
Improving Inference for Neural Image Compression

Yibo Yang, Robert Bamler, Stephan Mandt

Department of Computer Science

University of California, Irvine

{yibo.yang, rbamler, mandt}@uci.edu

Abstract

We consider the problem of lossy image compression with deep latent variable models. State-of-the-art methods [Ballé et al., 2018, Minnen et al., 2018, Lee et al., 2019] build on hierarchical variational autoencoders (VAEs) and learn inference networks to predict a compressible latent representation of each data point. Drawing on the variational inference perspective on compression [Alemi et al., 2018], we identify three approximation gaps which limit performance in the conventional approach: an amortization gap, a discretization gap, and a marginalization gap. We propose remedies for each of these three limitations based on ideas related to iterative inference, stochastic annealing for discrete optimization, and bits-back coding, resulting in the first application of bits-back coding to lossy compression. In our experiments, which include extensive baseline comparisons and ablation studies, we achieve new state-of-the-art performance on lossy image compression using an established VAE architecture, by changing only the inference method.

1 Introduction

Deep learning methods are reshaping the field of data compression, and recently started to outperform state-of-the-art classical codecs on image compression [Minnen et al., 2018]. Besides useful in its own right, image compression can be seen as a stepping stone towards better video codecs [Lombardo et al., 2019, Habibian et al., 2019], which can reduce a sizable amount of global internet traffic.

State-of-the-art neural methods for lossy image compression [Ballé et al., 2018, Minnen et al., 2018, Lee et al., 2019] learn a mapping between images and latent variables with a variational autoencoder (VAE). An *inference network* maps a given image to a compressible latent representation, which a generative network can then map back to a reconstructed image. In fact, compression can be more broadly seen as a form of inference: to compress—or “encode”—data, one has to perform inference over a well-specified decompression—or “decoding”—algorithm.

In classical compression codecs, the *decoder* has to follow a well-specified procedure to ensure interoperability between different implementations of the same codec. By contrast, the *encoding* process is typically not uniquely defined: different encoder implementations of the same codec often compress the same input data to different bitstrings. For example, `pngcrush` [Randers-Pehrson, 1997] typically produces smaller PNG files than `ImageMagick`. Yet, both programs are standard-compliant PNG encoder implementations as both produce a compressed file that decodes to the same image. In other words, both encoder implementations perform *correct* inference over the standardized decoder specification, but use different inference algorithms with different performance characteristics.

The insight that better inference leads to better compression performance even within the same codec motivates us to reconsider how inference is typically done in neural data compression with VAEs. In this paper, we show that the conventional *amortized* inference [Kingma and Welling, 2013, Rezende et al., 2014] in VAEs leaves substantial room for improvement when used for data compression.

We propose an improved inference method for data compression tasks with three main innovations:

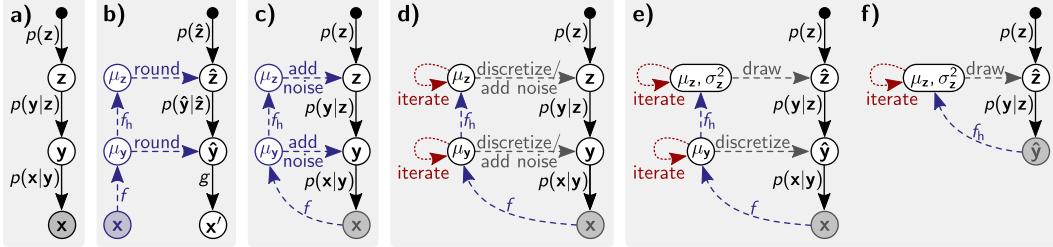


Figure 1: Graphical model and control flow charts. a) generative model with hyperlatents \mathbf{z} , latents \mathbf{y} , and image \mathbf{x} [Minné et al., 2018]; b) conventional method for compression (dashed blue, see Eq. 1) and decompression (solid black); c) common training objective due to [Ballé et al., 2017] (see Eq. 3); d) proposed hybrid amortized (dashed blue) / iterative (dotted red) inference (Section 3.1); e)-f) inference in the proposed lossy bitsback method (Section 3.3); the encoder first executes e) and then keeps $\hat{\mathbf{y}}$ fixed while executing f); the decoder reconstructs $\hat{\mathbf{y}}$ and then executes f) to get bits back.

1. *Improved amortization:* The amortized inference strategy in VAEs speeds up training, but is restrictive at compression time. We draw a connection between a recently proposed iterative procedure for compression [Campos et al., 2019] to the broader literature of VI that closes the amortization gap, which provides the basis of the following two novel inference methods.
2. *Improved discretization:* Compression requires discretizing the latent representation from VAEs, because only discrete values can be entropy coded. As inference over discrete variables is difficult, existing methods typically relax the discretization constraint in some way during inference and then discretize afterwards. We instead propose a novel method based on a stochastic annealing scheme that performs inference directly over discrete points.
3. *Improved entropy coding:* In lossless compression with latent-variable models, bits-back coding [Wallace, 1990, Hinton and Van Camp, 1993] allows approximately coding the latents with the *marginal* prior. It is so far believed that bits-back coding is incompatible with lossy compression [Habibian et al., 2019] because it requires inference on the *decoder* side, which does not have access to the exact (undistorted) input data. We propose a remedy to this limitation, resulting in the first application of bits-back coding to lossy compression.

We evaluate the above three innovations on an otherwise unchanged architecture of an established model and compare against a wide range of baselines and ablations. Our proposals significantly improve compression performance and results in a new state of the art in lossy image compression.

The rest of the paper is structured as follows: Section 2 summarizes lossy compression with VAEs, for which Section 3 proposes the above three improvements. Section 4 reports experimental results. We conclude in Section 5. We review related work in each relevant subsection.

2 Background: Lossy Neural Image Compression as Variational Inference

In this section, we summarize an existing framework for lossy image compression with deep latent variable models, which will be the basis of three proposed improvements in Section 3.

Related Work. Ballé et al. [2017] and Theis et al. [2017] were among the first to recognize a connection between the rate-distortion objective of lossy compression and the loss function of a certain kind of Variational Autoencoders (VAEs), and to apply it to end-to-end image compression. [Ballé et al., 2018] proposes a hierarchical model, upon which current state-of-the-art methods [Minné et al., 2018, Lee et al., 2019] further improve by adding an auto-regressive component. For simplicity, we adopt the VAE of [Minné et al., 2018] without the auto-regressive component, reviewed in this section below, which has also been a basis for recent compression research [Johnston et al., 2019].

Generative Model. Figure 1a shows the generative process of an image \mathbf{x} . The Gaussian likelihood $p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}; g(\mathbf{y}; \theta), \sigma_x^2 I)$ has a fixed variance σ_x^2 and its mean is computed from latent variables \mathbf{y} by a deconvolutional neural network (DNN) g with weights θ . A second DNN g_h with weights θ_h outputs the parameters (location and scale) of the prior $p(\mathbf{y}|\mathbf{z})$ conditioned on ‘hyperlatents’ \mathbf{z} .

Compression and Decompression. Figure 1b illustrates compression (“encoding”, dashed blue) and decompression (“decoding”, solid black) as proposed in [Minnen et al., 2018]. The encoder passes a target image \mathbf{x} through trained inference networks f and f_h with weights ϕ and ϕ_h , respectively. The resulting continuous latent representations, (μ_y, μ_z) , are rounded (denoted $\lfloor \cdot \rfloor$) to discrete (\hat{y}, \hat{z}) ,

$$\hat{y} = \lfloor \mu_y \rfloor; \quad \hat{z} = \lfloor \mu_z \rfloor \quad \text{where} \quad \mu_y = f(\mathbf{x}; \phi); \quad \mu_z = f_h(f(\mathbf{x}; \phi); \phi_h). \quad (1)$$

The encoder then entropy-codes \hat{z} and \hat{y} , using discretized versions $P(\hat{z})$ and $P(\hat{y}|\hat{z})$ of the hyperprior $p(z)$ and (conditional) prior $p(y|z)$, respectively, as entropy models. This step is simplified by a clever restriction on the shape of p , such that it agrees with the discretized P on all integers (see [Ballé et al., 2017, 2018, Minnen et al., 2018]). The decoder then recovers \hat{z} and \hat{y} and obtains a lossy image reconstruction $\mathbf{x}' := \arg \max_{\mathbf{x}} p(\mathbf{x}|\hat{\mathbf{y}}) = g(\hat{\mathbf{y}}; \theta)$.

Model Training. Let \mathbf{x} be sampled from a training set of images. To train the above VAE for lossy compression, Theis et al. [2017], Minnen et al. [2018] consider minimizing a rate-distortion objective:

$$\begin{aligned} \mathcal{L}_\lambda(\lfloor \mu_y \rfloor, \lfloor \mu_z \rfloor) &= \mathcal{R}(\lfloor \mu_y \rfloor, \lfloor \mu_z \rfloor) + \lambda \mathcal{D}(\lfloor \mu_y \rfloor, \mathbf{x}) \\ &= \underbrace{-\log_2 p(\lfloor \mu_z \rfloor)}_{\substack{\text{information content} \\ \text{of } \hat{z} = \lfloor \mu_z \rfloor}} - \underbrace{\log_2 p(\lfloor \mu_y \rfloor | \lfloor \mu_z \rfloor)}_{\substack{\text{information content of} \\ \hat{y} = \lfloor \mu_y \rfloor \text{ given } \hat{z} = \lfloor \mu_z \rfloor}} + \lambda \underbrace{\| \mathbf{x} - g(\lfloor \mu_y \rfloor; \theta) \|_2^2}_{\text{distortion } \mathcal{D}} \end{aligned} \quad (2)$$

where (μ_y, μ_z) is computed from the inference networks as in Eq. 1, and the parameter $\lambda > 0$ controls the trade-off between bitrate \mathcal{R} under entropy coding, and distortion (reconstruction error) \mathcal{D} . As the rounding operations prevent gradient-based optimization, Ballé et al. [2017] propose to replace rounding during training by adding uniform noise from the interval $[-\frac{1}{2}, \frac{1}{2}]$ to each coordinate of μ_y and μ_z (see Figure 1c). This is equivalent to sampling from uniform distributions $q(y|x)$ and $q(z|x)$ with a fixed width of one centered around μ_y and μ_z , respectively. One thus obtains the following relaxed rate-distortion objective, for a given data point \mathbf{x} ,

$$\tilde{\mathcal{L}}_\lambda(\theta, \theta_h, \phi, \phi_h) = \mathbb{E}_{q(y|x) q(z|x)} \left[-\log_2 p(z) - \log_2 p(y|z) + \lambda \| \mathbf{x} - g(y; \theta) \|_2^2 \right]. \quad (3)$$

Connection to Variational Inference. As pointed out in [Ballé et al., 2017], the relaxed objective in Eq. 3 is the negative evidence lower bound (NELBO) of variational inference (VI) if we identify $\lambda = 1/(2\sigma_x^2 \log 2)$. This draws a connection between lossy compression and VI [Blei et al., 2017, Zhang et al., 2019]. We emphasize a distinction between variational *inference* and variational *expectation maximization* (EM) [Beal and Ghahramani, 2003]: whereas variational EM trains a model (and employs VI as a subroutine), VI is used to compress data using a trained model. A central result of this paper is that improving inference in a fixed generative model at compression time already suffices to significantly improve compression performance.

3 Novel Inference Techniques for Data Compression

This section presents our main contributions. We identify three approximation gaps in VAE-based compression methods (see Section 2): an amortization gap, a discretization gap, and a marginalization gap. Recently, the amortization gap was considered by [Campos et al., 2019]; we expand on this idea in Section 3.1, bringing it under a wider framework of algorithms in the VI literature. We then propose two specific methods that improve inference at compression time: a novel inference method over discrete representations (Section 3.2) that closes the discretization gap, and a novel lossy bits-back coding method (Section 3.3) that closes the marginalization gap.

3.1 Amortization Gap and Hybrid Amortized-Iterative Inference

Amortization Gap. Amortized variational inference [Kingma and Welling, 2013, Rezende et al., 2014] turns optimization over *local* (per-data) variational parameters into optimization over the *global* weights of an inference network. In the VAE of Section 2, the inference networks f and f_h map an image \mathbf{x} to local parameters μ_y and μ_z of the variational distributions $q(y|x)$ and $q(z|x)$, respectively (Eq. 1). Amortized VI speeds up training by avoiding an expensive inner inference loop of variational EM, but it leads to an amortization gap [Cremer et al., 2018, Krishnan et al., 2018], which is the difference between the value of the NELBO (Eq. 3) when μ_y and μ_z are obtained from the inference networks, compared to its true minimum when μ_y and μ_z are directly minimized. Since

the NELBO approximates the rate-distortion objective (Section 2), the amortization gap translates into sub-optimal performance when amortization is used at compression time. As discussed below, this gap can be closed by refining the output of the inference networks by iterative inference.

Related Work. The idea of combining amortized inference with iterative optimization has been studied and refined by various authors [Hjelm et al., 2016, Kim et al., 2018, Krishnan et al., 2018, Marino et al., 2018]. While not formulated in the language of variational autoencoders and variational inference, Campos et al. [2019] apply a simple version of this idea to compression. Drawing on the connection to hybrid amortized-iterative inference, we show that the method can be drastically improved by addressing the discretization gap (Section 3.2) and the marginalization gap (Section 3.3).

Hybrid Amortized-Iterative Inference. We reinterpret the proposal in [Campos et al., 2019] as a basic version of a hybrid amortized-iterative inference idea that changes inference only at compression time but not during model training. Figure 1d illustrates the approach. When compressing a target image \mathbf{x} , one initializes μ_y and μ_z from the trained inference networks f and f_h , see Eq. 1. One then treats μ_y and μ_z as *local* variational parameters, and minimizes the NELBO (Eq. 3) over (μ_y, μ_z) with the reparameterization trick and stochastic gradient descent. This approach thus separates inference *at test time* from inference *during model training*. We show in the next two sections that this simple idea forms a powerful basis for new inference approaches that drastically improve compression performance.

3.2 Discretization Gap and Stochastic Gumbel Annealing (SGA)

Discretization Gap. Compressing data to a bitstring is an inherently *discrete* optimization problem. As discrete optimization in high dimensions is difficult, neural compression methods instead optimize some relaxed objective function (such as the NELBO $\tilde{\mathcal{L}}_\lambda$ in Eq. 3) over continuous representations (such as μ_y and μ_z), which are then discretized afterwards for entropy coding (see Eq. 1). This leads to a *discretization gap*: the difference between the true rate-distortion objective \mathcal{L}_λ at the discretized representation $(\hat{\mathbf{y}}, \hat{\mathbf{z}})$, and the relaxed objective function $\tilde{\mathcal{L}}_\lambda$ at the continuous approximation (μ_y, μ_z) .

Related Work. Current neural compression methods use a differentiable approximation to discretization *during model training*, such as Straight-Through Estimator (STE) [Bengio et al., 2013, Oord et al., 2017, Yin et al., 2019], adding uniform noise [Ballé et al., 2017] (see Eq. 3), stochastic binarization [Toderici et al., 2016], and soft-to-hard quantization [Agustsson et al., 2017]. Discretization *at compression time* was addressed in [Yang et al., 2020] without assuming that the training procedure takes discretization into account, whereas our work makes this additional assumption.

Stochastic Gumbel Annealing (SGA). We refine the hybrid amortized-iterative inference approach of Section 3.1 to close the discretization gap: for a given image \mathbf{x} , we aim to find its *discrete* (in our case, integer) representation $(\hat{\mathbf{y}}, \hat{\mathbf{z}})$ that optimizes the rate-distortion objective \mathcal{L}_λ in Eq. 2, this time re-interpreted as a function of $(\hat{\mathbf{y}}, \hat{\mathbf{z}})$ directly. Our proposed annealing scheme approaches this discretization optimization problem with the help of continuous proxy variables μ_y and μ_z , which we initialize from the inference networks as in Eq. 1. We limit the following discussion to the latents μ_y and $\hat{\mathbf{y}}$, treating the hyperlatents μ_z and $\hat{\mathbf{z}}$ analogously. For each dimension i of μ_y , we map the continuous coordinate $\mu_{y,i} \in \mathbb{R}$ to an integer coordinate $\hat{y}_i \in \mathbb{Z}$ by rounding either up or down. Let $r_{y,i}$ be a one-hot vector that indicates the rounding direction, with $r_{y,i} = (1, 0)$ for rounding down (denoted $\lfloor \cdot \rfloor$) and $r_{y,i} = (0, 1)$ for rounding up (denoted $\lceil \cdot \rceil$). Thus, the result of rounding is the inner product, $\hat{y}_i = r_{y,i} \cdot (\lfloor \mu_{y,i} \rfloor, \lceil \mu_{y,i} \rceil)$. Now we let $r_{y,i}$ be a Bernoulli variable with a “tempered” distribution

$$q_\tau(r_{y,i} | \mu_{y,i}) \propto \begin{cases} \exp\{-\psi(\mu_{y,i} - \lfloor \mu_{y,i} \rfloor)/\tau\} & \text{if } r_{y,i} = (1, 0) \\ \exp\{-\psi(\lceil \mu_{y,i} \rceil - \mu_{y,i})/\tau\} & \text{if } r_{y,i} = (0, 1) \end{cases} \quad (4)$$

with temperature parameter $\tau > 0$, and $\psi : [0, 1] \rightarrow \mathbb{R}$ is an increasing function satisfying $\lim_{\alpha \rightarrow 1} \psi(\alpha) = \infty$ (this is chosen to ensure continuity of the resulting objective (5); we used $\psi = \tanh^{-1}$). Thus, as $\mu_{y,i}$ approaches an integer, the probability of rounding to that integer approaches one. Defining $q_\tau(r_y | \mu_y) = \prod_i q_\tau(r_{y,i} | \mu_{y,i})$, we thus minimize the stochastic objective

$$\tilde{\mathcal{L}}_{\lambda, \tau}(\mu_y, \mu_z) = \mathbb{E}_{q_\tau(r_y | \mu_y) q_\tau(r_z | \mu_z)} [\mathcal{L}_\lambda(r_y \cdot (\lfloor \mu_y \rfloor, \lceil \mu_y \rceil), r_z \cdot (\lfloor \mu_z \rfloor, \lceil \mu_z \rceil))] \quad (5)$$

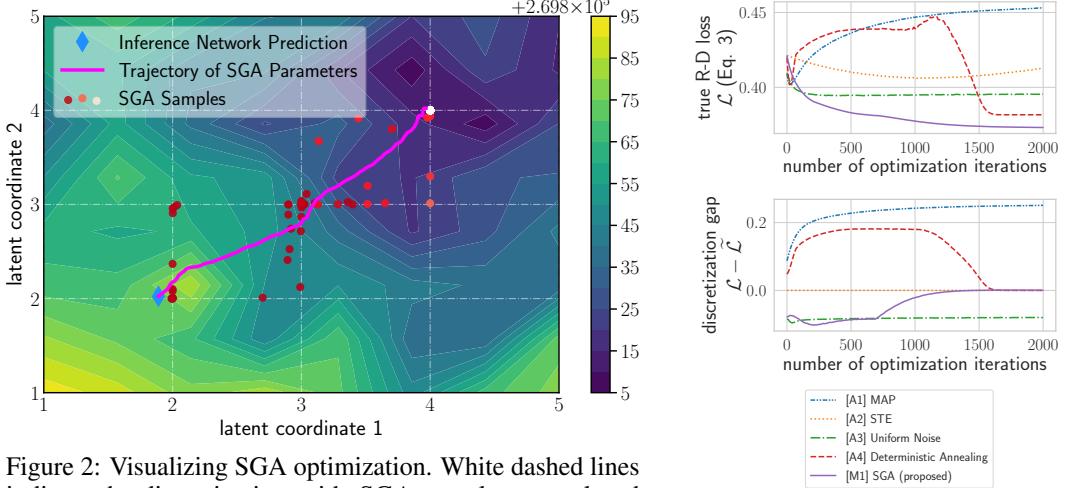


Figure 2: Visualizing SGA optimization. White dashed lines indicate the discretization grid. SGA samples are colored by temperature (lighter color corresponds to lower temperature). 10 Gumbel-softmax samples were used to approximate expectations in the objective (5), and temperature $\tau = 0.1$.

Figure 3: Comparing Stochastic Gumbel Annealing (Section 3.2) to alternatives (Table 1, [A1]-[A4]).

where we reintroduce the hyperlatents \mathbf{z} , and $r_{\mathbf{y}} \cdot (\lfloor \mu_{\mathbf{y}} \rfloor, \lceil \mu_{\mathbf{y}} \rceil)$ denotes the discrete vector $\hat{\mathbf{y}}$ obtained by rounding each coordinate $\mu_{\mathbf{y},i}$ of $\mu_{\mathbf{y}}$ according to its rounding direction $r_{\mathbf{y},i}$. At any temperature τ , the objective in Eq. 5 smoothly interpolates the R-D objective in Eq. 2 between all integer points. We minimize Eq. 5 over $(\mu_{\mathbf{y}}, \mu_{\mathbf{z}})$ with stochastic gradient descent, propagating gradients through Bernoulli samples via the Gumbel-softmax trick [Jang et al., 2016, Maddison et al., 2016] using, for simplicity, the same temperature τ as in q_{τ} . We note that in principle, REINFORCE [Williams, 1992] can also work instead of Gumbel-softmax. We anneal τ towards zero over the course of optimization, such that the Gumbel approximation becomes exact and the stochastic rounding operation converges to the deterministic one in Eq. 1. We thus name this method ‘‘Stochastic Gumbel Annealing’’ (SGA).

Figure 2 illustrates the loss function (Eq. 5) landscape of SGA near its converged solution, for a given Kodak image. Here, we visualize along the two coordinates of $\mu_{\mathbf{y}}$ whose optimization trajectories have the most variance among all coordinates, and plot the optimization trajectory of $\mu_{\mathbf{y}}$ (magenta line), and stochastically rounded samples (red dots) computed by the inner product $r_{\mathbf{y}} \cdot (\lfloor \mu_{\mathbf{y}} \rfloor, \lceil \mu_{\mathbf{y}} \rceil)$. The final SGA sample converges to the deterministically rounded solution (white dot, (4, 4)), which is significantly better than the initialization predicted by inference network (blue diamond, near (2, 2)).

Comparison to Alternatives. Alternatively, we could have started with Eq. 2, and optimized it as a function of $(\mu_{\mathbf{y}}, \mu_{\mathbf{z}})$, using existing discretization methods to relax rounding; the optimized $(\mu_{\mathbf{y}}, \mu_{\mathbf{z}})$ would subsequently be rounded to $(\hat{\mathbf{y}}, \hat{\mathbf{z}})$. However, our method outperforms all these alternatives. Specifically, we tried (with shorthands [A1]-[A4] for ‘‘ablation’’, referring to Table 1 of Section 4): [A1] **MAP**: completely ignoring rounding during optimization; [A2] **Straight-Through Estimator (STE)** [Bengio et al., 2013]: rounding only on the forward pass of automatic differentiation; [A3] **Uniform Noise** [Campos et al., 2019]: optimizing Eq. 3 over $(\mu_{\mathbf{y}}, \mu_{\mathbf{z}})$ also at compression time; and [A4] **Deterministic Annealing**: turning the stochastic rounding operations of SGA into deterministic weighted averages, by pushing the expectation operators w.r.t q in Eq. 5 to *inside* the arguments of \mathcal{L}_{λ} , resulting in the loss function $\mathcal{L}_{\lambda}(\mathbb{E}_{q_{\tau}(r_{\mathbf{y}}|\mu_{\mathbf{y}})}[r_{\mathbf{y}} \cdot (\lfloor \mu_{\mathbf{y}} \rfloor, \lceil \mu_{\mathbf{y}} \rceil)], \mathbb{E}_{q_{\tau}(r_{\mathbf{z}}|\mu_{\mathbf{z}})}[r_{\mathbf{z}} \cdot (\lfloor \mu_{\mathbf{z}} \rfloor, \lceil \mu_{\mathbf{z}} \rceil)])$ and an algorithm more similar to [Agustsson et al., 2017].

We tested the above discretization methods on images from the Kodak data set [Kodak], using a pre-trained hyperprior model (Section 2). Figure 3 (left) shows learning curves for the true rate-distortion objective \mathcal{L}_{λ} , evaluated via rounding the intermediate $(\mu_{\mathbf{y}}, \mu_{\mathbf{z}})$ throughout optimization. Figure 3 (right) shows the discretization gap $\mathcal{L}_{\lambda} - \tilde{\mathcal{L}}_{\lambda}$, where $\tilde{\mathcal{L}}_{\lambda}$ is the relaxed objective of each method. Our proposed SGA method closes the discretization gap and achieves the lowest true rate-distortion loss among all methods compared, using the same initialization. *Deterministic Annealing* also closed the discretization gap, but converged to a worse solution. *MAP* naively converged to a

non-integer solution, and *STE* consistently diverged even with a tiny learning rate, as also observed by [Yin et al. \[2019\]](#) (following their results, we changed the gradient of the backward pass from identity to that of ReLU or clipped ReLU; neither helped). *Uniform Noise* produced a consistently negative discretization gap, as the initial μ_y was close to 0, and the optimization did not move it sufficiently far away from the nearby integer solution.

3.3 Marginalization Gap and Lossy Bits-Back Coding

Marginalization Gap. To compress an image x with the hierarchical model of Section 2, ideally we would only need to encode and transmit its latent representation \hat{y} but not the hyper-latents \hat{z} , as only \hat{y} is needed for reconstruction. This would require entropy coding with (a discretization of) the *marginal prior* $p(y) = \int p(z) p(y|z) dz$, which unfortunately is computationally intractable. Therefore, the standard compression approach reviewed in Section 2 instead encodes some hyperlatent \hat{z} using the discretized hyperprior $P(\hat{z})$ first, and then encodes \hat{y} using the entropy model $P(\hat{y}|\hat{z})$. This leads to a *marginalization gap*, which is the difference between the information content $-\log_2 P(\hat{z}, \hat{y}) = -\log_2 P(\hat{z}) - \log_2 P(\hat{y}|\hat{z})$ of the transmitted tuple (\hat{z}, \hat{y}) and the information content of \hat{y} alone,

$$-\log_2 P(\hat{z}, \hat{y}) - (-\log_2 P(\hat{y})) = -\log_2 P(\hat{z}|\hat{y}), \quad (6)$$

i.e., the marginalization gap is the information content of the hyperlatent \hat{z} under the ideal posterior $P(\hat{z}|\hat{y})$, in the (discrete) latent generative process $\hat{z} \rightarrow \hat{y}$.

Related Work. A similar marginalization gap has been addressed in *lossless* compression by bits-back coding [[Wallace, 1990](#), [Hinton and Van Camp, 1993](#)] and made more practical by the BB-ANS algorithm [[Townsend et al., 2019a](#)]. Recent work has improved its efficiency in hierarchical latent variable models [[Townsend et al., 2019b](#), [Kingma et al., 2019](#)], and extended it to flow models [[Ho et al., 2019](#)]. To our knowledge, bits-back coding has not yet been used in lossy compression. This is likely since bits-back coding requires Bayesian posterior inference on the *decoder* side, which seems incompatible with lossy compression, as the decoder does not have access to the undistorted data x .

Lossy Bits-Back Coding. We extend the bits-back idea to lossy compression by noting that the discretized latent representation \hat{y} is encoded losslessly (Figure 1b) and thus amenable to bits-back coding. A complication arises as bits-back coding requires that the encoder’s inference over z can be exactly reproduced by the decoder (see below). This requirement is violated by the inference network in Eq. 1, where $\mu_z = f_h(f(x; \phi); \phi_h)$ depends on x , which is not available to the decoder in lossy compression. It turns out that the naive fix of setting instead $\mu_z = f_h(\hat{y}; \phi_h)$ would hurt performance by more than what bits-back saves (see ablations in Section 4). We propose instead a two-stage inference algorithm that cuts the dependency on x after an initial joint inference over \hat{y} and z .

Bits-back coding bridges the marginalization gap in Eq. 6 by encoding a limited number of bits of some additional side information (e.g., an image caption or a previously encoded image of a slide show) into the choice of \hat{z} using an entropy model $Q(\hat{z}|\mu_z, \sigma_z^2)$ with parameters μ_z and σ_z^2 . We obtain $Q(\hat{z}|\mu_z, \sigma_z^2)$ by discretizing a Gaussian variational distribution $q(z|\mu_z, \sigma_z^2) = \mathcal{N}(z; \mu_z, \text{diag}(\sigma_z^2))$, thus extending the last layer of the hyper-inference network f_h (Eq. 1) to output now a tuple (μ_z, σ_z^2) of means and diagonal variances. This replaces the form of variational distribution $q(z|x)$ of Eq 3 as the restriction to a box-shaped distribution with fixed width is not necessary in bits-back coding. Additionally, we drop the restriction on the shape of the hyperprior $p(z)$ reviewed in Section 2, simply adopting the flexible density model as proposed in [[Ballé et al., 2018](#)] without convolving it with a uniform distribution, as μ_z is no longer discretized to integers. We train the resulting VAE by minimizing the NELBO. Since $q(z|\mu_y, \sigma_y^2)$ now has a variable width, the NELBO has an extra term compared to Eq. 3 that subtracts the entropy of $q(z|\mu_z, \sigma_z^2)$, reflecting the expected number of bits we ‘get back’. Thus, the NELBO is again a relaxed rate-distortion objective, where now the *net* rate is the compressed file size minus the amount of embedded side information.

Algorithm 1 describes lossy compression and decompression with the trained model. Subroutine `encode` initializes the variational parameters μ_y , μ_z , and σ_z^2 conditioned on the target image x using the trained inference networks (line 1). It then jointly performs SGA over \hat{y} by following Section 3.2, and Black-Box Variational Inference (BBVI) over μ_z and σ_z^2 by minimizing the the NELBO (line 2). At the end of the routine, the encoder *decodes* the provided side information ξ (an arbitrary bitstring) into \hat{z} using $Q(\hat{z}|\mu_z, \sigma_z^2)$ as entropy model (line 4) and then encodes \hat{z} and \hat{y} as usual (line 5).

Algorithm 1: Proposed lossy bits-back coding method (Section 3.3 and Figure 1e-f).

Global Constants: Trained hierarchical VAE with model $p(\mathbf{z}, \mathbf{y}, \mathbf{x}) = p(\mathbf{z}) p(\mathbf{y}|\mathbf{z}) p(\mathbf{x}|\mathbf{y})$ and inference networks $f(\cdot; \phi)$ and $f_h(\cdot; \phi_h)$, see Figure 1e (f is only used in subroutine `encode`).

Subroutine `encode`(image \mathbf{x} , side information ξ) \mapsto returns compressed bitstring \mathbf{s}

- 1 Initialize $\mu_y \leftarrow f(\mathbf{x}; \phi)$ and $(\mu_z, \sigma_z^2) \leftarrow f_h(\mu_y; \phi_h)$. \triangleright Figure 1e (blue)
- 2 Optimize over $\hat{\mathbf{y}}$ using SGA (Section 3.2), and over μ_z and σ_z^2 using BBVI. \triangleright Figure 1e (red)
- 3 Reset $(\mu_z, \sigma_z^2) \leftarrow \text{reproducible_BBVI}(\hat{\mathbf{y}})$. \triangleright See below.
- 4 Decode side information ξ into $\hat{\mathbf{z}}$ using $Q(\hat{\mathbf{z}}|\mu_z, \sigma_z^2)$ as entropy model.
- 5 Encode $\hat{\mathbf{z}}$ and $\hat{\mathbf{y}}$ into \mathbf{s} using $P(\hat{\mathbf{z}})$ and $P(\hat{\mathbf{y}}|\hat{\mathbf{z}})$ as entropy models, respectively.

Subroutine `decode`(compressed bitstring \mathbf{s}) \mapsto returns (lossy reconstruction \mathbf{x}' , side info ξ)

- 6 Decode \mathbf{s} into $\hat{\mathbf{z}}$ and $\hat{\mathbf{y}}$; then get reconstructed image $\mathbf{x}' = \arg \max_{\mathbf{x}} p(\mathbf{x}|\hat{\mathbf{y}})$.
- 7 Set $(\mu_z, \sigma_z^2) \leftarrow \text{reproducible_BBVI}(\hat{\mathbf{y}})$. \triangleright See below.
- 8 Encode $\hat{\mathbf{z}}$ into ξ using $Q(\hat{\mathbf{z}}|\mu_z, \sigma_z^2)$ as entropy model.

Subroutine `reproducible_BBVI`(discrete latents $\hat{\mathbf{y}}$) \mapsto returns variational parameters (μ_z, σ_z^2)

- 9 Initialize $(\mu_z, \sigma_z^2) \leftarrow f_h(\hat{\mathbf{y}}; \phi_h)$; seed random number generator reproducibly. \triangleright Figure 1f (blue)
 - 10 Refine μ_z and σ_z^2 by running BBVI for fixed $\hat{\mathbf{y}}$. \triangleright Figure 1f (red)
-

The important step happens on line 3: up until this step, the fitted variational parameters μ_z and σ_z^2 depend on the target image \mathbf{x} due to their initialization on line 1. This would prevent the decoder, which does not have access to \mathbf{x} , from reconstructing the side information ξ by encoding $\hat{\mathbf{z}}$ with the entropy model $Q(\hat{\mathbf{z}}|\mu_z, \sigma_z^2)$. Line 3 therefore re-fits the variational parameters μ_z and σ_z^2 in a way that is exactly reproducible based on $\hat{\mathbf{y}}$ alone. This is done in the subroutine `reproducible_BBVI`, which performs BBVI in the prior model $p(\mathbf{z}, \mathbf{y}) = p(\mathbf{z})p(\mathbf{y}|\mathbf{z})$, treating $\mathbf{y} = \hat{\mathbf{y}}$ as observed and only \mathbf{z} as latent. Although we reset μ_z and σ_z^2 on line 3 immediately after optimizing them on line 2, optimizing jointly over both $\hat{\mathbf{y}}$ and (μ_z, σ_z^2) on line 2 allows the method to find a better $\hat{\mathbf{y}}$.

The decoder decodes $\hat{\mathbf{z}}$ and $\hat{\mathbf{y}}$ as usual (line 6). Since the subroutine `reproducible_BBVI` depends only on $\hat{\mathbf{y}}$ and uses a fixed random seed (line 9), calling it from the decoder on line 7 yields the exact same entropy model $Q(\hat{\mathbf{z}}|\mu_z, \sigma_z^2)$ as used by the encoder, allowing the decoder to recover ξ (line 8).

4 Experiments

We demonstrate the empirical effectiveness of our approach by applying two variants of it—with and without bits-back coding—to an established (but not state-of-the-art) base model [Minnen et al., 2018]. We improve its performance drastically, achieving an average of over 15% BD rate savings on Kodak and 20% on Tecnick [Asuni and Giachetti, 2014], outperforming the previous state-of-the-art of both classical and neural lossy image compression methods. We conclude with ablation studies.

Proposed Methods. Table 1 describes all compared methods, marked as [M1]-[M9] for short. We tested two variants of our method ([M1] and [M2]): SGA builds on the exact same model and training procedure as in [Minnen et al., 2018] (the *Base Hyperprior* model [M3]) and changes only how the trained model is used for compression by introducing hybrid amortized-iterative inference [Campos et al., 2019] (Section 3.1) and Stochastic Gumbel Annealing (Section 3.2). *SGA+BB* adds to it bits-back coding (Section 3.3), which requires changing the inference model over hyperlatents \mathbf{z} to admit a more flexible variational distribution $q(\mathbf{z}|\mathbf{x})$, and no longer restricts the shape of $p(\mathbf{z})$, as discussed in Section 3.3. The two proposed variants address different use cases: SGA is a “standalone” variant of our method that compresses individual images, while *SGA+BB* encodes images with additional side information. In all results, we used Adam [Kingma and Ba, 2015] for optimization, and annealed the temperature of SGA by an exponential decay schedule, and found good convergence without per-model hyperparameter tuning. We provide details in the Supplementary Material.

Baselines. We compare to the *Base Hyperprior* model from [Minnen et al., 2018] without our proposed improvements ([M3] in Table 1), two state-of-the-art neural methods ([M4] and [M5]), two

Table 1: Compared methods ([M1]-[M9]) and ablations ([A1]-[A6]). We propose two variants: [M1] compresses images, and [M2] compresses images + side information via bits-back coding.

Name	Explanation and Reference (for baselines)
ours	[M1] SGA Proposed standalone variant: same trained model as [M3], SGA (Section 3.2) at compression time.
	[M2] SGA + BB Proposed variant with bits-back coding: both proposed improvements of Sections 3.2, and 3.3.
	[M3] Base Hyperprior Base method of our two proposals, reviewed in Section 2 of the present paper [Minnen et al., 2018].
	[M4] Context + Hyperprior Like [M3] but with an extra context model defining the prior $p(\mathbf{y} \mathbf{z})$ [Minnen et al., 2018].
	[M5] Context-Adaptive Context-Adaptive Entropy Model proposed in [Lee et al., 2019].
	[M6] Hyperprior Scale-Only Like [M3] but the hyperlatents \mathbf{z} model only the scale (not the mean) of $p(\mathbf{y} \mathbf{z})$ [Ballé et al., 2018].
	[M7] CAE Pioneering ‘Compressive Autoencoder’ model proposed in [Theis et al., 2017].
	[M8] BPG 4:4:4 State-of-the-art classical lossy image compression codec ‘Better Portable Graphics’ [Bellard, 2014].
	[M9] JPEG 2000 Classical lossy image compression method [Adams, 2001].
ablations	[A1] MAP Like [M1] but with continuous optimization over $\mu_{\mathbf{y}}$ and $\mu_{\mathbf{z}}$ followed by rounding instead of SGA.
	[A2] STE Like [M1] but with straight-through estimation (i.e., round only on backpropagation) instead of SGA.
	[A3] Uniform Noise Like [M1] but with uniform noise injection instead of SGA (see Section 3.1) [Campos et al., 2019].
	[A4] Deterministic Annealing Like [M1] but with deterministic version of Stochastic Gumbel Annealing
	[A5] BB without SGA Like [M2] but without optimization over $\hat{\mathbf{y}}$ at compression time.
	[A6] BB without iterative inference Like [M2] but without optimization over $\hat{\mathbf{y}}$, $\mu_{\mathbf{z}}$, and $\sigma_{\mathbf{z}}^2$ at compression time.

other neural methods ([M6] and [M7]), the state-of-the art classical codec BPG [M8], and JPEG 2000 [M9]. We reproduced the *Base Hyperprior* results, and took the other results from [Ballé et al.].

Results. Figure 4 compares the compression performance of our proposed method to existing baselines on the *Kodak* dataset, using the standard Peak Signal-to-Noise Ratio (PSNR) quality metric (higher is better), averaged over all images for each considered quality setting $\lambda \in [0.001, 0.08]$. The bitrates in our own results ([M1-3], [A1-6]) are based on rate estimates from entropy models, but do not differ significantly from actual file sizes produced by our codec implementation (which has negligible ($< 0.5\%$) overhead). The left panel of Figure 4 plots PSNR vs. bitrate, and the right panel shows the resulting BD rate savings [Bjontegaard, 2001] computed relative to BPG as a function of PSNR for readability (higher is better). The BD plot cuts out CAE [M7] and JPEG 2000 [M9] at the bottom, as they performed much worse. Overall, both variants of the proposed method (blue and orange lines in Figure 4) improve substantially over the *Base Hyperprior* model (brown), and outperform the previous state-of-the-art. We report similar results on the Tecnick dataset [Asuni and Giachetti, 2014] in the Supplementary Material. Figure 6 shows a qualitative comparison of a compressed image (in the order of BPG, SGA (proposed), and the *Base Hyperprior*), in which we see that our method notably enhances the baseline image reconstruction at comparable bitrate, while avoiding unpleasant visual artifacts of BPG.

Ablations. Figure 5 compares BD rate improvements of the two variants of our proposal (blue and orange) to six alternative choices (ablations [A1]-[A6] in Table 1), measured relative to the *Base Hyperprior* model [Minnen et al., 2018] (zero line; [M3] in Table 1), on which our proposals build. [A1]-[A4] replace the discretization method of SGA [M1] with four alternatives discussed at the end of Section 3.2. [A5] and [A6] are ablations of our proposed bits-back variant SGA+BB [M2], which remove iterative optimization over $\hat{\mathbf{y}}$, or over $\hat{\mathbf{y}}$ and $q(\mathbf{z})$, respectively, at compression time. Going from red [A3] to orange [A2] to blue [M1] traces our proposed improvements over [Campos et al., 2019] by adding SGA (Section 3.2) and then bits-back (Section 3.3). It shows that iterative inference

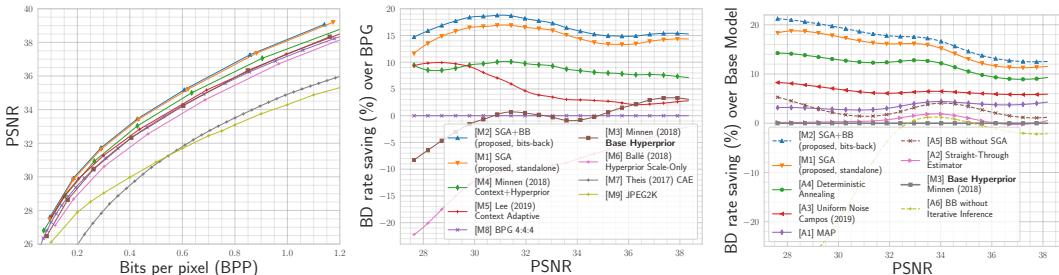


Figure 4: Compression performance comparisons on *Kodak* against existing baselines. Left: R-D curves. Right: BD rate savings (%) relative to BPG. Legend shared; higher values are better in both.

Figure 5: BD rate savings of various ablation methods relative to *Base Hyperprior* on *Kodak*.

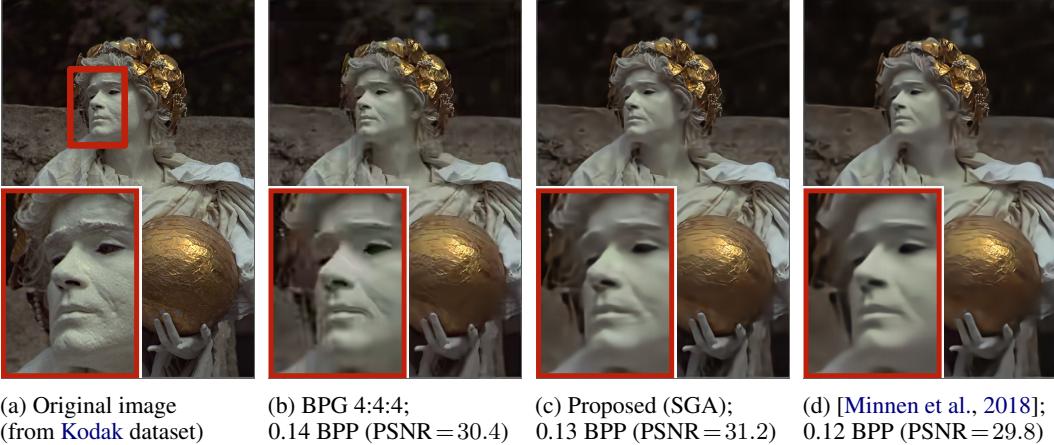


Figure 6: Qualitative comparison of lossy compression performance. Our method (c; [M1] in Table 1) significantly boosts the visual quality of the *Base Hyperprior* method (d; [M3] in Table 1) at similar bit rates. It sharpens details of the hair and face (see insets) obscured by the baseline method (d), while avoiding the ringing artifacts around the jaw and ear pendant produced by a classical codec (b).

(Section 3.1) and SGA contribute approximately equally to the performance gain, whereas the gain from bits-back coding is smaller. Interestingly, lossy bits-back coding without iterative inference and SGA actually hurts performance [A6]. As Section 3.3 mentioned, this is likely because bits-back coding constrains the posterior over \mathbf{z} to be conditioned only on $\hat{\mathbf{y}}$ and not on the original image \mathbf{x} .

5 Discussion

Starting from the variational inference view on data compression, we proposed three enhancements to the standard inference procedure in VAEs: hybrid amortized-iterative inference, Stochastic Gumbel Annealing, and lossy bits-back coding, which translated to dramatic performance gains on lossy image compression. Improved inference provides a new promising direction to improved compression, orthogonal to modeling choices (e.g., with auto-regressive priors [Minnen et al., 2018], which can harm decoding efficiency). Although lossy-bits-back coding in the present VAE only gave relatively minor benefits, it may reach its full potential in more hierarchical architectures as in [Kingma et al., 2019]. Similarly, carrying out iterative inference also at *training time* may lead to even more performance improvement, with techniques like iterative amortized inference [Marino et al., 2018].

Broader Impacts

Improved image and video compression is becoming more and more important as our lives become more digital. For instance, video compression is of enormous societal relevance, as over 80% of web traffic is due to video streaming, and the share of video data is expected to increase even further in the future [Cisco, 2017]. Moreover, there is an explosion of different new data formats that need to be efficiently compressed; examples include high resolution medical images, lidar data, 3D graphics data, DNA sequences, etc. It would be very costly to design custom codecs for such data, making learnable compression indispensable. Ultimately, better compression algorithms, both for data and models, will also facilitate the possibility to carry-out machine learning on local devices, as opposed to large centralized servers. This may address many of our privacy concerns.

Currently, neural compression does come with some downsides compared to hand-engineered codecs, such as higher computation/energy cost, and lack of performance guarantees, which we hope can be overcome by active research in this area. Our work on improving the encoding procedure already has the potential to lower the *overall* energy consumption of a neural codec, for applications where the increased cost of encoding can be amortized across many file transfer/decoding operations and translate to net savings, e.g., on a content hosting website that has millions of visitors per day.

Acknowledgments and Disclosure of Funding

We thank Yang Yang for valuable feedback on the manuscript. Yibo Yang acknowledges funding from the Hasso Plattner Foundation. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001120C0021. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA). Furthermore, this work was supported by the National Science Foundation under Grants 1928718, 2003237 and 2007719, and by Qualcomm. Stephan Mandt consulted for Disney and Google.

References

- Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *International Conference on Learning Representations*, 2018.
- David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *Advances in Neural Information Processing Systems*, pages 10771–10780, 2018.
- Jooyoung Lee, Seunghyun Cho, and Seung-Kwon Beack. Context-adaptive entropy model for end-to-end optimized image compression. In *the 7th Int. Conf. on Learning Representations*, May 2019.
- Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A Saurous, and Kevin Murphy. Fixing a broken ELBO. In *International Conference on Machine Learning*, pages 159–168. PMLR, 2018.
- Salvator Lombardo, Jun Han, Christopher Schroers, and Stephan Mandt. Deep generative video compression. In *Advances in Neural Information Processing Systems 32*, pages 9287–9298, 2019.
- Amirhossein Habibian, Ties van Rozendaal, Jakub M Tomczak, and Taco S Cohen. Video compression with rate-distortion autoencoders. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7033–7042, 2019.
- Glenn Randers-Pehrson. Mng: a multiple-image format in the png family. *World Wide Web Journal*, 2(1):209–211, 1997.
- Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286, 2014.
- Joaquim Campos, Simon Meierhans, Abdelaziz Djelouah, and Christopher Schroers. Content adaptive optimization for neural image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019.
- Chris S Wallace. Classification by minimum-message-length inference. In *International Conference on Computing and Information*, pages 72–81. Springer, 1990.
- Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13, 1993.
- Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. *International Conference on Learning Representations*, 2017.
- Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. *International Conference on Learning Representations*, 2017.
- Nick Johnston, Elad Eban, Ariel Gordon, and Johannes Ballé. Computationally efficient neural image compression. *arXiv preprint arXiv:1912.08771*, 2019.

- David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.
- Cheng Zhang, Judith Butepage, Hedvig Kjellstrom, and Stephan Mandt. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 2019.
- Matthew J Beal and Zoubin Ghahramani. The variational Bayesian EM algorithm for incomplete data: with application to scoring graphical model structures. In *Bayesian Statistics 7: Proceedings of the Seventh Valencia International Meeting*, volume 7, pages 453–464. Oxford University Press, 2003.
- Chris Cremer, Xuechen Li, and David Duvenaud. Inference suboptimality in variational autoencoders. In *International Conference on Machine Learning*, pages 1078–1086, 2018.
- Rahul Krishnan, Dawen Liang, and Matthew Hoffman. On the challenges of learning with inference networks on sparse, high-dimensional data. In *International Conference on Artificial Intelligence and Statistics*, pages 143–151, 2018.
- Devon Hjelm, Russ R Salakhutdinov, Kyunghyun Cho, Nebojsa Jojic, Vince Calhoun, and Junyoung Chung. Iterative refinement of the approximate posterior for directed belief networks. In *Advances in Neural Information Processing Systems*, pages 4691–4699, 2016.
- Yoon Kim, Sam Wiseman, Andrew Miller, David Sontag, and Alexander Rush. Semi-amortized variational autoencoders. In *International Conference on Machine Learning*, pages 2678–2687, 2018.
- Joe Marino, Yisong Yue, and Stephan Mandt. Iterative amortized inference. In *Proceedings of the 35th International Conference on Machine Learning*, pages 3403–3412, 2018.
- Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.
- Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *arXiv preprint arXiv:1711.00937*, 2017.
- Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley Osher, Yingyong Qi, and Jack Xin. Understanding straight-through estimator in training activation quantized neural nets. *arXiv preprint arXiv:1903.05662*, 2019.
- George Toderici, Sean M O’Malley, Sung Jin Hwang, Damien Vincent, David Minnen, Shumeet Baluja, Michele Covell, and Rahul Sukthankar. Variable rate image compression with recurrent neural networks. *International Conference on Learning Representations*, 2016.
- Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc V Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. In *Advances in Neural Information Processing Systems*, 2017.
- Yibo Yang, Robert Bamler, and Stephan Mandt. Variational Bayesian Quantization. In *International Conference on Machine Learning*, 2020.
- Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.
- Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.
- Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.
- Eastman Kodak. Kodak lossless true color image suite (PhotoCD PCD0992). URL <http://r0k.us/graphics/kodak>.
- James Townsend, Tom Bird, and David Barber. Practical lossless compression with latent variables using bits back coding. *arXiv preprint arXiv:1901.04866*, 2019a.

- James Townsend, Thomas Bird, Julius Kunze, and David Barber. Hilloc: Lossless image compression with hierarchical latent variable models. *arXiv preprint arXiv:1912.09953*, 2019b.
- Friso H Kingma, Pieter Abbeel, and Jonathan Ho. Bit-swap: Recursive bits-back coding for lossless compression with hierarchical latent variables. *arXiv preprint arXiv:1905.06845*, 2019.
- Jonathan Ho, Evan Lohn, and Pieter Abbeel. Compression with flows via local bits-back coding. In *Advances in Neural Information Processing Systems*, pages 3874–3883, 2019.
- N. Asuni and A. Giachetti. TESTIMAGES: A large-scale archive for testing visual devices and basic image processing algorithms (SAMPLING 1200 RGB set). In *STAG: Smart Tools and Apps for Graphics*, 2014. URL https://sourceforge.net/projects/testimages/files/OLD/OLD_SAMPLING/testimages.zip.
- Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic gradient descent. In *International Conference on Learning Representations*, 2015.
- Fabrice Bellard. BPG specification, 2014. URL https://bellard.org/bpg/bpg_spec.txt. (accessed June 3, 2020).
- Michael D Adams. The jpeg-2000 still image compression standard. 2001.
- Johannes Ballé, Sung Jin Hwang, Nick Johnston, and David Minnen. Tensorflow-compression: Data compression in tensorflow. URL <https://github.com/tensorflow/compression>.
- Gisle Bjontegaard. Calculation of average psnr differences between rd-curves. *VCEG-M33*, 2001.
- Visual Network Index Cisco. Forecast and methodology, 2016-2021. *White Paper*, 2017.

Supplementary Material to Improving Inference for Neural Compression

S1 Stochastic Annealing

Here we provide conceptual illustrations of our stochastic annealing idea on a simple example.

Consider the following scalar optimization problem over integers,

$$\underset{z \in \mathbb{Z}}{\text{minimize}} \quad f(z) = z^2$$

Following our stochastic annealing method in Section 3.2, we let $x \in \mathbb{R}$ be a continuous proxy variable, and $r \in \{(1, 0), (0, 1)\}$ be the one-hot vector of stochastic rounding direction. We let r follow a Bernoulli distribution with temperature $\tau > 0$,

$$q_\tau(r|x) = \begin{cases} \exp\{-\psi(x - \lfloor x \rfloor)/\tau\}/C(x, \tau) & \text{if } r = (1, 0) \\ \exp\{-\psi(\lceil x \rceil - x)/\tau\}/C(x, \tau) & \text{if } r = (0, 1) \end{cases}$$

where

$$C(x, \tau) = \exp\{-\psi(x - \lfloor x \rfloor)/\tau\} + \exp\{-\psi(\lceil x \rceil - x)/\tau\}$$

is the normalizing constant.

The stochastic optimization problem for SGA is then

$$\begin{aligned} \underset{x \in \mathbb{R}}{\text{minimize}} \quad & \ell_\tau(x) := \mathbb{E}_{q_\tau(r|x)}[f(r \cdot (\lfloor x \rfloor, \lceil x \rceil))] \\ &= \frac{\exp\{-\psi(x - \lfloor x \rfloor)/\tau\}}{C(x, \tau)} f(\lfloor x \rfloor) + \frac{\exp\{-\psi(\lceil x \rceil - x)/\tau\}}{C(x, \tau)} f(\lceil x \rceil) \end{aligned}$$

Below we plot the rounding probability $q_\tau(r = (0, 1)|x)$ and the SGA objective ℓ_τ as a function of x , at various temperatures.

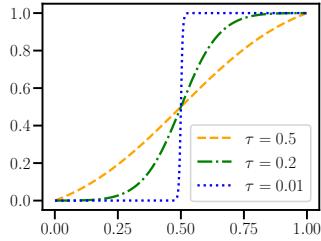


Figure S1: Illustration of the probability $q_\tau(r = (0, 1)|x)$ of rounding *up*, on the interval $(0, 1)$, at various temperatures.

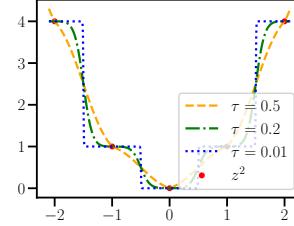


Figure S2: Graphs of stochastic annealing objective ℓ_τ at various temperatures. ℓ_τ interpolates the discrete optimization problem between integers, and approaches $\text{round}(x)^2$ as $\tau \rightarrow 0$.

S2 Model Architecture and Training

As mentioned in the main text, the *Base Hyperprior* model uses the same architecture as in Table 1 of Minnen et al. [2018] except without the ‘‘Context Prediction’’ and ‘‘Entropy Parameters’’ components (this model was referred to as ‘‘Mean & Scale Hyperprior’’ in this work). The model is mostly already implemented in `bmshj2018.py`¹ from Ballé et al., Ballé et al. [2018], and we modified their implementation to double the number of output channels of the `HyperSynthesisTransform` to predict both the mean and (log) scale of the conditional prior $p(\mathbf{y}|\mathbf{z})$ over the latents \mathbf{y} .

As mentioned in Section 3.3 and 4, lossy bits-back in this model makes the following modifications:

¹<https://github.com/tensorflow/compression/blob/master/examples/bmshj2018.py>

1. The number of output channels of the hyper inference network is doubled to compute both the mean and diagonal (log) variance of Gaussian $q(\mathbf{z}|\mathbf{x})$;
2. The hyperprior $p(\mathbf{z})$ is no longer restricted to the form of a flexible density model convolved with the uniform distribution on $[-0.5, 0.5]$; instead it simply uses the flexible density, as described in the Appendix of Ballé et al. [2018].

We trained the above models on CLIC-2018² images using minibatches of eight 256×256 randomly-cropped image patches, for $\lambda \in \{0.001, 0.0025, 0.005, 0.01, 0.02, 0.04, 0.08\}$. Following Ballé et al. [2018], we found that increasing the number of latent channels helps avoid the “bottleneck” effect and improve rate-distortion performance at higher rates, and increased the number of latent channels from 192 to 256 for models trained with $\lambda = 0.04$ and 0.08 . All models were trained for 2 million steps, except the ones with $\lambda = 0.001$ which were trained for 1 million steps, and $\lambda = 0.08$ which were trained for 3 million steps. Our code and pre-trained models can be found at <https://github.com/mandt-lab/improving-inference-for-neural-image-compression>.

S3 Hyper-Parameters of Various Methods Considered

Given a pre-trained model (see above) and image(s) \mathbf{x} to be compressed, our experiments in Section 4 explored hybrid amortized-iterative inference to improve compression performance. Below we provide hyper-parameters of each method considered:

Stochastic Gumbel Annealing. In all experiments using SGA (including the *SGA+BB* experiments, which optimized over $\hat{\mathbf{y}}$ using SGA), we used the Adam optimizer with an initial learning rate of 0.005, and an exponentially decaying temperature schedule $\tau(t) = \min(\exp\{-ct\}, 0.5)$, where t is the iteration number, and $c > 0$ controls the speed of the decay. We found that lower c generally increases the number of iterations needed for convergence, but can also yield slightly better solutions; in all of our experiments we set $c = 0.001$, and obtained good convergence with 2000 iterations.

Temperature Annealing Schedule for SGA. We initially used a naive temperature schedule, $\tau(t) = \tau_0 \exp\{-ct\}$ for SGA, where τ_0 is the initial temperature (typically set to 0.5 to simulate soft quantization), and found that aggressive annealing with a large decay factor c can lead to suboptimal solutions, as seen with $c = 0.002$ or $c = 0.001$ in the left subplot of Figure S3. We found that we can overcome the suboptimality from fast annealing by fixing the temperature to τ_0 for some initial number t_0 of steps (until the R-D objective roughly converges) before annealing. We demonstrate this in the right subplot of Figure S3, where we modify the naive schedule to $\tau(t) = \min\{\tau_0, \tau_0 \exp\{-c(t - t_0)\}\}$, so that SGA roughly converges after $t_0 = 700$ steps at temperature $\tau_0 = 0.5$ before annealing; as can be seen, the results are robust to different choices of decay factor c , with $c = 0.002$ and $c = 0.001$ giving comparably good results to $c = 0.0005$ but with faster convergence.

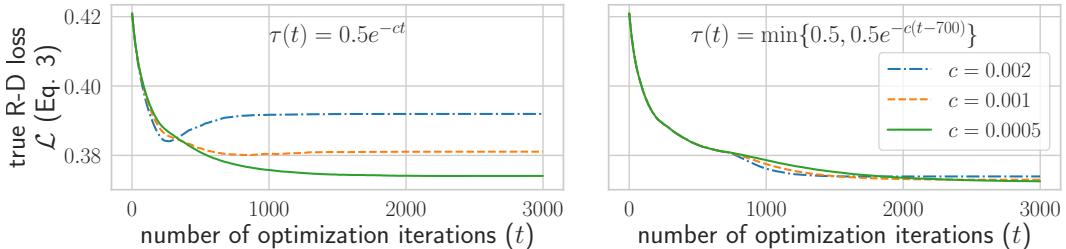


Figure S3: Comparing different annealing schedules for SGA (Section 3.2).

Lossy bits-back. In the *SGA+BB* experiments, line 2 of the `encode` subroutine of Algorithm 1 performs joint optimization w.r.t. μ_y , μ_z , and σ_z^2 with Black-box Variational Inference (BBVI), using the reparameterization trick to differentiate through Gumbel-softmax samples for μ_y , and Gaussian samples for (μ_z, σ_z^2) . We used Adam with an initial learning rate of 0.005, and ran 2000 stochastic

²<https://www.compression.cc/2018/challenge/>

gradient descent iterations; the optimization w.r.t. μ_y (SGA) used the same temperature annealing schedule as above. The `reproducible_BBVI` subroutine of Algorithm 1 used Adam with an initial learning rate of 0.003 and 2000 stochastic gradient descent iterations; we used the same random seed for the encoder and decoder for simplicity, although a hash of \hat{y} can also be used instead.

Alternative Discretization Methods In Figure 3 and ablation studies of Section 4, all alternative discretization methods were optimized with Adam for 2000 iterations to compare with SGA. The initial learning rate was 0.005 for *MAP*, *Uniform Noise*, and *Deterministic Annealing*, and 0.0001 for *STE*. All methods were tuned on a best-effort basis to ensure convergence, except that *STE* consistently encountered convergence issues even with a tiny learning rate (see [Yin et al., 2019]). The rate-distortion results for *MAP* and *STE* were calculated with early stopping (i.e., using the intermediate (\hat{y}, \hat{z}) with the lowest true rate-distortion objective during optimization), just to give them a fair chance. Lastly, the comparisons in Figure 3 used the *Base Hyperprior* model trained with $\lambda = 0.0025$.

S4 Additional Results

On Kodak, we achieve the following BD rate savings: 17% for *SGA+BB* and 15% for *SGA* relative to *Base Hyperprior*; 17% for *SGA+BB* and 15% for *SGA* relative to BPG. On Tecnick, we achieve the following BD rate savings: 20% for *SGA+BB* and 19% for *SGA* relative to *Base Hyperprior*; 22% for *SGA+BB* and 21% for *SGA* relative to BPG.

Our experiments were conducted on a Titan RTX GPU with 24GB RAM; we observed about a $100\times$ slowdown from our proposed iterative inference methods (compared to standard encoder network prediction), similar to what has been reported in [Campos et al., 2019]. Note that our proposed standalone variant [M1] with SGA changes only compression and does not affect *decompression* speed (which is more relevant, e.g., for images on a website with many visitors).

Below we provide an additional qualitative comparison on the *Kodak* dataset, and report detailed rate-distortion performance on the *Tecnick* [Asuni and Giachetti, 2014] dataset.

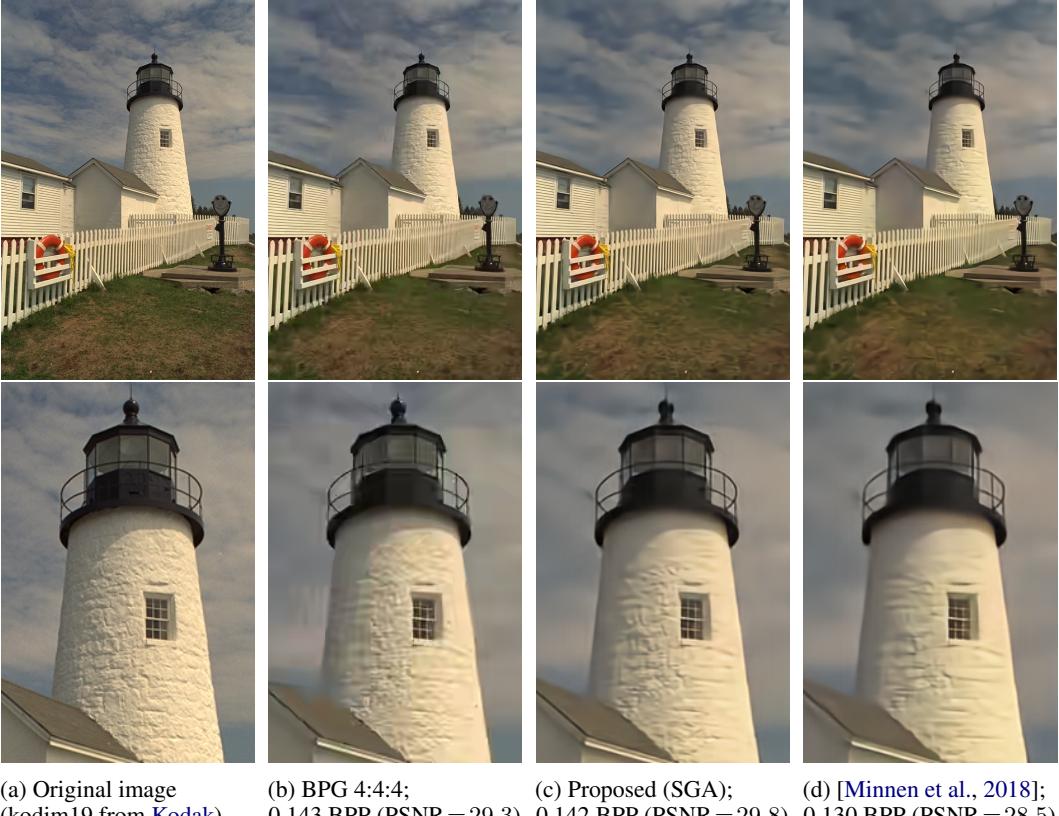


Figure S4: Qualitative comparison of lossy compression performance on an image from the **Kodak** dataset. Figures in the bottom row focus on the same cropped region of images in the top row. Our method (c; [M1] in Table 1) significantly boosts the visual quality of the *Base Hyperprior* method (d; [M3] in Table 1) at similar bit rates. Unlike the learning-based methods (subfigures c,d), the classical codec BPG (subfigure b) introduces blocking and geometric artifacts near the rooftop, and ringing artifacts around the contour of the lighthouse.

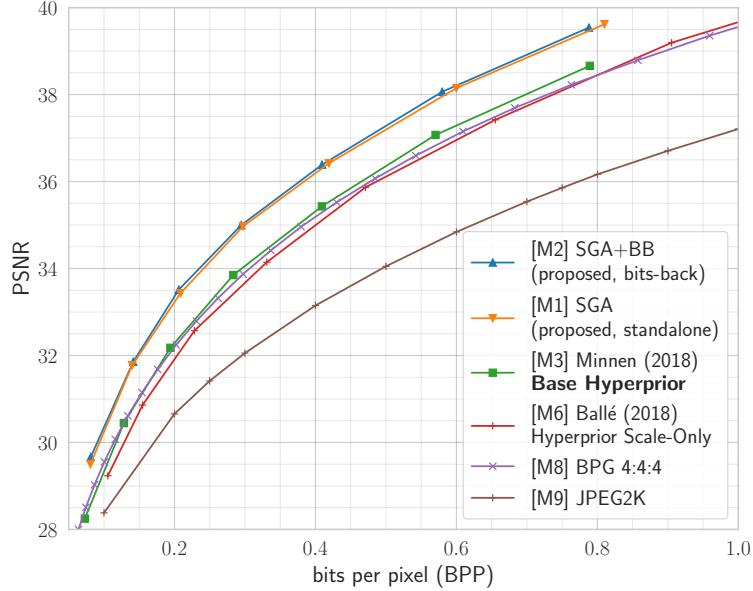


Figure S5: Rate-distortion performance comparisons on the Tecnick [[Asuni and Giachetti, 2014](#)] dataset against existing baselines. Image quality measured in Peak Signal-to-Noise Ratio (PSNR) in RGB; higher values are better.

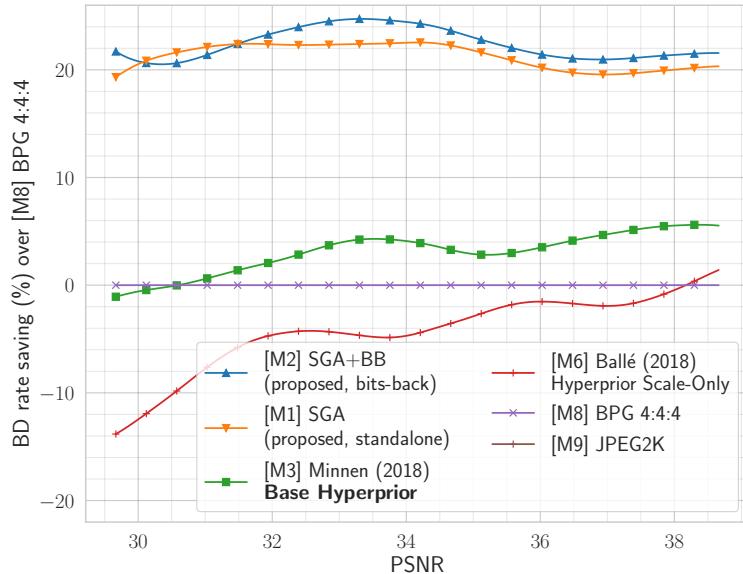


Figure S6: BD rate savings (%) relative to BPG as a function of PSNR, computed from R-D curves (Figure S5) on Tecnick. Higher values are better.