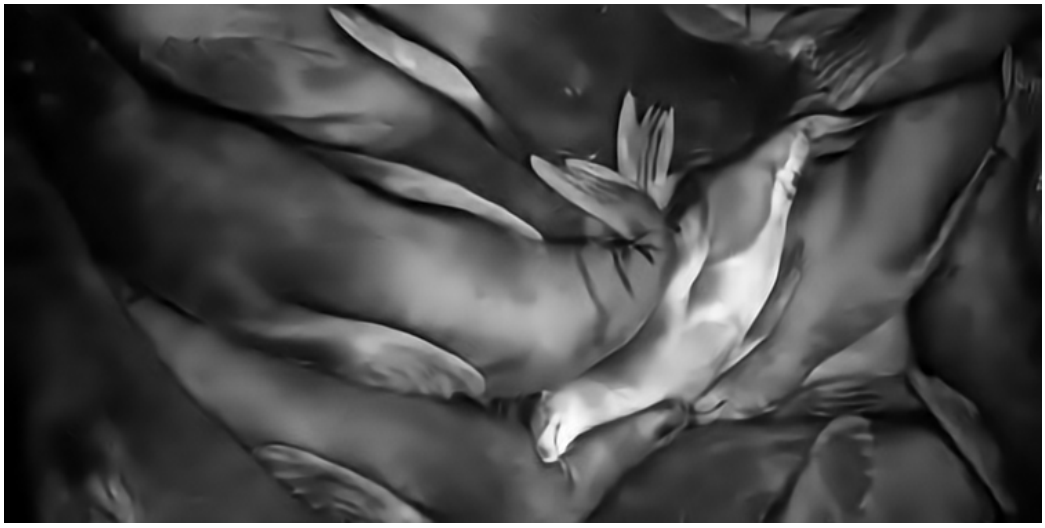
JPEG, 5047 bytes (0.143 bit/px), RMSE: 9.55, PSNR: 28.53 dB

Proposed method, 4546 bytes (0.129 bit/px), RMSE: 7.18, PSNR: 31.01 dB

JPEG 2000, 4554 bytes (0.129 bit/px), RMSE: 7.91, PSNR: 30.17 dB

Figure 15: Grayscale example, 752×376 pixels.

JPEG, 4462 bytes (0.126 bit/px), RMSE: 9.00, PSNR: 29.05 dB



Proposed method, 3877 bytes (0.110 bit/px), RMSE: 6.59, PSNR: 31.75 dB

JPEG 2000, 3877 bytes (0.110 bit/px), RMSE: 6.99, PSNR: 31.24 dB

Figure 16: Grayscale example, 752×376 pixels.

JPEG, 7058 bytes (0.200 bit/px), RMSE: 12.32, PSNR: 26.32 dB

Proposed method, 6635 bytes (0.188 bit/px), RMSE: 9.23, PSNR: 28.83 dB

JPEG 2000, 6691 bytes (0.189 bit/px), RMSE: 9.22, PSNR: 28.83 dB

Figure 17: Grayscale example, 752×376 pixels.

JPEG, 13393 bytes (0.379 bit/px), RMSE: 12.05, PSNR: 26.51 dB

Proposed method, 13334 bytes (0.377 bit/px), RMSE: 9.30, PSNR: 28.76 dB

JPEG 2000, 13494 bytes (0.382 bit/px), RMSE: 10.29, PSNR: 27.88 dB

Figure 18: Grayscale example, 752×376 pixels.