Variable-Bitrate Neural Compression via Bayesian Arithmetic Coding

Yibo Yang [*1] **Robert Bamler** [*1] **Stephan Mandt** [1]

Abstract

Deep Bayesian latent variable models have enabled new approaches to both model and data compression. Here, we propose a new algorithm for compressing latent representations in deep probabilistic models, such as variational autoencoders, in post-processing. The approach thus separates model design and training from the compression task. Our algorithm generalizes arithmetic coding to the continuous domain, using adaptive discretization accuracy that exploits estimates of posterior uncertainty. A consequence of the "plug and play" nature of our approach is that various rate-distortion trade-offs can be achieved with a single trained model, eliminating the need to train multiple models for different bit rates. Our experimental results demonstrate the importance of taking into account posterior uncertainties, and show that image compression with the proposed algorithm outperforms JPEG over a wide range of bit rates using only a single machine learning model. Further experiments on Bayesian neural word embeddings demonstrate the versatility of the proposed method.

1. Introduction

Probabilistic latent-variable models have become a mainstay of modern machine learning. Scalable approximate Bayesian inference methods, in particular Black Box Variational Inference (Ranganath et al., 2014; Rezende et al., 2014), have spurred the development of increasingly large and expressive probabilistic models, including deep generative probabilistic models such as variational autoencoders (Kingma & Welling, 2014b) and Bayesian neural networks (MacKay, 1992; Blundell et al., 2015). One natural application of deep latent variable modeling is data compression, and recent work has focused on end-to-end procedures that optimize a model for a particular compression objective. Here, we study a related but different problem: given a *trained* model, what is the best way to encode the information contained in its continuous latent variables?

As we demonstrate, our proposed solution provides an entirely new "plug & play" approach to lossy compression that separates the compression task from modeling and training. Our method can be applied to both model and data compression, and allows tuning the trade-off between bitrate and reconstruction quality without the need to retrain the model.

Compression aims to best describe some data in as few bits as possible. For continuous-valued data like natural images, videos, or distributed representations, digital compression is necessarily *lossy*, as arbitrary real numbers cannot be perfectly represented by a finite number of bits. Lossy compression algorithms therefore typically find a discrete approximation of some semantic representation of the data, which is then encoded with a lossless compression method.

In classical lossy compression methods such as JPEG or MP3, the semantic representation is carefully designed to support compression at variable bitrates. By contrast, state-of-the-art deep learning based approaches to lossy data compression (Ballé et al., 2017; 2018; Rippel & Bourdev, 2017; Mentzer et al., 2018; Lombardo et al., 2019) are trained to minimize a distortion metric at a fixed bitrate. To support variable-bitrate compression, one has to train several models for different bitrates. While training several models may be viable in many cases, a bigger issue of this approach is the increase in decoder size as the decoder has to store the parameters of not one but several deep neural networks for each bitrate setting. In applications like video streaming under fluctuating connectivity, the decoder further has to load a new deep learning model into memory every time a change in bandwidth requires adjusting the bitrate.

By contrast, we propose a lossy neural compression method that decouples training from compression, and that enables variable-bitrate compression with a single model. We generalize a classical entropy coding algorithm, Arithmetic Coding (Witten et al., 1987; MacKay, 2003), from discrete data to the continuous domain. At the heart of the proposed Bayesian Arithmetic Coding algorithm is an adaptive quantization scheme that exploits posterior uncertainty estimates to automatically reduce the accuracy of latent variables for which the model is uncertain anyway. This strategy is analo-

[*]Equal contribution [1]Department of Computer Science, University of California, Irvine. Correspondence to: Yibo Yang <yibo.yang@uci.edu>, Robert Bamler <rbamler@uci.edu>.

gous to the way humans communicate quantitative information. For example, Wikipedia lists the population of Rome in 2017 with the specific number 2,879,728. By contrast, its population in the year 500 AD is estimated by the rounded number 100,000 because the high uncertainty would make a more precise number meaningless. Our ablation studies show that this posterior-informed quantization scheme is crucial to obtaining competitive performance.

In detail, our contributions are as follows:

- *A new algorithm.* We present a fundamentally novel approach to compressing latent variables in a variational inference framework. Our approach generalizes arithmetic coding from discrete to continuous distributions and takes posterior uncertainty into account.
- *Single-model compression at variable bitrates.* The decoupling of modeling and compression allows us to adjust the trade-off between bitrate and distortion in post-processing. This is in contrast to existing approaches to both data and model compression, which often require specialized models for each bitrate.
- *Automatic self-pruning.* Deep latent variable models often exhibit posterior collapse, i.e., the variational posterior collapses to the model prior. In our approach, latent dimensions with collapsed posteriors require close to zero bits, thus don't require manual pruning.
- *Competitive experimental performance.* We show that our method outperforms JPEG over a wide range of bitrates using only a single model. We also show that we can successfully compress a word embeddings with minimal loss, as evaluated on semantic reasoning task.

The paper is structured as follows: Section 2 discusses related work in neural compression; Section 3 describes our proposed Bayesian Arithmetic Coding algorithm. We give empirical results in Section 4, and conclude in Section 5.

2. Related Work

Compressing continuous-valued data is a classical problem in the signal processing community. Typically, a distortion measure (often the squared error) and a source distribution are assumed, and the goal is to design a quantizer that optimizes the rate-distortion (R-D) performance (Lloyd, 1982; Berger, 1972; Chou et al., 1989). Optimal vector quantization, although theoretically well-motivated (Gallager, 1968), is not tractable in high-dimensional spaces (Gersho & Gray, 2012) and not scalable in practice. Therefore most classical lossy compression algorithms map data to a suitably designed semantic representation, in such a way that coordinate-wise scalar quantization can be fruitfully applied.

Recent machine-learning-based data compression methods learn such hand-designed representation from data, but similar to classical methods, most such ML methods directly take quantization into account in the generative model design or training. Various approaches replace the non-differentiable quantization operation with either stochastic binarization (Toderici et al., 2016; 2017), additive uniform noise (Ballé et al., 2017; 2018; Habibian et al., 2019), or other differentiable approximation (Agustsson et al., 2017; Theis et al., 2017; Mentzer et al., 2018; Rippel & Bourdev, 2017); many such schemes result in uniform quantization of the latent variables, with the exception of (Agustsson et al., 2017), which optimizes for quantization grid points.

We depart from such approaches by considering quantization as a post-processing step that decouples quantization from model design and training. An important feature of our algorithm is a new quantization scheme that automatically adapts to different length scales in the representation space by exploiting posterior uncertainty estimates. To the best of our knowledge, the only prior work that uses posterior uncertainty for compression is in the context of bits-back coding (Honkela & Valpola, 2004; Townsend et al., 2019), but these works focus on lossless compression.

Most existing neural image compression methods require training a separate machine learning model for each desired bitrate setting (Ballé et al., 2017; 2018; Mentzer et al., 2018; Theis et al., 2017; Lombardo et al., 2019). In fact, Alemi et al. (2018) showed that any particular fitted VAE model only targets one specific point on the rate-distortion curve. One approach has the same goal of variable-bitrate single-model compression in mind as methods based on recurrent VAEs (Gregor et al., 2016; Toderici et al., 2016; 2017; Johnston et al., 2018), which use dedicated model architecture for progressive image reconstruction; but instead focus more broadly on lossy compression for any given generative model, designed and trained for specific application purposes (possibly other than compression).

3. Posterior-Informed Variable-Bitrate Compression

We now propose an algorithm for compressing latent variables in trained models. After describing the problem setup and assumptions (Subsection 3.1), we briefly review Arithmetic Coding (Subection 3.2). Subsection 3.3 describes our proposed lossy compression algorithm, which generalizes Arithmetic Coding to the continuous domain.

3.1. Problem Setup

Generative Model and Variational Inference. We consider a wide class of generative probabilistic models with data $\mathbf{x}$ and unknown (or "latent") variables $\mathbf{z} \in \mathbb{R}^K$ from

some continuous latent space with dimension K. The generative model is defined by a joint probability distribution,

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})\, p(\mathbf{x}|\mathbf{z}) \tag{1}$$

with a prior $p(\mathbf{z})$ and a likelihood $p(\mathbf{x}|\mathbf{z})$. Although our presentation focuses on unsupervised representation learning, our framework also captures the supervised setup.[1]

Our proposed compression method uses $\mathbf{z}$ as a proxy to describe the data $\mathbf{x}$. This requires "solving" Eq. 1 for $\mathbf{z}$ given $\mathbf{x}$, i.e., inferring the posterior $p(\mathbf{z}|\mathbf{x}) = p(\mathbf{x}, \mathbf{z}) / \int p(\mathbf{x}, \mathbf{z})\, d\mathbf{z}$. Since exact Bayesian inference is often intractable, we resort to Variational Inference (VI) (Jordan et al., 1999; Blei et al., 2017; Zhang et al., 2019), which approximates the posterior by a so-called variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$ by minimizing the Kullback-Leibler divergence $D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \,||\, p(\mathbf{z}|\mathbf{x}))$ over a set of variational parameters ϕ.

Factorization Assumptions. We assume that both the prior $p(\mathbf{z})$ and the variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$ are fully factorized (mean-field assumption). For concreteness, our examples use a Gaussian variational distribution. Thus,

$$p(\mathbf{z}) = \textstyle\prod_{i=1}^{K} p(z_i); \qquad \text{and} \tag{2}$$

$$q_\phi(\mathbf{z}|\mathbf{x}) = \textstyle\prod_{i=1}^{K} \mathcal{N}(z_i; \mu_i(\mathbf{x}), \sigma_i^2(\mathbf{x})), \tag{3}$$

where $p(z_i)$ is a prior for the i^{th} component of $\mathbf{z}$, and the means μ_i and standard deviations σ_i together comprise the variational parameters ϕ over which VI optimizes.[2]

Prominently, the model class defined by Eqs. 1-3 includes variational autoencoders (VAEs) (Kingma & Welling, 2014a) for data compression, but we stress that the class is much wider, capturing also Bayesian neural nets (MacKay, 2003), probabilistic word embeddings (Barkan, 2017; Bamler & Mandt, 2017), matrix factorization (Mnih & Salakhutdinov, 2008), and topic models (Blei et al., 2003).

Protocol Overview. We consider two parties in communication, a sender and a receiver. Given a probabilistic model (Eq. 1), the goal is to transmit a data sample $\mathbf{x}$ as efficiently as possible. Both parties have access to the model, but only the sender has access to $\mathbf{x}$, which it uses to fit a variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$. It then uses the algorithm proposed below to select a latent variable vector $\hat{\mathbf{z}}$ that has high probability under q_ϕ, and that can be encoded into a compressed bitstring, which gets transmitted to the receiver. The receiver losslessly decodes the compressed bitstring back into $\hat{\mathbf{z}}$ and uses the likelihood $p(\mathbf{x}|\hat{\mathbf{z}})$ to generate a reconstructed data point $\hat{\mathbf{x}}$, typically setting $\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} p(\mathbf{x}|\hat{\mathbf{z}})$.

[1] For supervised learning with labels y, we would consider a conditional generative model $p(y, \mathbf{z}|\mathbf{x}) = p(y|\mathbf{z}, \mathbf{x})\, p(\mathbf{z})$ with conditional likelihood $p(y|\mathbf{z}, \mathbf{x})$, where $\mathbf{z}$ are the model parameters, treated as a Bayesian latent variable with associated prior $p(\mathbf{z})$.

[2] These parameters are often amortized by a neural network (in which case μ_i and σ_i depend on $\mathbf{x}$), but don't have to (in which case μ_i and σ_i do not depend on $\mathbf{x}$ and are directly optimized).

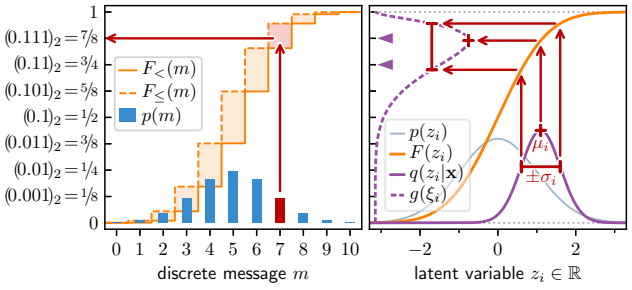


Figure 1. Comparison of standard Arithmetic Coding (AC, left) and Bayesian AC (right, proposed). Both methods use a prior CDF (orange) to map nonuniformly distributed data to a number $\xi \sim \mathcal{U}(0, 1)$, and require an uncertainty region for truncation.

The rest of this section describes how the proposed algorithm selects $\hat{\mathbf{z}}$ and encodes it into a compressed bitstring.

3.2. Background: Arithmetic Coding

Our lossy compression algorithm, introduced in Section 3.3 below, generalizes a lossless compression algorithm, *arithmetic coding* (AC) (Witten et al., 1987; MacKay, 2003), from discrete data to the continuous space of latent variables $\mathbf{z} \in \mathbb{R}^K$. To get there, we first review the main idea of AC that our proposed algorithm borrows.

AC is an instance of so-called entropy coding. It uniquely maps messages $m \in \mathcal{M}$ from a discrete set $\mathcal{M}$ to a compressed bitstring of some length $\mathcal{R}_m$ (the "bitrate"). Entropy coding exploits prior knowledge of the distribution $p(m)$ of messages to map probable messages to short bitstrings while spending more bits on improbable messages. This way, entropy coding algorithms aim to minimize the expected rate $\mathbb{E}_{p(m)}[\mathcal{R}_m]$. For lossless compression, the expected rate has a fundamental lower bound, the entroy $H = \mathbb{E}_{p(m)}[h(m)]$, where $h(m) = -\log_2 p(m)$ is the Shannon information content of m. AC provides near optimal lossless compression as it maps each message $m \in \mathcal{M}$ to a bitstring of length $\mathcal{R}_m = \lceil h(m) \rceil$, where $\lceil \cdot \rceil$ denotes the ceiling function.

AC is usually discussed in the context of streaming compression where m is a sequence of symbols from a finite alphabet, as AC improves on this task over the more widely known Huffman coding (Huffman, 1952). In our work, we focus on a different aspect of AC: its use of a cumulative probability distribution function to map a nonuniformly distributed random variable $m \sim p(m)$ to a number ξ that is nearly uniformly distributed over the interval $[0, 1)$.

Figure 1 (left) illustrates AC for a binomial-distributed message $m \in \{0, \dots, 10\}$ (the number of 'heads' in a sequence of ten coin flips). The solid and dashed orange lines show

the left and right sided cumulative distribution function,[3] $F_<(m) := \sum_{m'<m} p(m')$ and $F_\leq(m) := \sum_{m'\leq m} p(m')$, respectively. They define a partitioning of the interval $[0, 1)$ (vertical axis in Figure 1 (left)) into pairwise disjoint subintervals $\mathcal{I}_m := \big[F_<(m), F_\leq(m)\big)$ (orange squares). Since the intervals $\mathcal{I}_m$ are disjoint for all $m \in \mathcal{M}$, any number $\xi \in \mathcal{I}_m$ uniquely identifies a given message m. AC picks such a number $\hat{\xi} \in \mathcal{I}_m$ and encodes it into a string of bits b_κ, $\kappa \in \{0, \dots, \mathcal{R}_m\}$ by writing it in binary representation,

$$\hat{\xi} = (0.b_1 b_2 \dots b_{\mathcal{R}_m})_2 \quad \text{with bits } b_\kappa \in \{0, 1\}\ \forall \kappa. \tag{4}$$

Since any $\xi \in \mathcal{I}_m$ may be used to identify the message m, we can interpret the interval $\mathcal{I}_m$ as an *uncertainty region* in ξ-space. AC picks the number $\hat{\xi} \in \mathcal{I}_m$ with the shortest binary representation. This requires at most $\lceil h(m) \rceil$ bits because the numbers ξ that can be represented by Eq. 4 with $\mathcal{R}_m = \lceil h(m) \rceil$ form a uniform grid with spacing $2^{-\mathcal{R}_m} = 2^{-\lceil h(m) \rceil}$, which is at most as wide as the size of the interval, $|\mathcal{I}_m| = p(m) = 2^{-h(m)}$. The red arrows in Figure 1 (left) illustrate how AC would encode the message $m = 7$ in the toy example into the bitstring "111". Decoding works in the opposite direction and maps $\hat{\xi}$ back to m.

In the next section, we generalize AC to the continuous domain. As we will show, the concept of an "uncertainty region" in ξ-space becomes again crucial.

3.3. Bayesian Arithmetic Coding

We now present our proposed algorithm, Bayesian Arithmetic Coding (Bayesian AC), which generalizes standard AC from the domain of discrete messages m to the domain of continuous latent variables $\mathbf{z} \in \mathbb{R}^K$. Similar to AC, Bayesian AC exploits knowledge of a prior probability distribution $p(\mathbf{z})$ in combination with a (soft) uncertainty region to encode probable values of $\mathbf{z}$ into short bitstrings.

From Intervals to Distributions. The main ideas that Bayesian AC borrows from standard AC are as follows: (1) the use of a cumulative distribution function to map any non-uniformly distributed random variable to a uniformly distributed random variable ξ over the interval $(0, 1)$, and (2) the use of an "uncertainty region" to select a number on this interval to encode the message with as few bits as possible. While AC was characterized by an interval $\mathcal{I}_m$ with hard boundaries, Bayesian AC softens this uncertainty region by drawing on posterior uncertainty.

We consider a single continuous latent variable $z_i \in \mathbb{R}$ with arbitrary prior $p(z_i)$. The cumulative (CDF) of the prior,

$$F(z_i) := \int_{-\infty}^{z_i} p(z_i')\, dz_i', \tag{5}$$

is shown in orange in Figure 1 (right). It maps $z_i \sim p(z_i)$ to $\xi_i \sim \mathcal{U}(0, 1)$. In contrast to the discrete case discussed in Section 3.2, where the prior CDF maps each message m to an entire interval $\mathcal{I}_m$, note that the CDF of a continuous random variable maps real numbers to real numbers.

Since $\xi_i \sim \mathcal{U}(0, 1)$ is almost surely an irrational number, its binary representation is infinitely long, and a practical encoding method has to truncate it to some finite length. We find an optimal truncation by generalizing the idea of the uncertainty region $\mathcal{I}_m$ to the continuous space. To this end, we consider the posterior uncertainty in z_i-space and map it to ξ_i-space. Approximating the posterior $p(z_i|\mathbf{x})$ by the variational distribution $q(z_i|\mathbf{x}) := \mathcal{N}(z_i; \mu_i(\mathbf{x}), \sigma_i^2(\mathbf{x}))$, see Eq. 3, we thus consider the function

$$g(\xi_i) := q(F^{-1}(\xi_i)\,|\,\mathbf{x}). \tag{6}$$

Here, F^{-1} is the inverse CDF (the quantile function), which maps ξ_i back to z_i. Note that g is not a normalized probability distribution, as Eq. 6 deliberately does not include the Jacobian $\nabla_{\xi_i} F^{-1}(\xi_i)$ because the final objective will be to maximize $q_\phi(\mathbf{z}|\mathbf{x})$ at a single point (see Eq. 7 below).

Intuition. The solid and dashed purple curves in figure Figure 1 (right) plot $q(z_i|\mathbf{x})$ and $g(\xi_i)$ on the horizontal and vertical axis, respectively. The red arrows illustrate how a finite uncertainty region $\mu_i(\mathbf{x}) \pm \sigma_i(\mathbf{x})$ in z_i-space is mapped to a finite width of g in ξ_i-space. Bayesian AC encodes a quantile $\hat{\xi}_i$ that has high value under g while at the same time having a short binary representation. The two purple arrowheads on the vertical axis point to two viable candidates, $\hat{\xi}_i = \frac{7}{8}$ and $\hat{\xi}_i = \frac{3}{4}$, that both lie within the uncertainty region. The choice between these two points poses a rate-distortion trade-off: while $\frac{7}{8} \equiv (0.111)_2$ has higher value under g (i.e., it identifies a point $\hat{z}_i = F^{-1}(\frac{7}{8})$ with higher approximate posterior probability $q(\hat{z}_i|\mathbf{x})$), the alternative $\hat{\xi}_i = \frac{3}{4} \equiv (0.11)_2$ can be encoded in fewer bits.

Optimizing the Rate-Distortion Trade-Off. Rather than considering a hard uncertainty region, Bayesian AC simply tries to find a point $\boldsymbol{\xi} \equiv (\xi_i)_{i=1}^K$ that identifies latent variables $\mathbf{z} \equiv (z_i)_{i=1}^K$ with high probability under the variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$ while being expressible in few bits. We thus express $\log q_\phi(\mathbf{z}|\mathbf{x})$ in terms of the coordinates $\xi_i = F(z_i)$ using Eq. 3,

$$\log q_\phi(\mathbf{z}|\mathbf{x}) = -\sum_{i=1}^{K} \frac{\big(F^{-1}(\xi_i) - \mu_i(\mathbf{x})\big)^2}{2\,\sigma_i^2(\mathbf{x})} + \text{cnst}. \tag{7}$$

For each dimension i, we restrict the quantile $\xi_i \in (0, 1)$ to the set of code points $\hat{\xi}_i$ that can be represented in binary via Eq. 4 with a finite but arbitrary bitlength $\mathcal{R}(\hat{\xi}_i)$. We define the total bitlength $\mathcal{R}(\hat{\boldsymbol{\xi}}) := \sum_{i=1}^K \mathcal{R}(\hat{\xi}_i)$, i.e., the

[3] If m is a sequence of symbols, $F_<$ and $F_\leq$ are defined by lexicographical order and can be constructed in a streaming manner.

length of the concatenation of all codes $\hat{\xi}_i$, $i \in \{1, \dots, K\}$ neglecting, for now, an overhead for delimiters (see below). Using a rate penalty parameter $\lambda > 0$ that is shared across all dimensions i, we minimize the rate-distortion objective

$$\mathcal{L}_\lambda(\hat{\boldsymbol{\xi}}|\mathbf{x}) = -\log q_\phi(\hat{\mathbf{z}}|\mathbf{x}) + \lambda \mathcal{R}(\hat{\boldsymbol{\xi}}) \tag{8}$$
$$= \sum_{i=1}^{K} \left[\frac{\left(F^{-1}(\hat{\xi}_i) - \mu_i(\mathbf{x})\right)^2}{2\,\sigma_i^2(\mathbf{x})} + \lambda \mathcal{R}(\hat{\xi}_i) \right] + \text{cnst.}$$

The optimization thus decouples across all latent dimensions i, and can be solved efficiently and in parallel by minimizing the K independent objective functions

$$\ell_\lambda(\hat{\xi}_i|\mathbf{x}) = \left(F^{-1}(\hat{\xi}_i) - \mu_i(\mathbf{x})\right)^2 + 2\lambda\,\sigma_i^2(\mathbf{x})\,\mathcal{R}(\hat{\xi}_i). \tag{9}$$

Although the bitlength $R(\hat{\xi}_i)$ is discontinuous (it counts the number of binary digits, see Eq. 4), $\ell_\lambda(\hat{\xi}_i|\mathbf{x})$ can be efficiently minimized over $\hat{\xi}_i$ using Algorithm 1. The algorithm iterates over all rates $r \in \{1, 2, \dots\}$ and searches for the code point $\hat{\xi}_i^*$ that minimizes $\ell(\hat{\xi}_i|\mathbf{x})$. For each r, the algorithm only needs to consider the two code points $\hat{\xi}_i^{r,\text{left}} \le \xi_i^\dagger$ and $\hat{\xi}_i^{r,\text{right}} \ge \xi_i^\dagger$ with rate at most r that enclose the optimum $\xi_i^\dagger := F(\mu_i(\mathbf{x}))$ and are closest to it; these two code points can be easily computed in constant time. The iteration terminates as soon as the maximally possible remaining increase in $\log q(z_i|\mathbf{x}) = \log g(\xi_i)$ is smaller than the minimum penalty for an increasing bitlength (in practice, the iteration rarely exceeds $r \approx 8$).

Encoding. After finding the optimal code points $(\hat{\xi}_i^*)_{i=1}^K$, they have to be encoded into a single bitstring. Simply concatenating the binary representations (Eq. 4) of all $\hat{\xi}_i^*$ would be ambiguous due to their variable lengths $\mathcal{R}(\hat{\xi}_i^*)$ (see detailed discussion in the Supplementary Material). Instead, we treat the code points as symbols from a discrete vocabulary and encode them via lossless entropy coding, e.g., standard Arithmetic Coding. The entropy coder requires a probabilistic model over all code points; here we simply use their empirical distribution. When using our method for model compression, this empirical distribution has to be transmitted to the receiver as additional header information that counts towards the total bitrate. For data compression, by contrast, we obtain the empirical distribution of code points on training data and include it in the decoder.

Discussion. The proposed algorithm adjusts the accuracy for each latent variable z_i based on two factors: (i) a *global* rate setting λ that is shared across all dimensions i; and (ii) a per-dimension posterior uncertainty estimate $\sigma_i(\mathbf{x})$. Point (i) allows tuning the rate-distortion trade-off whereas (ii) takes the anisotropy of the latent space into account.

Figure 2 illustrates the effect of anisotropy in latent space. The right panel plots the posterior of a toy Bayesian linear regression model $y = ax + b$ (see left panel) with only two latent variables $\mathbf{z} \equiv (a, b)$. Due to the elongated shape of the posterior, Bayesian AC uses a higher accuracy for a than for b. As a result, the algorithm encodes a point $\hat{\mathbf{z}}$ (orange dot in right panel) that is closer to the optimal (MAP) solution (green dot) along the a-axis than along the b-axis.

The purple dot in Figure 2 (right) compares to a more common quantization method, which simply rounds the MAP solution to the nearest point (which is then entropy coded) from a fixed grid with spacing $\delta > 0$. We tuned δ so that the resulting encoded point (purple dot) has the same distance to the optimum as our proposed solution (orange dot). Despite the equal distance to the optimum, Bayesian AC encodes model parameters with higher posterior probability. The resulting model fits the data better (left panel).

This concludes the description of the proposed Bayesian

Algorithm 1 Rate-Distortion Optimization for Dimension i

Input: Prior CDF $F(z_i)$, rate penalty $\lambda > 0$, variational mode $\mu_i(\mathbf{x})$ and variance $\sigma_i^2(\mathbf{x})$.
Output: Optimal code point $\hat{\xi}_i^* \equiv (0.b_1 b_2 \dots b_{\mathcal{R}(\hat{\xi}_i^*)})_2$.

Evaluate $\xi_i^\dagger \leftarrow F(\mu_i(\mathbf{x}))$.
Initialize $r \leftarrow 0$, $\hat{\xi}_i^* \leftarrow$ null, $\ell^* \leftarrow \infty$.
repeat
  Update $r \leftarrow r + 1$
  Set $\hat{\xi}_i^{r,\text{left}} \leftarrow 2^{-r}\lfloor 2^r \xi_i^\dagger \rfloor$, $\hat{\xi}_i^{r,\text{right}} \leftarrow 2^{-r}\lceil 2^r \xi_i^\dagger \rceil$.
  if $\hat{\xi}_i^{r,\text{left}} \neq 0$ **and** $\ell_\lambda(\hat{\xi}_i^{r,\text{left}}\,|\,\mathbf{x}) < \ell^*$ **then**
    Update $\hat{\xi}_i^* \leftarrow \hat{\xi}_i^{r,\text{left}}$, $\ell^* \leftarrow \ell_\lambda(\hat{\xi}_i^{r,\text{left}}\,|\,\mathbf{x})$.
  end if
  if $\hat{\xi}_i^{r,\text{right}} \neq 1$ **and** $\ell_\lambda(\hat{\xi}_i^{r,\text{right}}\,|\,\mathbf{x}) < \ell^*$ **then**
    Update $\hat{\xi}_i^* \leftarrow \hat{\xi}_i^{r,\text{right}}$, $\ell^* \leftarrow \ell_\lambda(\hat{\xi}_i^{r,\text{right}}\,|\,\mathbf{x})$.
  end if
until $\log g(\xi_i^\dagger) - \log g(\hat{\xi}_i^*) < \lambda(r + 1 - \mathcal{R}(\hat{\xi}_i^*))$.

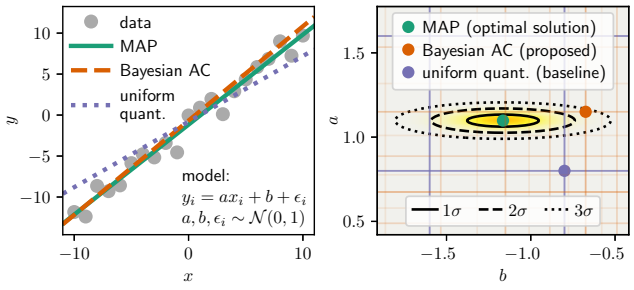


Figure 2. Effect of an anisotropic posterior distribution on lossy compression. Left: linear regression model with optimal fit (green) and fits from two lossy compression methods. Right: posterior distribution and positions of the compressed models in latent space. Although both compressed models are equally far away from the optimal solution (green dot), Bayesian AC (orange) fits the data better because it takes the anisotropy of the posterior into account.

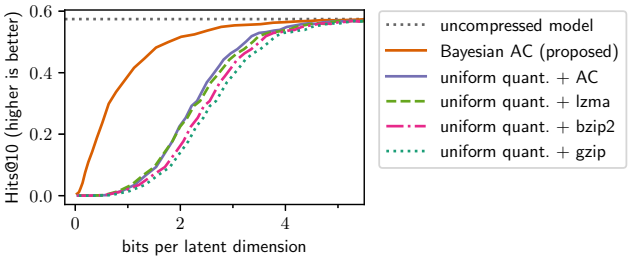


Figure 3. Performance of compressed word embeddings on a standard semantic and syntactic reasoning task (Mikolov et al., 2013a). Bayesian AC (orange, proposed) leads to much smaller file sizes at equal model performance over a wide range of performances.

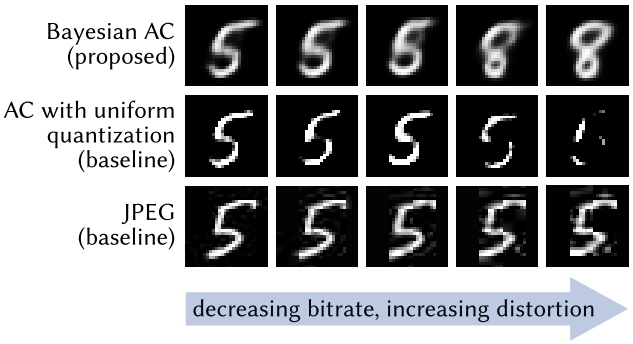


Figure 4. Qualitative behavior of three different image compression methods upon reducing the bitrate (bitrates are on different scales). While JPEG and uniformly quantizing a VAE see loss in pixel-level detail, Bayesian AC tends to preserve details but semantically confuses the encoded object with a generic one.

Arithmetic Coding algorithm. In the next section, we analyze the algorithm's behaviour experimentally and demonstrate its performance for variable-bitrate compression on both word embeddings and images.

4. Experiments

We tested our approach in two very different domains: word embeddings and images. For word embeddings, we measured the performance drop on a semantic reasoning task due to lossy compression. Our proposed Bayesian AC method significantly improves model performance over uniform discretization and compression with either Arithmetic Coding (AC), gzip, bzip2, or lzma at equal bitrate. For image compression, we show that a single standard VAE, compressed with Bayesian AC, outperforms JPEG and other baselines at a wide range of bitrates, both quantitatively and visually.

4.1. Compressing Word Embeddings

We consider the Bayesian Skip-gram model for neural word embeddings (Barkan, 2017), a probabilistic generative formulation of word2vec (Mikolov et al., 2013b) which interprets word and context embedding vectors as latent variables and associates them with Gaussian approximate posterior distributions. Point estimating the latent variables would result in classical word2vec. Even though the model was not specifically designed or trained with model compression taken into consideration, the proposed algorithm can successfully compress it in post-processing.

Experiment Setup. We implemented the Black Box VI version of the Bayesian Skip-gram model proposed in (Bamler & Mandt, 2017),[4] and trained the model on books published between 1980 and 2008 from the Google Books corpus (Michel et al., 2011), following the preprocessing described in (Bamler & Mandt, 2017) with a vocabulary of $V = 100{,}000$ words and embedding dimension $d = 100$.

[4] See Supplement for hyperparameters. Our code is available at https://github.com/mandt-lab/bayesian-ac/.

In the trained model, we observed that the distribution of posterior modes $\mu_{w,j}$ across all words w and all dimensions j of the embedding space was quite different from the prior. To improve the bitrate of our method, we used an "empirical prior" for encoding that is shared across all w and j; we chose a Gaussian $\mathcal{N}(0, \sigma_0^2)$ where σ_0^2 is the empirical variance of all variational means $(\mu_{w,j})_{w=1,\dots,V;\, j=1,\dots,d}$.

We compare our method's performance to a baseline that quantizes to a uniform grid and then uses the empirical distribution of quantized coordinates for lossless entropy coding. We also compare to uniform quantization baselines that replace the entropy coding step with the standard compression libraries gzip, bzip2, and lzma. These methods are not restricted by a factorized distribution of code points and could therefore detect and exploit correlations between quantized code points across words or dimensions.

We evaluate performance on the semantic and syntactic reasoning task proposed in (Mikolov et al., 2013a), a popular dataset of semantic relations like "Japan : yen = Russia : ruble" and syntactic relations like "amazing : amazingly = lucky : luckily", where the goal is to predict the last word given the first three words. We report Hits@10, i.e., the fraction of challenges for which the compressed model ranks the correct prediction among the top ten.

Results. Figure 3 shows the model performance on the semantic and syntactic reasoning tasks as a function of compression rate. Our proposed Bayesian AC significantly outperforms all baselines and reaches the same Hits@10 at less than half the bitrate over a wide range.[5]

[5] The uncompressed model performance (dotted gray line in Figure 3) is not state of the art. This is not a shortcoming of the compression method but merely of the model, and can be attributed to the smaller vocabulary and training set used compared to (Mikolov et al., 2013b) due to hardware constraints.

4.2. Image Compression

While Section 4.1 demonstrated the proposed Bayesian AC method for model compression, we now apply the same method to data compression using a variational autoencoder (VAE). We first provide a qualitative evaluation on MNIST, and then quantitative results on full resolution color images.

Model. For simplicity, we consider regular VAEs with a standard normal prior and Gaussian variational posterior. The generative network parameterizes a factorized Bernoulli or Gaussian likelihood model in the two experiments, respectively. Network architectures are described below and in more detail in Supplementary Material.

Baselines. We consider the following baselines:

- *Uniform quantization:* for a given image $\mathbf{x}$, we quantize each dimension of the posterior mean vector $\boldsymbol{\mu}(\mathbf{x})$ to a uniform grid. We report the bitrate for encoding the resulting quantized latent representation via standard entropy coding (e.g., arithmetic coding). Entropy coding requires prior knowledge of the probabilities of each grid point. Here, we use the empirical frequencies of grid points over a subset of the training set;
- *k-means quantization*: similar to "uniform quantization", but with the placement of grid points optimized via k-means clustering on a subset of the training data;
- *JPEG:* we used the libjpeg implementation packaged with the Python Pillow library, using default configurations (e.g., 4:2:0 subsampling), and we adjust the quality parameter to vary the rate-distortion trade-off;
- *Deep learning baseline:* we compare to Ballé et al. (2017), who directly optimized for the rate and distortion, training a separate model for each point on the R-D curve. In our large-scale experiment, we adopte their model architecture, so their performance essentially represents the end-to-end optimized performance upper bound for our method (which uses a single model).

4.2.1. Qualitative Analysis on Toy Experiment

We trained a simple VAE on MNIST digits, and compared our method to uniform quantization and JPEG.

Figure 4 shows example compressed digits, ranging from the highest to the lowest bitrate that each method allows. As the rate decreases, we see that unlike JPEG, which introduces pixel-level artifacts, both VAE-based methods were able to preserve semantic aspects of the original image. It is interesting to see how the performance degrades in the strongly compressed regime for Bayesian AC. With aggressive decrease in bitrate (as $\mathcal{R}(\hat{\boldsymbol{\xi}}) \to 0$), our method gradually "confuses" the original image with a generic image (8 being in the center of the embedding space), while preserving approximately the same level of sharpness.

4.2.2. Full-Resolution color image Compression

We apply our Bayesian AC method to a VAE trained on color images and demonstrate its practical image compression performance rivaling JPEG.

Model and Dataset. The inference and generative networks of the VAE are identical to the analysis and synthesis networks of Ballé et al. (2017), using 3 layers of 256 filters each in a convolutional architecture. We used a diagonal Gaussian likelihood model, whose mean is computed by the generative net and the variance σ^2 is fixed as a hyperparameter, similar to a β-VAE (Higgins et al., 2017) approach (σ^2 was tuned to 0.001 to ensure the VAE achieved overall good R-D trade-off; see (Alemi et al., 2018)). We trained the model on the same subset of the ImageNet dataset as used in (Ballé et al., 2017). We evaluated performance on the standard Kodak dataset (kod), a separate set of 24 uncompressed color images. As in the word embedding experiment, we also observed that using an empirical prior for our method improved the bitrate. We used the same generic density model as in (Ballé et al., 2018), fitting a different distribution for each latent channel, on samples of posterior means $\boldsymbol{\mu}$ (treating spatial dimensions as i.i.d.).

Results. As common in image compression work, we measure the distortion between the original and compressed image under two quality metrics (the higher the better): Peak Signal-to-Noise ratio (PSNR), and MS-SSIM (Wang et al., 2003), over all RGB channels. Figure 6 shows rate-distortion performance, where we averaged both the bits per pixel (BPP) and quality measure across all images in the Kodak dataset, for each fixed R-D trade-off setting (we obtained similar results when averaging only over the quality metrics for fixed bitrates). The results for Ballé et al. (2017) are taken from the paper and the authors' website.

We found that our method generally produced images with higher quality, both in terms of PSNR and perceptual quality, compared to JPEG and uniform quantization. Similar to (Ballé et al., 2017), our method avoids jarring artifacts, and introduces blurriness at low bitrate. See Figure 5 for example image reconstructions. For more examples and R-D curves on individual images, see Supplementary Material.

Although our results fall short of the end-to-end optimized rate-distortion performance of Ballé et al. (2017), it is worth emphasizing that our method allows operating anywhere on the R-D curve with a *single* trained VAE model, unlike Ballé et al. (2017), which requires costly optimization and storage of individual models for each point on the R-D curve.

(a) Original (b) JPEG MS-SSIM=0.813 (c) Bayesian AC MS-SSIM=0.933 (d) Uniform grid MS-SSIM=0.723 (e) Ball et al. MS-SSIM=0.958

Figure 5. Image reconstructions at matching bitrate (0.24 bits per pixel). Bayesian AC (c; proposed) outperforms AC with uniform quantization (d) and JPEG (b) and is comparable to the approach by (Ballé et al., 2017) (e) despite using a model that is not optimized for this specific bitrate. Uniform quantization here used a modified version of the VAE in Figure 6, using an additional conv layer with smaller dimensions to reduce the bitrate down to 0.24 (this was not possible in the original model even with the largest possible grid spacing).

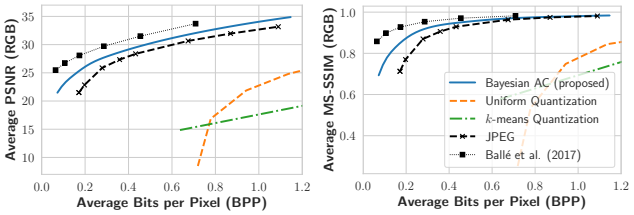


Figure 6. Aggregate rate-distortion performance on the Kodak dataset (higher is better). Bayesian AC (blue, proposed) outperforms JPEG for all tested bitrates with a single model. By contrast, (Ballé et al., 2017) (black squares) relies on individually optimized models for each bitrate that all have to be included in the decoder.

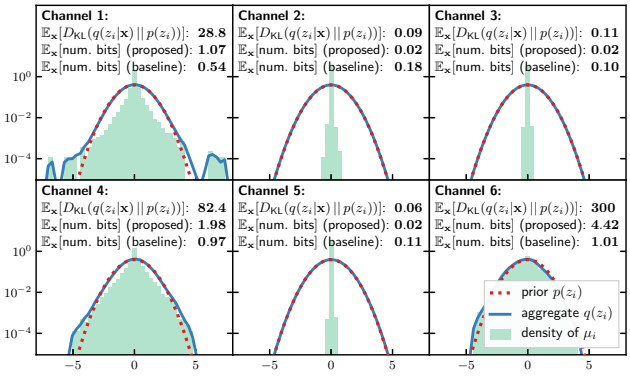


Figure 7. Variational posteriors and the encoding cost for the first 6 latent channels of an image-compression VAE trained with high β setting. "Baseline" refers to uniform quantization. The proposed Bayesian AC method wastes fewer bits on channels that exhibit posterior collapse (channels 2, 3, and 5) than the baseline method. It instead spends more bits on channels without posterior collapse.

Indifference to Posterior Collapse. A known issue in deep generative models such as VAEs is the phenomenon of posterior collapse, where the model ignores some subset of latent variables, and the corresponding variational posterior distributions collapse to closely match the prior. Since such collapsed dimensions do not contribute to the model's performance, they constitute an overhead in regular neural compression approaches and may need to be pruned.

One curious consequence of our approach is that it spends close to zero bits encoding the collapsed latent dimensions. As an illustration, we trained a VAE as used in the color image compression experiment with a high β setting to purposefully induce posterior collapse, and examine the average number of bits spent on various latent channels.

Figure 7 shows the prior $p(z_i)$, aggregated (approximate) posterior $q(z_i) := \mathbb{E}_{\mathbf{x}}[q(z_i|\mathbf{x})]$, and histograms of posterior means $\mu_i(\mathbf{x})$ for the first six channels of the VAE; all the quantities were averaged over an image batch and across latent spatial dimensions. We observe that channels 2, 3, and 5 appear to exhibit posterior collapse, as the aggregated posteriors closely match the prior while the posterior means tightly cluster at zero; this is also reflected by low average KL-divergence between the variational posterior $q(z_i|\mathbf{x})$ and prior $p(z_i)$, see text inside each panel. We observe that, for these collapsed channels, our method spends fewer bits on average than uniform quantization (baseline) at the same total bitrate, and more bits instead on channels 1, 4, and 6, which do not exhibit posterior collapse. The explanation is that a collapsed posterior has unusually high variance $\sigma_i^2(\mathbf{x})$, causing our model to refrain from long code words due to the high penalty $\propto \sigma_i^2(\mathbf{x})$ per bitrate $\mathcal{R}(\hat{\xi}_i)$ in Eq. 9.

5. Conclusions

We proposed a novel algorithm for lossy compression, based on a new quantization scheme that automatically adapts encoding accuracy to posterior uncertainty estimates. This decouples the task of compression from model design and training, and enables variable-bitrate compression for probabilistic generative models with mean-field variational distributions, in post-processing.

We empirically demonstrated the effectiveness of our approach for both model and data compression. Our proposed algorithm can be readily applied to many existing models. In particular, we believe it holds promise for compressing Bayesian neural networks.

Acknowledgements

Stephan Mandt acknowledges funding from DARPA (HR001119S0038), NSF (FW-HTF-RM), and Qualcomm.

References

The kodak photocd dataset. https://www.cns.nyu.edu/~lcv/iclr2017/. Accessed: 2020-01-09.

Agustsson, E., Mentzer, F., Tschannen, M., Cavigelli, L., Timofte, R., Benini, L., and Gool, L. V. Soft-to-hard vector quantization for end-to-end learning compressible representations. In *Advances in Neural Information Processing Systems*, 2017.

Alemi, A., Poole, B., Fischer, I., Dillon, J., Saurous, R. A., and Murphy, K. Fixing a broken elbo. In *International Conference on Machine Learning*, 2018.

Ballé, J., Laparra, V., and Simoncelli, E. P. End-to-end optimized image compression. *International Conference on Learning Representations*, 2017.

Ballé, J., Minnen, D., Singh, S., Hwang, S. J., and Johnston, N. Variational image compression with a scale hyperprior. *In ICLR*, 2018.

Bamler, R. and Mandt, S. Dynamic word embeddings. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 380–389, 2017.

Barkan, O. Bayesian neural word embedding. In *Association for the Advancement of Artificial Intelligence*, pp. 3135–3143, 2017.

Berger, T. Optimum quantizers and permutation codes. *IEEE Trans. Information Theory*, 18:759–765, 1972.

Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent dirichlet allocation. *Journal of machine Learning research*, 3: 993–1022, 2003.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Chou, P. A., Lookabaugh, T. D., and Gray, R. M. Entropy-constrained vector quantization. *IEEE Trans. Acoustics, Speech, and Signal Processing*, 37:31–42, 1989.

Gallager, R. G. *Information theory and reliable communication*, volume 2. Springer, 1968.

Gersho, A. and Gray, R. M. *Vector quantization and signal compression*, volume 159. Springer Science & Business Media, 2012.

Gregor, K., Besse, F., Rezende, D. J., Danihelka, I., and Wierstra, D. Towards conceptual compression, 2016.

Habibian, A., Rozendaal, T. v., Tomczak, J. M., and Cohen, T. S. Video compression with rate-distortion autoencoders. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 7033–7042, 2019.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework. *International Conference on Learning Representations*, 2017.

Honkela, A. and Valpola, H. Variational learning and bits-back coding: an information-theoretic view to bayesian learning. *IEEE transactions on Neural Networks*, 15(4): 800–810, 2004.

Huffman, D. A. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

Johnston, N., Vincent, D., Minnen, D., Covell, M., Singh, S., Chinen, T., Jin Hwang, S., Shor, J., and Toderici, G. Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4385–4393, 2018.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014a.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, pp. 1–9, 2014b.

Lloyd, S. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

Lombardo, S., Han, J., Schroers, C., and Mandt, S. Deep generative video compression. In *Advances in Neural Information Processing Systems*, pp. 9283–9294, 2019.

MacKay, D. J. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

MacKay, D. J. *Information theory, inference and learning algorithms*. Cambridge University Press, 2003.

Mentzer, F., Agustsson, E., Tschannen, M., Timofte, R., and Van Gool, L. Conditional probability models for deep image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4394–4402, 2018.

Michel, J.-B., Shen, Y. K., Aiden, A. P., Veres, A., Gray, M. K., Pickett, J. P., Hoiberg, D., Clancy, D., Norvig, P., Orwant, J., et al. Quantitative analysis of culture using millions of digitized books. *science*, 331(6014):176–182, 2011.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pp. 3111–3119, 2013b.

Mnih, A. and Salakhutdinov, R. R. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pp. 1257–1264, 2008.

Ranganath, R., Gerrish, S., and Blei, D. M. Black box variational inference. In *International Conference on Artificial Intelligence and Statistics*, pp. 814–822, 2014.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pp. 1278–1286, 2014.

Rippel, O. and Bourdev, L. Real-time adaptive image compression, 2017.

Theis, L., Shi, W., Cunningham, A., and Huszár, F. Lossy image compression with compressive autoencoders. *International Conference on Learning Representations*, 2017.

Toderici, G., O'Malley, S. M., Hwang, S. J., Vincent, D., Minnen, D., Baluja, S., Covell, M., and Sukthankar, R. Variable rate image compression with recurrent neural networks. *International Conference on Learning Representations*, 2016.

Toderici, G., Vincent, D., Johnston, N., Hwang, S. J., Minnen, D., Shor, J., and Covell, M. Full resolution image compression with recurrent neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5435–5443, 2017.

Townsend, J., Bird, T., and Barber, D. Practical lossless compression with latent variables using bits back coding. *arXiv preprint arXiv:1901.04866*, 2019.

Wang, Z., Simoncelli, E. P., and Bovik, A. C. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pp. 1398–1402. Ieee, 2003.

Witten, I. H., Neal, R. M., and Cleary, J. G. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.

Zhang, C., Butepage, J., Kjellstrom, H., and Mandt, S. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

Supplementary Material to "Variable-Bitrate Neural Compression via Bayesian Arithmetic Coding"

Yibo Yang [*1] **Robert Bamler** [*1] **Stephan Mandt** [1]

This document provides details of the proposed compression method (Section S1), model hyperparameters (Section S2), and additional examples of compressed images (Section S3).

S1. Delimitation Overhead

We elaborate on the "encoding" paragraph of Section 3.3 of the main paper. After finding a quantized code point $\hat{\xi}_i$ for each dimension $i \in \{1, \dots, K\}$ of the latent space, these code points have to be losslessly encoded into a single bitstring for transmission or storage. We experimented with two encoding schemes, described in Subsections S1.1 and S1.2 below. Subsection S1.3 provides further analysis.

S1.1. Encoding via Concatenation

We first describe an encoding scheme that we did not end up using, but that makes it easier to understand the objective function of Bayesian AC (Eq. 8 of the main text). This encoding scheme concatenates the binary representations of $\hat{\xi}_i$ (Eq. 4 of the main text) for all $i \in \{1, \dots, K\}$ in to a single bitstring. As each dimension i contributes $\mathcal{R}(\hat{\xi}_i)$ bits to the concatenated bitstring, this encoding scheme justifies the rate penalty term "$\lambda\mathcal{R}(\hat{\xi}_i)$" in Eq. 4 of the main text.

One also has to transmit the rates $\mathcal{R}(\hat{\xi}_i)$ (in compressed form using traditional entropy coding) so that the decoder can split the concatenated bitstring at the correct positions. While this incurs some overhead, the variable-bitlength representation of $\hat{\xi}_i$ also saves one bit per dimension i because the last bit in the binary representation of each $\hat{\xi}_i$ does not need to be transmitted as it is always a one (otherwise, the optimization algorithm in Bayesian AC would favor an equivalent shorter binary representation of $\hat{\xi}_i$).

S1.2. Encoding via Standard Entropy Coding

The actual encoding scheme we ended up using does not deal with the binary representation of each $\hat{\xi}_i$ explicitly. Instead, we treat each $\hat{\xi}_i$ as a discrete symbol and directly encode the sequence $(\hat{\xi}_i)_{i=1}^K$ of symbols via entropy coding (e.g., standard arithmetic coding). The entropy coder needs a model of the probability $p(\hat{\xi}_i)$ of each symbol. For model compression, we use the empirical frequencies, which we transmit as extra header information that counts towards the total bitrate. For data compression, we estimate the frequencies on training data and include them in the decoder.

Transmitting the empirical frequencies lead to a negligible overhead in the word embeddings experiment. Only a few hundred code points (depending on λ) had nonzero frequencies, so that the compressed file size was dominated by the encoding of $K = Vd = 10^7$ quantized latent variables.

S1.3. Justification of the Rate Penalty Term $\lambda\mathcal{R}(\hat{\xi}_i)$

All experimental results are reported with the encoding scheme of Section S1.2 as it lead to slightly lower bitrates in practice. A peculiarity of this encoding scheme is that it ignores the length $\mathcal{R}(\hat{\xi}_i)$ of the binary representation of each $\hat{\xi}_i$. For a sequence of symbols $\hat{\boldsymbol{\xi}} \equiv (\hat{\xi}_i)_{i=1}^K$ with an i.i.d. entropy model $p(\hat{\xi}_i)$, an optimal entropy coder (such as arithmetic coding) achieves the total bitrate $\mathcal{R}(\hat{\boldsymbol{\xi}}) = \lceil h(\hat{\boldsymbol{\xi}}) \rceil$ with the information content

$$h(\hat{\boldsymbol{\xi}}) = \sum_{i=1}^{K} h(\hat{\xi}_i) = -\sum_{i=1}^{K} \log_2 p(\hat{\xi}_i). \tag{S1}$$

In particular, Eq. S1 does not depend on $\mathcal{R}(\hat{\xi}_i)$. This poses the question whether the rate penalty term "$\lambda\mathcal{R}(\hat{\xi}_i)$" in the Bayesian AC objective (Eq. 8 of the main text) is justified. Ideally, the algorithm would minimize $\lambda h(\hat{\xi}_i) = -\lambda \log_2 p(\hat{\xi}_i)$ instead, but this quantity is unknown until the quantizations $(\hat{\xi}_i)_{i=1}^K$ and therefore the empirical frequencies $p(\hat{\xi}_i)$ are obtained. Our experiments suggest that $\mathcal{R}(\hat{\xi}_i)$ is a useful proxy for the eventual value of $h(\hat{\xi}_i)$.

Figure S1 plots the rate estimate $\mathcal{R}(\hat{\xi}_i)$, i.e., the integer number of bits in the binary representation of $\hat{\xi}_i$ (x-axis) against the actual contribution $h(\hat{\xi}_i) = -\log_2 p(\hat{\xi}_i)$ to the total bitrate according to Eq. S1 (y-axis). The figure shows experimental data for compressed word embeddings at 1.32 bits per latent dimension. We make the following observations:

[*]Equal contribution [1]Department of Computer Science, University of California, Irvine. Correspondence to: Yibo Yang <yibo.yang@uci.edu>, Robert Bamler <rbamler@uci.edu>.

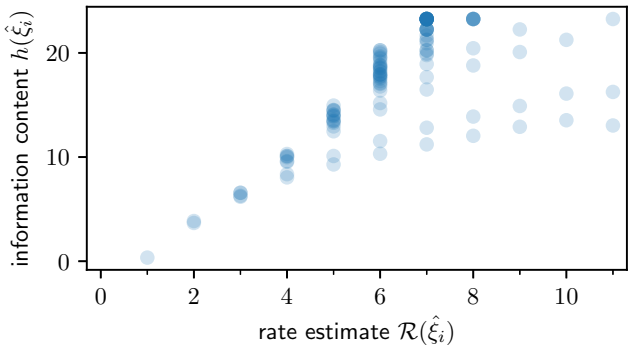


Figure S1. Relation between rate estimate $\mathcal{R}(\hat{\xi}_i)$ and the actual contribution $h(\hat{\xi}_i)$ of code point $\hat{\xi}_i$ to the total bitrate under entropy coding. The approximate affine linear relationship justifies minimizing $\mathcal{R}(\hat{\xi}_i)$ as a proxy for $h(\hat{\xi}_i)$ in Bayesian AC.

- For most code points $\hat{\xi}_i$, the dependency between $h(\hat{\xi}_i)$ and $\mathcal{R}(\hat{\xi}_i)$ can be approximated by an affine linear function, thus justifying the use of $\mathcal{R}(\hat{\xi}_i)$ in the optimization of Bayesian AC.

- The slope of the approximate linear dependency is larger than one. This may be understood by the penalty term $\lambda\mathcal{R}(\hat{\xi}_i)$ in the objective function (Eq. 8 of the main text), which causes the method to avoid code points $\hat{\xi}_i$ with large rate estimates $\mathcal{R}(\hat{\xi}_i)$, thus reducing their empirical frequencies $p(\hat{\xi}_i)$ and increasing their information content $h(\hat{\xi}_i) = -\log_2 p(\hat{\xi}_i)$. This observation does not invalidate the use of $\mathcal{R}(\hat{\xi}_i)$ as an estimate for $h(\hat{\xi}_i)$ since the different slope can be absorbed in a rescaling of the parameter λ

- For rates $\mathcal{R}(\hat{\xi}_i) \geq 4$, there are two code points for each rate with considerably lower information content. These code points correspond to the two extremes for each rate, i.e., $\hat{\xi}_i$ closest to zero or one, respectively. The observation that the two extremes have lower information content (i.e., higher empirical frequencies) can be explained by the fact that the empirical prior distribution whose CDF we use to map latent variables z_i to quantiles ξ_i does not fully capture the true distribution of variational means. Indeed, experiments with a more long tailed empirical prior distribution lead to marginally better performance, but the simplicity of a Gaussian empirical prior seemed more valuable to us.

S2. More Experimental Details

S2.1. Word Embeddings

The word embeddings experiment involved only minimal hyperparameter tuning, and we only optimized for performance of the uncompressed model since the goal of the experiment was to test the proposed compression method on a model that was not tuned for compression. We trained for 10^5 iterations with minibatches of 10^4 randomly drawn words and contexts due to hardware constraints. We tried learning rates 0.1 and 1 and chose 0.1.

S2.2. Toy VAE for MNIST Images

The VAE's inference network has two convolutional layers followed by a fully connected layer. The two conv layers use 32 and 64 filters respectively, with kernel size 3, stride size 2, and ReLU activation. The fully connected layer has output dimension 10 so that μ and σ^2 each has dimension 5.

The generative network architecture mirrors the inference network but in reverse, starting with a dense layer mapping 5 dimensional latent variables to 1568 dimensional, treated as 32-channel 7x7 activations, and followed by two deconvolutional layers of 64 and 32 filters (with identical padding and stride as the convolutional layers). The output is deconvolved with a single 3x3 filter with sigmoid activation function. For each pixel, the (scalar) output of the last layer parameterizes the likelihood of the pixel being white.

We trained the network on binarized MNIST for 100 epochs, using the Adam optimizer with learning rate 10^{-4}.

S2.3. VAE for Color Image Compression

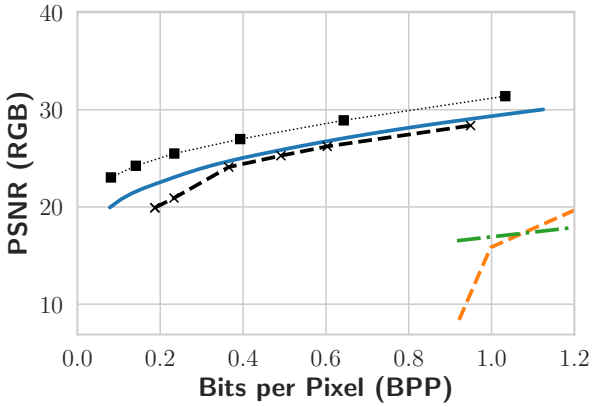
As mentioned in the main text, the VAE uses a fully convolutional architecture with 3 layers of 256 filters each, same as in (Balle et al., 2017); see latter for detailed descriptions. We tuned the variance σ^2 of the likelihood model on a logarithmic grid from 10^{-4} to 0.1 and set it to 0.001. The VAE was trained on the same dataset as in (Balle et al., 2017) for 2 million steps, using Adam with learning rate 10^{-4}.

In the image compression R-D curves, λ ranges from 2^{-6} to 2^{16}. In Figure 5 of the main text, λ was set to 17.5 for Bayesian AC to match the bitrate of the other methods. The uniform quantization result was obtained with 4 quantization levels, on a separately tuned model that had an additional convolutional layer of 64 channels. The additional conv layer was to reduce the latent dimensionality, as uniform quantization could not achieve bitrates lower than 0.5 even with only 2 grid points in the original 3-layer model.

S3. Additional Image Compression Examples

Starting on the next page, we provide detailed compression results for individual images from the Kodak dataset. For each image, we show the rate-distortion performance by various methods, followed by reconstructions using our proposed method and JPEG at equal bitrate.[1]

[1]The present version of this document contains a subset of example images due to a file size limit on arXiv submissions.



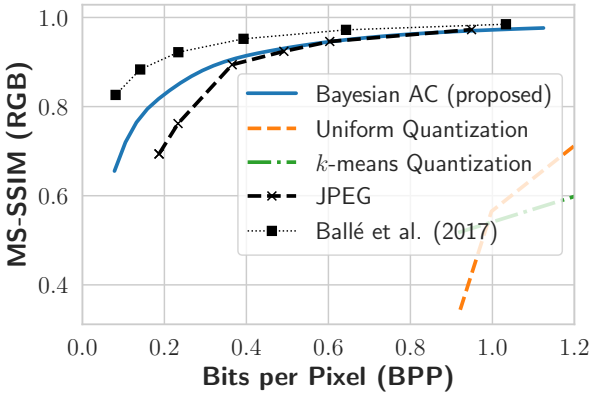
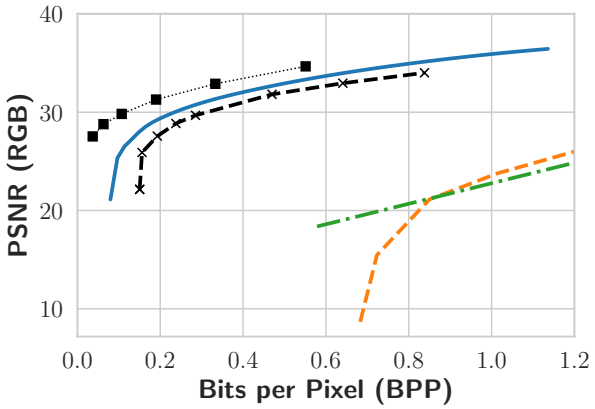


Figure S2. Proposed. bits-per-pixel: 0.27, PSNR: 23.595, MS-SSIM: 0.871

Figure S3. JPEG. bits-per-pixel: 0.27, PSNR: 22.015, MS-SSIM: 0.816



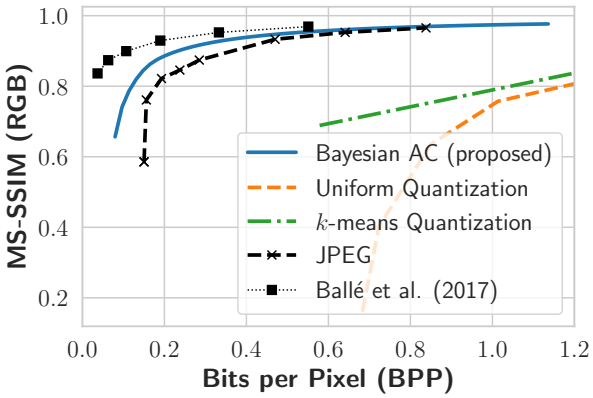
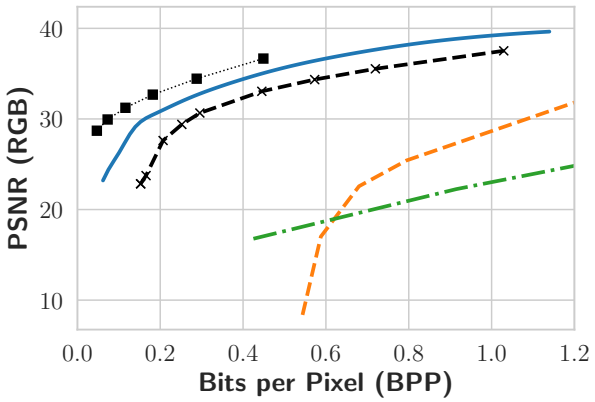


Figure S4. Proposed. bits-per-pixel: 0.19, PSNR: 29.226, MS-SSIM: 0.882

Figure S5. JPEG. bits-per-pixel: 0.19, PSNR: 26.658, MS-SSIM: 0.747



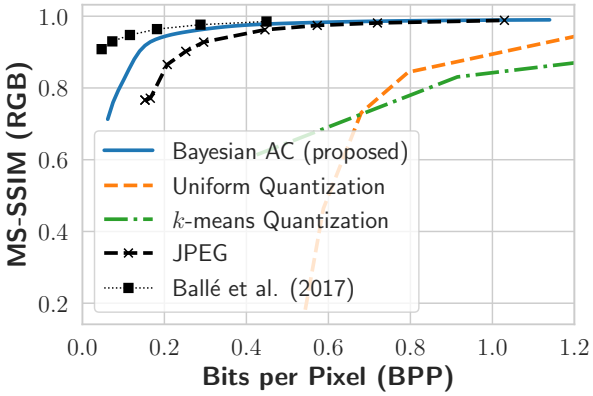
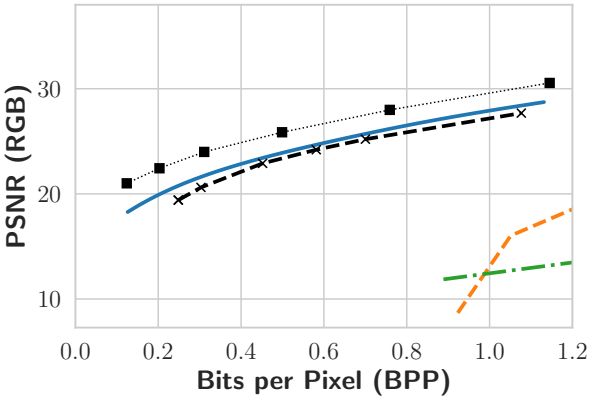


Figure S6. Proposed. bits-per-pixel: 0.19, PSNR: 30.664, MS-SSIM: 0.94

Figure S7. JPEG. bits-per-pixel: 0.19, PSNR: 26.201, MS-SSIM: 0.842



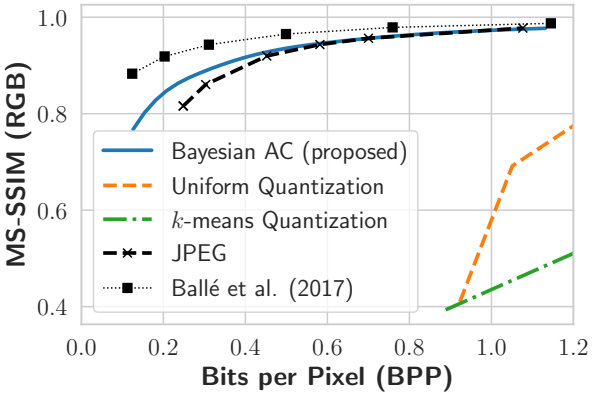
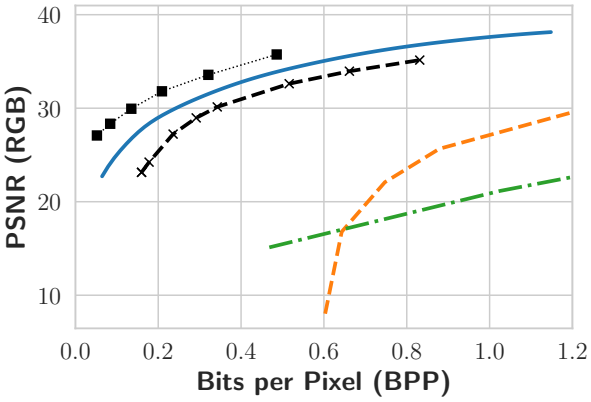
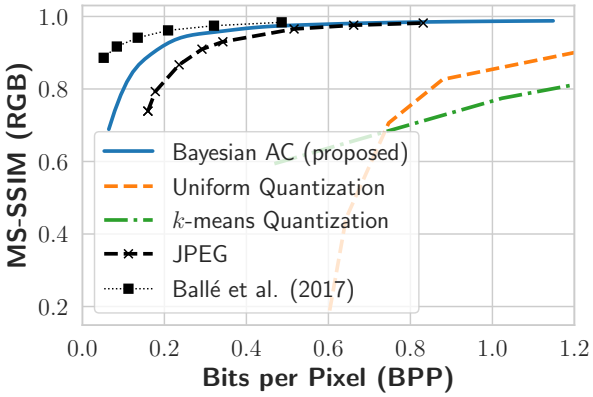


Figure S8. Proposed. bits-per-pixel: 0.34, PSNR: 22.177, MS-SSIM: 0.903

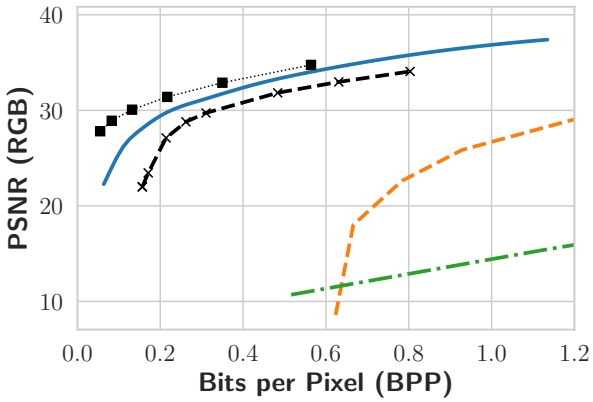
Figure S9. JPEG. bits-per-pixel: 0.34, PSNR: 21.331, MS-SSIM: 0.882





(a) Proposed. bits-per-pixel: 0.22, PSNR: 29.584, MS-SSIM: 0.935

(b) JPEG. bits-per-pixel: 0.22, PSNR: 26.543, MS-SSIM: 0.846



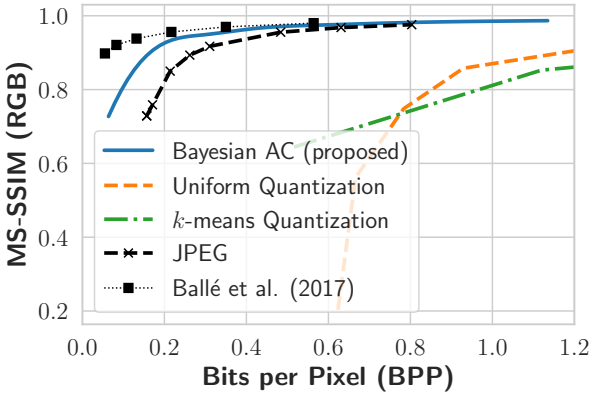
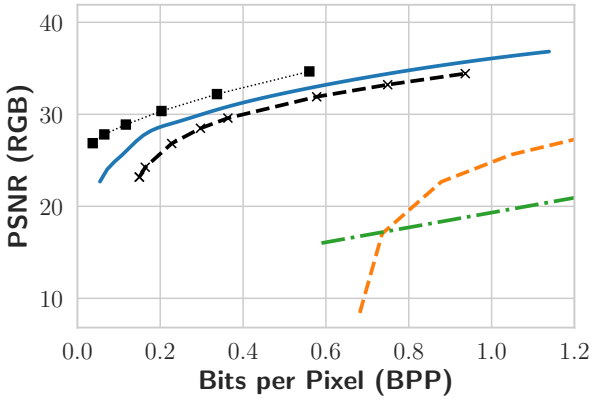


Figure S11. Proposed. bits-per-pixel: 0.2, PSNR: 29.523, MS-SSIM: 0.928

Figure S12. JPEG. bits-per-pixel: 0.2, PSNR: 26.6, MS-SSIM: 0.838



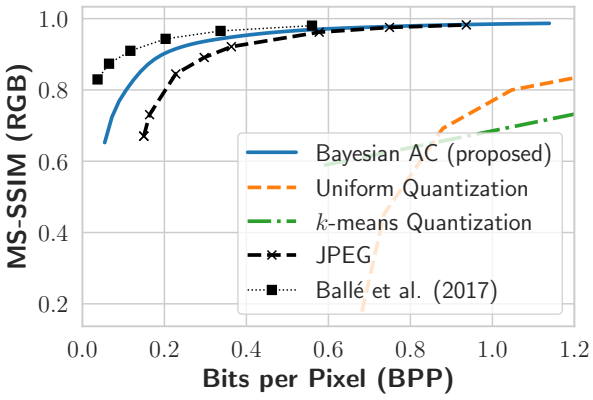
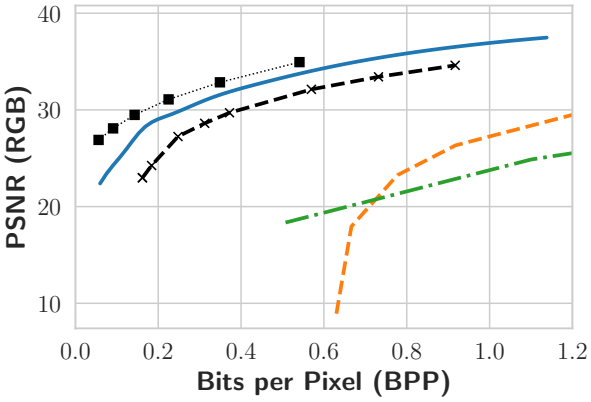
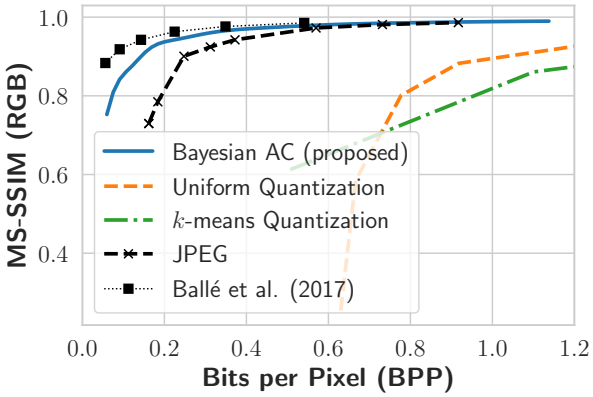
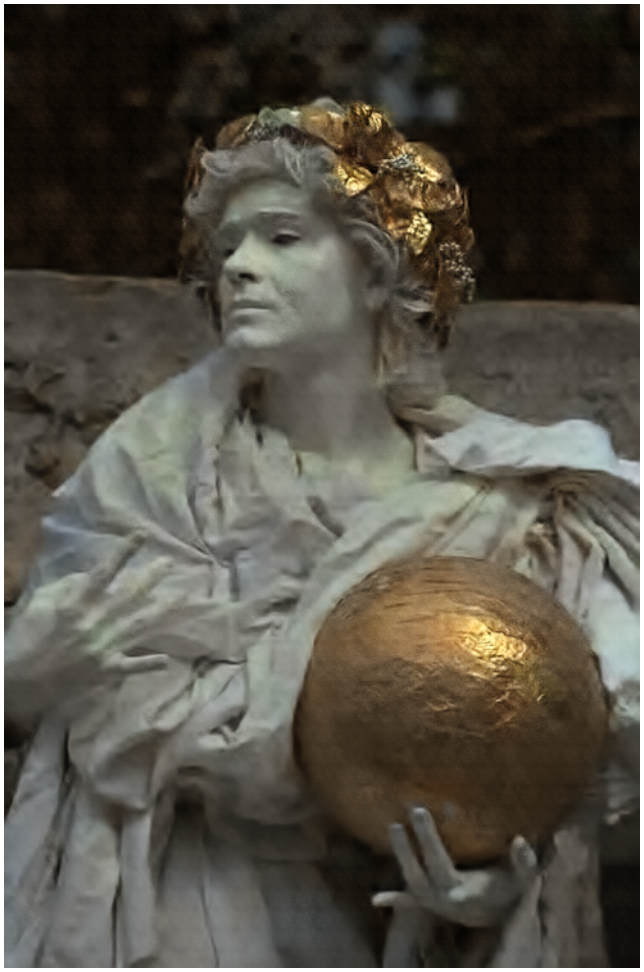


Figure S13. Proposed. bits-per-pixel: 0.21, PSNR: 28.77, MS-SSIM: 0.908

Figure S14. JPEG. bits-per-pixel: 0.21, PSNR: 26.247, MS-SSIM: 0.818

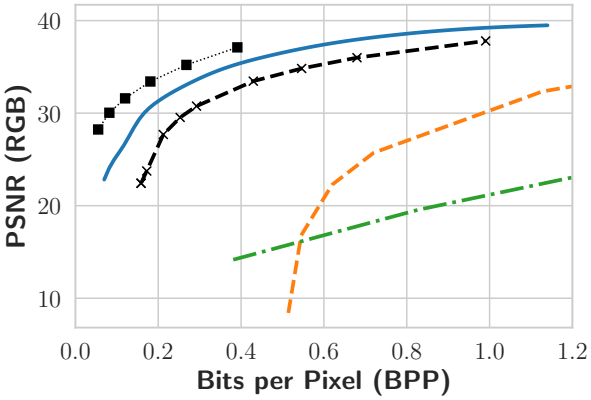






(a) Proposed. bits-per-pixel: 0.2, PSNR: 29.006, MS-SSIM: 0.936

(b) JPEG. bits-per-pixel: 0.2, PSNR: 25.389, MS-SSIM: 0.835



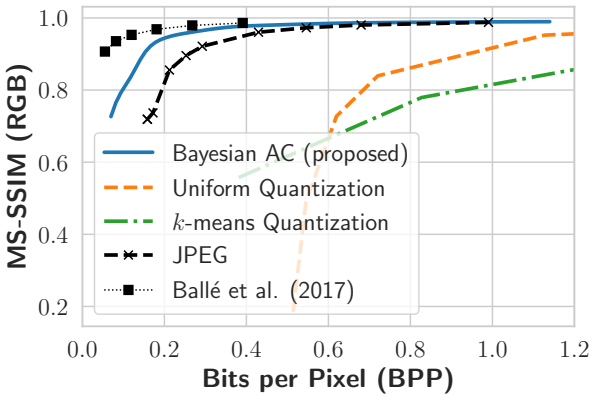
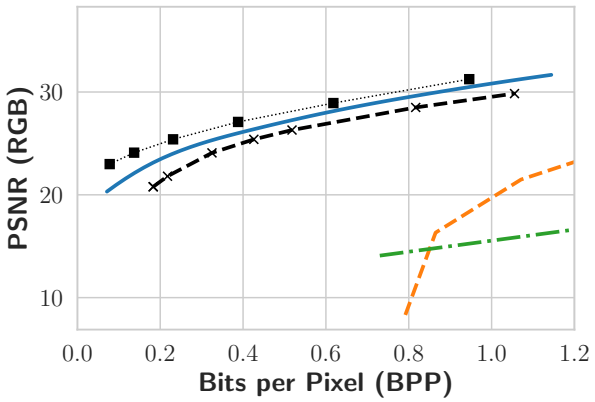


Figure S16. Proposed. bits-per-pixel: 0.2, PSNR: 31.425, MS-SSIM: 0.945

Figure S17. JPEG. bits-per-pixel: 0.2, PSNR: 26.976, MS-SSIM: 0.833



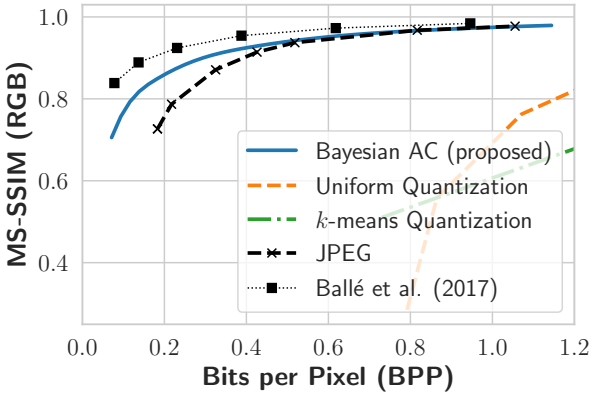


Figure S18. Proposed. bits-per-pixel: 0.24, PSNR: 24.254, MS-SSIM: 0.882

Figure S19. JPEG. bits-per-pixel: 0.24, PSNR: 22.562, MS-SSIM: 0.818