VARIATIONAL IMAGE COMPRESSION WITH A SCALE HYPERPRIOR

Anonymous authors
Paper under double-blind review

ABSTRACT

We describe an end-to-end trainable model for image compression based on variational autoencoders. The model incorporates a hyperprior into the generative model to effectively capture spatial dependencies in the latent representation. This hyperprior relates to side information, a concept universal to virtually all modern image codecs but largely unexplored for image compression with artificial neural networks (ANNs). Unlike existing autoencoder compression methods, our model trains a powerful entropy model jointly with the underlying autoencoder. We demonstrate that this model leads to state-of-the-art image compression when measuring visual quality using the popular MS-SSIM index, and yields rate–distortion performance surpassing published ANN-based methods when evaluated using a more traditional metric based on squared error (PSNR).

1 INTRODUCTION

Recent machine learning methods for lossy image compression have generated significant interest in both the machine learning and image processing communities (Rippel and Bourdev, 2017; Johnston et al., 2017; Ballé et al., 2016; Minnen et al., 2017). The majority of these methods are based on autoencoders with a continuous latent representation that is quantized and then losslessly compressed using *entropy coding* to create the bitstream.

The expected code length of an optimal entropy coder is the cross entropy between the true distribution of the latent representation and the probability model used by the entropy coder (the *entropy model*). The minimal code length is achieved when the observed distribution and the model distribution are identical, and any divergence can lead to suboptimal compression performance. In all of the existing machine learning methods for image compression, the entropy model used to encode the latent representation is assumed to be fixed and fully known by both the encoder and decoder. Empirically, however, the distribution of the latent representation of natural images varies significantly from image to image as well as spatially within a single image. This observation implies a potential mismatch between the entropy model and the local distributions within each image, which means that it may be possible to improve existing compression models by making the entropy model image-dependent and spatially adaptive.

Conventional compression methods achieve this by transmitting *side information* that allows for more effective entropy coding. For instance, while JPEG always uses fixed-size blocks of 8×8 pixels, more recent methods such as WebP and BPG (derived from HEVC) partition each image into variable-size blocks, convey the partition structure to the decoder, and then transmit the actual block representations. Because the encoder can optimize the partitioning for each image, it can be used to achieve higher compression rates without degrading image quality.

In conventional codecs, the structure of this side information is hand-designed. In contrast, the models presented in this paper learn to generate side information that adapts to the image content. Essentially, they learn a representation of the entropy model in the same way that the underlying model learns a representation of the image. Furthermore, because our models are optimized end-to-end, they minimize the total expected code length by learning to balance the amount of side information with the expected improvement of the entropy model.

This optimization problem can be formalized by expressing it in terms of variational autoencoders (VAEs), probabilistic generative models augmented with approximate inference models (Diederik P. Kingma and Welling, 2014). Ballé et al. (2016) and Theis et al. (2017) previously identified that

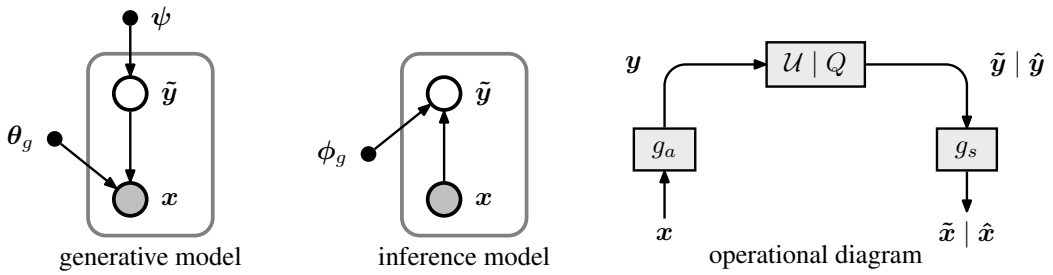

Figure 1: Left: Representation of a transform coding model as a generative Bayesian model, and a corresponding variational inference model. Nodes represent random variables or parameters, and arrows indicate conditional dependence. Right: Diagram showing the operational structure of the compression model. Arrows indicate the flow of data, and boxes represent transformations of the data. Boxes labeled $\mathcal{U} \mid Q$ represent either addition of uniform noise applied during training (producing vectors labeled with a tilde), or quantization and arithmetic coding/decoding during testing (producing vectors labeled with a hat).

Figure 2: Left: an image from the Tecnick dataset. Center: visualization of a subset of the responses $\hat{\boldsymbol{y}}$ of a nonlinear transform coding method to that image. Note that even though the model assumes independence between the spatial neighbors, there is clearly visible structure around edges and textured regions, indicating that a dependency structure exists. Right: visualization of the normalized responses of a hierarchical model obtained by dividing the model responses $\boldsymbol{y}$ elementwise by the model estimate of their standard deviation $\hat{\boldsymbol{\sigma}}$. Note how this reduces the apparent structure.

some autoencoder-based compression methods are formally equivalent to VAEs, where the entropy model corresponds to the prior on the latent representation. Here, we use this formalism to show that side information can be viewed as a prior on the parameters of the entropy model making them *hyperpriors* of the latent representation.

Specifically, we extend the model presented in Ballé et al. (2016) with a hyperprior that captures the fact that spatially neighboring elements of the latent representation vary together in their scales. We demonstrate that our model leads to state-of-the-art image compression when measured using the MS-SSIM quality index, and it provides significantly better rate-distortion performance compared to other machine learning methods when measured using PSNR. Finally, we present novel results studying the effects of training the same model class using different distortion losses.

2 Compression with Variational Models

In the *transform coding* approach to image compression, the encoder transforms the image vector $\boldsymbol{x}$ using a parametric analysis transform $g_a(\boldsymbol{x}; \boldsymbol{\phi}_g)$ into a latent representation $\boldsymbol{y}$, which is then quantized to form $\hat{\boldsymbol{y}}$. Because $\hat{\boldsymbol{y}}$ is discrete valued, it can be losslessly compressed using entropy coding techniques such as arithmetic coding () and transmitted as a sequence of bits. On the other side, the decoder recovers $\hat{\boldsymbol{y}}$ from the compressed signal, and subjects it to a parametric synthesis transform $g_s(\hat{\boldsymbol{y}}; \boldsymbol{\theta}_g)$ to recover the reconstructed image $\hat{\boldsymbol{x}}$. In the context of this paper, we think

of the transforms g_a and g_s as generic parameterized functions, such as artificial neural networks. The parameters $\boldsymbol{\theta}_g$ and $\boldsymbol{\phi}_g$ then encapsulate the weights of the neurons, etc. (refer to Section 4 for details).

Compression can be cast as a rate–distortion optimization problem. Rate is the expected code length (bit rate) of the compressed representation: assuming the entropy coding technique is operating at high efficiency, this can be approximated as the Shannon entropy of $\hat{\boldsymbol{y}}$. Distortion is the expected difference between the reconstruction $\hat{\boldsymbol{x}}$ and the original image $\boldsymbol{x}$. A trade-off exists between rate and distortion where a higher rate allows for a lower distortion and vice versa. Various compression methods can be viewed as minimizing a weighted sum of these two quantities. The minimization problem is parameterized by λ, the weight on the distortion term. Different applications require a different trade-off between the rate and distortion, and hence different values of λ.

In order to be able to use gradient descent methods to optimize the performance of the model over the parameters of the transforms ($\boldsymbol{\theta}_g$ and $\boldsymbol{\phi}_g$), the problem needs to be relaxed, because due to the quantization, gradients with respect to $\boldsymbol{\phi}_g$ are zero almost everywhere. Approximations that have been investigated include substituting the gradient of the quantizer (Theis et al., 2017), and substituting additive uniform noise for the quantizer itself during training (J. Ballé et al., 2016). Here, we follow the latter method, which switches back to actual quantization for applying the model as a compression method. We denote the quantities derived from this approximation with a tilde, as opposed to a hat; for instance, $\tilde{\boldsymbol{y}}$ represents the "noisy" representation, and $\hat{\boldsymbol{y}}$ the quantized representation.

The relaxed problem can be formally represented as a variational autoencoder (Diederik P. Kingma and Welling, 2014); that is, a probabilistic generative model of the image combined with an approximate inference model (figure 1). The synthesis transform is linked to the generative model ("generating" a reconstructed image from the latent representation), and the analysis transform to the inference model ("inferring" the latent representation from the source image). In variational inference, the true posterior $p_{\tilde{\boldsymbol{y}}|\boldsymbol{x}}(\tilde{\boldsymbol{y}} \mid \boldsymbol{x})$ is approximated with a parametric variational density $q(\tilde{\boldsymbol{y}} \mid \boldsymbol{x})$ by minimizing the Kullback–Leibler divergence

$$D_{\mathrm{KL}}[q \parallel p_{\tilde{\boldsymbol{y}}|\boldsymbol{x}}] = \cancelto{0}{\mathbb{E}_{\tilde{\boldsymbol{y}}\sim q} \log q(\tilde{\boldsymbol{y}} \mid \boldsymbol{x})} \underbrace{- \mathbb{E}_{\tilde{\boldsymbol{y}}\sim q} \log p_{\boldsymbol{x}|\tilde{\boldsymbol{y}}}(\boldsymbol{x} \mid \tilde{\boldsymbol{y}})}_{\text{weighted distortion}} \underbrace{- \mathbb{E}_{\tilde{\boldsymbol{y}}\sim q} \log p_{\tilde{\boldsymbol{y}}}(\tilde{\boldsymbol{y}})}_{\text{rate}} + \text{const.} \tag{1}$$

By matching the parametric density functions to the transform coding framework, we can appreciate that the minimization of the KL divergence is equivalent to optimizing the compression model for rate–distortion performance. We have indicated here that the first term will evaluate to zero, and the second and third term correspond to the weighted distortion and the bit rate, respectively. Let's take a closer look at each of the terms.

First, the mechanism of "inference" is computing the the analysis transform of the image and adding uniform noise (as a stand-in for quantization), thus:

$$\begin{aligned} q(\tilde{\boldsymbol{y}} \mid \boldsymbol{x}, \boldsymbol{\phi}_g) &= \prod_i \mathcal{U}\big(\tilde{y}_i \mid y_i - \tfrac{1}{2}, y_i + \tfrac{1}{2}\big) \\ &\quad \text{with } \boldsymbol{y} = g_a(\boldsymbol{x}; \boldsymbol{\phi}_g), \end{aligned} \tag{2}$$

where $\mathcal{U}$ denotes the uniform distribution. Since the width of the uniform distribution is constant, the first term in the KL divergence technically evaluates to zero, and can be dropped from the loss function.

For the sake of argument, assume for a moment that the likelihood is given by:

$$\begin{aligned} p_{\boldsymbol{x}|\tilde{\boldsymbol{y}}}(\boldsymbol{x} \mid \tilde{\boldsymbol{y}}, \boldsymbol{\theta}_g) &= \mathcal{N}\big(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}, (2\lambda)^{-1}\mathbf{1}\big) \\ &\quad \text{with } \tilde{\boldsymbol{x}} = g_s(\tilde{\boldsymbol{y}}; \boldsymbol{\theta}_g) \end{aligned} \tag{3}$$

The log likelihood then works out to be the squared difference between $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$, the output of the synthesis transform; weighted by λ. Minimizing the second term in the KL divergence is thus equivalent to minimizing the expected distortion of the reconstructed image. Minimizing squared error is equivalent to choosing a Gaussian distribution; other distortion metrics may have an equivalent distribution, but this is not guaranteed, as not all metrics necessarily correspond to a normalized density function.

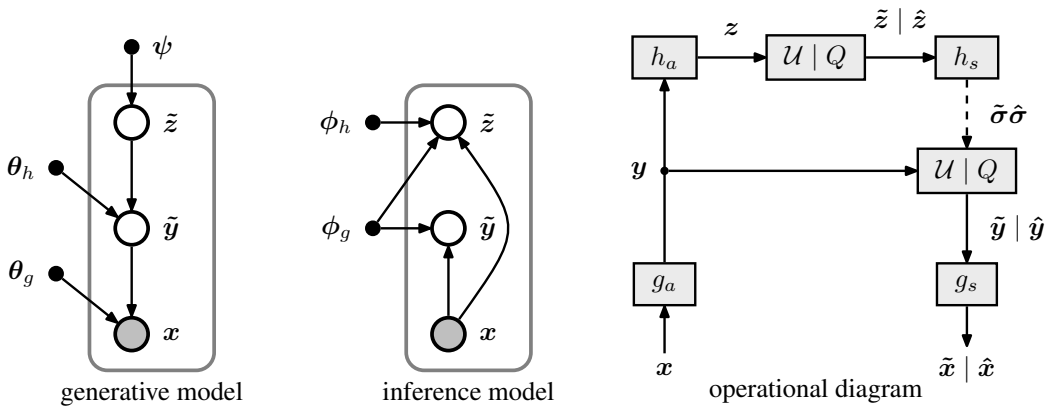

Figure 3: As in figure 1, but extended with a hyperprior.

The third term in the KL divergence is easily seen to be identical to the differential cross-entropy between the variational density q and the prior. It reflects the cost of encoding $\tilde{\boldsymbol{y}}$, as produced by the inference model, assuming $p_{\tilde{\boldsymbol{y}}}$ as the entropy model. Under certain provisions, this closely approximates the discrete (Shannon) cross-entropy of encoding $\hat{\boldsymbol{y}}$ (for details, see Ballé et al., 2016), which represents the expected bitrate of an optimal arithmetic coder. Most previous work assumes that $p_{\tilde{\boldsymbol{y}}}$ is a parametric, but factorized distribution, i.e., assumes independence between the elements of $\tilde{\boldsymbol{y}}$ (Theis et al., 2017; Ballé et al., 2016).

The center panel in figure 2 visualizes a subset of the quantized responses ($\hat{\boldsymbol{y}}$) of a compression model trained in this way. Visually, it is clear that the choice of a factorized distribution is a stark simplification: the responses are highly clustered in areas of high contrast; i.e., around edges, or within textured regions.

This implies a probabilistic coupling between the responses which is not represented in models with a fully factored prior. We would expect a better model fit and, consequently, a better compression performance, if the model captured these dependencies. Introducing a hyperprior is an elegant way of achieving this.

3 Introduction of a scale hyperprior

As evident from the center pane of figure 2, there are significant spatial dependencies among $\hat{y}$. A standard way to model dependencies between a set of target variables is to introduce latent variables conditioned on which the target variables are assumed to be independent (Bishop, 1998). We introduce an additional set of random variables $\tilde{\boldsymbol{z}}$ to capture the spatial dependencies and propose the following extension of the model (figure 3):

Each element $\tilde{y}_i$ is modeled as a zero-mean Gaussian with its own standard deviation σ_i, where the standard deviations are predicted by applying a parametric transform h_s to $\tilde{\boldsymbol{z}}$:

$$p_{\tilde{\boldsymbol{y}}|\tilde{\boldsymbol{z}}}(\tilde{\boldsymbol{y}} \mid \tilde{\boldsymbol{z}}, \boldsymbol{\theta}_h) = \prod_i \Big(\mathcal{N}\big(0, \tilde{\sigma}_i^2\big) * \mathcal{U}\big(-\tfrac{1}{2}, \tfrac{1}{2}\big) \Big)(\tilde{y}_i) \quad \text{with } \tilde{\boldsymbol{\sigma}} = h_s(\tilde{\boldsymbol{z}}; \boldsymbol{\theta}_h) \tag{4}$$

Note that we convolve each Gaussian with a standard uniform density. This is to take into account the uniform noise added by the inference model – if we modeled $\tilde{y}_i$ simply as Gaussian, it would create a potential model mismatch between the variational encoder and decoder. This technique is explained in more detail in appendix 7.2.

We extend the inference model simply by stacking another parametric transform h_a on top of $\boldsymbol{y}$, effectively creating a single joint factored variational posterior, as follows:

$$q(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}} \mid \boldsymbol{x}, \boldsymbol{\phi}_g, \boldsymbol{\phi}_h) = \prod_i \mathcal{U}\big(\tilde{y}_i \mid y_i - \tfrac{1}{2}, y_i + \tfrac{1}{2}\big) \cdot \prod_j \mathcal{U}\big(\tilde{z}_j \mid z_j - \tfrac{1}{2}, z_j + \tfrac{1}{2}\big) \quad \text{with } \boldsymbol{y} = g_a(\boldsymbol{x}; \boldsymbol{\phi}_g), \boldsymbol{z} = h_a(\boldsymbol{y}; \boldsymbol{\phi}_h) \tag{5}$$

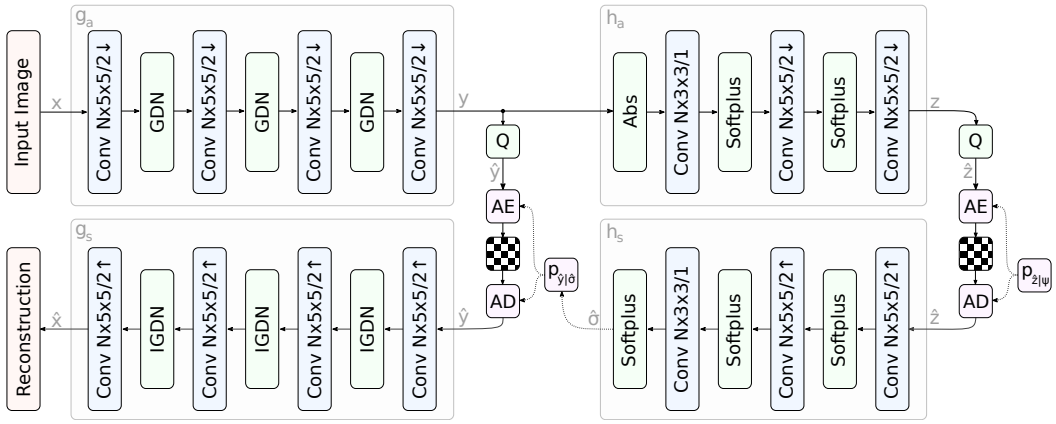

Figure 4: Network Architecture of our model. Convolutional layers are represented as (num_channels $\times$ height $\times$ width / stride). The left half is akin to an autoencoder with Q representing a quantization block and AE, AD representing arithmetic encoder and arithmetic decoder blocks, respectively. The right half corresponds to the network learning hyperpriors with $p_{\hat{y}|\hat{\sigma}}$ representing priors dependent on hyperpriors and $p_{\hat{z}|\psi}$ representing the distribution of hyperparameters.

This follows the intuition that the responses $\boldsymbol{y}$ should be sufficient to estimate the spatial distribution of the standard deviations. As we have no prior beliefs about the hyperprior on $\tilde{\boldsymbol{z}}$, we model it using a non-parametric, fully factorized density model (refer to appendix 7.1 for details):

$$p_{\tilde{\boldsymbol{z}}|\boldsymbol{\psi}}(\tilde{\boldsymbol{z}} \mid \boldsymbol{\psi}) \quad = \quad \prod_i \Big(p_{z_i|\boldsymbol{\psi}^{(i)}}\big(\boldsymbol{\psi}^{(i)}\big) * \mathcal{U}\big(-\tfrac{1}{2}, \tfrac{1}{2}\big)\Big)(\tilde{z}_i) \tag{6}$$

where the vectors $\boldsymbol{\psi}^{(i)}$ encapsulate the parameters of each marginal distribution $p_{z_i|\boldsymbol{\psi}^{(i)}}$ (we denote all these parameters collectively as $\boldsymbol{\psi}$). Again, we use the uniform noise distribution to reduce model mismatch (appendix 7.2).

The loss function of this model works out to be:

$$\begin{aligned} D_{\mathrm{KL}}[q \parallel p_{\tilde{\boldsymbol{y}},\tilde{\boldsymbol{z}}|\boldsymbol{x}}] = \mathbb{E}_{\tilde{\boldsymbol{y}}\sim q} \log q(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}} \mid \boldsymbol{x}) - \mathbb{E}_{\tilde{\boldsymbol{y}}\sim q} \log p_{\boldsymbol{x}|\tilde{\boldsymbol{y}}}(\boldsymbol{x} \mid \tilde{\boldsymbol{y}}) \\ - \mathbb{E}_{\tilde{\boldsymbol{y}}\sim q} \log p_{\tilde{\boldsymbol{y}}|\tilde{\boldsymbol{z}}}(\tilde{\boldsymbol{y}} \mid \tilde{\boldsymbol{z}}) - \mathbb{E}_{\tilde{\boldsymbol{y}}\sim q} \log p_{\tilde{\boldsymbol{z}}}(\tilde{\boldsymbol{z}}) + \mathrm{const}. \end{aligned} \tag{7}$$

Again, the first term is constant, since q is a product of uniform densities with constant width. The second term (the likelihood) encapsulates the distortion, as before. The third and fourth term measure the conditional cross-entropy/cross-entropy needed to encode $\tilde{\boldsymbol{y}}$ and $\tilde{\boldsymbol{z}}$, respectively.

The right-hand panel in figure 3 illustrates how the model is used as a compression method. The encoder subjects the input image $\boldsymbol{x}$ to g_a, yielding the responses $\boldsymbol{y}$ with spatially varying standard deviations. The responses are fed into h_a, summarizing the distribution of standard deviations in $\boldsymbol{z}$. $\boldsymbol{z}$ is then quantized, compressed, and sent over the channel. The encoder then uses the quantized vector $\hat{\boldsymbol{z}}$ to estimate $\hat{\boldsymbol{\sigma}}$, the spatial distribution of standard deviations, and uses it to compress and transmit the quantized responses $\hat{\boldsymbol{y}}$.

The decoder first recovers $\hat{\boldsymbol{z}}$ from the compressed signal. It then uses h_s to obtain $\hat{\boldsymbol{\sigma}}$, which provides it with the correct probability estimates to successfully recover $\hat{\boldsymbol{y}}$ as well. It then feeds $\hat{\boldsymbol{y}}$ into g_s to obtain the reconstructed image.

4 Experiments

To assess the compression performance of our proposed model, we conducted a number of experiments using the Tensorflow framework.

4.1 Experimental setup

We set up the transforms g_a, g_s, h_a, and h_s as alternating compositions of linear and nonlinear functions, as is common in artificial neural networks (figure 4). Specifically, g_a and g_s are composed

of convolutions and GDN/IGDN nonlinearities (Ballé et al., 2015; Ballé et al., 2016).[1] h_a and h_s are composed of convolutions and $\mathrm{softplus}$ nonlinearities.

The fully factorized nonparametric model $p_{z_i|\boldsymbol{\psi}^{(i)}}$ is described in appendices 7.1 and 7.2. To maintain the translation invariance, all elements of $\boldsymbol{z}$ with the same channel index are assumed to follow the same marginal distribution. This allows the model to be used with arbitrary image sizes. Arithmetic coding is implemented using a simple non-adaptive binary arithmetic coder. Each element of $\hat{\boldsymbol{y}}$ and $\hat{\boldsymbol{z}}$ is independently converted to its representation as a binary integer and arithmetically encoded from the most significant to the least significant bit. Since the spatial distribution of standard deviations ($\hat{\boldsymbol{\sigma}}$) is known to the decoder by the time decoding of $\hat{\boldsymbol{y}}$ is attempted, the arithmetic coder does not need to handle conditional dependencies. It also does not need to be separately trained, since the binary probabilities needed for encoding are a direct function of the probability mass functions of $\hat{\boldsymbol{y}}$ and $\hat{\boldsymbol{z}}$, and the probability mass functions in turn are direct functions of their "noisy" counterparts $\tilde{\boldsymbol{y}}$, $\tilde{\boldsymbol{z}}$ by design (J. Ballé et al., 2016). This is particulary important for $\hat{\boldsymbol{y}}$. Since the prior is conditioned on $\hat{\boldsymbol{\sigma}}$, the probability mass functions $P_{\hat{y}_i}$ need to be constructed "on the fly" during decoding of an image:

$$P_{\hat{y}_i}(\hat{y}_i \mid \hat{\sigma}_i) = p_{\tilde{y}_i}(\hat{y}_i \mid \hat{\sigma}_i) = \Big(\mathcal{N}(0, \hat{\sigma}_i) * \mathcal{U}\big(-\tfrac{1}{2}, \tfrac{1}{2}\big)\Big)(\hat{y}_i) = \int_{\hat{y}_i - 1/2}^{\hat{y}_i + 1/2} \mathcal{N}(y \mid 0, \hat{\sigma}_i)\, \mathrm{d}y \quad (8)$$

which can be evaluated in closed form.

The models were trained on a body of color JPEG images with heights/widths between 3000 and 5000 pixels, comprising approximately 1 million images scraped from the world wide web. Images with excessive saturation were screened out to reduce the number of non-photographic images. To reduce existing compression artifacts, the images were further downsampled by a randomized factor, such that the minimum of their height and width equaled between 640 and 1200 pixels. Then, randomly placed 256×256 pixel crops of these downsampled images were extracted. Minibatches of 8 of these crops at a time were used to perform stochastic gradient descent using the Adam algorithm (D. P. Kingma and Ba, 2014) with a learning rate of 10^{-4}. Common machine learning techniques such as batch normalization or learning rate decay were found to have no beneficial effect (this may be due to the local normalization properties of GDN, which contain global normalization as a special case).

With this setup, we trained a total of 32 separate models: half of the models with a hyperprior and half without; half of the models with mean squared error as the distortion metric (as described in the previous section), and half on the MS-SSIM distortion index (Wang, Eero P Simoncelli, et al., 2003); finally, each of these combinations with 8 different values of λ in order to cover a range of rate–distortion tradeoffs.

4.2 Experimental results

We evaluate the compression performance of all models on the publicly available Kodak dataset (Kodak, n.d.). Summarized rate–distortion curves are shown in figure 5. Results for individual images, as well as summarized comparisons to a wider range of existing methods such as JPEG 2000, WebP, Johnston et al. (2017), are provided in appendices 7.3 and 7.4. We quantify image distortion using peak signal-to-noise ratio (PSNR) and MS-SSIM. Each curve represents the rate–distortion tradeoffs for a given set of models, across different values of λ. Since MS-SSIM yields values between 0 (worst) and 1 (best), and most of the compared methods achieve values well above 0.9, we converted the quantity to decibels in order to improve legibility.

Interestingly, but maybe not surprisingly, results differ substantially depending on which distortion metric is used in the loss function during training. When measuring distortion in PSNR (figure 5, top), both our models perform poorly if they have been optimized for MS-SSIM. However, when optimized for squared error, the model with fixed prior outperforms existing conventional codecs such as JPEG, as well as other machine learning based methods which have been trained for squared error (Theis et al., 2017; Ballé et al., 2016). Note that other published methods not shown here underperform compared to the ones that are shown, or have not made their data available to us. Our fixed prior model does not outperform BPG, an encapsulation of HEVC (Bross et al., 2012) targeted

[1] We used the Tensorflow implementation of GDN/IGDN, as documented at `https://www.tensorflow.org/versions/r1.4/api_docs/python/tf/contrib/layers/GDN`.

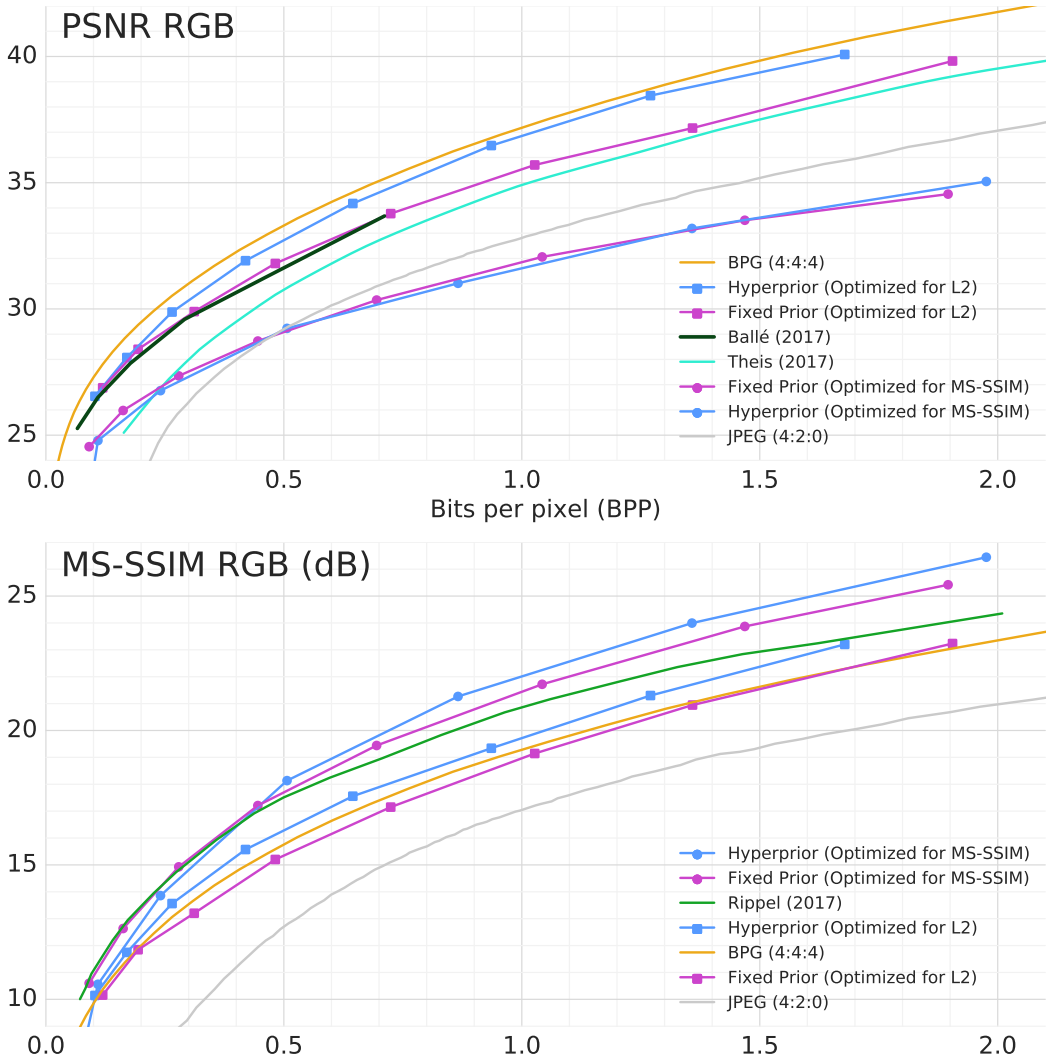

Figure 5: Rate–distortion curves aggregated over the Kodak dataset. For both the PSNR (top) and MS-SSIM (bottom) graphs, we see that matching the training loss to the evaluation metric provides a significant improvement. For PSNR, our hyperprior model trained on L2 loss outperforms all other machine learning methods for which we have evaluation results. For MS-SSIM, the hyperprior model outperforms all conventional codecs and exceeds Rippel and Bourdev (2017), the current state-of-the-art model for MS-SSIM, above 0.4 bpp. Please see Figures 8 and 9 in Appendix 7.3 for full-page RD curves that include a wider range of codecs and machine learning models.

at still image compression. When training our hyperprior model for squared error, we roughly match BPG performance, with better results at higher bitrates than lower ones, but still substantially outperforming all published machine learning based methods.

When measuring distortion using MS-SSIM (figure 5, bottom), conventional codecs such as JPEG and BPG end up at the lower end of the performance ranking. This is not surprising, since these methods have been optimized for squared error (with hand-selected constraints intended to ensure that squared error optimization doesn't go against visual quality). To the best of our knowledge, the state-of-the-art for compression performance in terms of MS-SSIM is Rippel and Bourdev (2017). Surprisingly, it is matched at low bitrates, and substantially surpassed at higher bitrates, by our fixed prior model, even though the model presented in Rippel and Bourdev (2017) is conceptually much more complex (due to its multiscale architecture, GAN loss, and context-adaptive entropy coder). The hyperprior model adds even further gains above 0.5 bpp, although it loses some at the lowest

Figure 6: The visual artifacts generated at low bit rates depend on the training loss. The top figure (0.17974 bpp, PSNR=27.46, MS-SSIM=0.9724) was generated by the hyperprior model using an MS-SSIM loss, while the bottom figure (0.17997 bpp, PSNR=31.77, MS-SSIM=0.9706) was trained using L2 loss.

bitrates. We believe that this may be caused by a suboptimal architecture choice for the low bitrate models (such as number of channels or layers).[2]

What is the qualitative difference of image reconstructions when optimizing for different distortion losses? When comparing images compressed to similar bit rates by models optimized with an MS-SSIM distortion loss compared to a squared loss (figure 6), we find that the overall fidelity in terms of how much detail is preserved appears similar. However, the spatial distribution of detail changes substantially. MS-SSIM, like its predecessor SSIM, is a metric designed to model contrast perception. Compared to a simple squared loss, it has the effect of attenuating the error in image regions with high contrast, and boosting the error in regions with low contrast, because the human visibility threshold varies with local contrast (Wang, Bovik, et al., 2004). The result is that the compression model trained for MS-SSIM assigns more detail to lower contrast regions, such as the grass, and removes detail from high contrast regions, such as the text on the side of the airplane. More frequently than expected, this is inconsistent with human expectations: because semantic relevance is often assigned to high-contrast areas (such as text, or salient objects), the squared-error optimized models produce subjectively better reconstructions in some cases. It is important to note that neither distortion metric is powerful enough to capture semantics, which makes the choice of distortion loss a difficult one.

Prior work on machine learning based image compression has shown that extending the transform coding concept from linear to nonlinear transforms fundamentally improves the qualitative nature of compression artifacts (Ballé et al., 2016; Theis et al., 2017; Johnston et al., 2017). It appears that nonlinear transforms with higher computational capacity adapt better to the statistics of natural images, imitating properties of the data distribution better than linear transforms even under severe rate limitations. When training a model with or without the hyperprior for equal values of λ, we find no changes to the qualitative nature of the artifacts. Rather, the hyperprior model simply tends to produce image reconstructions with improved detail and a lower bit rate than the corresponding model with a fixed prior (figure 5; refer to appendix 7.4 for visual results).

5 Related work

An early exploration of hierarchical generative models for compression of small images is found in Gregor et al. (2016). However, the aspect of quantization is not thoroughly considered, and hence, no actual compression method is designed. Theis et al. (2017) approaches the problem of generating gradient descent directions for quantization functions by replacing their (unhelpful) gradient with the identity function, and derives a differentiable upper bound for the discrete rate term. J. Ballé et al. (2016) replace the quantizer with additive uniform noise during training, and the discrete rate term with differential entropy. This method allows a formal connection to variational autoencoders; however, no bound for the rate approximation is given. Instead, Ballé et al. (2016) verify empirically that the discrete entropy cost matches the differentiable one.

Our variational model is perhaps most closely related to ladder VAEs (Sønderby et al., 2016). However, we choose different parametric forms to accommodate the approximation of the quantization and entropy coding process.

Wainwright and Eero P. Simoncelli, 2000 observe that linear filter responses (i.e., wavelet coefficients obtained by filtering an image) follow heavy-tailed marginal distributions, but can be represented as conditionally Gaussian when groups of neighboring coefficients are linked by a common scale multiplier. That is, the distributions of the filter responses can be modeled as gaussian scale mixtures. Our model can be seen as an extension where the filter responses are replaced with responses of a nonlinear transform, spatially local model is generalized to a global one, and an approximate inference model is added. Theis et al. (2017) use discrete Gaussian scale mixtures as a fully factored density model of the latent representation. Crucially, they model all elements as independent, not fully exploiting their statistical dependencies.

Santurkar et al. (2017) formulate their model in a hybrid VAE-GAN framework, adopting a stepwise training scheme where a decoder is first trained using an adversarial loss. It is then fixed and

[2]Note to reviewers: we will investigate the issue and update a revision of this paper with our findings, and potentially a fix.

a encoder is trained to minimize the reconstruction from decoder. However, the step-wise training breaks the end-to-end paradigm and may lead to sub-optimal models.

Baig and Torresani, 2017 propose a colorization based compression scheme where, given the luminance channel, color channels are predicted by making use of some model specific side information. The luminance channel is compressed using a traditional method. The proposed method exhibits significant color distortions at low bit rate and is limited by the compression method used for luminance for preserving details.

Among traditional approaches, JPEG (Wallace, 1991) is the most common lossy image compression method (Bull, 2014). However, more advanced standards have since been developed including JPEG2000, WebP (Google, 2010), and Better Portable Graphics (BPG) (Bellard, 2014).

6 DISCUSSION

We implement a variational image compression model, conceptually identical to the model presented in (Ballé et al., 2016), and augment it with a more powerful entropy model by introducing a hyperprior on its local scale parameters. The hyperprior is trained end-to-end with the rest of the model. This sets our work apart from other image compression models based on latent image representations:

In classical transform coding methods, compression researchers have exploited dependency in the latent variables (e.g., DCT or wavelet coefficients) by carefully hand-engineering entropy codes modeling the dependencies in the quantized regime (Taubman and Marcellin, 2002). This is a much more difficult problem than relying on an entropy coder that assumes independence between all the elements of $\hat{\boldsymbol{y}}$; transitioning to nonlinear transforms whose parameters are determined through training (and thus may be different for each re-training) only aggravates this problem.

Toderici et al. (2017) model images directly with a binarized latent representation, which technically removes the need for a separate entropy coding step. However, it corresponds to a very inflexible prior (a uniform prior on a binary representation, with no trainable parameters). The models compensate for this by introducing higher capacity transforms (e.g., based on recurrent networks). Johnston et al. (2017) improve this by designing an entropy coder in a way that adapts to the spatial probability structure of the latent representation. However, no feedback (in terms of gradients) is returned from the entropy coder back to the prior during training. This breaks the paradigm of end-to-end optimization, and may lead to suboptimal results.

Rippel and Bourdev (2017) impose a hand-designed energy function without trainable parameters as the prior for training the autoencoder. They also train an adaptive entropy coder without feedback to the autoencoder. The fact that our fixed prior model outperforms their method when optimized on the same metric, despite their state-of-the-art results, may point towards this disconnect.

Like all recent image compression methods based on machine learning, our method can be directly optimized for distortion losses that are more complex than simpler losses such as mean squared error. As one of the first studies in this emerging field, we examine the effect of optimizing for one of the most popular perceptual metrics, MS-SSIM, and compare it to optimizing for squared loss. Figure 6 demonstrates that the results can show significant variation in terms of visual quality, depending on image content, which implies that unless human rating experiments are conducted to provide more reliable data, it is wise to compare methods based on more than a single type of metric.

Our model, when trained on the appropriate loss, has the capacity to surpass the state of the art on MS-SSIM, but does not quite reach the performance of a heavily optimized traditional method such as BPG on PSNR (while outperforming all other methods based on machine learning). This discrepancy may indicate that methods based on machine learning have not yet reached the expressive power of traditional methods. As such, the introduction of a hyperprior – or, in traditional terms, side information – is an elegant way of introducing more powerful priors, and a big step in the right direction.

REFERENCES

Baig, Mohammad Haris and Lorenzo Torresani (2017). "Multiple hypothesis colorization and its application to image compression". In: *Computer Vision and Image Understanding*. URL: http://www.sciencedirect.com/science/article/pii/S1077314217300267.

Ballé, J, V Laparra, and E P Simoncelli (2016). "Density Modeling of Images using a Generalized Normalization Transformation". In: *Int'l. Conf. on Learning Representations (ICLR2016)*. URL: https://arxiv.org/abs/1511.06281.

Ballé, Valero Laparra, and Eero P. Simoncelli (2015). "Density Modeling of Images Using a Generalized Normalization Transformation". In: *arXiv e-prints*. Published as a conference paper at the 4th International Conference for Learning Representations, San Juan, 2016. arXiv: 1511.06281.

– (2016). "End-to-end Optimized Image Compression". In: *arXiv e-prints*. 5th Int. Conf. for Learning Representations.

Bellard, F. (2014). *BPG Image Format (http://bellard.org/bpg/)*. Accessed: 2017-01-30. URL: http://bellard.org/bpg/.

Bishop, Christopher M (1998). "Latent variable models". In: *Learning in graphical models*. Springer, pp. 371–403.

Bross, Benjamin et al. (2012). *High Efficiency Video Coding (HEVC) text specification draft 6*. JCTVC-H1003.

Bull, David (2014). "Communicating Pictures". In: *Communicating Pictures*. Academic Press. ISBN: 978-0-12-405906-1. DOI: http://dx.doi.org/10.1016/B978-0-12-405906-1.00014-3.

Google (2010). *WebP: Compression Techniques (http://developers. google.com/speed/webp/docs/-compression)*. Accessed: 2017-01-30. URL: http://developers.google.com/speed/webp/docs/compression.

Gregor, Karol et al. (2016). "Towards Conceptual Compression". In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee et al. Curran Associates, Inc., pp. 3549–3557.

Johnston, Nick et al. (2017). "Improved Lossy Image Compression with Priming and Spatially Adaptive Bit Rates for Recurrent Networks". In: *arXiv preprint arXiv:1703.10114*.

Kingma, D. P. and J. Ba (2014). "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980. URL: http://arxiv.org/abs/1413.6980.

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

Kodak, Eastman (n.d.). *Kodak Lossless True Color Image Suite (PhotoCD PCD0992)*. URL: http://r0k.us/graphics/kodak/.

Minnen, David et al. (2017). "SPATIALLY ADAPTIVE IMAGE COMPRESSION USING A TILED DEEP NETWORK". In: *IEEE International Conference on Image Processing*, pp. 2796–2800.

Rippel, Oren and Lubomir Bourdev (2017). "Real-Time Adaptive Image Compression". In: *The 34th International Conference on Machine Learning*.

Santurkar, Shibani, David Budden, and Nir Shavit (2017). "Generative Compression". In: *arXiv:1703.01467*. URL: https://arxiv.org/abs/1703.01467.

Sønderby, Casper Kaae et al. (2016). "Ladder variational autoencoders". In: *Advances in Neural Information Processing Systems*, pp. 3738–3746.

Taubman, David S. and Michael W. Marcellin (2002). *JPEG 2000 – Image Compression Fundamentals, Standards and Practice*.

Theis, L. et al. (2017). "Lossy Image Compression with Compressive Autoencoders". In: *Int'l. Conf. on Learning Representations*.

Toderici, George et al. (2017). "Full Resolution Image Compression with Recurrent Neural Networks". In: *CVPR* abs/1608.05148. URL: http://arxiv.org/abs/1608.05148.

Wainwright, Martin J. and Eero P. Simoncelli (2000). "Scale Mixtures of Gaussians and the Statistics of Natural Images". In: *Advances in Neural Information Processing Systems 12*, pp. 855–861.

Wallace, Gregory K. (1991). "The JPEG still picture compression standard". In: *Communications of the ACM*, pp. 30–44.

Wang, Zhou, Alan Conrad Bovik, et al. (2004). "Image Quality Assessment: From Error Visibility to Structural Similarity". In: 13.4. DOI: 10.1109/TIP.2003.819861.

Wang, Zhou, Eero P Simoncelli, and Alan C Bovik (2003). "Multiscale structural similarity for image quality assessment". In: *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Seventh Asilomar Conference on*. Vol. 2. IEEE, pp. 1398–1402.

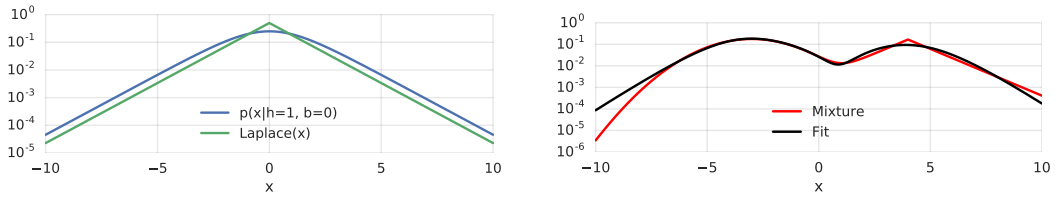

Figure 7: A visualization of the non-parametric model on log scale. Left compares the special case of $N = 1$ (eq. 23) with Laplacian distribution. Note the similar heavy tail behavior with a "smoothed" mode. Right shows a fit of the model (with $N = 2$) to a ground truth Gaussian+Laplacian mixture density. The non-parametric model is able to produce a reasonable approximation for the ground truth density. For details see section 7.1.

7 Appendix

7.1 Fully factorized, non-parametric density model

J. Ballé et al. (2016) use a non-parametric piecewise linear model to capture the marginals of the prior. By increasing the number of samples per unit interval, this can in principle be used to model any density with arbitrary precision. However, it has two practical issues: The range of non-zero probability must be finite and known ahead of time, and its implementation is non-trivial with existing automatic differentiation frameworks, both due to numerical issues with normalization and the fact that it typically relies on discrete operations such as array indexing. For the compression models presented in this paper, we instead use the following model based on the cumulative.

We define a density $p : \mathbb{R} \to \mathbb{R}^+$ using its cumulative $c : \mathbb{R} \to [0, 1]$ by satisfying the following constraints:

$$c(-\infty) = 0; \quad c(\infty) = 1; \quad p(x) = \frac{\partial c(x)}{\partial x} \geq 0 \tag{9}$$

Note that the monotonicity constraint of the cumulative is established by requiring the density function p to be non-negative. Suppose the cumulative is a composition of functions. Then the density can be written using the chain rule of calculus:

$$c = f_N \circ f_{N-1} \cdots f_1 \tag{10}$$

$$p = f'_N \cdot f'_{N-1} \cdots f'_1 \tag{11}$$

where we write the derivative of f_n as f'_n. We'll allow the f_n to be vector functions:

$$f_n : \mathbb{R}^{d_n} \to \mathbb{R}^{r_n} \tag{12}$$

In general, the f'_n are Jacobian matrices, and the dots are matrix multiplications. To ensure $p(x)$ is univariate, the domain of f_1 and the range of f_N need to be one dimensional ($d_1 = r_N = 1$).

To guarantee that $p(x)$ is a density, we just need f_N to map to the range between 0 and 1, and ensure that $p(x) \geq 0$. To do that, let us require all the Jacobian elements to be non-negative:

$$\forall i, j, n : [f'_n]_{ij} \geq 0 \tag{13}$$

Then the matrix product computing $p(x)$ is non-negative as well, and we have defined a valid density.

An effective choice of f_n is the following (as a shorthand, we define $\tanh$, $\mathrm{sigmoid}$, and $\mathrm{softplus}$ as elementwise functions, when applied to vectors or matrices):

$$f_n(\boldsymbol{x}) = g_n\big(\boldsymbol{H}^{(n)}\boldsymbol{x} + \boldsymbol{b}^{(n)}\big) \qquad 1 \leq n < N \tag{14}$$

$$f_N(\boldsymbol{x}) = \mathrm{sigmoid}\big(\boldsymbol{H}^{(N)}\boldsymbol{x} + \boldsymbol{b}^{(N)}\big) \tag{15}$$

where $\boldsymbol{H}^{(n)}$ are matrices, $\boldsymbol{b}^{(n)}$ are vectors, and g_n are nonlinearities defined as

$$g_n(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{a}^{(n)} \odot \tanh(\boldsymbol{x}) \tag{16}$$

where $\boldsymbol{a}^{(n)}$ is a vector and $\odot$ denotes elementwise multiplication. The rationale behind this particular nonlinearity is that it allows to expand or contract the space near $x = 0$. $\boldsymbol{a}^{(n)}$ controls the rate

of expansion (when positive) or contraction (when negative). If $\boldsymbol{a}^{(n)}$ were fixed to a positive value, "peaks" in the density would become easier to model than "troughs".

The derivatives work out as follows:

$$f_n'(\boldsymbol{x}) = \operatorname{diag} g_n'\big(\boldsymbol{H}^{(n)}\boldsymbol{x} + \boldsymbol{b}^{(n)}\big) \cdot \boldsymbol{H}^{(n)} \qquad 1 \leq n < N, \text{with} \tag{17}$$

$$g_n'(\boldsymbol{x}) = 1 + \boldsymbol{a}^{(n)} \odot \tanh'(\boldsymbol{x}) \qquad \text{and} \tag{18}$$

$$f_N'(\boldsymbol{x}) = \operatorname{sigmoid}'\big(\boldsymbol{H}^{(N)}\boldsymbol{x} + \boldsymbol{b}^{(N)}\big) \cdot \boldsymbol{H}^{(N)} \tag{19}$$

For the derivatives to be non-negative, we need to constrain $\boldsymbol{H}^{(n)}$ to have all non-negative elements, and the elements of $\boldsymbol{a}^{(n)}$ to be lower bounded by -1. This is easily done by reparameterization:

$$\boldsymbol{H}^{(n)} = \operatorname{softplus}\big(\hat{\boldsymbol{H}}^{(n)}\big) \tag{20}$$

$$\boldsymbol{a}^{(n)} = \tanh\big(\hat{\boldsymbol{a}}^{(n)}\big) \tag{21}$$

where the quantities with the hat are the actual parameters.

In the special case that $N = 1$, we obtain an interesting special case:

$$c(x) = \operatorname{sigmoid}\big(hx + b\big) \tag{22}$$

$$p(x) = \frac{h}{2} \cdot \frac{1}{1 + \cosh(hx + b)} \tag{23}$$

$$\log p(x) = -h \operatorname{softplus}(hx + b) - h \operatorname{softplus}(-hx - b) \tag{24}$$

which is a density function similar to a "smoothed" Laplacian, with identical tail behavior. A plot of this density function, as well as a plot of a fit of this model to a "toy" mixture density is provided in figure 7.

It may seem odd to define a density function as an explicit derivative; however, in an automatic differentiation framework, this operation is very easy to implement, and the resulting density function is normalized by construction. We have found the model to fit well to arbitrary densities, and perform just as well as the piecewise linear model in the context of compression models.

For all experiments in this paper, we used $N = 4$, with the dimensionalities $r_1 = r_2 = r_3 = 3$. Each marginal distribution model is associated with its own set of parameters $\boldsymbol{a}^{(n)}$, $\boldsymbol{b}^{(n)}$, $\boldsymbol{H}^{(n)}$ (which, together, form $\boldsymbol{\psi}^{(i)}$).

7.2 Modeling priors with added uniform noise

We model both the prior $p_{\tilde{\boldsymbol{y}}|\tilde{\boldsymbol{z}}}$ and the hyperprior $p_{\tilde{\boldsymbol{z}}}$ using densities that are convolved with a standard uniform density function. This is to make sure that the priors have enough flexibility to match the variational posterior (q). To see this, consider the following special case:

The inference model may effectively "disable" part of the representation by collapsing some of its dimensions. When this happens, the marginal distribution of individual elements of $\tilde{\boldsymbol{y}}$ or $\tilde{\boldsymbol{z}}$ will collapse to a standard uniform density due to the added uniform noise. To correctly capture the expected entropy cost of this distribution (zero – the quantized version of the element will have a deterministic value independent of the image), the prior or hyperprior needs to be able to model a uniform distribution, or it will assign a cross entropy to these elements that is non-zero, which may lead to suboptimal results.

Due to its infinitely steep edges, the uniform distribution is a corner case for not only the Gaussian density model, but also the non-parametric marginal model described in appendix 7.1. To fix this, we incorporate the added noise directly into the prior/hyperprior by convolving the underlying density model p with a standard uniform:

$$\begin{aligned}\tilde{p}(\tilde{y}) &= \Big(p * \mathcal{U}\big(-\tfrac{1}{2}, \tfrac{1}{2}\big)\Big)(\tilde{y}) \\ &= \int_{-\infty}^{\infty} p(y)\, \mathcal{U}\big(\tilde{y} - y| - \tfrac{1}{2}, \tfrac{1}{2}\big)\, \mathrm{d}y\end{aligned}$$

$$
\begin{aligned}
&= \int_{\tilde{y}-\frac{1}{2}}^{\tilde{y}+\frac{1}{2}} p(y) \, \mathrm{d}y \\
&= c\big(\tilde{y} + \tfrac{1}{2}\big) - c\big(\tilde{y} - \tfrac{1}{2}\big),
\end{aligned} \tag{25}
$$

where c is the cumulative of the underlying density model.

Since the non-parametric model is defined via its cumulative, and the cumulative of a Gaussian is available in most computational frameworks, this is easy to implement in practice.

7.3 PERFORMANCE COMPARISONS FOR THE KODAK IMAGE SET

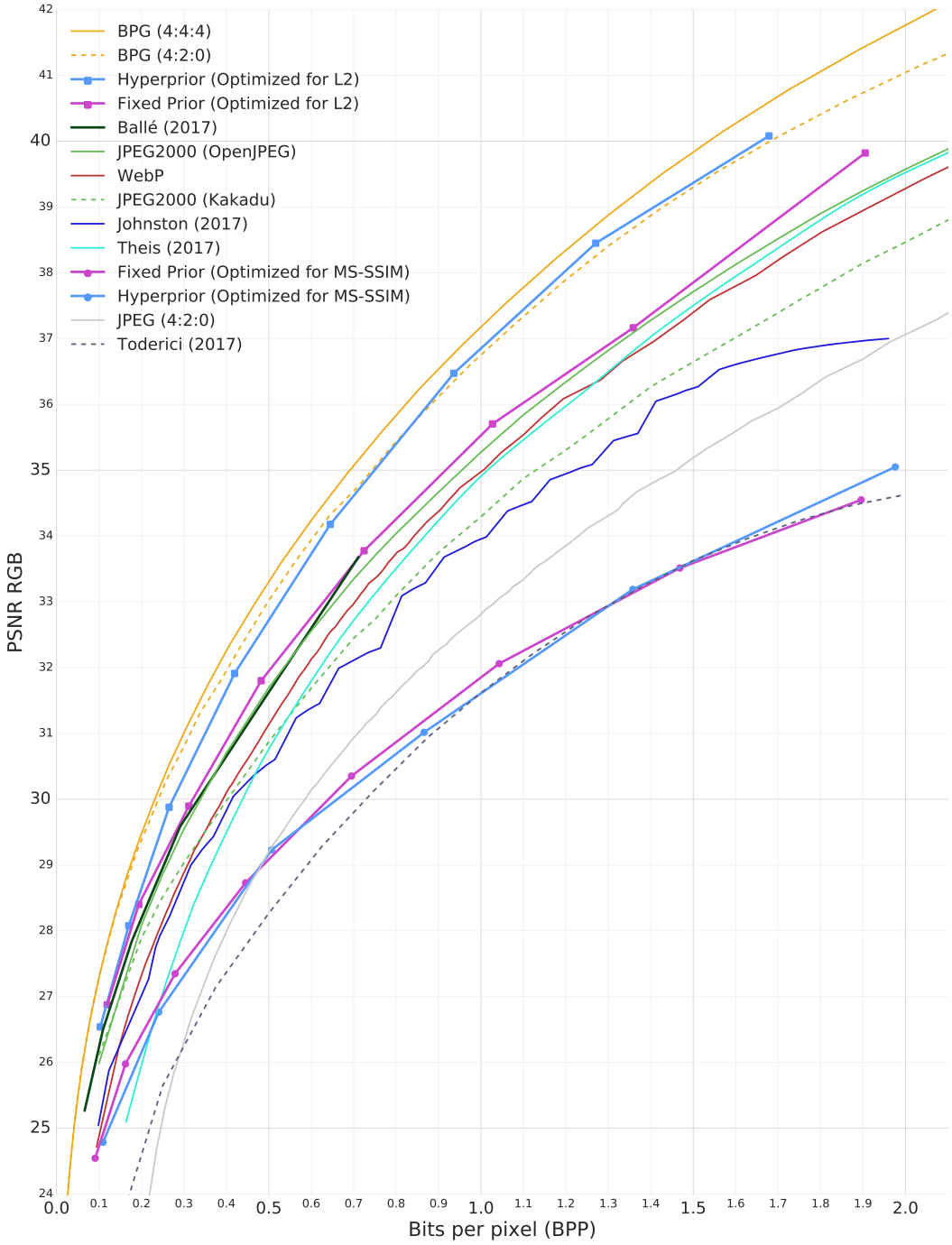

Figure 8: Rate-distortion curves for PSNR covering a wide range of conventional codecs and machine learning methods. We see that our hyperprior model (blue circles) outperforms most conventional codecs (JPEG, JPEG2000, and WebP) as well as all machine learning methods by a wide margin. As expected, we also see a significant advantage for our models on PSNR when trained on L2 (squares) compared to MS-SSIM (circles).

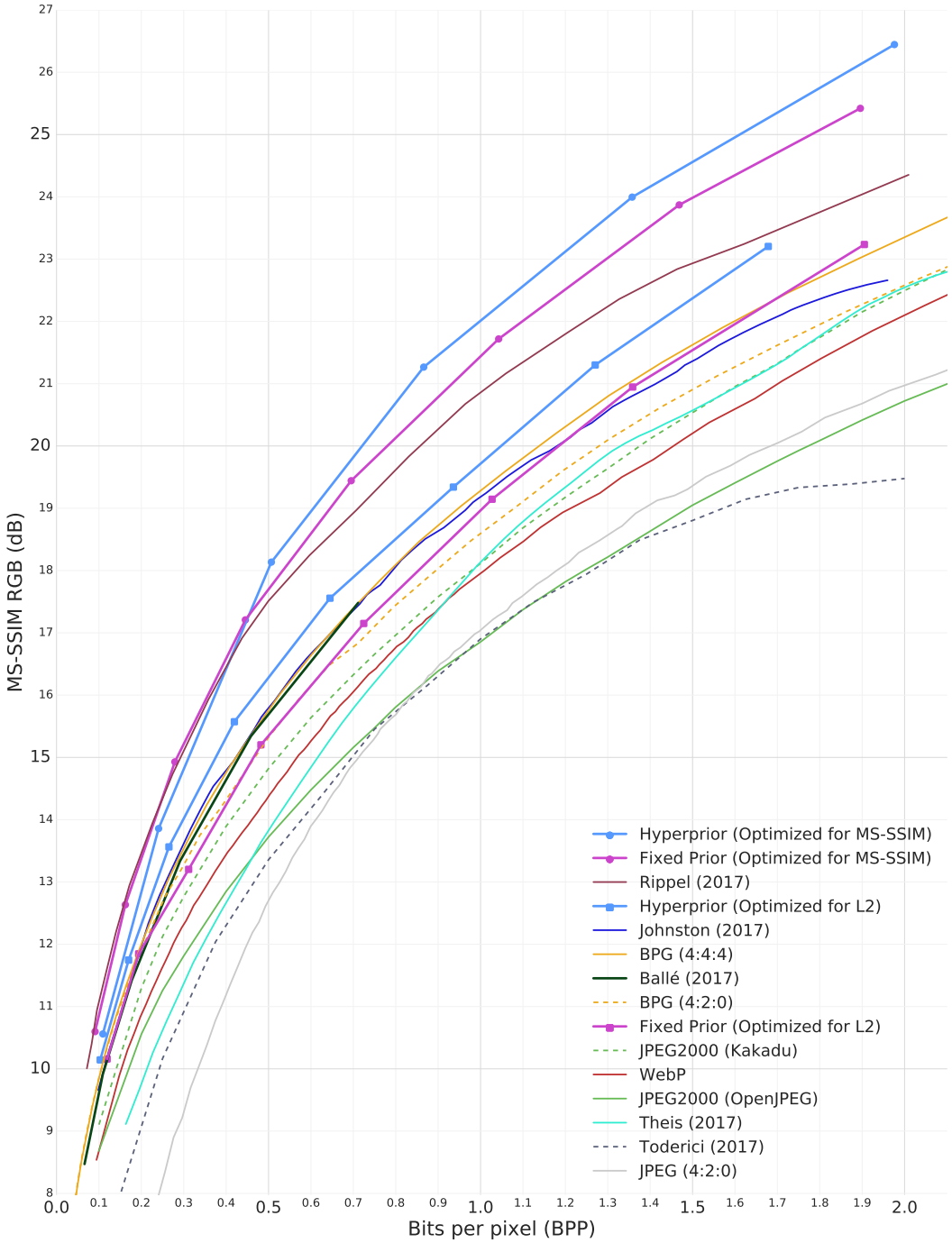

Figure 9: Rate-distortion curves for MS-SSIM covering a wide range of conventional codecs and machine learning methods. When trained on MS-SSIM, our models (both fixed prior and hyperprior) outperform Rippel and Bourdev (2017) above 0.4 bpp as well as all conventional and other machine learning methods at all bit rates. Furthermore, even when trained using L2 loss, our hyperprior model (blue squares) yields higher MS-SSIM scores than all of the conventional codecs.

7.4 PERFORMANCE COMPARISONS FOR INDIVIDUAL KODAK IMAGES

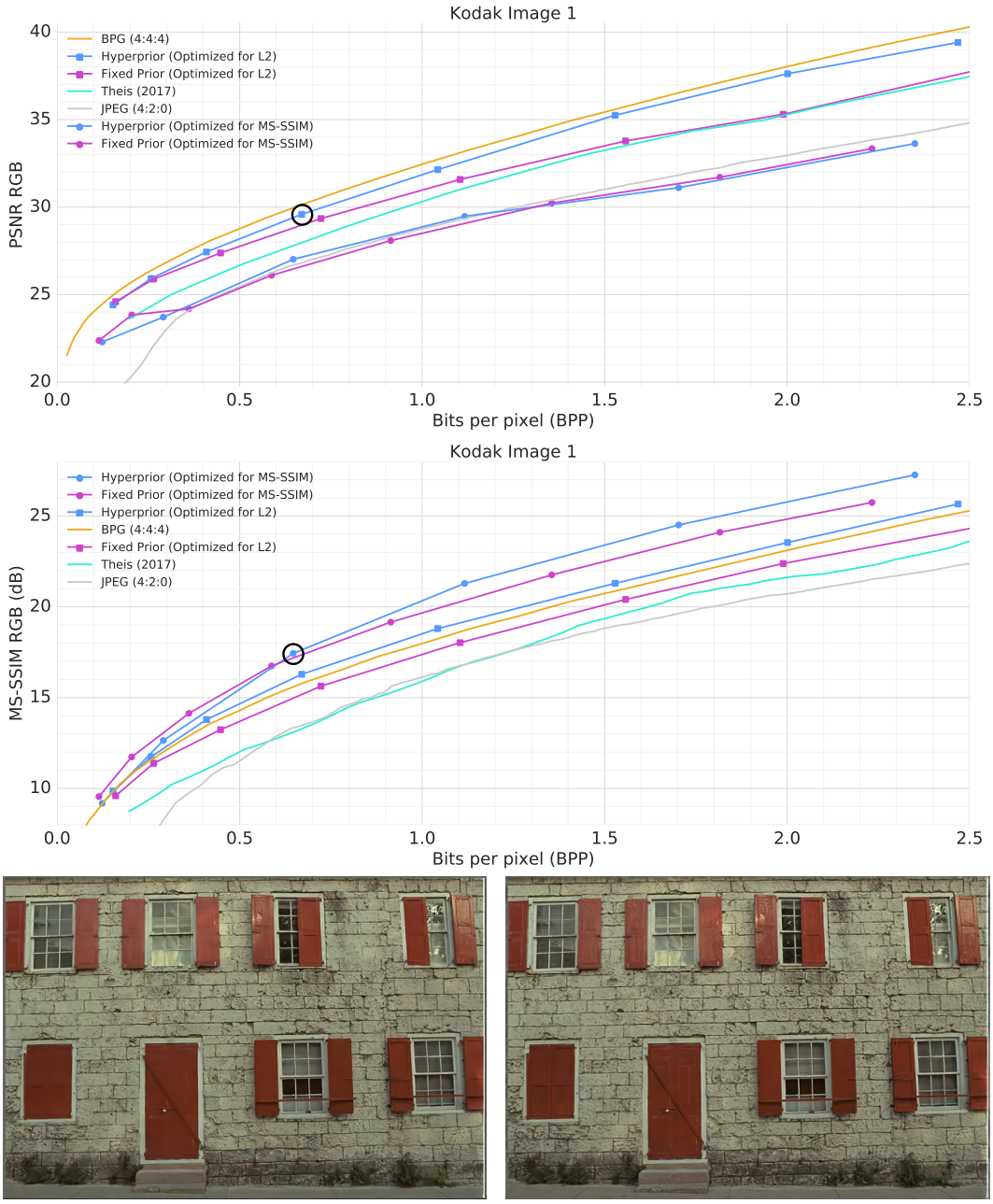

Figure 10: Results for Kodak image 01: PSNR and MS-SSIM rate-distortion curves (top), and example reconstructions for the hyperprior model optimized for L2 (bottom-left) and MS-SSIM (bottom-right). Best viewed on a computer screen. Images correspond to circled points in the plots.

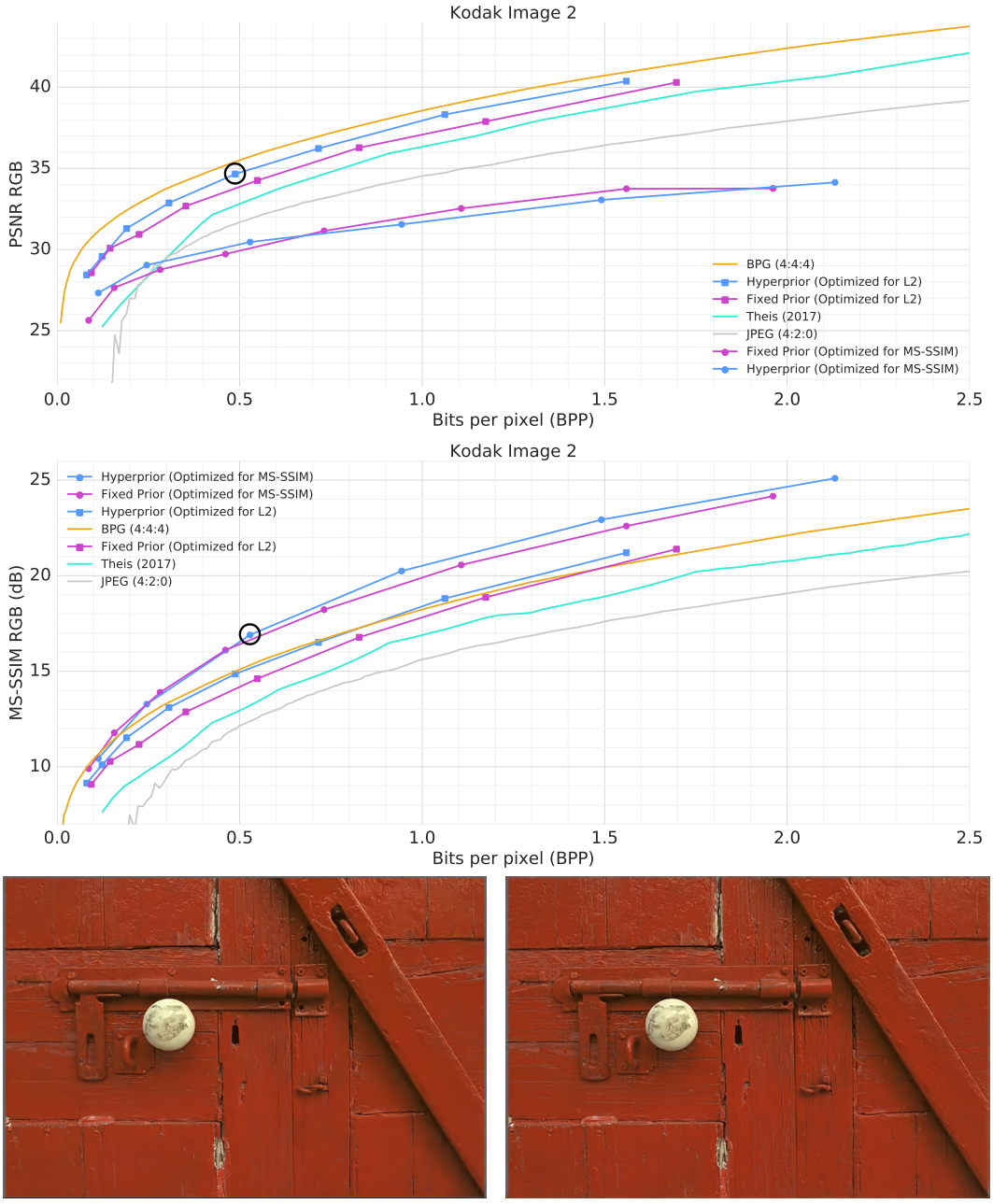

Figure 11: Results for Kodak image 02: PSNR and MS-SSIM rate-distortion curves (top), and example reconstructions for the hyperprior model optimized for L2 (bottom-left) and MS-SSIM (bottom-right). Best viewed on a computer screen. Images correspond to circled points in the plots.

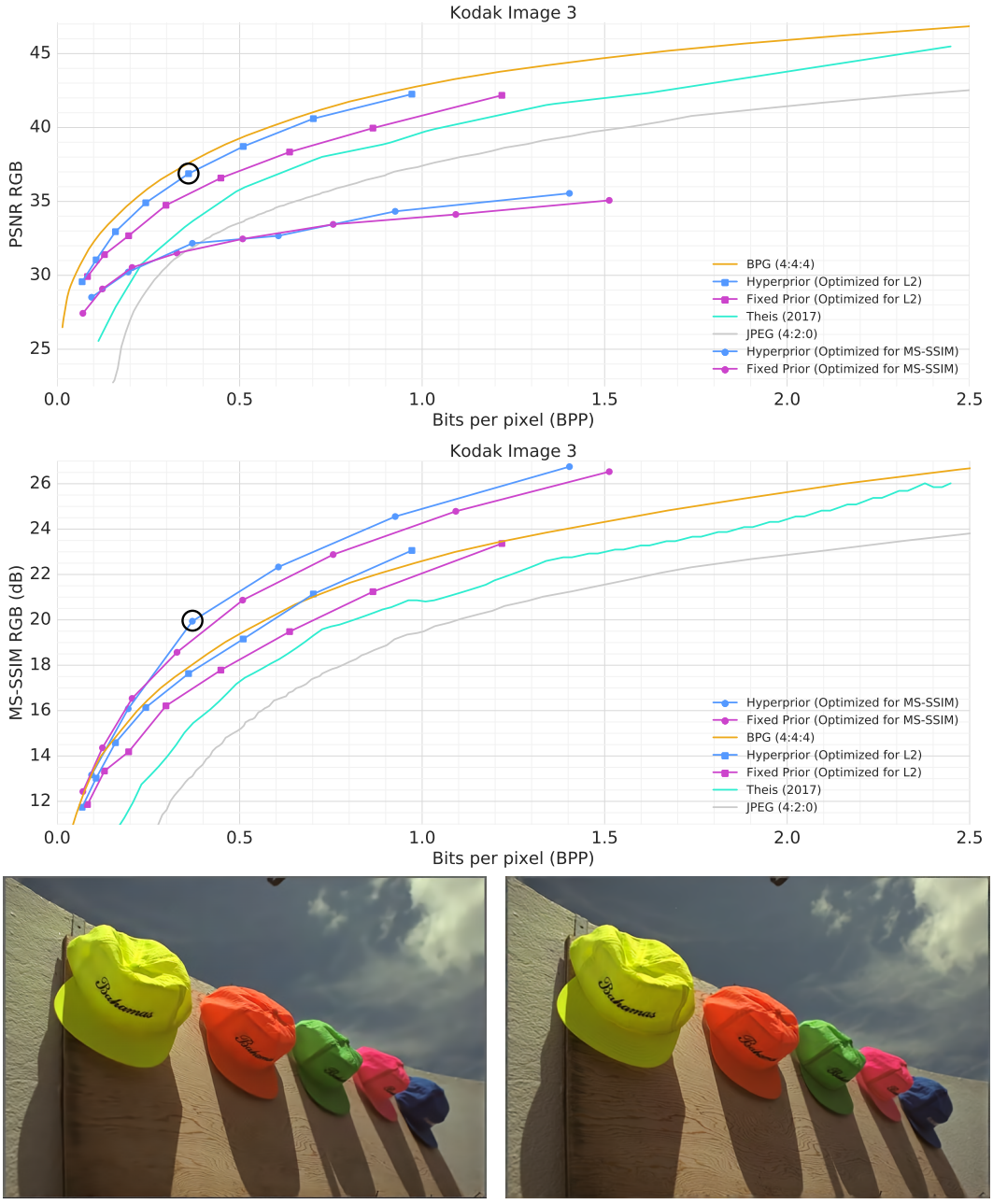

Figure 12: Results for Kodak image 03: PSNR and MS-SSIM rate-distortion curves (top), and example reconstructions for the hyperprior model optimized for L2 (bottom-left) and MS-SSIM (bottom-right). Best viewed on a computer screen. Images correspond to circled points in the plots.

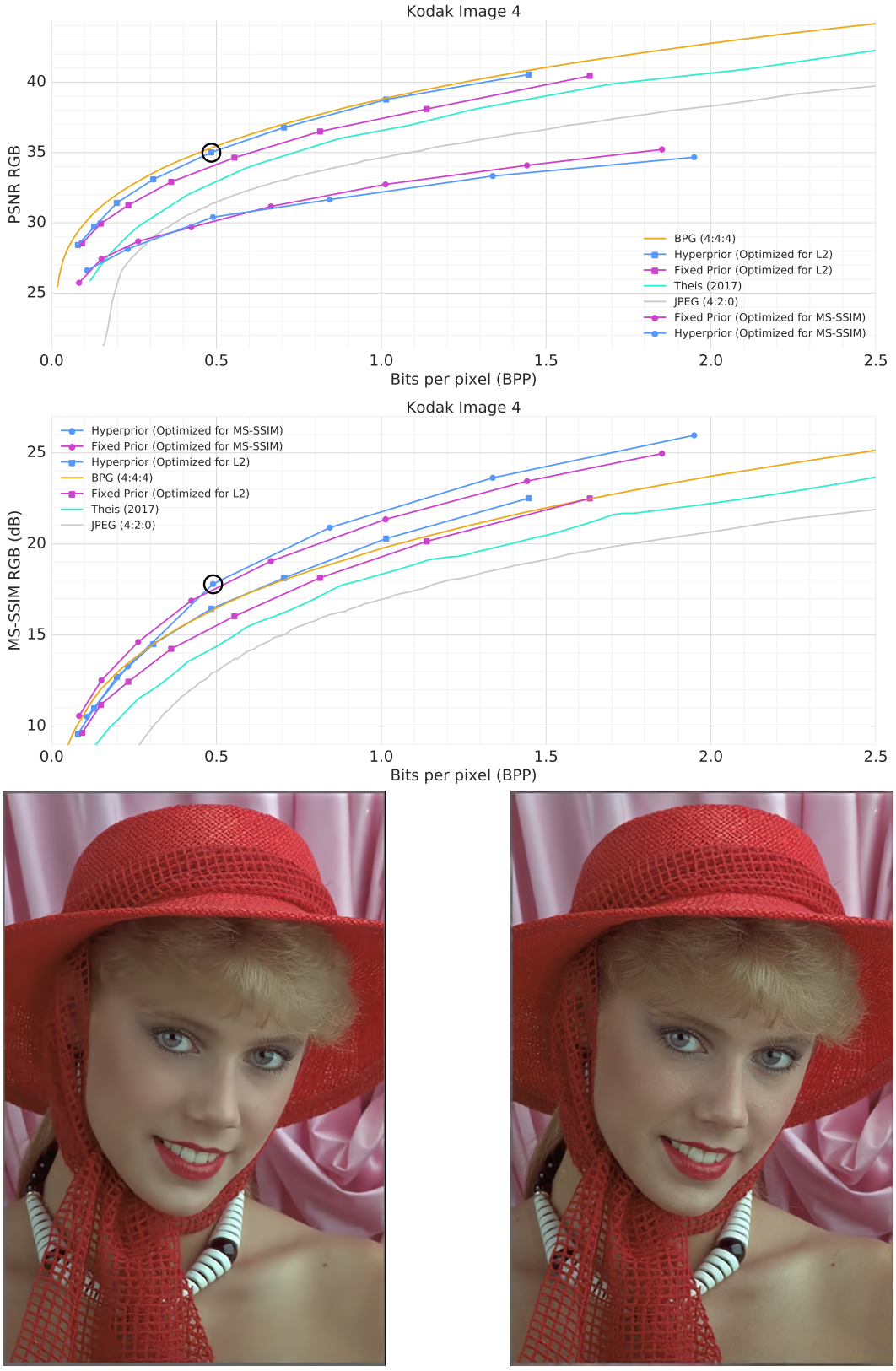

Figure 13: Results for Kodak image 04: PSNR and MS-SSIM rate-distortion curves (top), and example reconstructions for the hyperprior model optimized for L2 (bottom-left) and MS-SSIM (bottom-right). Best viewed on a computer screen. Images correspond to circled points in the plots.

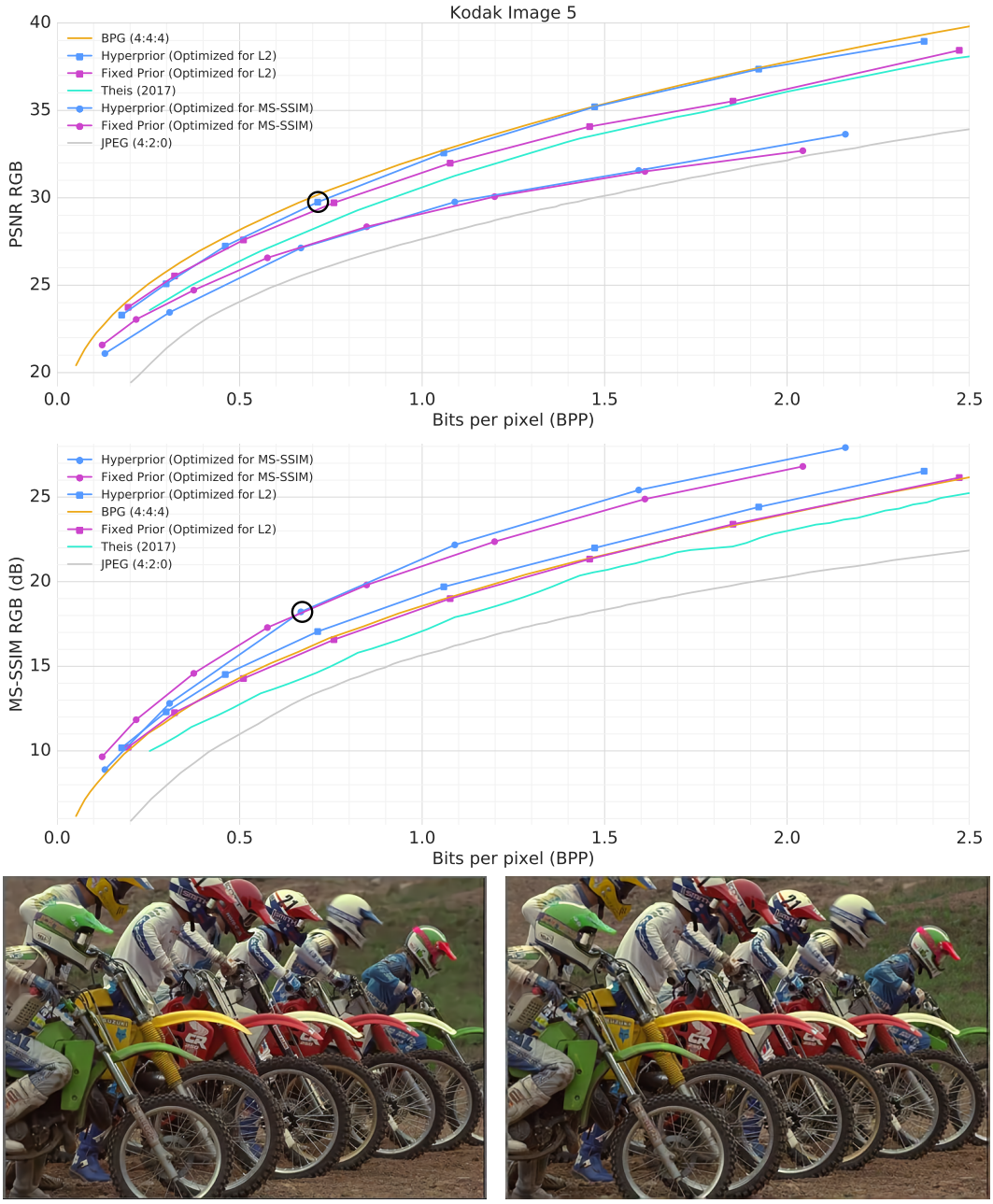

Figure 14: Results for Kodak image 05: PSNR and MS-SSIM rate-distortion curves (top), and example reconstructions for the hyperprior model optimized for L2 (bottom-left) and MS-SSIM (bottom-right). Best viewed on a computer screen. Images correspond to circled points in the plots.

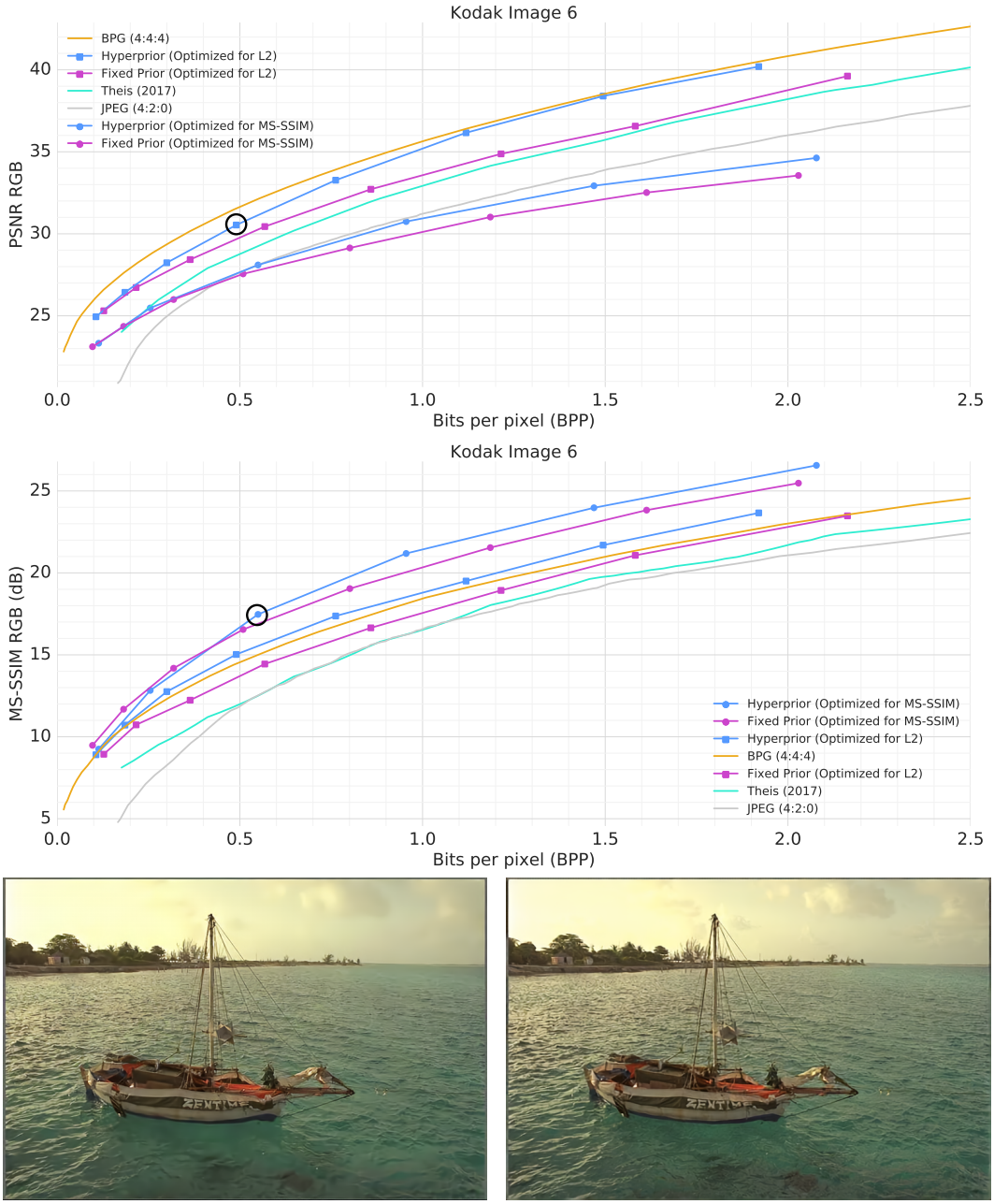

Figure 15: Results for Kodak image 06: PSNR and MS-SSIM rate-distortion curves (top), and example reconstructions for the hyperprior model optimized for L2 (bottom-left) and MS-SSIM (bottom-right). Best viewed on a computer screen. Images correspond to circled points in the plots.

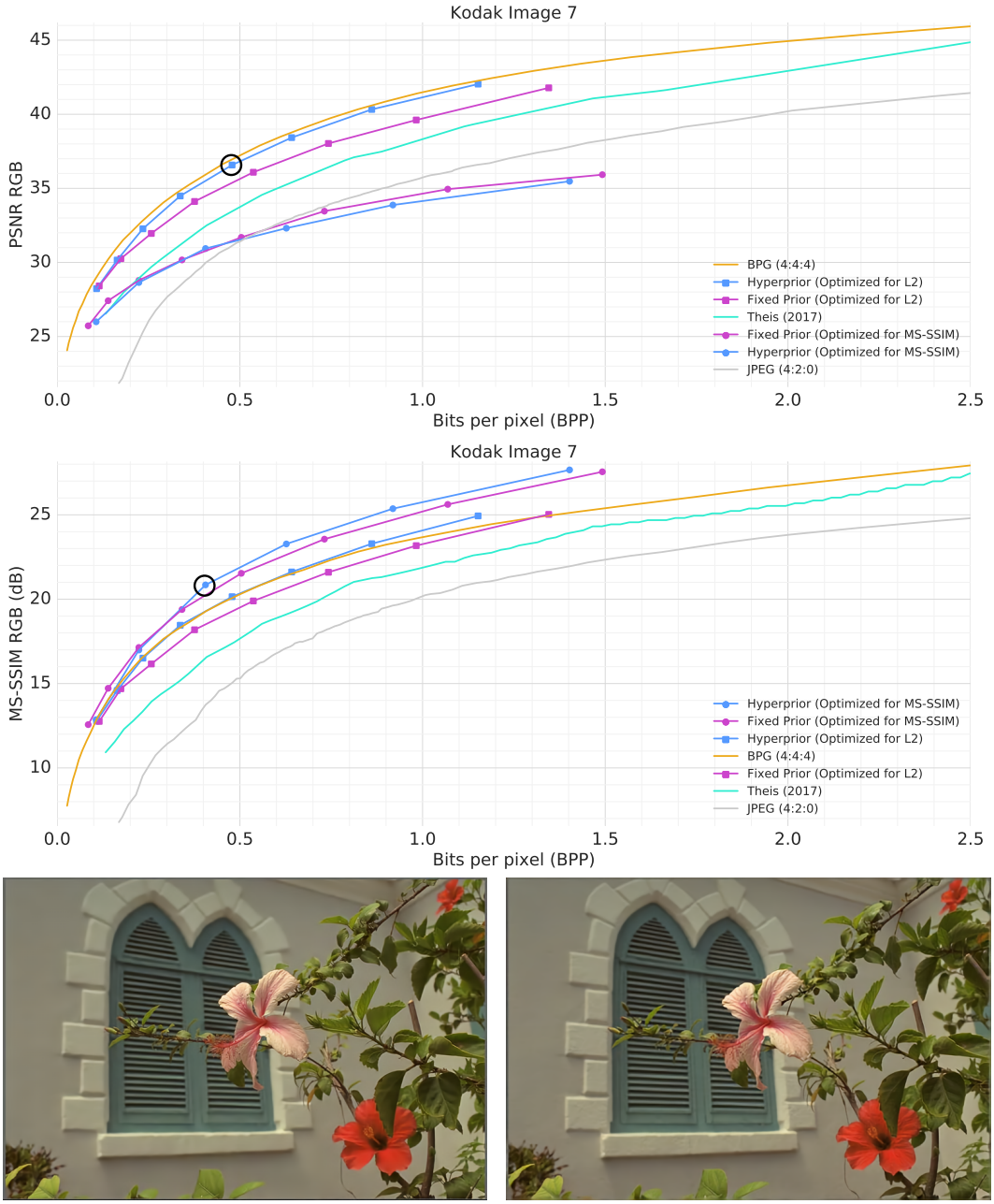

Figure 16: Results for Kodak image 07: PSNR and MS-SSIM rate-distortion curves (top), and example reconstructions for the hyperprior model optimized for L2 (bottom-left) and MS-SSIM (bottom-right). Best viewed on a computer screen. Images correspond to circled points in the plots.

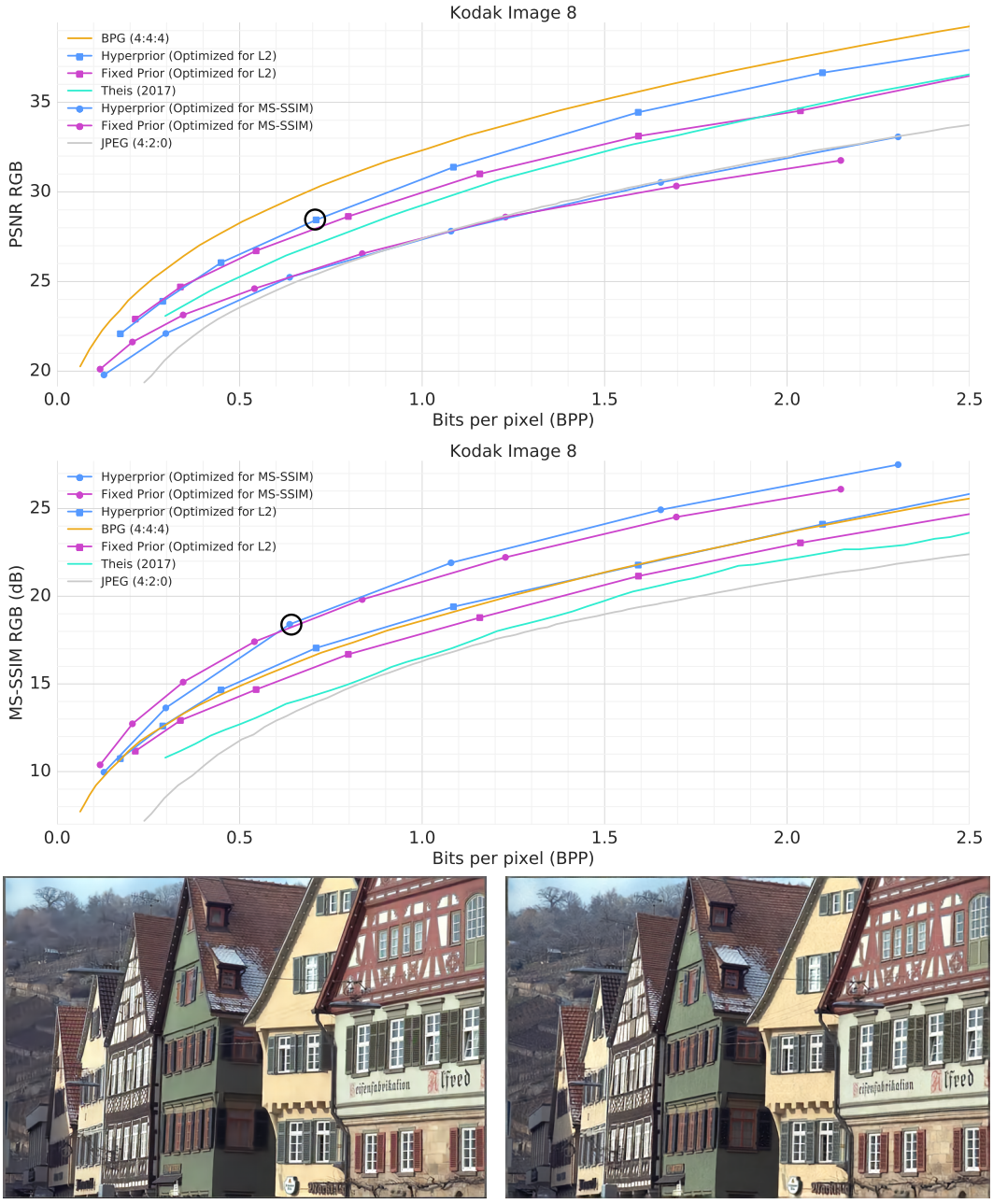

Figure 17: Results for Kodak image 08: PSNR and MS-SSIM rate-distortion curves (top), and example reconstructions for the hyperprior model optimized for L2 (bottom-left) and MS-SSIM (bottom-right). Best viewed on a computer screen. Images correspond to circled points in the plots.

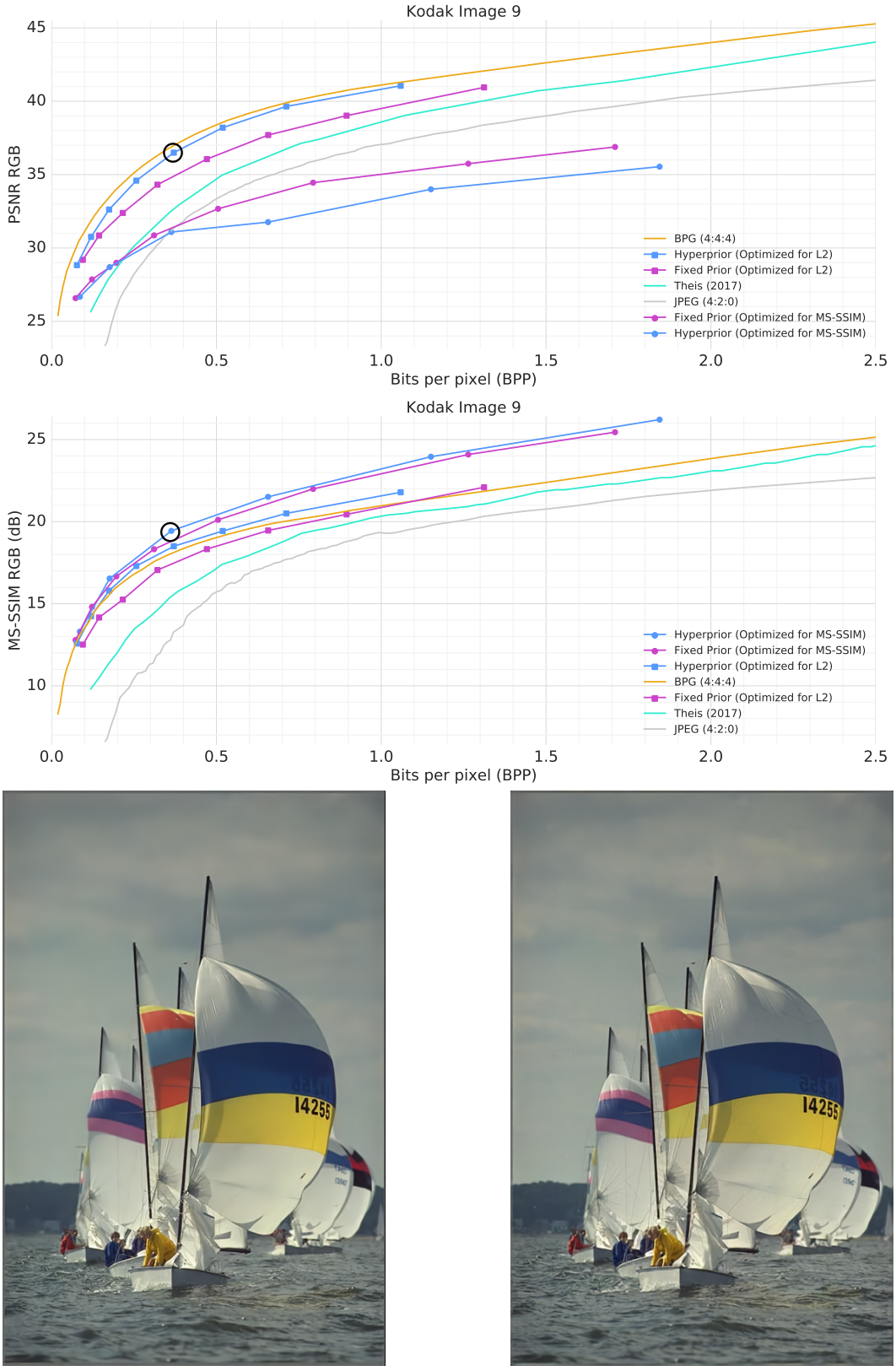

Figure 18: Results for Kodak image 09: PSNR and MS-SSIM rate-distortion curves (top), and example reconstructions for the hyperprior model optimized for L2 (bottom-left) and MS-SSIM (bottom-right). Best viewed on a computer screen. Images correspond to circled points in the plots.

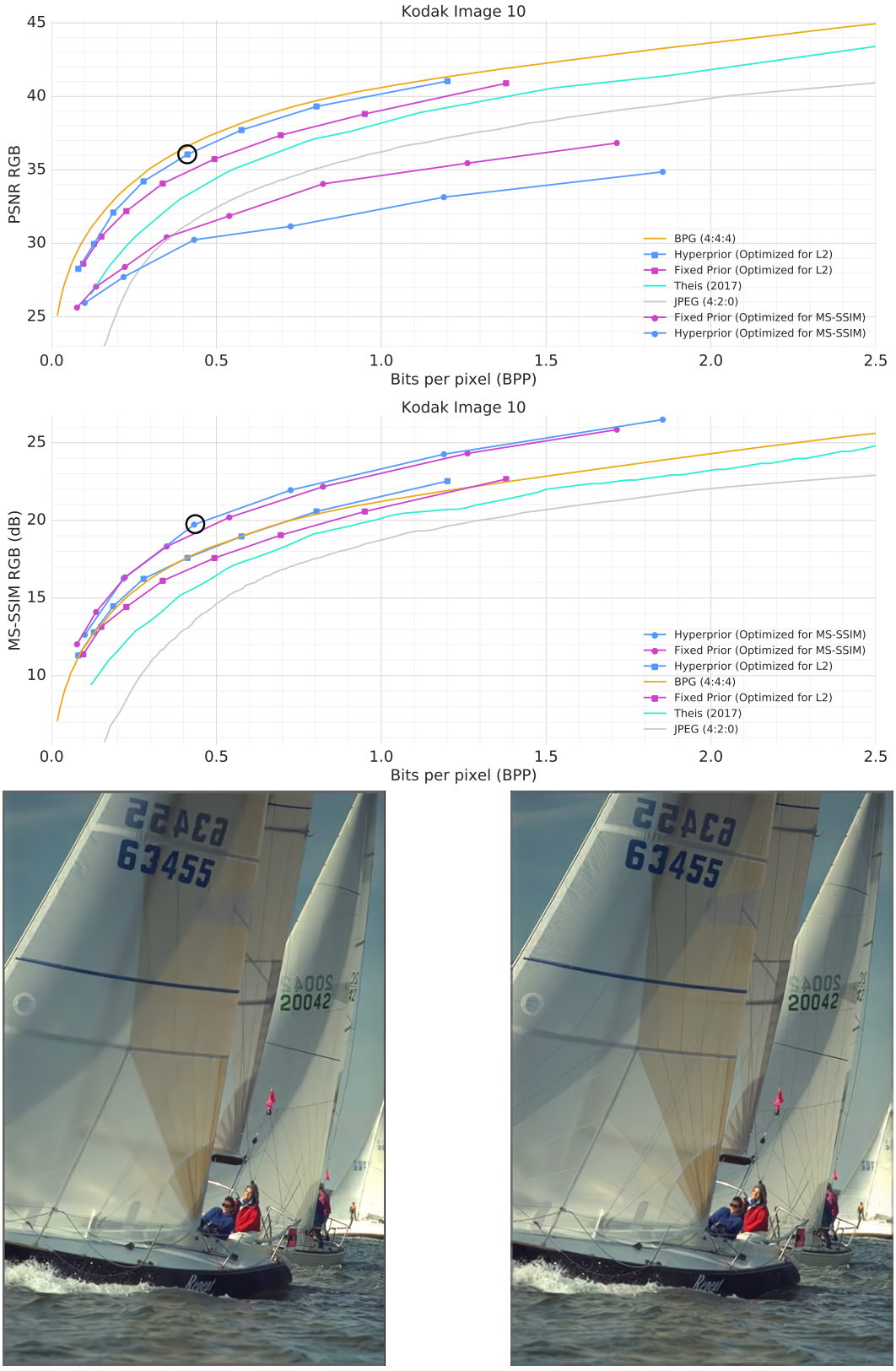

Figure 19: Results for Kodak image 10: PSNR and MS-SSIM rate-distortion curves (top), and example reconstructions for the hyperprior model optimized for L2 (bottom-left) and MS-SSIM (bottom-right). Best viewed on a computer screen. Images correspond to circled points in the plots.

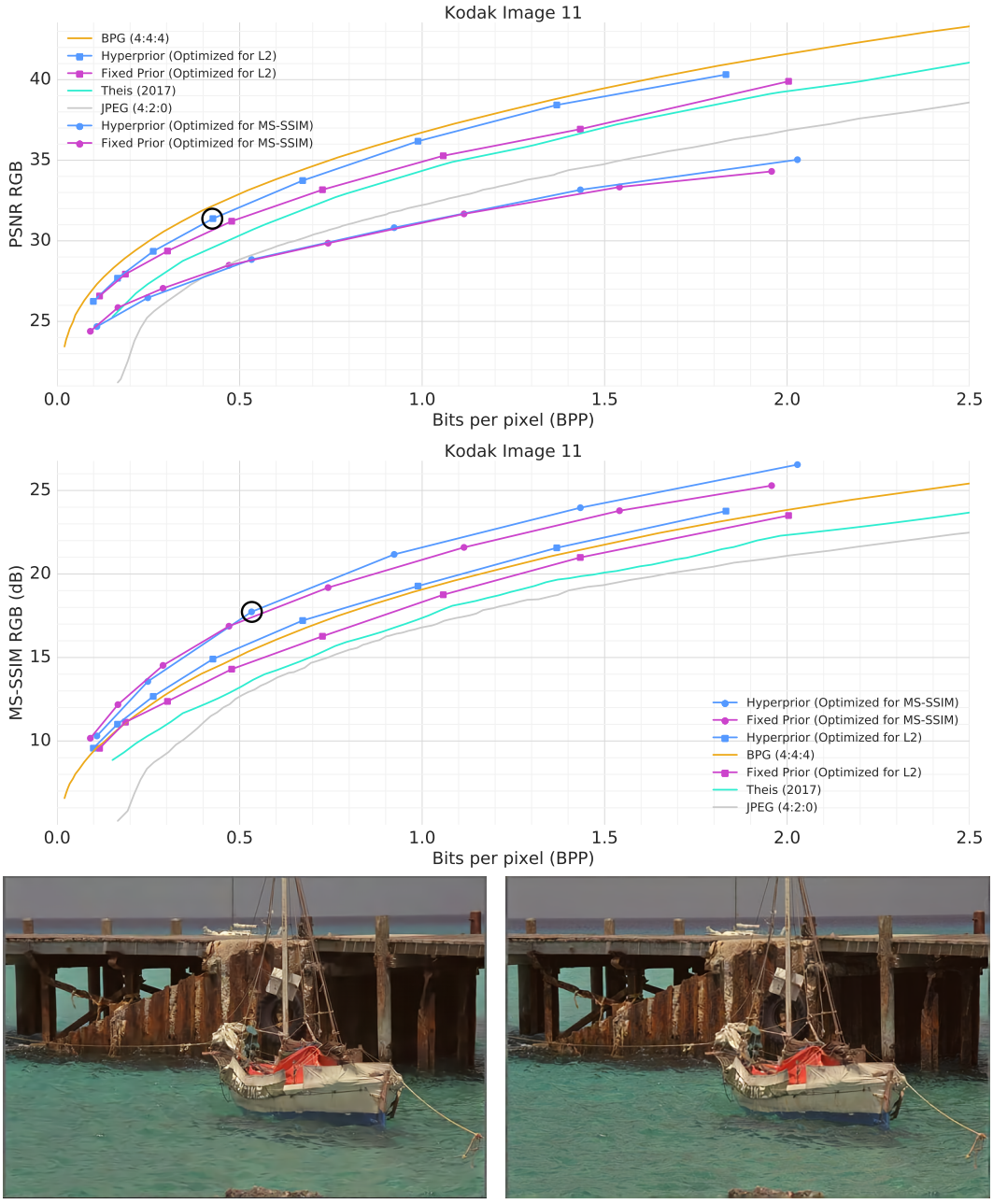

Figure 20: Results for Kodak image 11: PSNR and MS-SSIM rate-distortion curves (top), and example reconstructions for the hyperprior model optimized for L2 (bottom-left) and MS-SSIM (bottom-right). Best viewed on a computer screen. Images correspond to circled points in the plots.

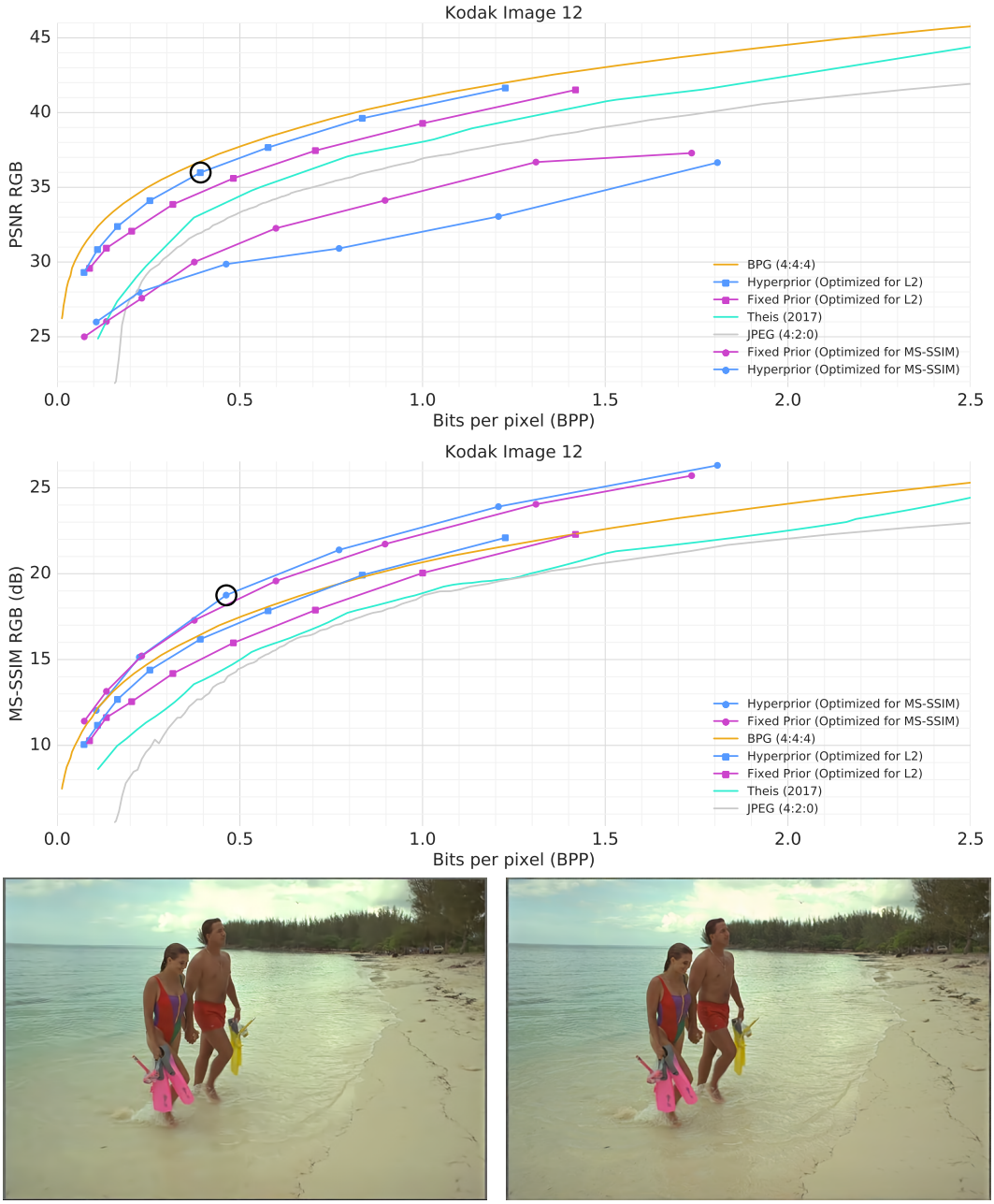

Figure 21: Results for Kodak image 12: PSNR and MS-SSIM rate-distortion curves (top), and example reconstructions for the hyperprior model optimized for L2 (bottom-left) and MS-SSIM (bottom-right). Best viewed on a computer screen. Images correspond to circled points in the plots.

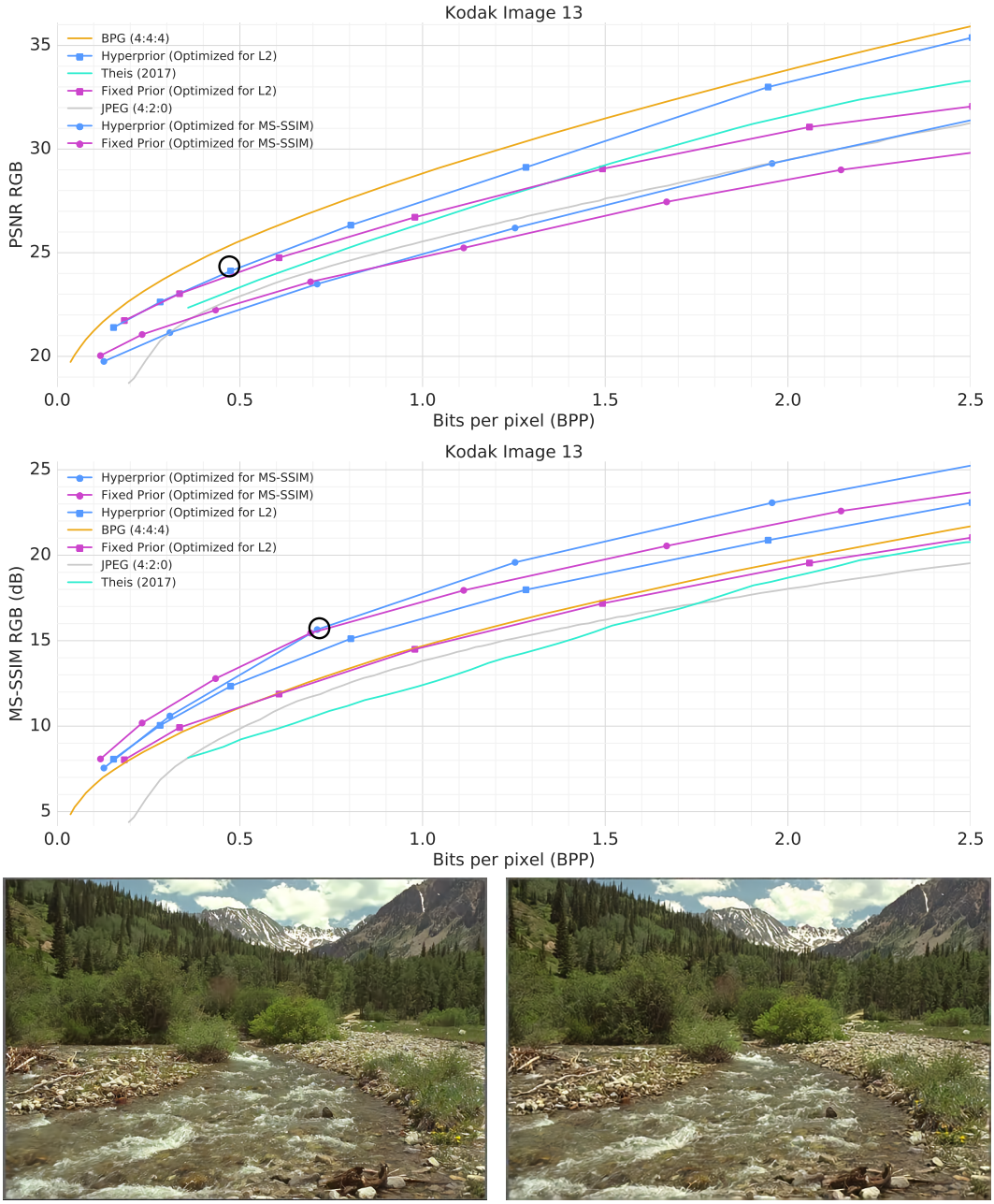

Figure 22: Results for Kodak image 13: PSNR and MS-SSIM rate-distortion curves (top), and example reconstructions for the hyperprior model optimized for L2 (bottom-left) and MS-SSIM (bottom-right). Best viewed on a computer screen. Images correspond to circled points in the plots.

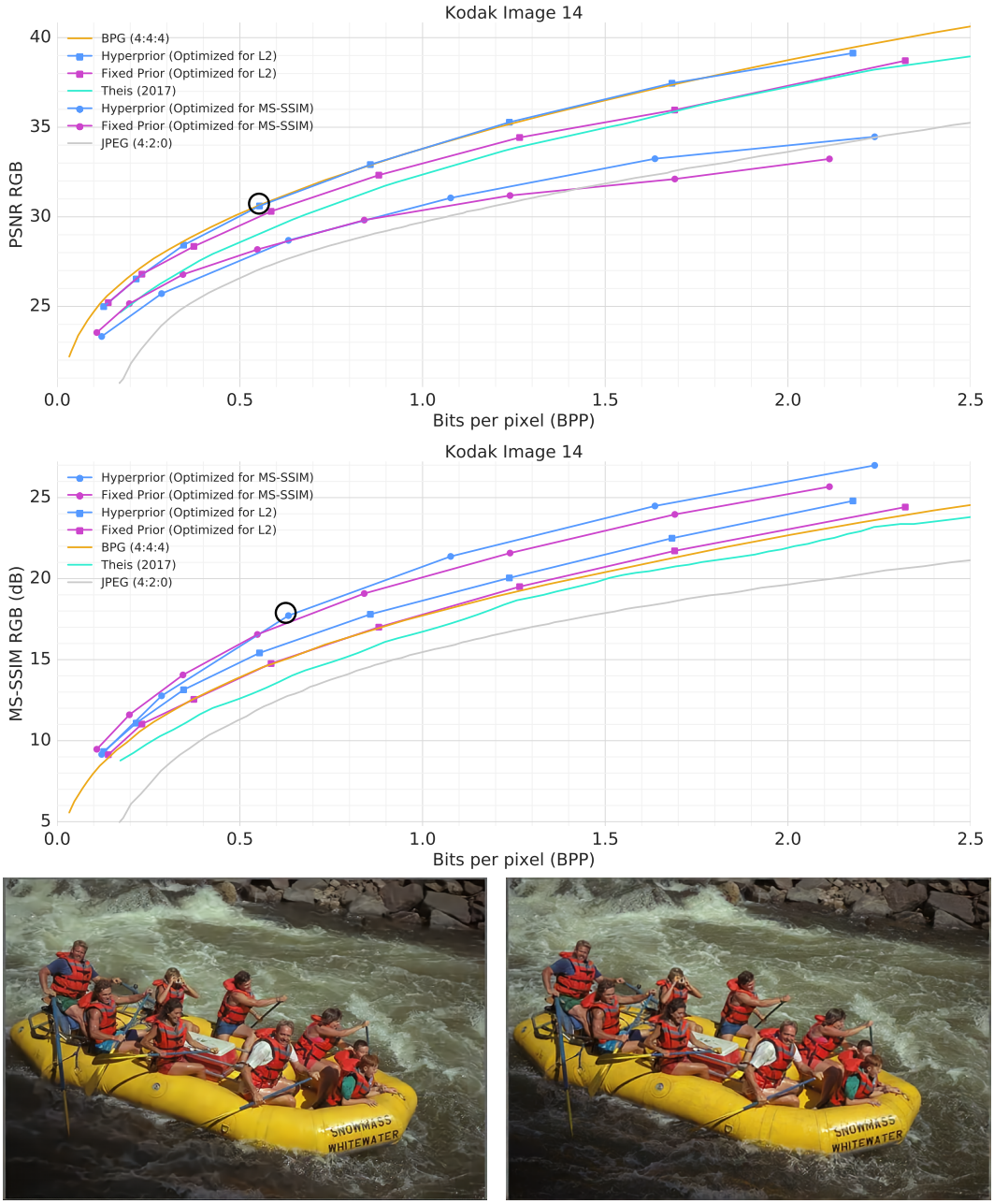

Figure 23: Results for Kodak image 14: PSNR and MS-SSIM rate-distortion curves (top), and example reconstructions for the hyperprior model optimized for L2 (bottom-left) and MS-SSIM (bottom-right). Best viewed on a computer screen. Images correspond to circled points in the plots.

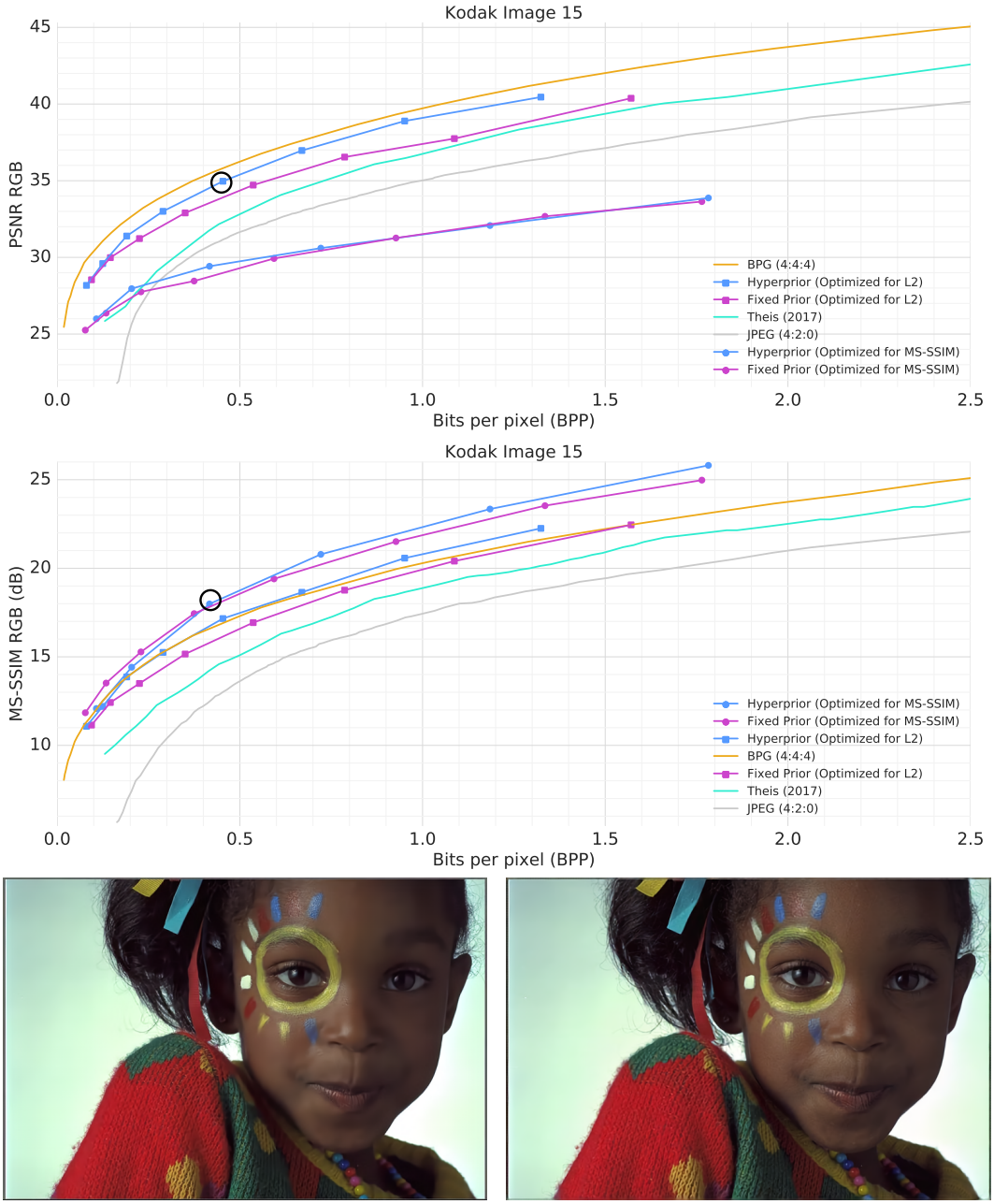

Figure 24: Results for Kodak image 15: PSNR and MS-SSIM rate-distortion curves (top), and example reconstructions for the hyperprior model optimized for L2 (bottom-left) and MS-SSIM (bottom-right). Best viewed on a computer screen. Images correspond to circled points in the plots.

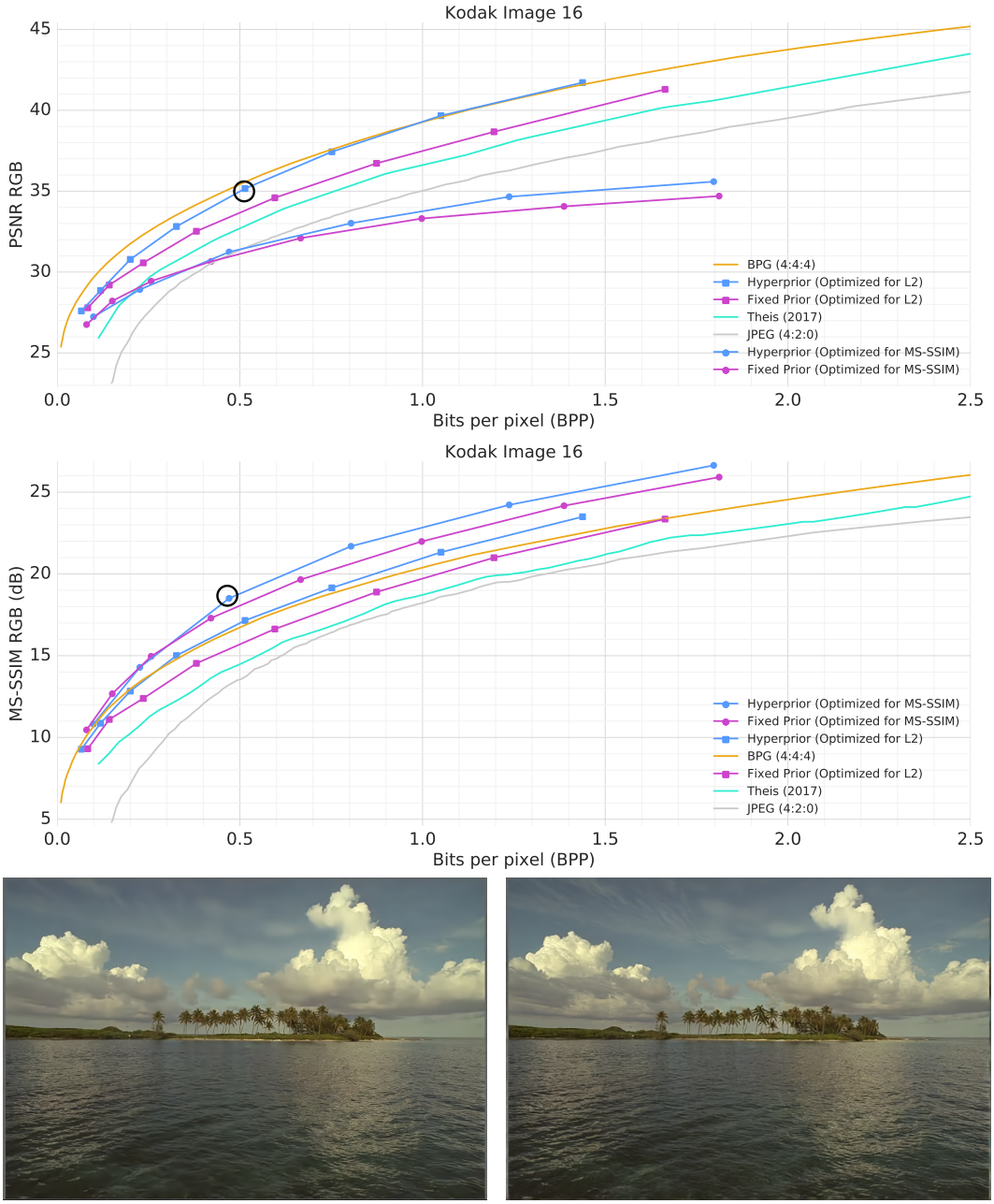

Figure 25: Results for Kodak image 16: PSNR and MS-SSIM rate-distortion curves (top), and example reconstructions for the hyperprior model optimized for L2 (bottom-left) and MS-SSIM (bottom-right). Best viewed on a computer screen. Images correspond to circled points in the plots.

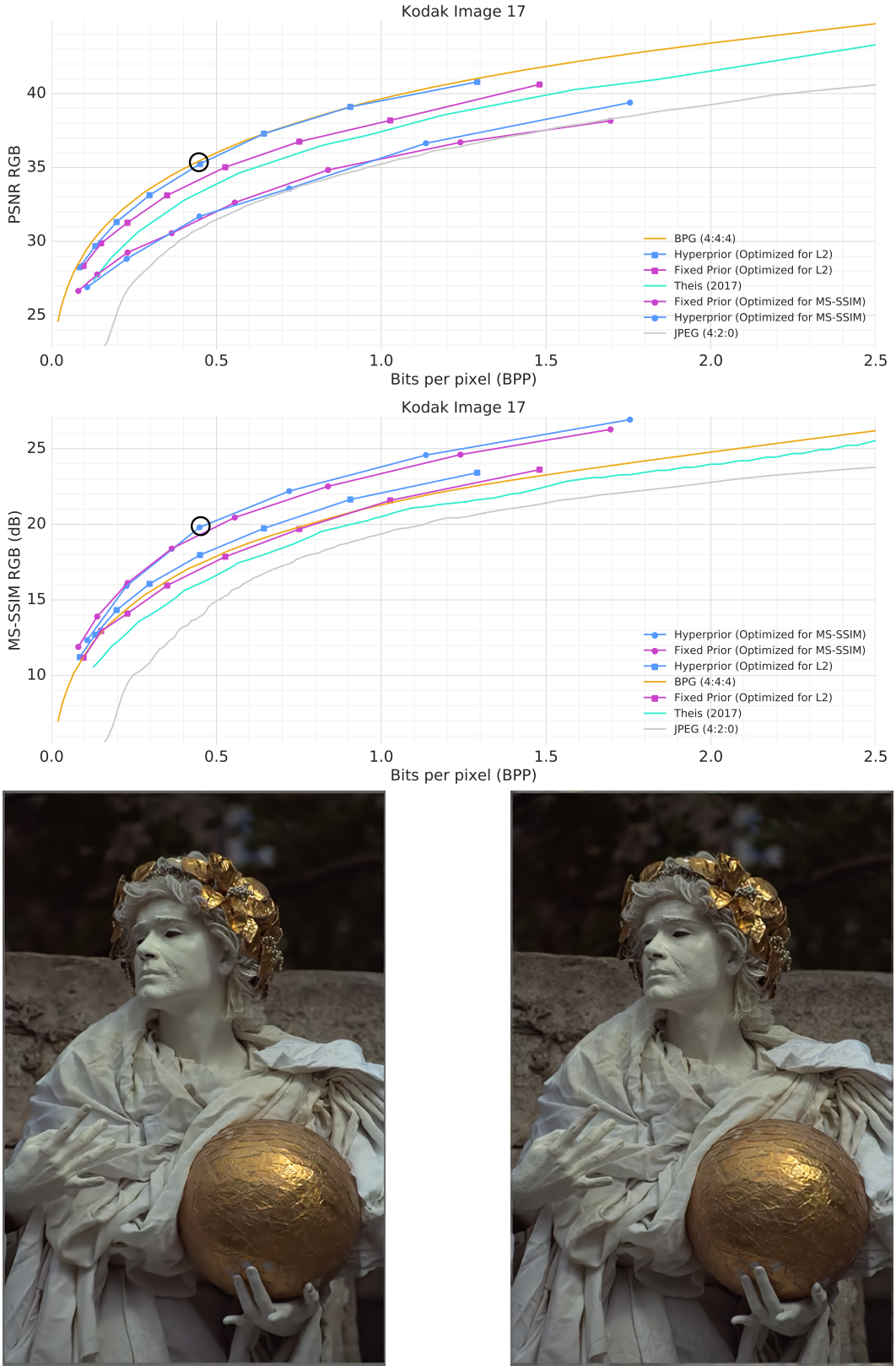

Figure 26: Results for Kodak image 17: PSNR and MS-SSIM rate-distortion curves (top), and example reconstructions for the hyperprior model optimized for L2 (bottom-left) and MS-SSIM (bottom-right). Best viewed on a computer screen. Images correspond to circled points in the plots.

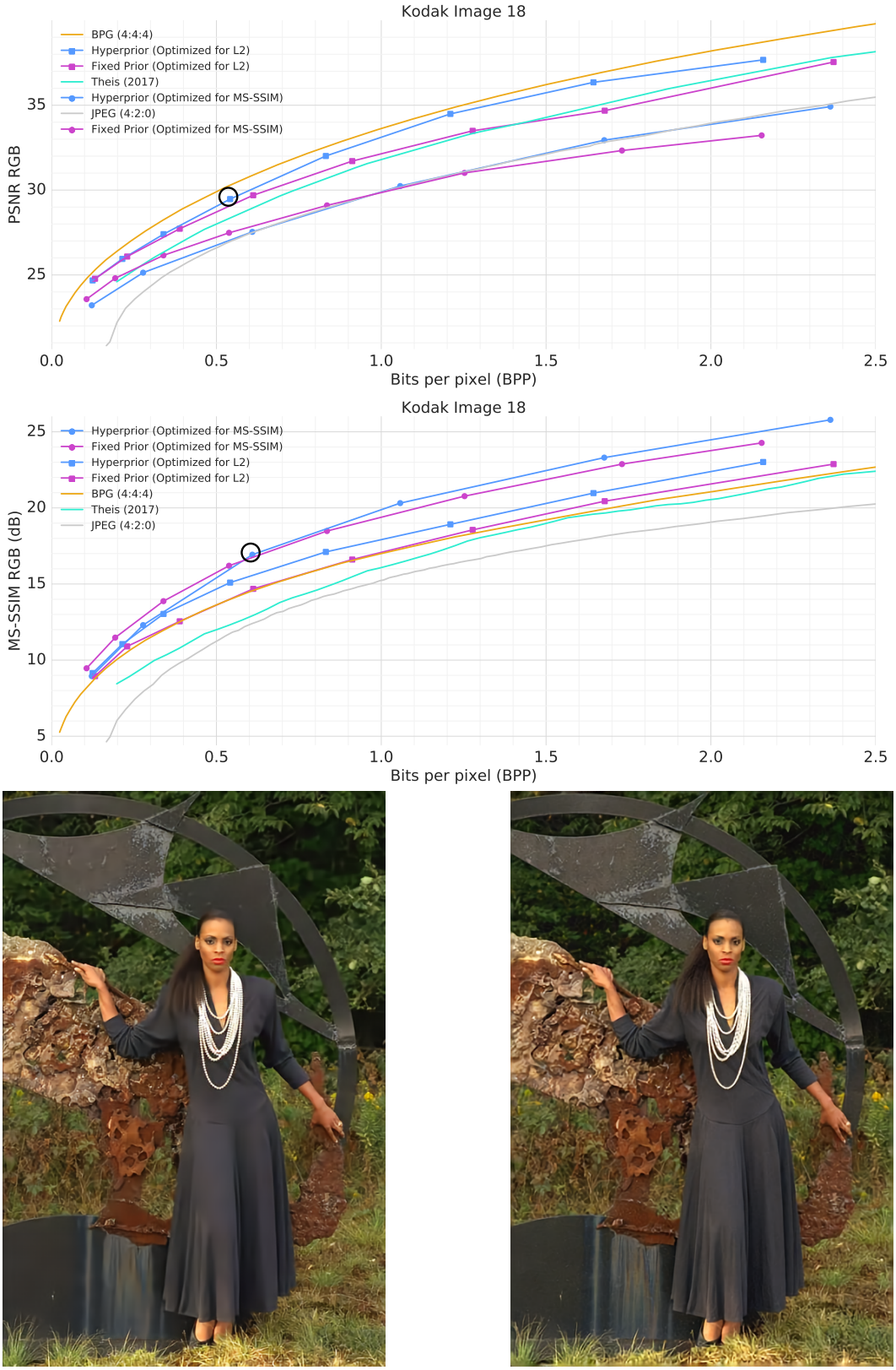

Figure 27: Results for Kodak image 18: PSNR and MS-SSIM rate-distortion curves (top), and example reconstructions for the hyperprior model optimized for L2 (bottom-left) and MS-SSIM (bottom-right). Best viewed on a computer screen. Images correspond to circled points in the plots.

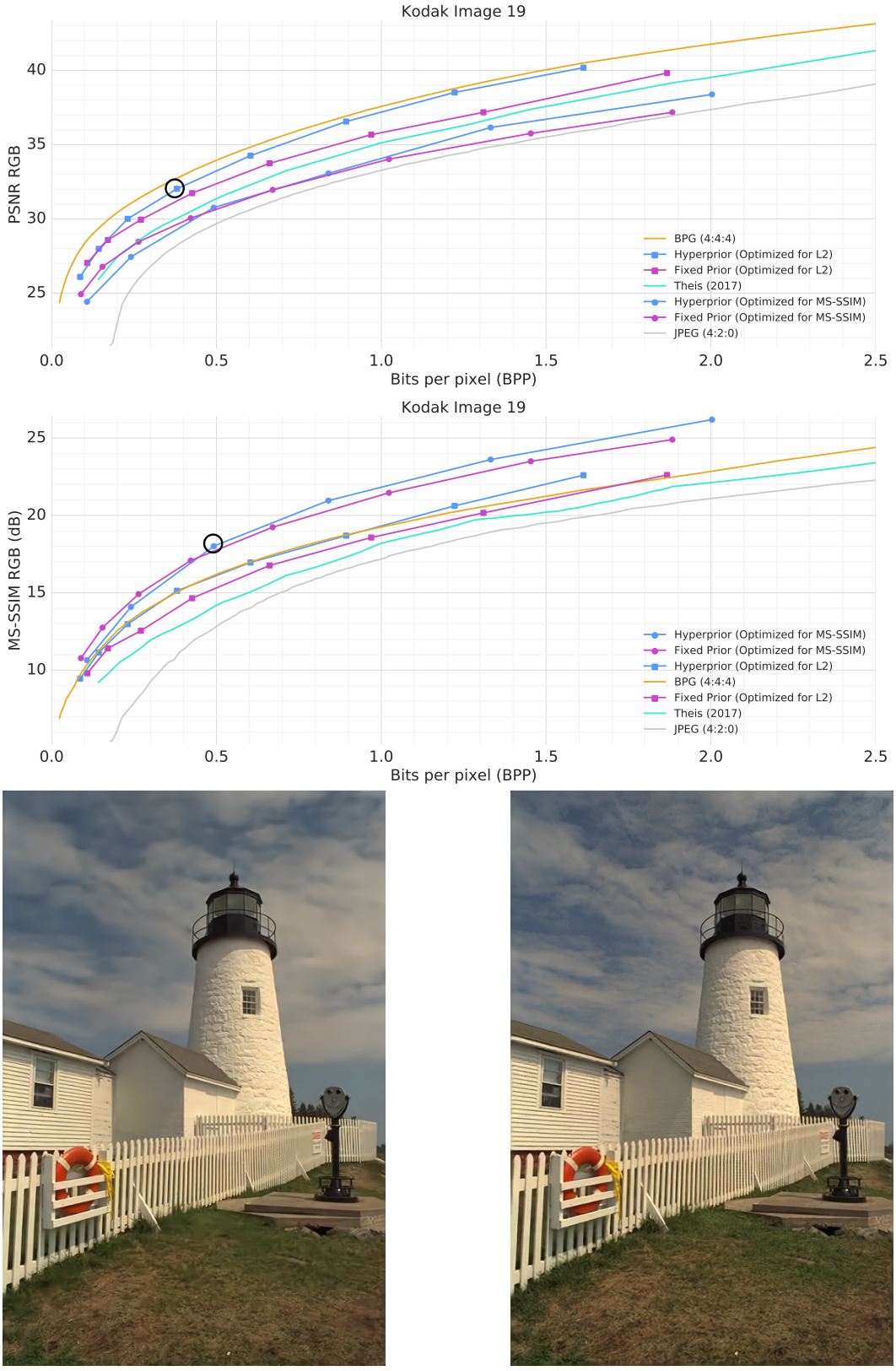

Figure 28: Results for Kodak image 19: PSNR and MS-SSIM rate-distortion curves (top), and example reconstructions for the hyperprior model optimized for L2 (bottom-left) and MS-SSIM (bottom-right). Best viewed on a computer screen. Images correspond to circled points in the plots.

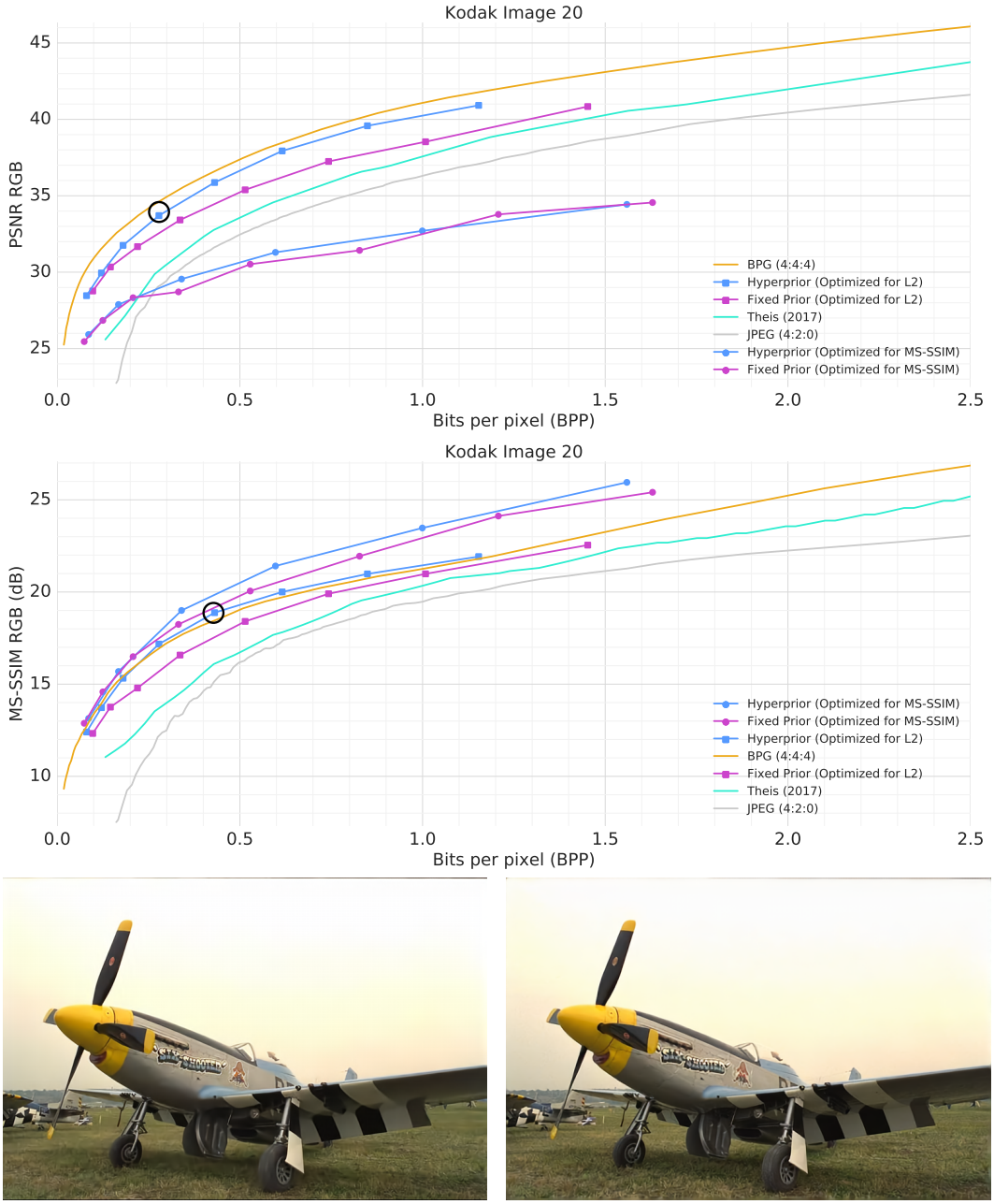

Figure 29: Results for Kodak image 20: PSNR and MS-SSIM rate-distortion curves (top), and example reconstructions for the hyperprior model optimized for L2 (bottom-left) and MS-SSIM (bottom-right). Best viewed on a computer screen. Images correspond to circled points in the plots.

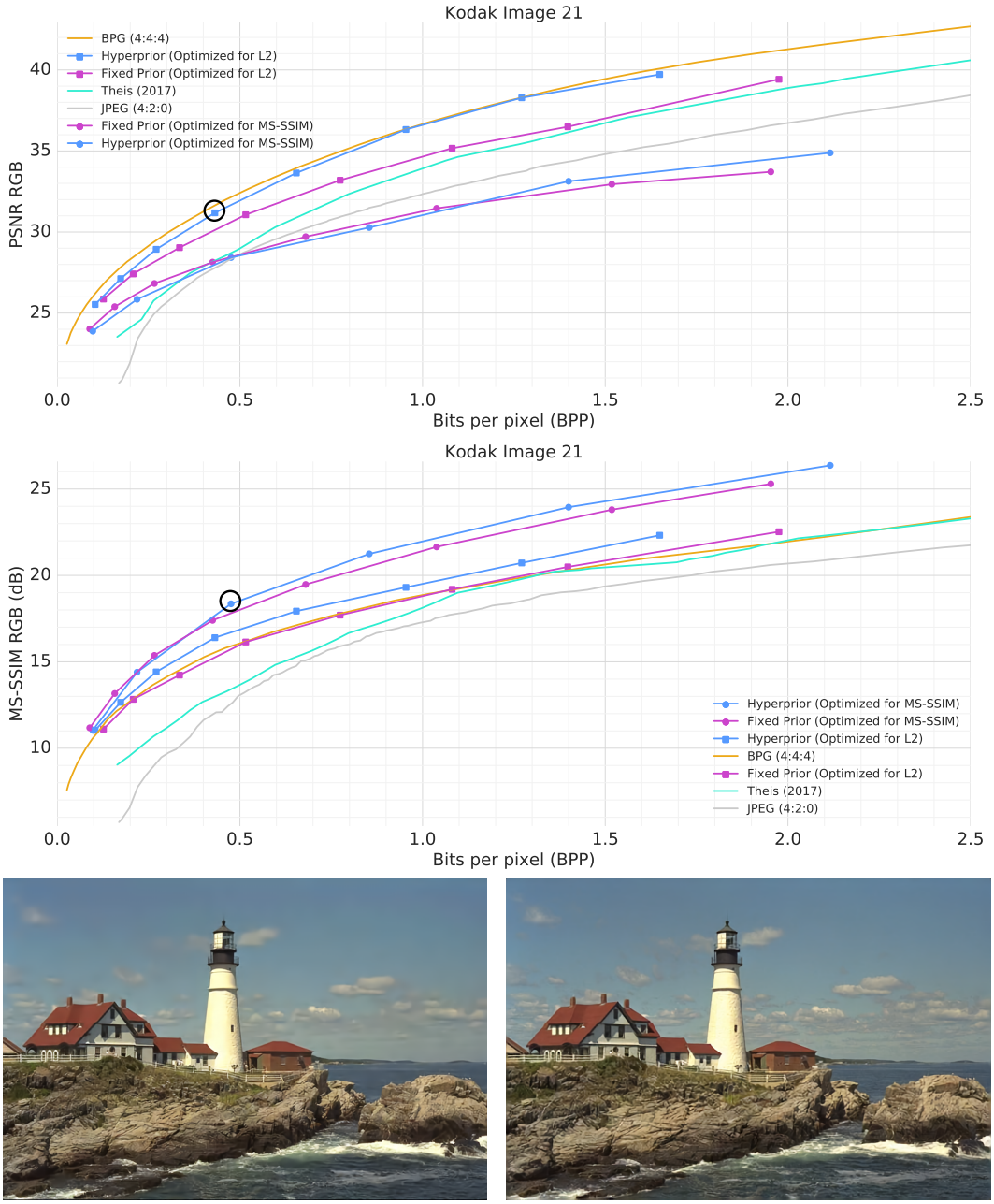

Figure 30: Results for Kodak image 21: PSNR and MS-SSIM rate-distortion curves (top), and example reconstructions for the hyperprior model optimized for L2 (bottom-left) and MS-SSIM (bottom-right). Best viewed on a computer screen. Images correspond to circled points in the plots.

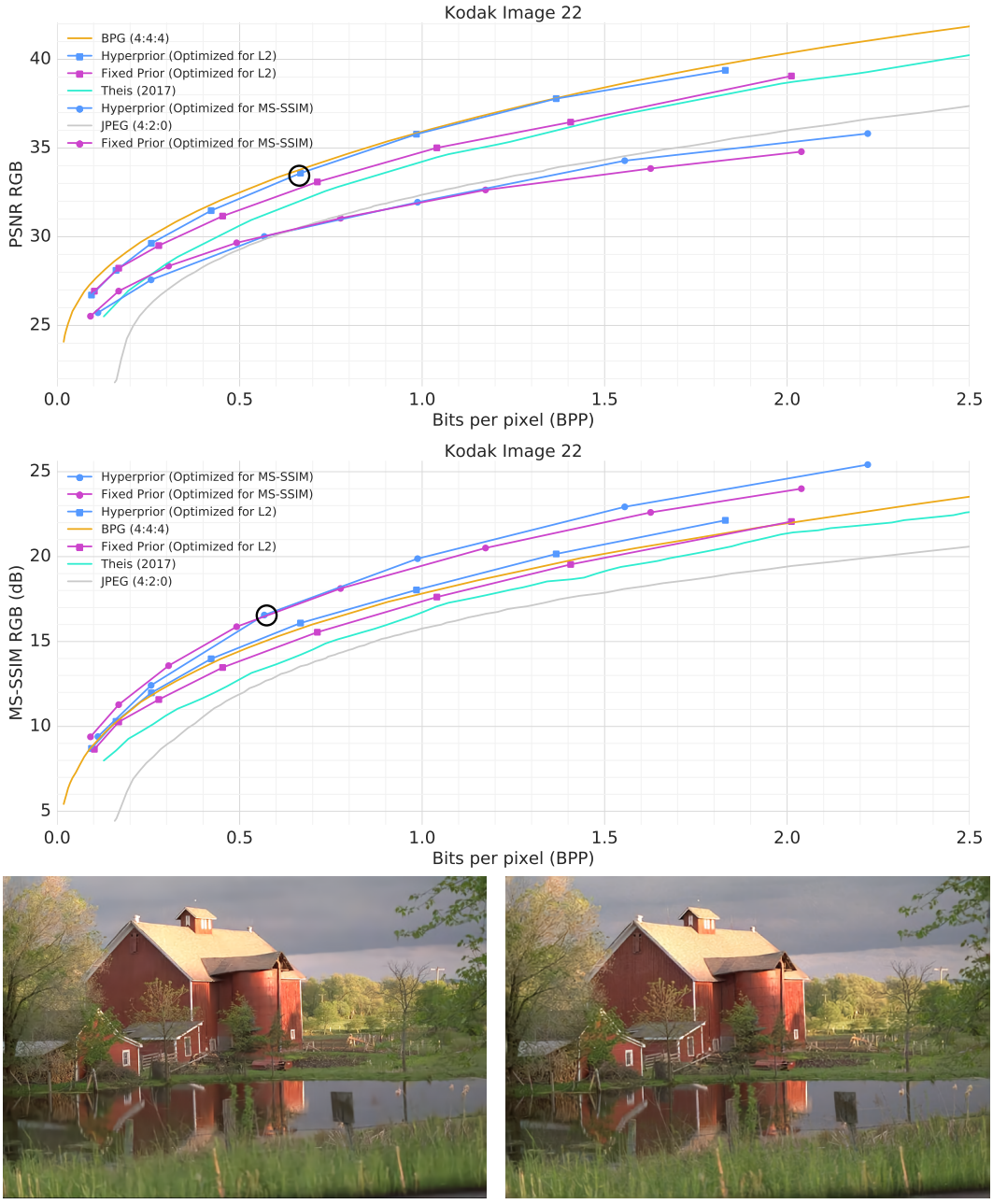

Figure 31: Results for Kodak image 22: PSNR and MS-SSIM rate-distortion curves (top), and example reconstructions for the hyperprior model optimized for L2 (bottom-left) and MS-SSIM (bottom-right). Best viewed on a computer screen. Images correspond to circled points in the plots.

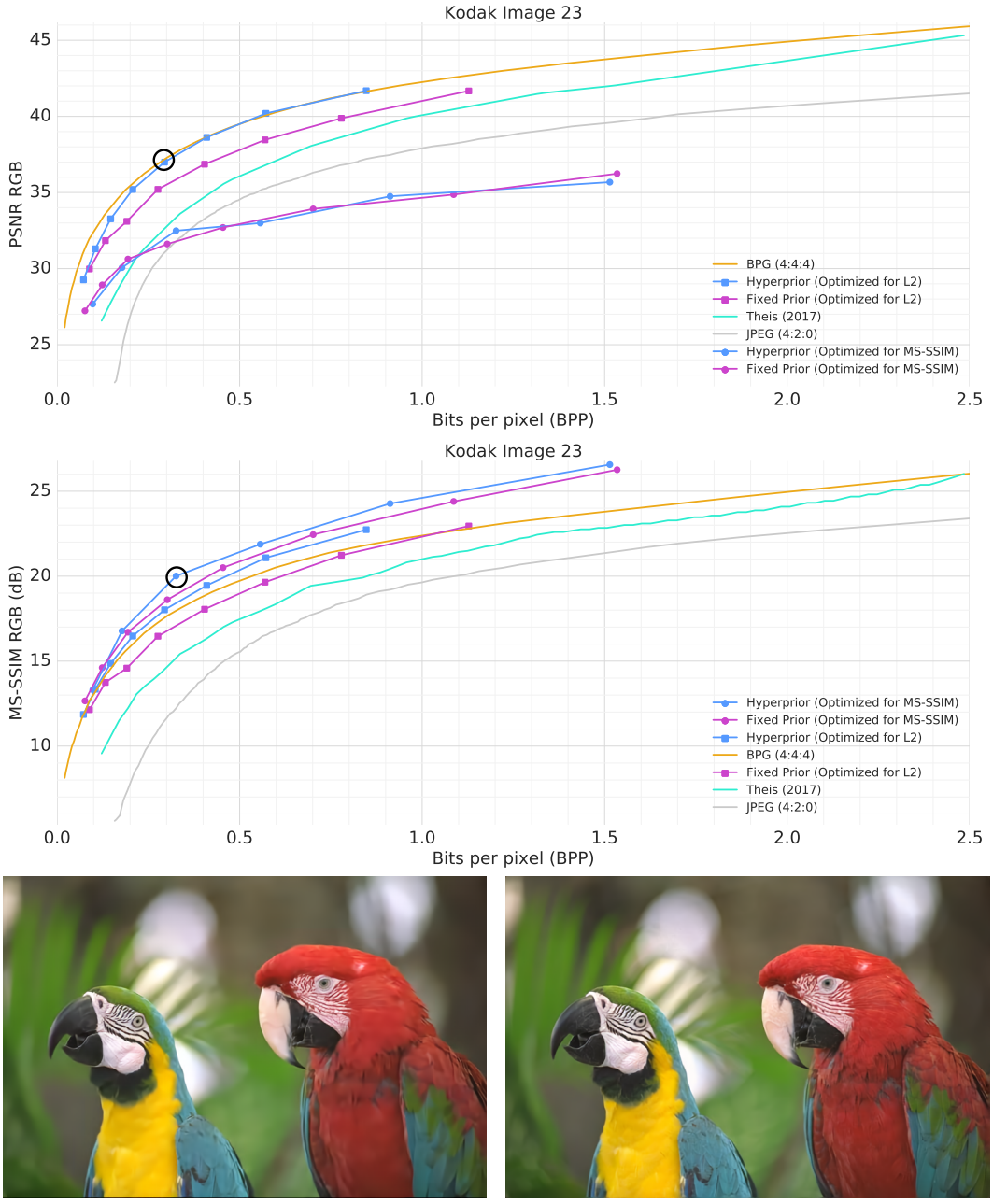

Figure 32: Results for Kodak image 23: PSNR and MS-SSIM rate-distortion curves (top), and example reconstructions for the hyperprior model optimized for L2 (bottom-left) and MS-SSIM (bottom-right). Best viewed on a computer screen. Images correspond to circled points in the plots.

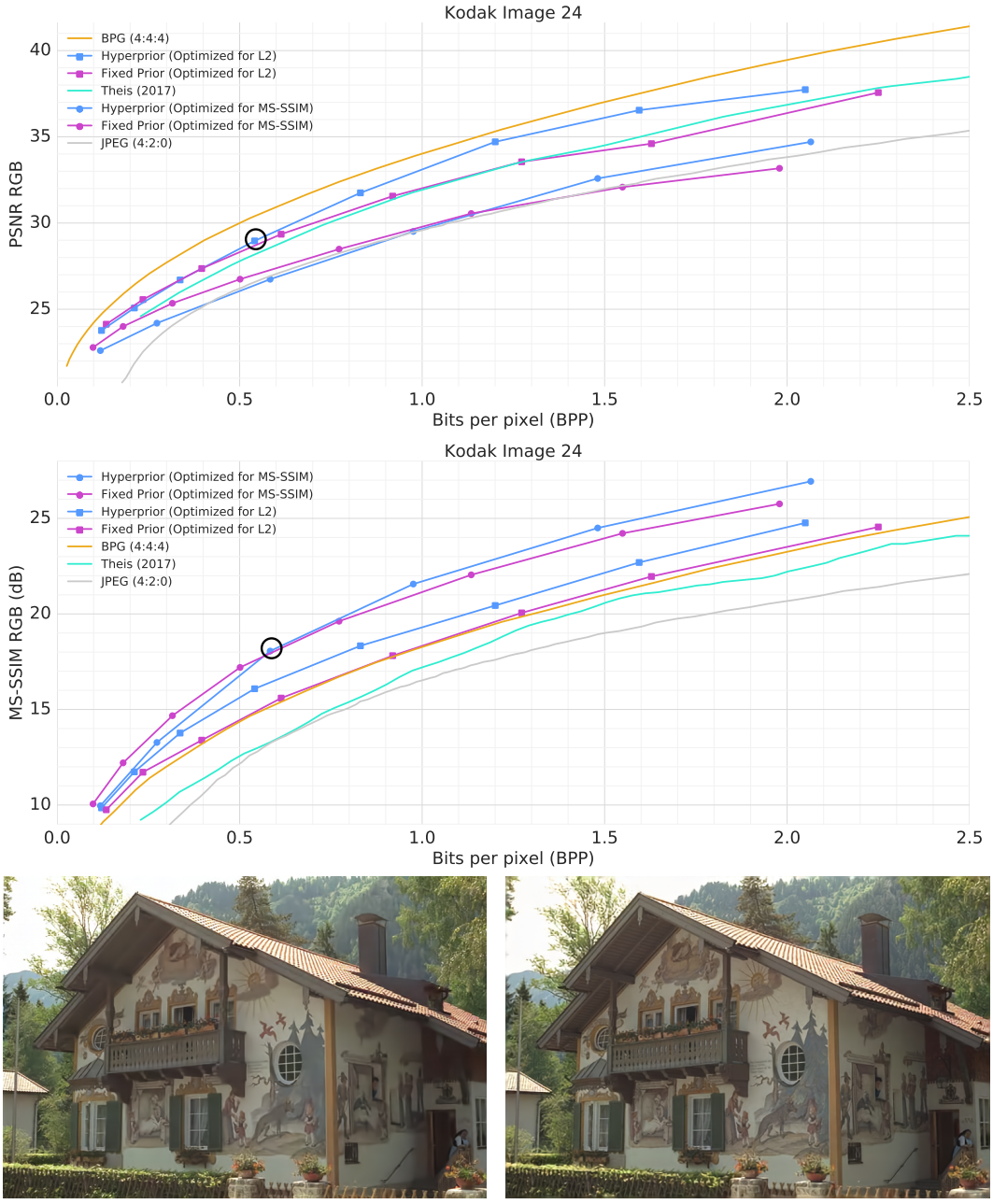

Figure 33: Results for Kodak image 24: PSNR and MS-SSIM rate-distortion curves (top), and example reconstructions for the hyperprior model optimized for L2 (bottom-left) and MS-SSIM (bottom-right). Best viewed on a computer screen. Images correspond to circled points in the plots.