# IMPROVING LOSSLESS COMPRESSION RATES VIA MONTE CARLO BITS-BACK CODING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Latent variable models have been successfully applied in lossless compression with the bits-back coding algorithm. However, bits-back suffers from an increase in the bitrate equal to the KL divergence between the approximate posterior and the true posterior. In this paper, we show how to remove this gap asymptotically by deriving bits-back schemes from tighter variational bounds. The key idea is to exploit extended space representations of Monte Carlo estimators of the marginal likelihood. Naively applied, our schemes would require more initial bits than the standard bits-back coder, but we show how to drastically reduce this additional cost with couplings in the latent space. We demonstrate improved lossless compression rates in a variety of settings.

## 1 INTRODUCTION

In principle, lossless data compression can be improved with better generative models for approximating the data generating distribution. From the panoply of generative models, latent variable models are particularly attractive for compression applications, and have facilitated some of the most successful learned compressors for large scale natural images (see Yang et al. (2020) for lossy, and Townsend et al. (2020) for lossless), because they are flexible and typically easy to parallelize.

The bits-back coding algorithm (Hinton & Van Camp, 1993) makes lossless compression with latent variable models tractable and efficient. However, since bits-back uses a variational approximation $q(z \mid x)$ to the true posterior $p(z \mid x)$, it adds roughly $D_{\mathrm{KL}}(q(z \mid x) \,\|\, p(z \mid x))$ bits to the model's optimal rate. This seems unimprovable for a fixed $q$ if approximating $p(z \mid x)$ is difficult or expensive.

In this paper, we show how to remove (asymptotically) the $D_{\mathrm{KL}}$ gap of bits-back schemes for (just about) any fixed $q$. Our method is based on recent work that derives tighter variational bounds using Monte Carlo estimators of the marginal likelihood (e.g., Burda et al., 2015). The idea is that $q$ and $p$ can be lifted into an extended latent space (e.g., Andrieu et al., 2010) such that the $D_{\mathrm{KL}}$ over the extended latent space goes to zero and the overall bitrate approaches $-\log p(x)$. For example, our simplest extended bits-back method, based on importance sampling, introduces $N$ identically distributed latents $z_i$ and a categorical random variable that picks from $z_i$ to approximate $p(z \mid x)$. We also define extended bits-back schemes based on more advanced Monte Carlo methods (AIS, Neal, 2001; SMC, Doucet et al., 2001). Adding $\mathcal{O}(N)$ latent variables increases the initial bit cost of our methods if naively applied. We introduce a novel technique to reduce this cost to $\mathcal{O}(\log N)$ for some of our coders using couplings in latent space. So, our coders extend bits-back onto a time-bitrate tradeoff with a negligible additional initial bit cost. We test our methods in a variety of settings and explore the factors that affect the rate savings in our settings.

## 2 BACKGROUND

Lossless compression with latent variable models can be practically achieved using the bits-back coding algorithm (Hinton & Van Camp, 1993), in particular the recently proposed efficient coding method Bits-Back with Asymmetric Numeral System (BB-ANS, Townsend et al., 2019). We introduce it briefly here and include a detailed introduction is in Appendix A.

ANS is an efficient and near optimal entropy coder. ANS stores data in a stack-like 'message' structure; encode pushes symbols onto the message and decode pops symbols from the message. ANS has an important property: it is LIFO (i.e., the encoded symbols will be decoded in the opposite

order), which makes it compatible with the bits-back algorithm. For our purposes, the ANS message can be thought of as a store of randomness. Given a message $m$ with enough bits, regardless of $m$'s provenance, we can decode from $m$ using any distribution $p$. This will return a random symbol $x$ and remove roughly $-\log p(x)$ bits from $m$. Conversely, we can encode a symbol $x$ onto $m$ with $p$, which will increase $m$'s length by roughly $-\log p(x)$. See Fig. 4 in Appendix A.

A latent variable model is specified in terms of a joint distribution $p(x, z)$ between a discrete observation random variable (or symbol) $x \in \mathbb{S}$ and a latent discrete random variable $z \in \mathbb{S}'$. We assume that the joint distribution of latent variable models factorizes as $p(z)p(x \mid z)$ and both $p(z)$ and $p(x \mid z)$ are tractable. However, computing the marginal $p(x) = \sum_z p(z)p(x \mid z)$ is often intractable. This fact means that we cannot directly encode $x$ onto $m$. A naive strategy would be for the sender to pick some $z \in \mathbb{S}'$, and encode $(x, z)$ using $p$, which would require $-\log p(x, z)$ bits; however, this involves communicating the symbol $z$, which is redundant information.

BB-ANS gets a better bitrate by compressing sequences of symbols in a chain and by having the sender *decode* latents $z$ from the intermediate message state, rather than picking $z$; see Fig. 1. Suppose that we have already pushed some symbols onto a message $m$. BB-ANS uses an approximate posterior $q(z \mid x)$ such that if $p(x, z) = 0$ then $q(z \mid x) = 0$. To encode a symbol $x$ onto $m$, the sender first pops $z$ from $m$ using $q(z \mid x)$. Then, they push $(x, z)$ onto $m$ using $p(x, z)$. The new message $m'$ has approximately $-\log p(x, z) + \log q(z \mid x)$ more bits than $m$. $m'$ is then used in exactly the same way for the next symbol. The per-symbol rate saving over the naive method is $-\log q(z \mid x)$ bits. However, for the first symbol, an initial message is needed, causing a one-time overhead. Thus, we define the *net bitrate* to be the expected increase in message length per symbol and the *initial bits* to be the number of bits needed to initialize the message. The distribution of the latent $z$, popped with $q(z \mid x)$, is approximately a sample from $q$ (Townsend et al., 2019). Then net bitrate of BB-ANS is the (negative) 'evidence lower bound' (ELBO)

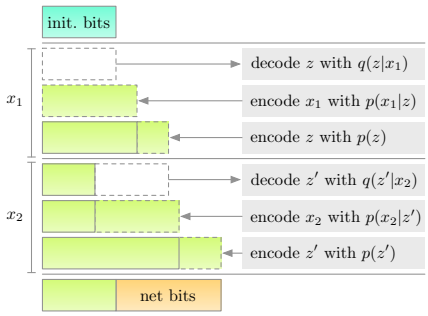



Figure 1: Encoding $(x_1, x_2)$ requires an initial source of bits (light blue), but bits-back reduces its total bit consumption by using intermediate messages as the initial source of bits for encoding $x_2$. Coloured bars represent the bits of the current message. The net bits used (light orange) is close to the negative ELBO.

$$\mathbb{E}_{z \sim q(z \mid x)} \left[ -\log p(x, z) + \log q(z \mid x) \right] = -\log p(x) + D_{\mathrm{KL}}(q(z \mid x) \,\|\, p(z \mid x)), \tag{1}$$

which assumes $z \sim q(z \mid x)$. Thus, thereafter we refer to vanilla BB-ANS as BB-ELBO.

## 3 Monte Carlo Bits-Back Coding

The net bitrate of bits-back is ideally the negative ELBO, which seems difficult to improve without finding a better $q$. However, the ELBO may be a loose bound on $\log p(x)$. Recent work in variational inference shows how to bridge the gap from the ELBO to the marginal log-likelihood with tighter variational bounds (e.g., Burda et al., 2015; Domke & Sheldon, 2018), motivating the question: *can we derive bits-back coders from those tighter bounds and approach the model's optimal rate?* In this section we provide an affirmative answer with a framework called Monte Carlo bits-back coding (McBits). We first motivate our framework with simple examples. Details are in Appendix B.

### 3.1 Bits-Back Importance Sampling

The simplest of our McBits coders is based on importance sampling (IS). IS samples $N$ particles $z_i \sim q(z_i \mid x)$ i.i.d. and uses the importance weights $p(x, z_i)/q(z_i \mid x)$ to estimate $p(x)$. The corresponding variational bound (IWAE, Burda et al., 2015) is the log-average importance weight:

$$-\mathbb{E}_{\{z_i\}_{i=1}^N} \left[ \log \left( \sum_{i=1}^N \frac{1}{N} \frac{p(x, z_i)}{q(z_i \mid x)} \right) \right] \geq -\log p(x). \tag{2}$$

IS provides a consistent estimator of $p(x)$. If the importance weights are bounded, the left-hand side of equation (2) converges monotonically to $-\log p(x)$ (Burda et al., 2015).

Surprisingly, equation (2) is actually the ELBO between a different model and a different approximate posterior on an extended space (Andrieu et al., 2010; Cremer et al., 2017; Domke & Sheldon, 2018).

In particular, consider an extended latent space $\mathbb{S}'^N \times \{1 \,..\, N\}$ that includes the configurations of the $N$ particles $\{z_i\}_{i=1}^N$ and an index $j \in \{1 \,..\, N\}$. The left-hand side of equation (2) can be re-written as the (negative) ELBO between a pair of distributions $P$ and $Q$ defined over this extended latent space, which are given in Alg. 3 in Appendix B.4. Briefly, given $x$, $Q$ samples $N$ particles $z_i \sim q(z_i \,|\, x)$ i.i.d. and selects one of them by sampling an index $j$ with probability $\tilde{w}_j \propto p(x, z_j)/q(z_j \,|\, x)$. The distribution $P$ pre-selects the special $j$th particle uniformly at random, samples its value $z_j \sim p(z_j)$ from the prior of the underlying model, and samples $x \sim p(x \,|\, z_j)$ given $z_j$. The remaining $z_i \sim q(z_i \,|\, x)$ for $i \neq j$ are sampled from the underlying approximate posterior. Because $P$ and $Q$ use $q$ for all but the special $j$th particle, most of the terms in the difference of the log-mass functions cancel, and all that remains is equation (2). See Appendix B.4.

Once we identify the left-hand side of equation (2) as a negative ELBO over the extended space, we can derive a bits-back scheme that achieves an expected net bitrate equal to equation (2). We call this the Bits-Back Importance Sampling (BB-IS) coder, and it is visualized in Fig. 2. To encode a symbol $x$, we first decode $N$ particles $z_i$ and the index $j$ with the $Q$ process by translating each 'sample' to 'decode'. Then we encode $x$, the particles $z_i$, and the index $j$ jointly with the $P$ process by translating each 'sample' to 'encode' in *reverse order*. By reversing $P$ at encoding time, we ensure the receiver decodes with $P$ in the right order.

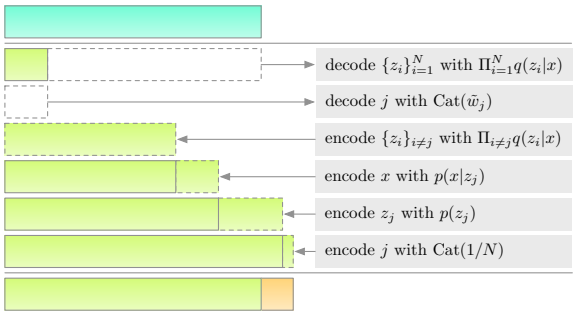



Figure 2: Bits-Back Importance Sampling (BB-IS)

Ideally BB-IS's asymptotic net bitrate is close to the left-hand side of equation (2), which converges to the marginal log-likelihood (Burda et al., 2015). Ultimately, as $N \to \infty$, it reaches the cross-entropy (model's optimal rate). Unfortunately, BB-IS requires roughly $-\log \tilde{w}_j - \sum_{i=1}^N \log q(z_i \,|\, x) \in \mathcal{O}(N)$ initial bits, because each decoded random variable $z_i$ needs to remove some bits from $m$. To avoid this, we design Bits-Back Coupled Importance Sampling (BB-CIS), which achieves a net bitrate comparable to BB-IS while reducing the initial bit cost to $\mathcal{O}(\log N)$ by coupling the latent variables. BB-CIS achieves this by decoding a *single* common random number which is shared by the $z_i$ and designing an encoding process that matches the modified decoding process. Details in Appendix B.5.

### 3.2 General Framework

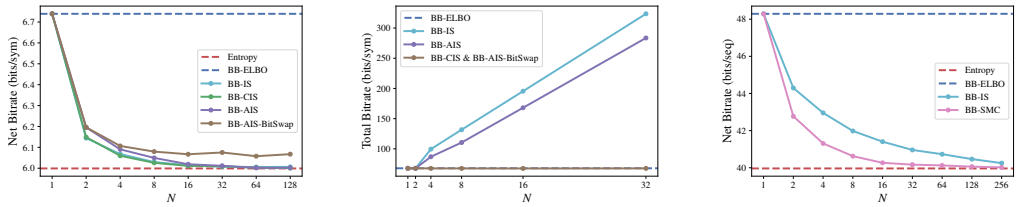
Monte Carlo bits-back coders generalize BB-IS and BB-CIS, which are built from extended latent space representations of Monte Carlo estimators of the marginal likelihood. Let $\hat{p}_N(x)$ be a positive unbiased estimator that can be simulated with $\mathcal{O}(N)$ random variables, i.e., $\mathbb{E}[\hat{p}_N(x)] = p(x)$. A variational bound on the log-marginal likelihood can be derived from $\hat{p}_N(x)$ by Jensen's inequality, i.e., $-\mathbb{E}[\log \hat{p}_N(x)] \geq -\log p(x)$. If $\hat{p}_N(x)$ is strongly consistent in $N$ and $-\log \hat{p}_N(x)$ satisfies a uniform integrability condition, then $-\mathbb{E}[\log \hat{p}_N(x)] \to -\log p(x)$ (Maddison et al., 2017). This framework captures recent efforts on tighter variational bounds (Burda et al., 2015; Maddison et al., 2017; Naesseth et al., 2018; Le et al., 2018; Domke & Sheldon, 2018; Caterini et al., 2018).

As with BB-IS and BB-CIS, the key step is to identify an extended latent space representation of $\hat{p}_N(x)$. Let $\mathcal{Z} \sim Q(\mathcal{Z} \,|\, x)$ be a set of random variables (often including those needed to compute $\hat{p}_N(x)$). If there exists a 'target' distribution $P(x, \mathcal{Z})$ over $x$ and $\mathcal{Z}$ with marginal $p(x)$ such that

$$\hat{p}_N(x) = \frac{P(x, \mathcal{Z})}{Q(\mathcal{Z} \,|\, x)}, \tag{3}$$

then the McBits coder, which decodes $\mathcal{Z}$ with $Q(\mathcal{Z} \,|\, x)$ and encodes $(x, \mathcal{Z})$ with $P(x, \mathcal{Z})$, will achieve a net bitrate of $-\log \hat{p}_N(x)$. If the log estimator converges in expectation to $-\log p(x)$, then ideally $D_{\mathrm{KL}}(Q(\mathcal{Z} \,|\, x) \,\|\, P(\mathcal{Z} \,|\, x)) \to 0$ and the McBits coder will achieve the model's optimal rate.

The challenge is to identify $\mathcal{Z}$, $Q$, and $P$. While Monte Carlo estimators of $p(x)$ get quite elaborate, many of them admit such extended latent space representations (e.g., Neal, 2001; Andrieu et al., 2010; Finke, 2015; Domke & Sheldon, 2018). These constructions are techniques for proving the unbiasedness of the estimators, but in McBits they become bits-back schemes themselves. In Appendix B, we derive Bits-Back Sequential Monte Carlo (BB-SMC) from SMC (Andrieu et al., 2010) and Bits-Back Annealed Importance Sampling (BB-AIS) from AIS (Neal, 2001). We also




(a) As $N \to \infty$, the net bitrate converges to the entropy for most coders on the toy mixture model.

(b) The initial bit cost (reflected in the total bitrate after the first symbol) is controlled by some McBits coders, but not others.

(c) As $N \to \infty$, the net bitrates of BB-IS and BB-SMC converge to the entropy on the toy HMM, but BB-SMC converges much faster.

Figure 3: The bitrate of McBits coders converges (in the number of particles or AIS steps $N$) to the entropy when using the data generating distribution. The initial bit cost of naive coders scales like $\mathcal{O}(N)$, but coupled and BitSwap variants significantly reduce it. Bitrates are bits/sym.

t

Table 1: BB-IS leads to more improved compression rates in transfer learning settings. The net bitrates (bits/dim) of BB-IS on EMNIST test sets using a VAE trained on different EMNIST splits.

| Trained on | MNIST | | Letters | |
|---|---|---|---|---|
| Compressing | MNIST | Letters | MNIST | Letters |
| BB-ELBO | 0.236 | 0.310 | 0.257 | 0.250 |
| BB-IS (5) | 0.231 | 0.289 | 0.249 | 0.243 |
| BB-IS (50) | 0.228 | 0.280 | 0.244 | 0.239 |
| Savings | 3.4% | **9.7%** | **5.1%** | 4.4% |

derive a coupled variant of BB-SMC and a BitSwap (Kingma et al., 2019) variant of BB-AIS to reduce the initial bit cost. In Appendix C, we discuss their computational cost. All coders require $\mathcal{O}(N)$ time, but the IS- and SMC-based coders are amenable to parallelization.

## 4 Experiments

We first studied the empirical properties of McBits coders on synthetic data and then highlighted the superiority of McBits coders in a transfer setting on image compression. Details and more experiments, including musical piece and lossy image compression, are in Appendix D.

**Empirical properties** We assessed the convergence properties and initial bit cost of our McBits coders on synthetic data. All coders were evaluated using the true data generating distribution with a uniform approximate posterior, ensuring a large mismatch with the true posterior.

First, a dataset of 5000 symbols was generated i.i.d. from a mixture model with alphabet sizes 64 and 256 for the observations and latents, respectively. The net bitrates of BB-IS, BB-CIS, and BB-AIS converged to the entropy (optimal rate) as $N$ increased, as shown in Fig. 3a. The indistinguishable gap between BB-CIS and BB-IS illustrates that particle coupling did not lead to a deterioration of net bitrate. BB-AIS-BitSwap did not converge, likely due to the dirty bits issue (as discussed in Appendix D.1). We then quantified the initial bit cost by computing the *total* bitrate (i.e, the net bitrate plus initial bits per symbol) after the first symbol. As shown in Fig. 3b, it increased linearly with $N$ for BB-IS and BB-AIS, but remained fixed for BB-CIS and BB-AIS-BitSwap.

Our second experiment was with a dataset of 5000 symbol subsequences generated i.i.d. from a small hidden Markov model (HMM). We used 10 timesteps with alphabet sizes 16 and 32 for the observations and latents, respectively. BB-ELBO, BB-IS, and BB-SMC were evaluated. The net bitrates of BB-IS and BB-SMC converged to the entropy, but BB-SMC converged much faster, illustrating the effectiveness of resampling particles for compressing sequential data (Fig. 3c).

**Superiority in transfer setting** We assessed BB-IS in a transfer learning setting on practical image compression. We used an alphanumeric extension of MNIST, called EMNIST (Cohen et al., 2017), dynamically binarized following Salakhutdinov & Murray (2008), and a VAE with 1 stochastic layer as in Burda et al. (2015). We trained models on the standard EMNIST-Letters and EMNIST-MNIST splits and evaluated compression performance on both test sets. BB-IS achieved greater rate savings than BB-ELBO when transferred to an out-of-distribution setting (Table 1). This illustrates that BB-IS may be particularly useful in more practical compression settings where the data distribution is different from that of the training data.

## REFERENCES


Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B*, 72(3):269–342, 2010.

Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016.

Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*, 2018.

Jean Bérard, Pierre Del Moral, and Arnaud Doucet. A lognormal central limit theorem for particle approximations of normalizing constants. *Electronic Journal of Probability*, 19, 2014.

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*, 2012.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Anthony L Caterini, Arnaud Doucet, and Dino Sejdinovic. Hamiltonian variational auto-encoder. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 8178–8188, 2018.

Frédéric Cérou, Pierre Del Moral, and Arnaud Guyader. A nonasymptotic theorem for unnormalized Feynman-Kac particle models. In *Annales de l'IHP Probabilités et statistiques*, volume 47, pp. 629–649, 2011.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. *Advances in Neural Information Processing Systems*, 28:2980–2988, 2015.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2921–2926. IEEE, 2017.

Thomas. M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications. 1991.

Chris Cremer, Quaid Morris, and David Duvenaud. Reinterpreting importance-weighted autoencoders. *arXiv preprint arXiv:1704.02916*, 2017.

Luc Devroye. Nonuniform random variate generation. *Handbooks in operations research and management science*, 13:83–121, 2006.

Justin Domke and Daniel R Sheldon. Importance weighting and variational inference. In *Advances in Neural Information Processing Systems*, pp. 4470–4479, 2018.

Arnaud Doucet, Nando De Freitas, and Neil Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo Methods in Practice*, pp. 3–14. Springer, 2001.

Jarek Duda. Asymmetric numeral systems. *arXiv preprint arXiv:0902.0271*, 2009.

Axel Finke. *On extended state-space constructions for Monte Carlo methods*. PhD thesis, 2015.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

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*, pp. 5–13, 1993.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

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.

Pavel Krajcevski, Srihari Pratapa, and Dinesh Manocha. Gst: Gpu-decodable supercompressed textures. *ACM Transactions on Graphics (TOG)*, 35(6):1–10, 2016.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Tuan Anh Le, Maximilian Igl, Tom Rainforth, Tom Jin, and Frank Wood. Auto-encoding sequential Monte Carlo. In *International Conference on Learning Representations*, 2018.

David J. C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

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.

Chris J Maddison, John Lawson, George Tucker, Nicolas Heess, Mohammad Norouzi, Andriy Mnih, Arnaud Doucet, and Yee Teh. Filtering variational objectives. In *Advances in Neural Information Processing Systems*, pp. 6573–6583, 2017.

David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *Advances in Neural Information Processing Systems*, pp. 10771–10780, 2018.

Christian Naesseth, Scott Linderman, Rajesh Ranganath, and David Blei. Variational sequential monte carlo. In Amos Storkey and Fernando Perez-Cruz (eds.), *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pp. 968–977. PMLR, 09–11 Apr 2018. URL `http://proceedings.mlr.press/v84/naesseth18a.html`.

Radford M Neal. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, 2001.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *arXiv preprint arXiv:1711.00937*, 2017.

Ruslan Salakhutdinov and Iain Murray. On the quantitative analysis of deep belief networks. In *Proceedings of the 25th International Conference on Machine learning*, pp. 872–879, 2008.

Casper Kaae Sønderby, Ben Poole, and Andriy Mnih. Continuous relaxation training of discrete latent variable image models. In *Beysian DeepLearning workshop, NIPS*, volume 201, 2017.

James Townsend. A tutorial on the range variant of asymmetric numeral systems, 2020.

James Townsend, Tom Bird, and David Barber. Practical lossless compression with latent variables using bits back coding. *ICLR*, 2019.

James Townsend, Thomas Bird, Julius Kunze, and David Barber. Hilloc: Lossless image compression with hierarchical latent variable models. *ICLR*, 2020.

Alastair J Walker. New fast method for generating discrete random numbers with arbitrary frequency distributions. *Electronics Letters*, 10(8):127–128, 1974.

Yibo Yang, Robert Bamler, and Stephan Mandt. Improving inference for neural image compression. *arXiv preprint arXiv:2006.04240*, 2020.

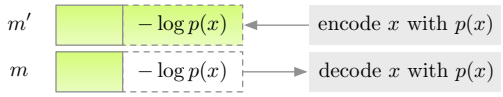



Figure 4: ANS is a last-in-first-out lossless coder. We adopt the visualizations of (Kingma et al., 2019): the green bars represent the message $m$, a stack that stores symbols $x$.

# A Detailed Background

## A.1 Asymmetric Numeral Systems

The goal of lossless compression is to find a short binary representation of the outcome of a discrete random variable $x \sim p_d(x)$ in a finite symbol space $x \in \mathbb{S}$. Achieving the best possible expected length, i.e., the entropy $H(p_d)$ of $p_d$, requires access to $p_d$, and typically a *model* probability mass function (PMF) $p(x)$ is used instead. In this case, the smallest achievable length is the *cross-entropy* of $p$ relative to $p_d$, $H(p_d, p) = -\sum_x p_d(x) \log p(x)$[1]. See MacKay (2003); Cover & Thomas (1991) for more detail.

*Asymmetric numeral systems* (ANS) are model-based coders that achieve near optimal compression rates on sequences of symbols (Duda, 2009). ANS stores data in a stack-like 'message' data structure, which we denote $m$, and provides an inverse pair of functions, $\mathrm{encode}_p$ and $\mathrm{decode}_p$, which each process one symbol $x \in \mathbb{S}$ at a time:

$$\begin{aligned} \mathrm{encode}_p &: m, x \mapsto m' \\ \mathrm{decode}_p &: m' \mapsto (m, x). \end{aligned} \tag{4}$$

Encode pushes $x$ onto $m$, and decode pops $x$ from $m'$. Both functions require access to routines for computing the cumulative distribution function (CDF) and inverse CDF of $p$. If $n$ symbols, drawn i.i.d. from $p_d$, are pushed onto $m$ with $p$ , the *bitrate* (bits/symbol) required to store the ANS message approaches $H(p_d, p) + \epsilon$ for some small error $\epsilon$ (Duda, 2009; Townsend, 2020). The exact sequence is recovered by popping $n$ symbols off the final $m$ with $p$.

For our purposes, the ANS message can be thought of as a store of randomness. Given a message $m$ with enough bits, regardless of $m$'s provenance, we can decode from $m$ using any distribution $p$. This will return a random symbol $x$ and remove roughly $-\log p(x)$ bits from $m$. Conversely, we can encode a symbol $x$ onto $m$ with $p$, which will increase $m$'s length by roughly $-\log p(x)$. If a sender produces a message $m$ through some sequence of encode or decode steps using distributions $p_i$, then a receiver, who has access to the $p_i$, can recover the sequence of encoded symbols and the initial message by reversing the order of operations and switching encodes with decodes. When describing algorithms, we often leave out explicit references to $m$, instead writing steps like '$\mathrm{encode}\ x$ with $p(x)$'. See Fig. 4.

## A.2 Bits-Back Compression with ANS

The class of latent variable models is highly flexible, and most operations required to compute with such models can be parallelized, making them an attractive choice for model-based coders. Bits-back coders, in particular Bits-Back with ANS (BB-ANS, Townsend et al., 2019), specialize in compression using latent variable models.

A latent variable model is specified in terms of a joint distribution $p(x, z)$ between $x$ and a latent discrete[2] random variable taking value in a symbol space $z \in \mathbb{S}'$. We assume that the joint distribution of latent variable models factorizes as $p(z)p(x \mid z)$ and that the PMFs, CDFs, and inverse CDFs, under $p(z)$ and $p(x \mid z)$ are tractable. However, computing the marginal $p(x) = \sum_z p(z)p(x \mid z)$ is often intractable. This fact means that we cannot directly encode $x$ onto $m$. A naive strategy would be for the sender to pick some $z \in \mathbb{S}'$, and encode $(x, z)$ using $p$, which would require $-\log p(x, z)$ bits; however, this involves communicating the symbol $z$, which is redundant information.

BB-ANS gets a better bitrate, by compressing sequences of symbols in a chain and by having the sender *decode* latents $z$ from the intermediate message state, rather than picking $z$; see Fig. 1.

[1] All logarithms in this paper are base 2.

[2] BB-ANS can easily be extended to continuous $z$, with negligible cost, by quantizing; see Townsend et al. (2019).

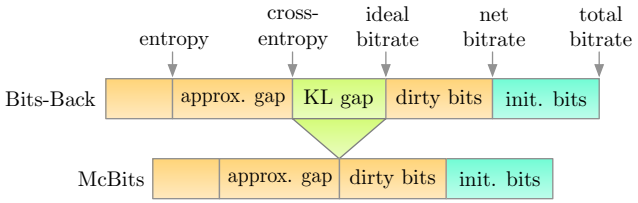



Figure 5: Monte Carlo Bits-Back coders reduce the KL gap to zero.

Suppose that we have already pushed some symbols onto a message $m$. BB-ANS uses an approximate posterior $q(z \mid x)$ such that if $p(x, z) = 0$ then $q(z \mid x) = 0$. To encode a symbol $x$ onto $m$, the sender first pops $z$ from $m$ using $q(z \mid x)$. Then they push $(x, z)$ onto $m$ using $p(x, z)$. The new message $m'$ has approximately $-\log p(x, z) + \log q(z \mid x)$ more bits than $m$. $m'$ is then used in exactly the same way for the next symbol. The per-symbol rate saving over the naive method is $-\log q(z \mid x)$ bits. However, for the first symbol, an initial message is needed, causing a one-time overhead.

### A.3 The Bitrate of Bits-Back Coders

When encoding a sequence of symbols, we define the *total bitrate* to be the number of bits in the final message per symbol encoded; the *initial bits* to be the number of bits needed to initialize the message; and the *net bitrate* to be the total bitrate minus the initial bits per symbol, which is equal to the expected increase in message length per symbol. As the number of encoded symbols grows, the total bitrate of BB-ANS will converge to the net bitrate.

One subtlety is that the BB-ANS bitrate depends on the distribution of the latent $z$, popped with $q(z \mid x)$. In an ideal scenario, where $m$ contains i.i.d. uniform Bernoulli distributed bits, $z$ will be an exact sample from $q$. Unfortunately, in practical situations, the exact distribution of $z$ is difficult to characterize. Nevertheless, Townsend et al. (2019) found the 'evidence lower bound' (ELBO)

$$\begin{aligned} &\mathbb{E}_{z \sim q(z \mid x)} \left[ -\log p(x, z) + \log q(z \mid x) \right] \\ &= -\log p(x) + D_{\mathrm{KL}}(q(z \mid x) \,\|\, p(z \mid x)), \end{aligned} \tag{5}$$

which assumes $z \sim q(z \mid x)$, to be an accurate predictor of BB-ANS's empirical compression rate; the effect of inaccurate samples (which they refer to as 'dirty bits') is typically less than $1\%$. So, in this paper we mostly elide the dirty bits issue, regarding (5) to be the net bitrate of BB-ANS, and hereafter we refer to vanilla BB-ANS as BB-ELBO. Taking the expectation under $p_d$, BB-ELBO achieves a net bitrate of approximately $H(p_d, p) + \mathbb{E}_{x \sim p_d}[D_{\mathrm{KL}}(q(z \mid x) \,\|\, p(z \mid x))]$.

## B Monte Carlo Bits-Back Coders

### B.1 Notation

- $\mathbb{I}(A)$ is the indicator function of the event $A$, i.e., $\mathbb{I}(A) = 1$ if $A$ holds and $0$ otherwise.
- $\mathbb{P}(A)$ is the probability of the event $A$.

### B.2 Assumptions

- We assume that, if $q(z \mid x) > 0$, then $p(x, z) > 0$.

### B.3 General Framework

**Algorithm 1:** Encode Procedure of McBits

**Procedure** `Encode` (*symbol $x$, message $m$*)
  decode $\mathcal{Z}$ with $Q(\mathcal{Z} \mid x)$
  encode $x$ and $\mathcal{Z}$ with $P(x, \mathcal{Z})$
  **return** $m'$

**Algorithm 2:** Decode Procedure of McBits

**Procedure** `Decode` (*message $m$*)
  decode $x$ and $\mathcal{Z}$ with $P(x, \mathcal{Z})$
  encode $\mathcal{Z}$ with $Q(\mathcal{Z} \mid x)$
  **return** $x$, $m'$

As discussed in the main body, the McBits coders can asymptotically drive the KL gap to zero, as illustrated in Fig. 5. Given the extended space representation of an unbiased estimator of the marginal

**Algorithm 3:** Extended Latent Space Representation of Importance Sampling

**Process** $Q(\mathcal{Z} \mid x)$
- sample $\{z_i\}_{i=1}^N \sim \prod_{i=1}^N q(z_i \mid x)$
- compute $\tilde{w}_i \propto \frac{p(x,z_i)}{q(z_i \mid x)}$
- sample $j \sim \mathrm{Cat}\,(\tilde{w}_j)$
- **return** $\{z_i\}_{i=1}^N, j$

**Process** $P(x, \mathcal{Z})$
- sample $j \sim \mathrm{Cat}(1/N)$
- sample $z_j \sim p(z_j)$
- sample $x \sim p(x \mid z_j)$
- sample $\{z_i\}_{i \neq j} \sim \prod_{i \neq j} q(z_i \mid x)$
- **return** $x, \{z_i\}_{i=1}^N, j$

**Algorithm 4:** Encode Procedure of BB-IS

**Procedure** `Encode` (*symbol* $x$*, message* $m$)
- decode $\{z_i\}_{i=1}^N$ with $\prod_{i=1}^N q(z_i \mid x)$
- decode $j$ with $\mathrm{Cat}\,(\tilde{w}_j)$
- encode $\{z_i\}_{i \neq j}$ with $\prod_{i \neq j} q(z_i \mid x)$
- encode $x$ with $p(x \mid z_j)$
- encode $z_j$ with $p(z_j)$
- encode $j$ with $\mathrm{Cat}(1/N)$
- **return** $m'$

likelihood, we can derive the general procedure of McBits coders as Alg. 1. The decode procedure could be easily derived from the encode procedure by simply reverse the order and switch 'encode' with 'decode', as in Alg. 2. Therefore, we do not tediously present the decode procedure for each McBits coder in the following subsections.

### B.4 Bits-Back Importance Sampling (BB-IS)

Importance sampling (IS) samples $N$ particles $z_i \sim q(z_i \mid x)$ i.i.d. and uses the average importance weight to estimate $p(x)$. The corresponding variational bound (IWAE, Burda et al., 2015) is the log-average importance weight which is given in equation (2). For simplicity, we denote importance weight as $w_i = p(x, z_i)/q(z_i \mid x)$ and normalized importance weight as $\tilde{w}_i = w_i / \sum_i w_i$. The basic importance sampling estimator of $p(x)$ is given by

$$\sum_{i=1}^N \frac{w_i}{N} = \sum_{i=1}^N \frac{1}{N} \frac{p(x, z_i)}{q(z_i \mid x)}. \tag{6}$$

**Extended latent space representation** The extended latent space representation of the importance sampling estimator is presented in Alg. 3, from which we can derive the proposal and target distribution as followed:

$$Q(\{z_i\}_{i=1}^N, j \mid x) = \tilde{w}_j \prod_{i=1}^N q(z_i \mid x) \tag{7}$$

$$P(x, \{z_i\}_{i=1}^N, j) = \frac{1}{N} p(x, z_j) \prod_{i \neq j} q(z_i \mid x). \tag{8}$$

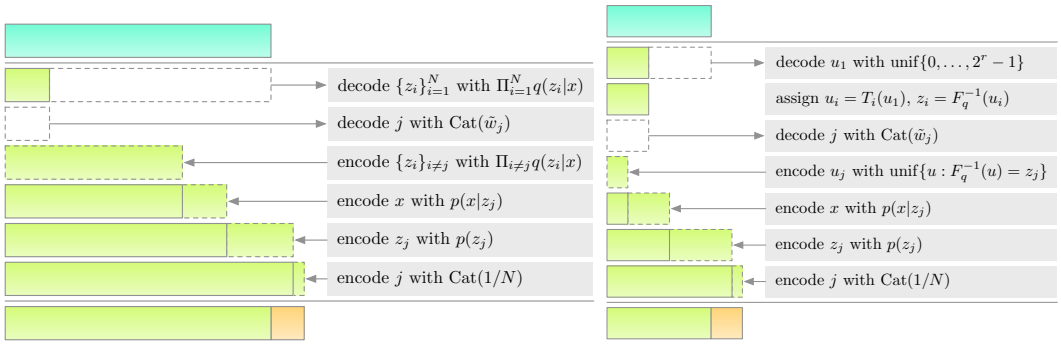
Now note,

$$\frac{P(x, \{z_i\}_{i=1}^N, j)}{Q(\{z_i\}_{i=1}^N, j \mid x)} = \frac{1}{N} \frac{\sum_{i=1}^N w_i}{w_j} \frac{p(x, z_j)}{q(z_j \mid x)} = \sum_{i=1}^N \frac{w_i}{N}, \tag{9}$$

which exactly gives us the IS estimator.

**BB-IS Coder** Based on the extended latent space representation, the Bits-Back Importance Sampling (BB-IS) coder is derived in Alg. 4 and visualized in Fig. 6a. The expected net bit length for encoding a symbol $x$ is:

$$-\mathbb{E}_{\{z_i\}_{i=1}^N, j} \left[ \log \frac{P(x, \{z_i\}_{i=1}^N, j)}{Q(\{z_i\}_{i=1}^N, j \mid x)} \right] = -\mathbb{E}_{\{z_i\}_{i=1}^N} \left[ \log \left( \sum_{i=1}^N \frac{w_i}{N} \right) \right]. \tag{10}$$




(a) Bits-Back Importance Sampling (b) Bits-Back Coupled Importance Sampling

Figure 6: The initial bit cost of encoding a single symbol $x$ with IS-based coders is reduced from $\mathcal{O}(N)$ to $\mathcal{O}(\log N)$ by coupling the latents with shared randomness. Both of these coders achieve a net bitrate that approaches $-\log p(x)$ as $N \to \infty$. We present BB-CIS together with BB-IS to highlight the reduction of initial bit cost.

Ignoring the dirty bits issue, i.e., $z_i \sim q(z_i \mid x)$ i.i.d., the expected net bit length exactly achieves the negative IWAE bound.

### B.5 Bits-Back Coupled Importance Sampling

Unfortunately, the BB-IS coder requires roughly $-\log \tilde{w}_j - \sum_{i=1}^N \log q(z_i \mid x) \in \mathcal{O}(N)$ initial bits. The reason is that each decoded random variable needs to remove some bits from $m$. Can this be avoided? Here, we design Bits-Back Coupled Importance Sampling (BB-CIS), which achieves a net bitrate comparable to BB-IS while reducing the initial bit cost to $\mathcal{O}(\log N)$.

BB-CIS is based on a reparameterization of the particles $z_i$ as deterministic functions of coupled uniform random variables. The method is a discrete analog of the inverse CDF technique for simulating non-uniform random variates (Devroye, 2006). Specifically, suppose that the latent space $\mathbb{S}'$ is totally ordered, and the probabilities of $q$ are approximated to an integer precision $r > 0$, i.e., for all $z \in \mathbb{S}'$

$$q(z \mid x) = \frac{q_z}{2^r}, \tag{11}$$

where $q_z$ is an integer. Note that this assumption is also required for ANS, so this is not an additional assumption. Define the unnormalized cumulative distribution function $F_q$ of $q$ as well as the following related objects:

$$F_q(z) = \sum_{z' \leq z} q_{z'} \tag{12}$$

$$F_q^{-1}(u) = \arg\min\{z \in \mathbb{S}' : F_q(z) > u\} \tag{13}$$

$$U(z) = \{u : F_q^{-1}(u) = z\}. \tag{14}$$

Notice that $|U(z)| = q_z$. Thus, if $u \sim \mathrm{unif}\{0 \,..\, 2^r - 1\}$, then

$$\begin{aligned}\mathbb{P}\left(F_q^{-1}(u) = z\right) &= \mathbb{P}\left(u \in U(z)\right) \\ &= \frac{|U(z)|}{2^r} \\ &= \frac{q_z}{2^r}.\end{aligned}$$

Therefore we can reparameterize the sampling process of $z$ as uniform sampling over $\{0 \,..\, 2^r - 1\}$ followed by the deterministic mapping $F_q^{-1}$. This is a classical approach in non-uniform random variate generation, specialized to this discrete setting. This is visualized in Fig. 7.

Coupled importance sampling (CIS) samples the latent variables by coupling their underlying uniforms. In particular, a coupled set of particles $z_i$ with marginals $q(z_i \mid x)$ can be simulated with a common random number by sampling a single uniform $u$ and setting $z_i = F_q^{-1}(T_i(u))$ for some functions $T_i : \{0 \,..\, 2^r - 1\} \to \{0 \,..\, 2^r - 1\}$. Intuitively, $T_i$ maps $u$ to the uniform $u_i$ underlying

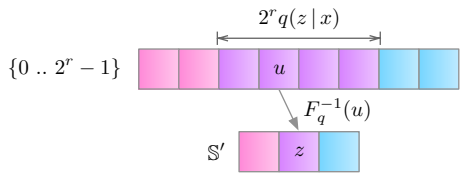



Figure 7: Mapping uniforms in $\{0 \,..\, 2^r - 1\}$ to samples in $\mathbb{S}'$. Colors represent the subsets mapped to $z$ of size $2^r q(z \mid x)$.

**Algorithm 5:** Extended Latent Space Representation of Coupled Importance Sampling

**Process** $Q(\mathcal{Z} \mid x)$
- sample $u_1 \sim \text{unif}\{0, \ldots, 2^r - 1\}$
- **for** $i = 1, \ldots, N$ **do**
  - assign $u_i = T_i(u_1)$
  - assign $z_i = F_q^{-1}(u_i)$
- compute $\tilde{w}_i \propto p(x, z_i)/q(z_i \mid x)$
- sample $j \sim \text{Cat}(\tilde{w}_i)$
- **return** $\{z_i\}_{i=1}^N, \{u_i\}_{i=1}^N, j$

**Process** $P(x, \mathcal{Z})$
- sample $j \sim \text{Cat}(1/N)$
- sample $z_j \sim p(z_j)$
- sample $x \sim p(x|z_j)$
- sample $u_j \sim \text{unif}\{u : F_q^{-1}(u) = z_j\}$
- **for** $i \neq j$ **do**
  - assign $u_i = T_i(T_j^{-1}(u_j))$
  - assign $z_i = F_q^{-1}(u_i)$
- **return** $x, \{z_i\}_{i=1}^N, \{u_i\}_{i=1}^N, j$

$z_i$. To ensure that $z_i$ have the same marginal, we require that $T_i$ are bijective functions. Precisely, if $u \sim \text{unif}\{0 \,..\, 2^r - 1\}$, then

$$\begin{aligned}
\mathbb{P}\left(T_i(u) = u'\right) &= \mathbb{P}\left(u = T_i^{-1}(u')\right) \\
&= \frac{\mathbb{I}\left(T_i^{-1}(u') \in \{0 \,..\, 2^r - 1\}\right)}{2^r} \\
&= \frac{1}{2^r}.
\end{aligned}$$

Thus, if $u \sim \text{unif}\{0 \,..\, 2^r - 1\}$, then $T_i(u) \overset{d}{=} u$ and $F_q^{-1}(T_i(u)) \sim q(z \mid x)$. For example, simple bijective operators $T_i$ can be defined as applying fixed "sampling shift" $\bar{u}_i \in \{0, 1, \ldots, 2^r - 1\}$ to $u$:

$$\begin{aligned}
T_i(u) &= (u + \bar{u}_i) \bmod 2^r \\
T_i^{-1}(u) &= (u - \bar{u}_i) \bmod 2^r.
\end{aligned}$$

For simplicity, we define $T_1$ to employ the zero shift $\bar{u}_1 = 0$, i.e., $T_1(u) = u$. We now have the definitions that we need to analyze the extended space construction for coupled importance sampling. Let $u_1 \sim \text{unif}\{0 \,..\, 2^{-}1\}$. As with IS, we denote importance weight as $w_i = p(x, F_q^{-1}(T_i(u_1)))/q(F_q^{-1}(T_i(u_1)) \mid x)$ and the normalized importance weight as $\tilde{w}_i = w_i / \sum_i w_i$. The coupled importance sampling estimator of $p(x)$ is given

$$\sum_{i=1}^N \frac{w_i}{N} = \sum_{i=1}^N \frac{1}{N} \frac{p(x, F_q^{-1}(T_i(u_1)))}{q(F_q^{-1}(T_i(u_1)) \mid x)}. \tag{15}$$

**Extended latent space representation** The extended latent space representations of the coupled importance sampling estimator is presented in Alg. 5. This extended space construction is novel, and we believe it may be useful for deriving other coupling schemes that reduce initial bit consumption. The $Q$ and $P$ processes have the following probability mass functions. For convenience, let $\mathbb{U} =$

**Algorithm 6:** Encode Procedure of BB-CIS

**Procedure** `Encode` (*symbol* $x$*, message* $m$)
- decode $u_1$ with $\text{unif}\{0, 1, \ldots, 2^r - 1\}$
- assign $u_i = T_i(u)$ and $z_i = F_q^{-1}(u_i)$ for $i \in \{1, \ldots, N\}$
- decode $j$ with $\text{Cat}\,(\tilde{w}_j)$
- encode $u_j$ with $\text{unif}\{u : F_q^{-1}(u) = z_j\}$
- encode $x$ with $p(x \,|\, z_j)$
- encode $z_j$ with $p(z_j)$
- encode $j$ with $\text{Cat}(1/N)$
- **return** $m'$

$\{0 \,..\, 2^r - 1\}$ and $z_i = F_q^{-1}(T_i(u_1))$.

$$Q(\{z_i\}_{i=1}^N, \{u_i\}_{i=1}^N, j \,|\, x) = \frac{\mathbb{I}\,(u_1 \in \mathbb{U})}{2^r} \prod_{i=2}^N \mathbb{I}\,(T_i(u_1) = u_i) \prod_{i=1}^N \mathbb{I}\left(F_q^{-1}(u_i) = z_i\right) \frac{w_j}{\sum_{i=1}^N w_i} \tag{16}$$

$$P(x, \{z_i\}_{i=1}^N, \{u_i\}_{i=1}^N, j) = \frac{1}{N} p(x, z_j) \frac{\mathbb{I}\,(u_j \in U(z_j))}{2^r q(z_j \,|\, x)} \prod_{i \neq j} \mathbb{I}\left(T_i\left(T_j^{-1}(u_j)\right) = u_i\right) \mathbb{I}\left(F_q^{-1}(u_i) = z_i\right), \tag{17}$$

where we used the fact that $q_{z_j} = 2^r q(z_j \,|\, x) = |\, U(z_j)\,|$. Because each $T_i$ is a bijection, we have that the range of $T_i(u_1)$ is all of $\mathbb{U}$, as $u_1$ ranges over $\mathbb{U}$. Thus,

$$\frac{\mathbb{I}\,(u_1 \in \mathbb{U})}{2^r} \prod_{i=2}^N \mathbb{I}\,(T_i\,(u_1) = u_i) = \frac{\mathbb{I}\,(u_j \in \mathbb{U})}{2^r} \prod_{i \neq j} \mathbb{I}\left(T_i\left(T_j^{-1}(u_j)\right) = u_i\right). \tag{18}$$

Moreover, for any $(u, z) \in \mathbb{U} \times \mathbb{S}'$,

$$\frac{\mathbb{I}\,(u \in \mathbb{U})}{2^r} \mathbb{I}\left(F_q^{-1}(u) = z\right) = \frac{\mathbb{I}\,(u \in U(z))}{2^r} = q(z \,|\, x) \frac{\mathbb{I}\,(u \in U(z))}{2^r q(z \,|\, x)}. \tag{19}$$

Thus,

$$Q(\{z_i\}_{i=1}^N, \{u_i\}_{i=1}^N, j \,|\, x) = \frac{\mathbb{I}\,(u_1 \in \mathbb{U})}{2^r} \prod_{i=2}^N \mathbb{I}\,(T_i(u_1) = u_i) \prod_{i=1}^N \mathbb{I}\left(F_q^{-1}(u_i) = z_i\right) \frac{w_j}{\sum_{i=1}^N w_i} \tag{20}$$

$$= \frac{\mathbb{I}\,(u_j \in \mathbb{U})}{2^r} \prod_{i \neq j} \mathbb{I}\left(T_i(T_j^{-1}(u_j)) = u_i\right) \prod_{i=1}^N \mathbb{I}\left(F_q^{-1}(u_i) = z_i\right) \frac{w_j}{\sum_{i=1}^N w_i} \tag{21}$$

$$= q(z_j \,|\, x) \frac{\mathbb{I}\,(u_j \in U(z_j))}{2^r q(z_j \,|\, x)} \prod_{i \neq j} \mathbb{I}\left(T_i(T_j^{-1}(u_j)) = u_i\right) \mathbb{I}\left(F_q^{-1}(u_i) = z_i\right) \frac{w_j}{\sum_{i=1}^N w_i}. \tag{22}$$

From this it directly follows that

$$\frac{P(x, \{z_i\}_{i=1}^N, \{u_i\}_{i=1}^N, j)}{Q(\{z_i\}_{i=1}^N, \{u_i\}_{i=1}^N, j \,|\, x)} = \frac{1}{N} \frac{\sum_{i=1}^N w_i}{w_j} \frac{p(x, z_j)}{q(z_j \,|\, x)} = \sum_{i=1}^N \frac{w_i}{N} = \sum_{i=1}^N \frac{1}{N} \frac{p(x, F_q^{-1}(T_i(u_1)))}{q(F_q^{-1}(T_i(u_1)) \,|\, x)}, \tag{23}$$

which is exactly the CIS estimator.

**BB-CIS coder** Based on the extended latent space representation, the Bits-Back Coupled Importance Sampling (BB-CIS) coder is derived in Alg. 6 and visualized in Fig. 6b. The expected net bit

length for encoding a symbol $x$ is:

$$
\begin{aligned}
-\mathbb{E}_{u_1,j}\left[\log \frac{P(x,\{z_i\}_{i=1}^N,\{u_i\}_{i=1}^N,j)}{Q(\{z_i\}_{i=1}^N,\{u_i\}_{i=1}^N,j\,|\,x)}\right] &= -\mathbb{E}_{u_1}\left[\log\left(\sum_{i=1}^N \frac{w_i}{N}\right)\right] \\
&= -\mathbb{E}_{u_1}\left[\log\left(\sum_{i=1}^N \frac{1}{N}\frac{p(x,F_q^{-1}(T_i(u_1)))}{q(F_q^{-1}(T_i(u_1))\,|\,x)}\right)\right].
\end{aligned} \tag{24}
$$

Here we explain BB-CIS in more detail to give intuition about the derivations of BB-CIS and its extended latent space representation. BB-CIS uses couplings in a latent decoding process that saves initial bits. It decodes $u_1$ with $\mathrm{unif}\{0\,..\,2^r-1\}$, sets $z_i = F_q^{-1}(T_i(u_1))$, and decodes the index $j$ of the special particle $z_j$ with $\mathrm{Cat}(\tilde{w}_j)$. This reduces the initial bit cost to $r - \log \tilde{w}_j \in \mathcal{O}(\log N)$. However, the challenge is showing that a net bitrate comparable to BB-IS is still achievable under such a reparameterization.

The coupled latent decoding process needs to be matched with an appropriate encoding process for $x$. Suppose that we finished encoding $x$, as with BB-IS, by encoding $\{z_i\}_{i\neq j}$ with $q(z_i\,|\,x)$ i.i.d. and encoding $(x, z_j, j)$ with $p(x, z_j)/N$. The net bitrate would be

$$
\mathbb{E}_{u_1}\left[-r - \sum_{i=1}^N \log q(F_q^{-1}(T_i(u_1)))\,|\,x) - \log\left(\sum_{i=1}^N \frac{1}{N}\frac{p(x,F_q^{-1}(T_i(u_1))))}{q(F_q^{-1}(T_i(u_1)))\,|\,x)}\right)\right]. \tag{25}
$$

This is clearly worse than BB-IS. The culprits are the $N-1$ latents that BB-IS pushes onto $m$, which is wasteful, because they are deterministically coupled. Fundamentally, the encoding process of BB-IS is not balanced with the initial bit savings of our coupled latent decoding process.

The solution is to design an encoding process over $\{z_i\}_{i=1}^N$, $\{u_i\}_{i=1}^N$, and $x$, which exactly matches the initial bit savings of the coupled decoding process. The idea is to encode just $(u_j, x, z_j, j)$, which is enough information for the receiver to reconstruct all other variables. The key is to design the encoding for $u_j$. Encoding $u_j$ with a uniform on $\{0\,..\,2^r-1\}$ is unnecessarily wasteful, because $z_j$ restricts the range of $u_j$. It turns out, that the best we can do is to encode $u_j$ with $\mathrm{unif}\{u : F_q^{-1}(u) = z_j\}$. Finally $(x, z_j, j)$ are encoded with $p(x, z_j)/N$. This gives the BB-CIS coder and its expected net bitrate exactly achieves equation (24) which is comparable to that of BB-IS.

The convergence of BB-CIS's net bitrate to $-\log p(x)$ will depend on the $T_i$. There are many possibilities, but one consideration is that both sender and receiver must share the $T_i$ (or a procedure for generating them). Another consideration is that the number of particles should never exceed $2^r$, $N \leq 2^r$. This is because we can execute numerical integration with a budget of $2^r$ particles. Assuming $T_i(u) = i$ for $i \in \{0\,..\,2^r-1\}$:

$$
\sum_{i=1}^N \frac{1}{N}\frac{p(x,F_q^{-1}(T_i(u_1)))}{q(F_q^{-1}(T_i(u_1))\,|\,x)} = \sum_{u=0}^{2^r-1}\frac{1}{2^r}\frac{p(x,F_q^{-1}(u))}{q(F_q^{-1}(u)\,|\,x)} = \sum_{z\in\mathbb{S}'} q(z\,|\,x)\frac{p(x,z)}{q(z\,|\,x)} = p(x). \tag{26}
$$

We briefly mention two possibilities:

1. Let $T_i(u) = (u + k_i) \bmod 2^r$ where $k_i \sim \mathrm{unif}\{0\,..\,2^r-1\}$ i.i.d. generated by a pseudo-random number generator where the sender and receiver share the seed. This will enjoy a convergence rate similar to BB-IS, because $\{(u + k_i) \bmod 2^r\}_{i=1}^N$ will be i.i.d. uniform on $\{0\,..\,2^r-1\}$ for $u \sim \mathrm{unif}\{0\,..\,2^r-1\}$. This is used in our experiments.
2. Inspired by the idea of numerical integration, let $T_i(u) = (u + k_i) \bmod 2^r$ where $k_i$ is the $i$th element of a random permutation of $\{0\,..\,2^r-1\}$. With this scheme, BB-CIS is performing numerical integration on a random permutation of $\{0\,..\,2^r-1\}$. The rate at which the net bitrate converges to $-\log p(x)$ could clearly be made worse by inefficient permutations. Randomizing the order may help avoid such inefficient permutations. We did not experiment with this.

### B.6 Bits-Back Annealed Importance Sampling

Annealed importance sampling (AIS) generalizes importance sampling by introducing a path of intermediate distributions between the tractable base distribution $q(z\,|\,x)$ and the unnormalized target

**Algorithm 7:** Extended Latent Space Representation of Annealed Importance Sampling

**Process** $Q(\mathcal{Z} \mid x)$
- sample $z_1 \sim q(z_1 \mid x) = \pi_0(z_1)$
- **for** $i = 1, \ldots, N-1$ **do**
  - sample $z_{i+1} \sim \mathcal{T}_i(z_{i+1} \mid z_i)$
- **return** $\{z_i\}_{i=1}^N$

**Process** $P(x, \mathcal{Z})$
- sample $z_N \sim p(z_N)$
- **for** $i = N-1, \ldots, 1$ **do**
  - sample $z_i \sim \tilde{\mathcal{T}}_i(z_i \mid z_{i+1})$
- sample $x \sim p(x \mid z_N)$
- **return** $x, \{z_i\}_{i=1}^N$

distribution $p(x, z)$. For AIS with $N$ steps, for $i \in \{0, \ldots, N\}$, define annealing distributions

$$\pi_i(z) \propto f_i(z) = q(z \mid x)^{1-\beta_i} p(x, z)^{\beta_i} \tag{27}$$

$$\beta_i \in [0, 1], \beta_0 = 0, \beta_i < \beta_{i+1}, \beta_N = 1. \tag{28}$$

Note that $\pi_0(z) = f_0(z) = q(z \mid x)$. Then define the MCMC transition operator $\mathcal{T}_i$ that leave the intermediate distribution $\pi_i$ invariant and its reverse $\tilde{\mathcal{T}}_i$:

$$\int \mathcal{T}_i(z' \mid z) \pi_i(z)\, dz = \pi_i(z') \tag{29}$$

$$\tilde{\mathcal{T}}_i(z' \mid z) = \mathcal{T}_i(z \mid z') \frac{\pi_i(z')}{\pi_i(z)} = \mathcal{T}_i(z \mid z') \frac{f_i(z')}{f_i(z)}. \tag{30}$$

Note, $\tilde{\mathcal{T}}_i$ is a normalized distribution. AIS samples a sequence of latents $\{z_i\}_{i=1}^N$ from base distribution and the MCMC transition kernels (see the Q process in Alg. 7) and obtains an unbiased estimate of $p(x)$ as the importance weight over the extended space.

$$\hat{p}_N(x) = \frac{f_1(z_1) f_2(z_2) \ldots f_N(z_N)}{f_0(z_1) f_1(z_2) \ldots f_{N-1}(z_N)}. \tag{31}$$

The corresponding variational bound is the log importance weight:

$$-\mathbb{E}_{\{z_i\}_{i=1}^N} \left[ \log \frac{f_1(z_1) f_2(z_2) \ldots f_N(z_N)}{f_0(z_1) f_1(z_2) \ldots f_{N-1}(z_N)} \right] \geq -\log p(x). \tag{32}$$

**Extended latent space representation** The extended space representation of AIS is derived in (Neal, 2001), as presented in Alg. 7. The extended latent variables contain all the latents $\{z_i\}_{i=1}^N$. Briefly, given $x$, the distribution $Q$ first samples $z_1$ from the base distribution $q(z_1 \mid x)$ and then sample $\{z_i\}_{i=2}^N$ sequentially from the transition kernel $\mathcal{T}_i(z_{i+1} \mid z_i)$. The distribution $P$ first samples $z_N$ from the prior $p(z_N)$, then samples $\{z_i\}_{i=N-1}^1$ from the reverse transition kernel $\tilde{\mathcal{T}}_i(z_i \mid z_{i+1})$ in the reverse order, and finally samples $x$ from $p(x \mid z_N)$. The proposal distribution and the target distribution can be derived as followed:

$$Q(\{z_i\}_{i=1}^N) = q(z_1 \mid x) \mathcal{T}_1(z_2 \mid z_1) \mathcal{T}_2(z_3 \mid z_2) \ldots \mathcal{T}_{N-1}(z_N \mid z_{N-1}) \tag{33}$$

$$= f_0(z_1) \mathcal{T}_1(z_2 \mid z_1) \mathcal{T}_2(z_3 \mid z_2) \ldots \mathcal{T}_{N-1}(z_N \mid z_{N-1}) \tag{34}$$

$$P(x, \{z_i\}_{i=1}^N) = p(x, z_N) \tilde{\mathcal{T}}_{N-1}(z_{N-1} \mid z_N) \tilde{\mathcal{T}}_{N-2}(z_{N-2} \mid z_{N-1}) \ldots \tilde{\mathcal{T}}_1(z_1 \mid z_2) \tag{35}$$

$$= f_N(z_N) \tilde{\mathcal{T}}_{N-1}(z_{N-1} \mid z_N) \tilde{\mathcal{T}}_{N-2}(z_{N-2} \mid z_{N-1}) \ldots \tilde{\mathcal{T}}_1(z_1 \mid z_2). \tag{36}$$

Now note

$$\frac{P(x, \{z_i\}_{i=1}^N)}{Q(\{z_i\}_{i=1}^N)} = \frac{f_N(z_N) \tilde{\mathcal{T}}_{N-1}(z_{N-1} | z_N) \tilde{\mathcal{T}}_{N-2}(z_{N-2} | z_{N-1}) \ldots \tilde{\mathcal{T}}_1(z_1 | z_2)}{f_0(z_1) \mathcal{T}_1(z_2 | z_1) \mathcal{T}_2(z_3 | z_2) \ldots \mathcal{T}_{N-1}(z_N | z_{N-1})} \tag{37}$$

$$= \frac{f_1(z_1) f_2(z_2) \ldots f_N(z_N)}{f_0(z_1) f_1(z_2) \ldots f_{N-1}(z_N)}, \tag{38}$$

which exactly gives us the AIS estimator. Note that we have used equation (30) above.

**Algorithm 8:** Encode Procedure of BB-AIS

**Procedure** `Encode` (*symbol $x$, message $m$*)
- decode $z_1$ with $q(z_1 \mid x) = \pi_0(z_1)$
- **for** $i = 1, \dots, N-1$ **do**
  - decode $z_{i+1}$ with $\mathcal{T}_i(z_{i+1} \mid z_i)$
- encode $x$ with $p(x \mid z_N)$
- **for** $i = 1, \dots, N-1$ **do**
  - encode $z_i$ with $\tilde{\mathcal{T}}_i(z_i \mid z_{i+1})$
- encode $z_N$ with $p(z_N)$
- **return** $m'$

**Algorithm 9:** Encode Procedure of BB-AIS with BitSwap

**Procedure** `Encode` (*symbol $x$, message $m$*)
- decode $z_1$ with $q(z_1 \mid x) = \pi_0(z_1)$
- **for** $i = 1, \dots, N-1$ **do**
  - decode $z_{i+1}$ with $\mathcal{T}_i(z_{i+1} \mid z_i)$
  - encode $z_i$ with $\tilde{\mathcal{T}}_i(z_i \mid z_{i+1})$
- encode $x$ with $p(x \mid z_N)$
- encode $z_N$ with $p(z_N)$
- **return** $m'$

**BB-AIS coder** Based on the extended space representation of AIS, the Bits-Back Annealed Importance Sampling coder is derived in Alg. 8 and visualized in Fig. 8a. The expected net bit length for encoding a symbol $x$ is:

$$-\mathbb{E}_{\{z_i\}_{i=1}^N}\left[\frac{\log P(x, \{z_i\}_{i=1}^N)}{\log Q(\{z_i\}_{i=1}^N \mid x)}\right] = -\mathbb{E}_{\{z_i\}_{i=1}^N}\left[\log \frac{f_1(z_1) f_2(z_2) \dots f_N(z_N)}{f_0(z_1) f_1(z_2) \dots f_{N-1}(z_N)}\right]. \tag{39}$$

Ignoring the dirty bits issue, i.e., $\{z_i\}_{i=1}^N \sim Q(\{z_i\}_{i=1}^N \mid x)$, the expected net bit length exactly achieves the negative AIS bound.

The BB-AIS coder has several practical issues. First is the computational cost. Since the intermediate distributions are usually not factorized over the latent dimensions, it is very inefficient to encode and decode the latents with entropy coders for high dimensional latents. Second is the increased initial bit cost as with BB-IS. More precisely, the initial bits that BB-AIS requires is as followed:

$$-\log q(z_1 \mid x) - \sum_{i=1}^{N-1} \log \mathcal{T}_i(z_{i+1} \mid z_i), \tag{40}$$

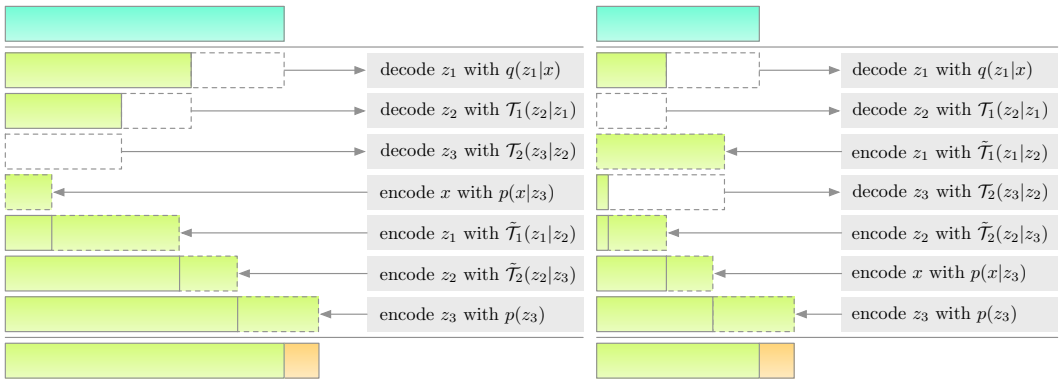
which scales like $\mathcal{O}(N)$. However, because the structure of BB-AIS is very similar to the hierarchical latent variable modelss with Markov structure in (Kingma et al., 2019), the BitSwap trick can be applied with BB-AIS to reduce the initial bit cost. The BB-AIS coder with BitSwap is derived in Alg. 9 and visualized in 8b. In particular, note that the latents $\{z_i\}_{i=1}^N$ in BB-AIS are encoded and decoded both in the forward time order, we can interleave the encode/decode operations such that $z_i$ are encoded with $z_{i+1}$ as soon as $z_{i+1}$ are decoded. Thus there are always at least $-\log \tilde{\mathcal{T}}_i(z_i \mid z_{i+1})$ available for decoding $z_{i+2}$ at the next step, and the initial bit cost is bounded by:

$$-\log q(z_1 \mid x) - \log \mathcal{T}_1(z_2 \mid z_1) + \sum_{i=1}^{N-2} \max(0, -\log \mathcal{T}_{i+1}(z_{i+2} \mid z_{i+1}) + \log \tilde{\mathcal{T}}_i(z_i \mid z_{i+1})). \tag{41}$$

Although BitSwap helps to reduce the initial bit cost, we find that it suffers more from the dirty bits issue than the naive implementation and affects the expected net bit length (see Fig. **??** for an empirical study). Therefore, using BB-AIS with BitSwap leads to a trade-off between net bitrate distortion and inital bit cost that depends on the number of symbols to be encoded.

### B.7 Bits-Back Sequential Monte Carlo

Sequential Monte Carlo (SMC) is a particle filtering method that combines importance sampling with resampling, and its estimate of marginal likelihood typically converges faster than importance




(a) Bits-Back Annealed Importance Sampling

(b) Bits-Back Annealed Importance Sampling with BitSwap

Figure 8: The encoding schemes of BB-AIS and BB-AIS with BitSwap both with $N = 3$ steps. Encoding a symbol $x$ with BB-AIS incurs a very large initial bit cost that scales like $\mathcal{O}(N)$. This can be significantly reduced by applying the BitSwap trick.

sampling for sequential latent variable models Cérou et al. (2011); Bérard et al. (2014). Suppose the observations are a sequence of $T$ random variables $\boldsymbol{x}_T \in \mathbb{S}^T$, where $\boldsymbol{x}_t := x_{1:t}$. Sequential latent variable models introduce a sequence of $T$ (unobserved) latent variables $\boldsymbol{z}_T \in \mathbb{S}'^T$ associated with $\boldsymbol{x}_T$, where $\boldsymbol{z}_t := z_{1:t}$. We assume the joint distribution $p(\boldsymbol{x}_T, \boldsymbol{z}_T)$ can be factored as:

$$p(\boldsymbol{x}_T, \boldsymbol{z}_T) = \prod_{t=1}^{T} f(z_t \,|\, \boldsymbol{x}_{t-1}, \boldsymbol{z}_{t-1}) g(x_t \,|\, \boldsymbol{x}_{t-1}, \boldsymbol{z}_t), \tag{42}$$

where $f$ and $g$ are (generalized) transition and emission distributions, respectively. For the case $t = 1$, $f(z_1 \,|\, \emptyset, \emptyset) = \mu(z_1)$ reduces to the prior distribution and $g(x_1 \,|\, \emptyset, z_1)$ only conditions on $z_1$. We also assume the approximate posterior $q(\boldsymbol{z}_T \,|\, \boldsymbol{x}_T)$ can be factorized as:

$$q(\boldsymbol{z}_T \,|\, \boldsymbol{x}_T) = \prod_{t=1}^{T} q(z_t \,|\, \boldsymbol{x}_t, \boldsymbol{z}_{t-1}). \tag{43}$$

For the case $t = 1$, $q(z_1 \,|\, x_1, \emptyset)$ only conditions on $x_1$.

SMC maintains a population of particle states $\boldsymbol{Z}_t := z_{1:t}^{1:N}$. Here we use subscript for indexing timestep $t$ and superscript for indexing particle $i$. In this appendix we describe the version of SMC that resamples at every iteration. Adaptive resampling schemes are possible (Doucet et al., 2001), but we omit them for clarity.

At each time step $t$, each particle independently samples an extension $z_t^i \sim q(z_t^i \,|\, \boldsymbol{x}_t, \boldsymbol{\tau}_{t-1}^i)$ where $\boldsymbol{\tau}_{t-1}^i$ is the ancestral trajectory of the $i$th particle at timestep $t$ (will be described in details later). The sampled extension $z_t^i$ is appended to $\boldsymbol{\tau}_{t-1}^i$ to form the trajectory of $i$th particle as $(\boldsymbol{\tau}_{t-1}^i, z_t^i)$. Then the whole population is resampled with probabilities in proportion to importance weights defined as followed:

$$w_t^i = \frac{f(z_t^i \,|\, \boldsymbol{x}_{t-1}, \boldsymbol{\tau}_{t-1}^i) g(x_t \,|\, \boldsymbol{x}_{t-1}, (\boldsymbol{\tau}_{t-1}^i, z_t^i))}{q(z_t^i \,|\, \boldsymbol{x}_t, \boldsymbol{\tau}_{t-1}^i)} \tag{44}$$

$$\tilde{w}_t^i = \frac{w_t^i}{\sum_{i=1}^N w_t^i}. \tag{45}$$

In particular, each particle samples a 'parent' index $A_t^i \sim \mathrm{Cat}(\tilde{w}_t^i)$. At the next timestep $t+1$, the $i$th particle 'inherits' the ancestral trajectory of the $A_t^i$th particle. More precisely,

$$\boldsymbol{\tau}_t^i = (\boldsymbol{\tau}_{t-1}^{A_t^i}, z_t^{A_t^i}). \tag{46}$$

Thus, at timestep $t+1$, given the parent index $A_t^i$, as well as the collection of particle states $\boldsymbol{Z}_t$ and previous ancestral indices $\boldsymbol{A}_{t-1} := A_{1:t-1}^{1:N}$, one can trace back the whole ancestral trajectory $\boldsymbol{\tau}_t^i$ by recursively applying equation (46).

**Algorithm 10:** Tracing Back the Ancestral Trajectory $\boldsymbol{\tau}_{t-1}^i$ for the $i$th Particle at Timestep $t$

**Procedure** `TraceBack` ($\boldsymbol{Z}_{t-1}, \boldsymbol{A}_{t-2}, A_{t-1}^i$)
  assign $B_{t-1}^i = A_{t-1}^i$
  **for** $k = t-2, \ldots, 1$ **do**
    assign $B_k^i = A_k^{B_{k+1}^i}$
  assign $\boldsymbol{\tau}_{t-1}^i = (z_1^{B_1^i}, z_2^{B_2^i}, \ldots, z_{t-1}^{B_{t-1}^i})$
  **return** $\boldsymbol{\tau}_{t-1}^i$

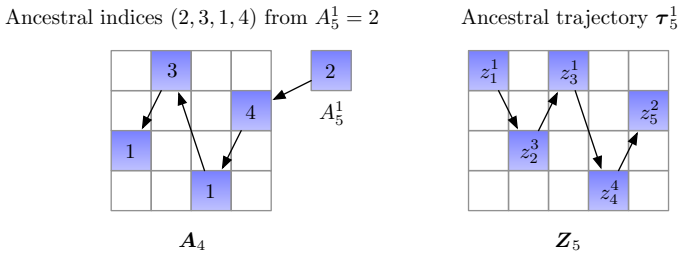



Figure 9: Trace back the ancestral trajectory $\boldsymbol{\tau}_{t-1}^i$ for the particle $i = 1$ at timestep $t = 6$.

For brevity and consistency, we provide the pseudocode (Alg. 10) and visualization (Fig. 9) for tracing back the ancestral trajectory $\boldsymbol{\tau}_{t-1}^i$ for the $i$th particle at timestep $t$. In particular, we first obtain the ancestral index $B_k^i$ at each previous timestep $k$ by using the fact

$$B_k^i = A_k^{B_{k+1}^i}. \tag{47}$$

Thus we can obtain a series of ancestral indices $(B_1^i, \ldots, B_{t-1}^i)$ in the backward time order and then obtain the ancestral trajectory by indexing $\boldsymbol{Z}_{t-1}$. Note that Alg. 10 is only for helping understand the SMC algorithm, in practice, typically the ancestral trajectories are book-kept and updated along the way using equation (47).

SMC obtains an unbiased estimate of $p(\boldsymbol{x}_T)$ using the intermediate importance weights:

$$\hat{p}_N(\boldsymbol{x}_T) = \prod_{t=1}^{T} \left( \frac{1}{N} \sum_{i=1}^{N} w_t^i \right). \tag{48}$$

The corresponding variational bound (FIVO, Maddison et al., 2017; Naesseth et al., 2018; Le et al., 2018) is

$$-\mathbb{E}_{\boldsymbol{Z}_T, \boldsymbol{A}_{T-1}} \left[ \sum_{t=1}^{T} \log \left( \frac{1}{N} \sum_{i=1}^{N} w_t^i \right) \right] \geq -\log p(\boldsymbol{x}_T). \tag{49}$$

**Extended latent space representation** The extended space representation of SMC is derived in (Andrieu et al., 2010), as presented in Alg. 11. The extended latent space variables contain the particle states $\boldsymbol{Z}_T$, ancestral indices $\boldsymbol{A}_{T-1}$, and a particle index $j$ used to pick one special particle trajectory. Briefly, the $Q$ distribution samples $\boldsymbol{Z}_T$ and $\boldsymbol{A}_{T-1}$ in the same manner as SMC, with an additional step that samples the particle index $j$ with probability proportional to the importance weight $w_T$. The $P$ distribution selects the ancestral indices of the special particle $(B_1, \ldots, B_T)$ uniformly, and samples the special particle trajectory $\boldsymbol{z}_T^* = (z_1^{B_1}, \ldots, z_T^{B_T})$ and the observations $\boldsymbol{x}_T$ jointly with the underlying model distribution. The remaining particle states and ancestral indices are sampled with the same distribution as $Q$. Note that the special particle trajectory $\boldsymbol{z}_T^* = \texttt{TraceBack}(\boldsymbol{Z}_T, \boldsymbol{A}_{T-1}, j)$.

From Alg. 11, the proposal distribution and the target distribution and be derived as followed:

$$Q(\boldsymbol{Z}_T, \boldsymbol{A}_{T-1}, j \,|\, \boldsymbol{x}_T) = \tilde{w}_T^j \prod_{i=1}^{N} q(z_1^i | x_1) \prod_{t=2}^{T} \prod_{i=1}^{N} \tilde{w}_{t-1}^{A_{t-1}^i} q(z_t^i \,|\, \boldsymbol{x}_t, \boldsymbol{\tau}_{t-1}^i) \tag{50}$$

**Algorithm 11:** Extended Latent Space Representation of Sequential Monte Carlo

**Process** $Q(\mathcal{Z} \mid \boldsymbol{x}_T)$
- **for** $t = 1, \ldots, T$ **do**
  - **if** $t \neq 1$ **then**
    - **for** $i = 1, \ldots, N$ **do**
      - sample $A_{t-1}^i \sim \mathrm{Cat}(\tilde{w}_{t-1})$
    - assign $\boldsymbol{A}_{t-1} = \left[\boldsymbol{A}_{t-2}, A_{t-1}^{1:N}\right]$
  - **for** $i = 1, \ldots, N$ **do**
    - assign $\boldsymbol{\tau}_{t-1}^i = \texttt{TraceBack}(\boldsymbol{Z}_{t-1}, \boldsymbol{A}_{t-2}, A_{t-1}^i)$
    - sample $z_t^i \sim q(z_t^i \mid \boldsymbol{x}_t, \boldsymbol{\tau}_{t-1}^i)$
  - assign $\boldsymbol{Z}_t = \left[\boldsymbol{Z}_{t-1}, z_t^{1:N}\right]$
- sample $j \sim \mathrm{Cat}(\tilde{w}_T)$
- **return** $\boldsymbol{Z}_T, \boldsymbol{A}_{T-1}, j$

**Process** $P(\boldsymbol{x}_T, \mathcal{Z})$
- **for** $t = 1, \ldots, T$ **do**
  - sample $B_t \sim \mathrm{Cat}(1/N)$
  - **if** $t \neq 1$ **then**
    - assign $A_{t-1}^{B_t} = B_{t-1}$
    - **for** $i \neq B_t$ **do**
      - sample $A_{t-1}^i \sim \mathrm{Cat}(\tilde{w}_{t-1})$
    - assign $\boldsymbol{A}_{t-1} = \left[\boldsymbol{A}_{t-2}, A_{t-1}^{1:N}\right]$
  - assign $\boldsymbol{\tau}_{t-1}^{B_t} = \texttt{TraceBack}(\boldsymbol{Z}_{t-1}, \boldsymbol{A}_{t-2}, B_{t-1})$
  - sample $z_t^{B_t} \sim f(z_t^{B_t} \mid \boldsymbol{x}_{t-1}, \boldsymbol{\tau}_{t-1}^{B_t})$
  - sample $x_t \sim g(x_t \mid \boldsymbol{x}_{t-1}, (\boldsymbol{\tau}_{t-1}^{B_t}, z_t^{B_t}))$
  - **for** $i \neq B_t$ **do**
    - assign $\boldsymbol{\tau}_{t-1}^i = \texttt{TraceBack}(\boldsymbol{Z}_{t-1}, \boldsymbol{A}_{t-2}, A_{t-1}^i)$
    - sample $z_t^i \sim q(z_t^i \mid \boldsymbol{x}_t, \boldsymbol{\tau}_{t-1}^i)$
  - assign $\boldsymbol{Z}_t = \left[\boldsymbol{Z}_{t-1}, z_t^{1:N}\right]$
- assign $j = B_T$
- **return** $\boldsymbol{x}_T, \boldsymbol{Z}_T, \boldsymbol{A}_{T-1}, j$

$$P(\boldsymbol{x}_T, \boldsymbol{Z}_T, \boldsymbol{A}_{T-1}, j) = \frac{1}{N^T}\mu(z_1^{B_1})g(x_1|z_1^{B_1})\prod_{t=2}^{T} f(z_t^{B_t} \mid \boldsymbol{x}_{t-1}, \boldsymbol{\tau}_{t-1}^{B_t})g(x_t \mid \boldsymbol{x}_{t-1}, (\boldsymbol{\tau}_{t-1}^{B_t}, z_t^{B_t}))$$

$$\prod_{i \neq B_1} q(z_1^i|x_1)\prod_{t=2}^{T}\prod_{i \neq B_t} \tilde{w}_{t-1}^{A_{t-1}^i} q(z_t^i \mid \boldsymbol{x}_t, \boldsymbol{\tau}_{t-1}^i) \tag{51}$$

$$= \frac{1}{N^T} p(\boldsymbol{x}_T, \boldsymbol{z}_T^*) \prod_{i \neq B_1} q(z_1^i|x_1)\prod_{t=2}^{T}\prod_{i \neq B_t} \tilde{w}_{t-1}^{A_{t-1}^i} q(z_t^i \mid \boldsymbol{x}_t, \boldsymbol{\tau}_{t-1}^i). \tag{52}$$

**Algorithm 12:** Encode Procedure of BB-SMC

**Procedure** `Encode` (*symbol* $\boldsymbol{x}_T$, *message* $m$)
- **for** $t = 1, \ldots, T$ **do**
  - **if** $t \neq 1$ **then**
    - **for** $i = 1, \ldots, N$ **do**
      - decode $A_{t-1}^i$ with $\mathrm{Cat}(\tilde{w}_{t-1})$
    - assign $\boldsymbol{A}_{t-1} = \left[\boldsymbol{A}_{t-2}, A_{t-1}^{1:N}\right]$
  - **for** $i = 1, \ldots, N$ **do**
    - assign $\boldsymbol{\tau}_{t-1}^i = \texttt{TraceBack}(\boldsymbol{Z}_{t-1}, \boldsymbol{A}_{t-2}, A_{t-1}^i)$
    - decode $z_t^i$ with $q(z_t^i \,|\, \boldsymbol{x}_t, \boldsymbol{\tau}_{t-1}^i)$
  - assign $\boldsymbol{Z}_t = \left[\boldsymbol{Z}_{t-1}, z_t^{1:N}\right]$
- decode $j$ with $\mathrm{Cat}(\tilde{w}_T)$
- set the ancestral lineage $B_T = j$ and $B_t = A_t^{B_{t+1}}$ for $t = T-1, \ldots, 1$
- **for** $t = T, \ldots, 1$ **do**
  - **for** $i \neq B_t$ **do**
    - encode $z_t^i$ with $q(z_t^i \,|\, \boldsymbol{x}_t, \boldsymbol{\tau}_{t-1}^i)$
  - encode $x_t$ with $g(x_t \,|\, \boldsymbol{x}_{t-1}, (\boldsymbol{\tau}_{t-1}^{B_t}, z_t^{B_t}))$
  - encode $z_t^{B_t}$ with $f(z_t^{B_t} \,|\, \boldsymbol{x}_{t-1}, \boldsymbol{\tau}_{t-1}^{B_t})$
  - **if** $t \neq 1$ **then**
    - **for** $i \neq B_t$ **do**
      - encode $A_{t-1}^i$ with $\mathrm{Cat}(\tilde{w}_{t-1})$
  - encode $B_t$ with $\mathrm{Cat}(1/N)$
- **return** $m'$

Now note,

$$\frac{P(\boldsymbol{x}_T, \boldsymbol{Z}_T, \boldsymbol{A}_{T-1}, j)}{Q(\boldsymbol{Z}_T, \boldsymbol{A}_{T-1}, j \,|\, \boldsymbol{x}_T)} = \frac{1}{N^T \tilde{w}_T^j} \frac{p(\boldsymbol{x}_T, \boldsymbol{z}_T^*)}{q(z_1^{B_1}|x_1) \prod_{t=2}^T \tilde{w}_{t-1}^{A_{t-1}^{B_t}} q(z_t^{B_t} \,|\, \boldsymbol{x}_t, \boldsymbol{\tau}_{t-1}^{B_t})} \tag{53}$$

$$= \frac{1}{N^T \tilde{w}_T^j} \frac{p(\boldsymbol{x}_T, \boldsymbol{z}_T^*)}{q(z_1^{B_1}|x_1) \prod_{t=2}^T \tilde{w}_{t-1}^{B_{t-1}} q(z_t^{B_t} \,|\, \boldsymbol{x}_t, \boldsymbol{\tau}_{t-1}^{B_t})} \tag{54}$$

$$= \frac{1}{N^T \prod_{t=1}^T \tilde{w}_t^{B_t}} \frac{\mu(z_1^{B_1}) g(x_1|z_1^{B_1}) \prod_{t=2}^T f(z_t^{B_t} \,|\, \boldsymbol{x}_{t-1}, \boldsymbol{\tau}_{t-1}^{B_t}) g(x_t \,|\, \boldsymbol{x}_{t-1}, (\boldsymbol{\tau}_{t-1}^{B_t}, z_t^{B_t}))}{q(z_1^{B_1}|x_1) \prod_{t=2}^T q(z_t^{B_t} \,|\, \boldsymbol{x}_t, \boldsymbol{\tau}_{t-1}^{B_t})} \tag{55}$$

$$= \frac{\prod_{t=1}^T w_t^{B_t}}{N^T \prod_{t=1}^T \tilde{w}_t^{B_t}} \tag{56}$$

$$= \prod_{t=1}^T \left(\frac{1}{N} \sum_{i=1}^N w_t^i\right), \tag{57}$$

which exactly gives us the SMC estimator. Note that we have used equation (47) and $B_T = j$ in the above derivation.

**BB-SMC coder** Based on the extended space representation of SMC, the Bits-Back Sequential Monte Carlo coder is derived in Alg. 12 and visualized in Fig. 10. Intuitively, BB-SMC first decodes all the extended latent variables $\mathcal{Z} = \{\boldsymbol{Z}_T, \boldsymbol{A}_{T-1}, j\}$ in the forward time order. Then it picks a special particle and traces its trajectory $\boldsymbol{z}_T^*$ backward in time. The distribution of the special trajectory can be seen as a non-parametric approximation of the true posterior. The special trajectory $\boldsymbol{z}_T^*$ is used to encode the sequential observations $\boldsymbol{x}_T$ and is itself encoded with with the model distribution. The special trajectory's ancestral indices are encoded using a uniform distribution. All other extended latent variables are encoded back with the same distribution as decoding and in the backward time

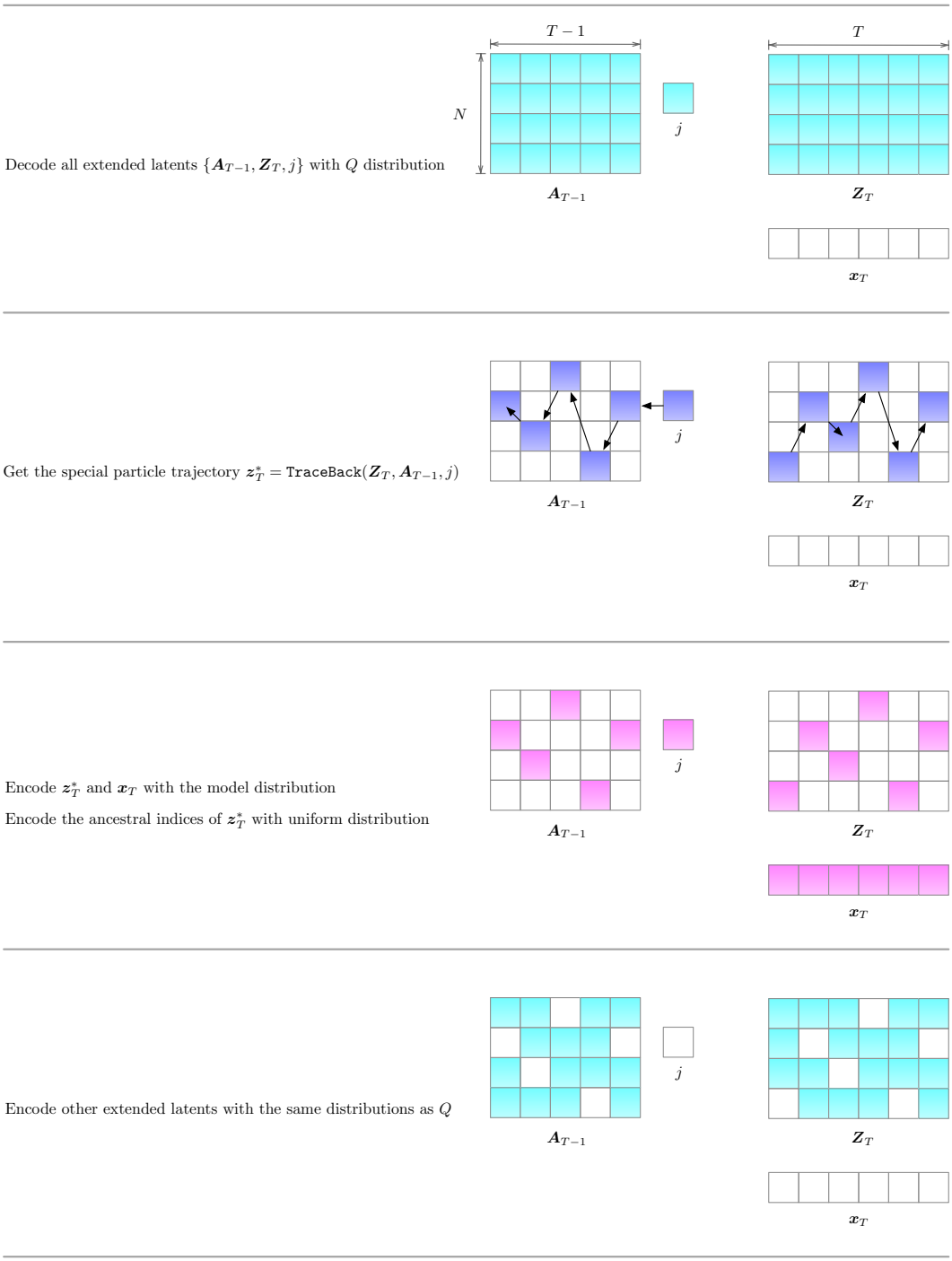



Figure 10: The visualization of the encode procedure of BB-SMC.

**Algorithm 13:** Encode Procedure of BB-CSMC

**Procedure** `Encode` (*symbol* $\boldsymbol{x}_T$*, message* $m$)
- **for** $t = 1, \ldots, T$ **do**
  - **if** $t \neq 1$ **then**
    - decode $v_{t-1}$ with $\mathrm{unif}\{0, \cdots, 2^r - 1\}$
    - assign $p_{t-1} = \mathrm{Cat}(\tilde{w}_{t-1})$
    - **for** $i = 1, \ldots, N$ **do**
      - assign $v_{t-1}^i = R_{t-1}^i(v_{t-1})$
      - assign $A_{t-1}^i = F_{p_{t-1}}^{-1}(v_{t-1}^i)$
    - assign $\boldsymbol{A}_{t-1} = \left[\boldsymbol{A}_{t-2}, A_{t-1}^{1:N}\right]$
  - decode $u_t$ with $\mathrm{unif}\{0, \cdots, 2^r - 1\}$
  - **for** $i = 1, \ldots, N$ **do**
    - assign $\boldsymbol{\tau}_{t-1}^i = \texttt{TraceBack}(\boldsymbol{Z}_{t-1}, \boldsymbol{A}_{t-2}, A_{t-1}^i)$
    - assign $u_t^i = T_t^i(u_t)$, $q_t^i = q(z_t^i \,|\, \boldsymbol{x}_t, \boldsymbol{\tau}_{t-1}^i)$
    - assign $z_t^i = F_{q_t^i}^{-1}(u_t^i)$
  - assign $\boldsymbol{Z}_t = \left[\boldsymbol{Z}_{t-1}, z_t^{1:N}\right]$
- decode $j$ with $\mathrm{Cat}(\tilde{w}_T)$
- set the ancestral lineage $B_T = j$ and $B_t = A_t^{B_{t+1}}$ for $t = T-1, \ldots, 1$
- **for** $t = T, \ldots, 1$ **do**
  - encode $u_t^{B_t}$ with $\mathrm{unif}\{u : F_{q_t^{B_t}}^{-1}(u) = z_t^{B_t}\}$
  - encode $x_t$ with $g(x_t \,|\, \boldsymbol{x}_{t-1}, (\boldsymbol{\tau}_{t-1}^{B_t}, z_t^{B_t}))$
  - encode $z_t^{B_t}$ with $f(z_t^{B_t} \,|\, \boldsymbol{x}_{t-1}, \boldsymbol{\tau}_{t-1}^{B_t})$
  - **if** $t \neq 1$ **then**
    - encode $v_{t-1}^{B_t}$ with $\mathrm{unif}\{v : F_{p_{t-1}}^{-1}(v) = A_{t-1}^{B_t}\}$
  - encode $B_t$ with $\mathrm{Cat}(1/N)$
- **return** $m'$

order. The expected net bit length for encoding a symbol $\boldsymbol{x}_T$ is:

$$-\mathbb{E}_{\boldsymbol{Z}_T, \boldsymbol{A}_{T-1}, j}\left[\frac{\log P(\boldsymbol{x}_T, \boldsymbol{Z}_T, \boldsymbol{A}_{T-1}, j)}{\log Q(\boldsymbol{Z}_T, \boldsymbol{A}_{T-1}, j \,|\, \boldsymbol{x}_T)}\right] = -\mathbb{E}_{\boldsymbol{Z}_T, \boldsymbol{A}_{T-1}}\left[\sum_{t=1}^{T} \log\left(\frac{1}{N}\sum_{i=1}^{N} w_t^i\right)\right]. \tag{58}$$

Ignoring the dirty bits issue, the expected net bit length exactly achieves the negative FIVO bound.

Note that our BB-SMC scheme can be easily extended to adaptive resampling setting where the particles are only resampled when a certain resampling criteria is satisfied. In adaptive schemes, the importance weights accumulate multiplicatively over time (between resampling events). One typical adaptive resampling criteria is applying resampling if the effective sample size (ESS) of the particles drops below $N/2$. Details can be found in (Doucet et al., 2001).

Adaptive resampling is possible, because the same resampling decisions made at encode time by the sender can be exactly recovered by the receiver. The encoding process of the sender produces the SMC state in the forward direction of time. The receiver also reconstructs the the SMC state in the forward direction of time. Thus, if the receiver has the same resampling criteria as the sender, they can recover exactly the SMC forward pass of the sender. Thus, we can make two major modifications to incorporate adaptive resampling to BB-SMC. First is that the parent indices $A_{t-1}^i$ are only decoded (and then encoded back) when the resampling criteria is satisfied at each timestep $t$, otherwise each particle inherits itself, i.e., set $A_{t-1}^i = i$. Second is that when resampling is not performed, the importance weights need to be accumulated and used for resampling next time. After resampling, the accumulated importance weights are reset to uniform for all particles.

**BB-CSMC coder** As with BB-IS, BB-SMC also suffers from a increased initial bits cost equal to $-\log Q(\boldsymbol{Z}_T, \boldsymbol{A}_{T-1}, j)$ that scales like $\mathcal{O}(NT)$, in contrast to the $\mathcal{O}(T)$ initial bit cost of BB-ELBO. Similarly, we can derive a coupled variant of BB-SMC which is called Bits-Back Coupled Sequential Monte Carlo (BB-CSMC).

BB-CSMC reparameterizes the particle states $z_t^i$ as deterministic functions of uniform random variables $u_t^i$ which are coupled by a common uniform $u_t$. At each timestep $t$, instead of directly decoding $z_t^i$ with their approximate posterior $q_t^i$, BB-CSMC first decodes $u_t$ and then obtain $u_t^i$ with bijective functions $T_t^i$, i.e., $u_t^i = T_t^i(u_t)$. Then the particle states $z_t^i$ are obtained as $z_t^i = F_{q_t^i}^{-1}(u_t^i)$, where the functions $F_{q_t^i}^{-1}$ are defined as equation (**??**). This is not enough to sufficiently reduce the $\mathcal{O}(NT)$ initial bit cost since the parent indices $A_{t-1}^i$ are also decoded (and thus require some initial bits) for each particle. Therefore, similarly for $A_{t-1}^i$, we also need to introduce the common uniform $v_{t-1}$ and the bijective functions $R_{t-1}^i$ to obtain $v_{t-1}^i = R_{t-1}^i(v_{t-1})$. $A_{t-1}^i$ are obtained as $A_{t-1}^i = F_{p_{t-1}}^{-1}(v_{t-1}^i)$ where $p_{t-1}$ is the categorical distribution defined by normalized importance weights $\tilde{w}_{t-1}$. This reduces the initial bit cost to $(2T-1)r - \log(\tilde{w}_T)$. Note that $r$ is lower bounded by $\log N$ because $2^r$ should be larger than $N$, the initial bit cost roughly scales like $\mathcal{O}(T \log N)$.

The encoding process of BB-CSMC should also be calibrated to match the modified decoding process, as with BB-CIS. In particular, after decoding the special particle index $j$, BB-CSMC only encodes $(x_t, B_t, v_{t-1}^{B_t}, u_t^{B_t}, z_t^{B_t})$ associated with the special particle trajectory. The uniform $v_{t-1}^{B_t}$ and $u_t^{B_t}$ are encoded using uniform distributions over restricted sets mapped from $A_{t-1}^{B_t} = B_{t-1}$ and $z_t^{B_t}$, respectively.

One can easily compute the net bitrate of BB-CSMC is:

$$-\mathbb{E}_{\{u_t\}_{t=1}^T, \{v_t\}_{t=1}^{T-1}} \left[ \sum_{t=1}^T \log \left( \frac{1}{N} \sum_{i=1}^N w_t^i \right) \right] \tag{59}$$

which is comparable to BB-SMC but uses only $\mathcal{O}(T \log N)$ initial bits.

## C Computational Cost

The computational complexity of our McBits coders can be distilled into 2 components: codec building and encoding/decoding operations. The former encompasses the operations of calculating the encoding/decoding distributions (which usually involves neural network computation) and constructing the data structure used to perform the latter. The number of times a codec is built is equal to the number of different PMFs involved in that particular coding scheme. For example, in BB-IS, the codec for the approximate posterior only needs to be built once for each encoded symbol as it is shared over particles, while the codecs involving the conditional likelihood are built $N$ times for $N$ particles. The codec for the categorical distribution is also built once for each encoded symbol. The latent prior codec is fixed and shared over encoded symbols, and therefore is built only once.

For most of our McBits coders (e.g., BB-IS and BB-AIS), the computational complexity scales like $\mathcal{O}(N)$, where $N$ is the number of particles or AIS steps. In BB-SMC, we are required to decode/encode $N$ times from a categorical distribution of alphabet size $N$ at each timestep. Naively implemented, this has computational complexity $\mathcal{O}(TN^2)$ (in contrast to the $\mathcal{O}(T)$ computational complexity of BB-ELBO), but we believe that this can be reduced to $\mathcal{O}(TN)$ by applying the alias method Walker (1974). In short, given any PMF of alphabet size $N$, the alias method pre-computes a data structure in $\mathcal{O}(N)$ that allows efficient sampling in $\mathcal{O}(1)$. Therefore, we could potentially use the alias method for decoding/encoding the ancestral indices from the categorical distribution that is shared over particles, which reduces the computational complexity to $\mathcal{O}(TN)$. In our experiments, we did not implement the alias method since the computational cost of decoding/encoding ancestral indices only accounted for a small fraction of the overall computational cost.

Furthermore, the computational costs of BB-IS and BB-SMC can be reduced by parallelizing over particles. To illustrate, the computation of neural networks to obtain the conditional likelihood codec can be easily batched over particles on GPUs. For the encoding/decoding operations of ANS, the computation is also parallelizable over particles by using a GPU implementation of ANS (Krajcevski et al., 2016). However, BB-AIS is not easily parallelizable due to its sequential nature.

Compressing with BB-IS on an Intel Xeon Processor (Skylake, IBRS) with a Tesla P100 GPU, we compressed the CIFAR-10 dataset at a rate of 10 images/second with $N = 64$ using the vectorized rANS implementation (CPU-based) Craystack Townsend et al. (2020).

# D EXPERIMENTAL DETAILS

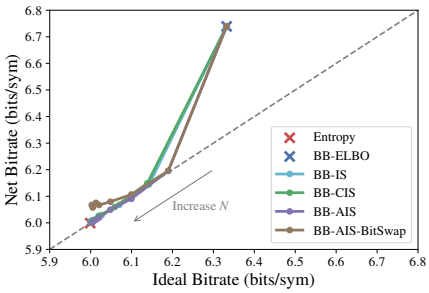
## D.1 LOSSLESS COMPRESSION ON SYNTHETIC DATA

We assessed the convergence properties, impact of dirty bits, and initial bit cost of our McBits coders on synthetic data. We sometimes evaluated the *ideal bitrate*, which for each coder is the corresponding estimated variational bound. Experimental details and results are presented below.

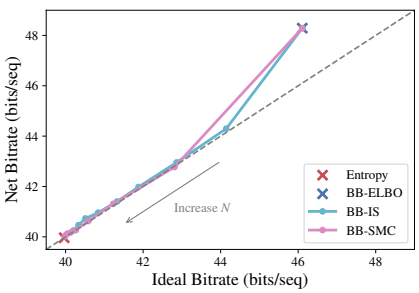
**Mixture Model** We used a mixture model with 1-dimensional observation and latent variables. The alphabet sizes of the observation and the latent were 64 and 256 respectively. The data generating distribution (prior and conditional likelihood distributions) counts were i.i.d. sampled random integers from the range $[1, 20]$ and then normalized. All the coders got access to the true data generating distribution of the model and used a uniform approximate posterior distribution. All coders were evaluated using a message consisting of 5000 symbols i.i.d. sampled from the model to compute the total and net bitrates. The ideal bitrate of each coder was computed by its corresponding (empirical) negative variational bound by resampling the $N$ particle system 100 times and averaging (except for the ELBO bound, which can be computed exactly).

**Hidden Markov Model** We used a hidden Markov model (HMM) with 1-dimensional observation and latent variables. The alphabet sizes of the observation and the latent were 16 and 32 respectively. The number of timesteps was set to 10. The data generating distribution (prior/transition/emission distributions) counts were i.i.d. sampled random integers from the range $[1, 20]$ and then normalized. All the coders got access to the true data generating distribution of the model and used a uniform approximate posterior distribution. All coders were evaluated using a message consisting of 5000 sequences i.i.d. sampled from the model to compute the total and net bitrates. The ideal bitrate of each coder was computed by its corresponding (empirical) negative variational bound of the sampled message, as with the mixture model. The entropy of the model was also computed empirically by the negative marginal likelihood of the sampled message which was computed by the forward algorithm.

**Additional Results** We measured the impact of dirty bits by plotting *ideal* versus *net* bitrates on the toy mixture and the toy HMM model in Fig. 11. The deviation of any point to the dashed line indicates the severity of dirty bits. Interestingly, most of our McBits coders appeared to 'clean' the bitstream as $N$ increased, i.e. the net bitrate converged to the entropy, as shown in Fig. 11a. Only BB-AIS-BitSwap did *not* clean the bitstream (Fig. 11a), indicating that the order of operations has a significant impact on the cleanliness of McBits coders.




(a) As $N \to \infty$, the net bitrates of all coders expect BB-AIS-BitSwap converge to the entropy on the toy mixture model. BB-AIS-BitSwap suffers from the dirty bits issue.




(b) As $N \to \infty$, the net bitrates of BB-IS and BB-SMC coders converge to the entropy on the toy HMM. Both coders do not suffer from dirty bits issue.

Figure 11: (a) & (b): The cleanliness plots of all evaluated coders on the toy mixture model and the toy hidden Markov model. The experimental setups were the same as with Fig. 3a. and Fig, 3c, respectively.

## D.2 LOSSLESS COMPRESSION ON IMAGES

We benchmarked the performance of BB-IS and BB-CIS on the standard train-test splits of two datasets: an alphanumeric extension of MNIST called EMNIST (Cohen et al., 2017), and CIFAR-10 (Krizhevsky, 2009). Experimental details and results are presented below.

**Datasets** We used two datasets for benchmarking lossless image compression: EMNIST (Cohen et al., 2017) and CIFAR-10. EMNIST dataset extends the MNIST dataset to handwritten digits. There are 6 different splits provided in the dataset and we used two of them in our experiments: MNIST and Letters. The EMNIST-MNIST split mimics the original MNIST dataset which contains 60,000 training and 10,000 test examples. The EMNIST-Letters split contains 124800 training and 20800 test examples. Both two splits were dynamically binarized following Salakhutdinov & Murray (2008). Specifically, the observations were randomly sampled from the Bernoulli distribution with expectations equal to the real pixel values. For the CIFAR-10 dataset, no additional preprocessing was applied.

**Model** For the EMNIST datasets, we used the VAE model with 1 stochastic layer as in Burda et al. (2015). The VAE model had 50 latents with a standard Gaussian prior and factorized Gaussian approximate posterior. The conditional likelihood distribution was modeled by the Bernoulli distribution which fit with the binarized observations. The training procudure was the same as Burda et al. (2015). For the CIFAR-10 dataset, we used the VQ-VAE model (Oord et al., 2017) with discrete latent variables. Categorical distributions were used for both the latents and the observations. We followed the experimental setup in Sønderby et al. (2017) and used the VQ-VAE model with 8 latent vairables per spatial dimension. The VQ-VAE model was trained with Gumbel-Softmax (Jang et al., 2016; Maddison et al., 2016) relaxation with a tempreture of 0.5 and a minibatch size of 32. The ADAM optimizer was used for training the model and the learning rate was tuned over $\{1 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}\}$.

**Discretization** To perform bits-back coding with continuous latent variables, we need to discretize the latent space and approximate the continuous prior and the approximate posterior with their discretized variants using the same set of bins. In MacKay (2003) and Townsend et al. (2019), they showed that the continuous latents could be discretized up to an arbitrary precision $\delta_z$ without affecting the *net* bitrate as we could also get those extra bits due to discretization "back". The notable effect is for the *total* bitrate since the initial bit cost would scale with the $-\log \delta_z$, which means that we prefer to use a reasonably small precision in practice. In our EMNIST experiments, continuous latent variables were used for the VAE model and need to be discretized for compression. We used the maximum entropy discretization in Townsend et al. (2019) and discretized the latent space into bins that had equal *mass* under the standard Gaussian prior for all the coders.

**Baselines** We included amortized-iterative inference method (Yang et al., 2020) which also improved the compression rate by bridging the gap of the ELBO bound and the marginal likelihood as a baseline in our experiments. When compressing a data example, it first initializes the *local* variational parameters (e.g., the mean and variance for factorized Gaussian approximate posterior in our experiments) from the trained VAE inference model. Then it optimizes the ELBO objective over the local variational parameters with stochastic gradient decent and uses the optimized variational parameters for compression. This method introduces expensive computation at the compression stage. In our experiments, we kept the number of optimization steps equal to the number of particles $N$ to roughly match the computation budget in terms of the number of quries to the VAE model. However, even so the computation cost of this method is more expensive than our BB-IS/BB-CIS coders because 1) BB-IS/BB-CIS coders with $N$ particles introduce $N$ queries to the VAE generative model and 1 query to the VAE inference model, while amortized-iterative inference with $N$ optimization steps requires $N$ quiries to the whole VAE model and $N$ back-propagation steps; 2) the computation of the BB-IS/BB-CIS coders can be potentially parallelized over the particles, but the computation of amortized-iterative inference needs to be done in $N$ sequential optimization steps. In our experiments, we used ADAM optimizer for optimizing the local variational parameters and the learning rate was tuned in the range $\{5 \times 10^{-4}, 1 \times 10^{-3}, 5 \times 10^{-3}, 1 \times 10^{-2}, 5 \times 10^{-2}\}$.

**Additional Results** We found that the variational bounds used to train the VAE models had an impact on compression performance. When using BB-IS with a model trained on the IWAE objective, equalizing the number of particles during compression and training resulted in better rates than BB-IS with an ELBO-trained VAE (Table 2). Therefore, we always use our McBits coders with models trained on the corresponding variational bound.

To study the potential trade-off of applying our BB-CIS coder, we compared the compression performance of BB-IS and BB-CIS with 50 particles on EMNIST datasets in Table 3. Although the BB-CIS coder adopts a near-random sampling strategy, its ideal bitrate and net bitrate match those of the BB-IS coder. And it is effective for reducing the initial bit cost, as the gap between the total

Table 2: BB-IS performs better on models that were trained with the same number of particles. The net bitrates (bits/dim) of BB-IS on EMNIST-MNIST and CIFAR-10 test sets. $N = 50$ for EMNIST-MNIST and $N = 10$ for CIFAR-10.

| | MNIST | | CIFAR-10 | |
|---|---|---|---|---|
| | ELBO | IWAE | ELBO | IWAE |
| BB-ELBO | 0.236 | 0.236 | 4.898 | 4.898 |
| BB-IS (5) | 0.232 | 0.231 | 4.866 | 4.827 |
| BB-IS ($N$) | 0.230 | **0.228** | 4.857 | **4.810** |
| Savings | 2.5% | 3.4% | 0.8% | 1.8% |

bitrate and the net bitrate is much smaller than that of the BB-IS coder. These observations illustrate that we can use BB-CIS as a drop-in replacement for BB-IS nearly for free.

Table 3: BB-CIS is better than BB-IS in terms of total bitrate while matching the ideal and net bitrates. Comparison was done on the EMNIST-MNIST and EMNIST-Letters test sets. All bitrates were measured in bits/dim.

| Method | MNIST | | | Letters | | |
|---|---|---|---|---|---|---|
| | Ideal | Net | Total | Ideal | Net | Total |
| BB-IS (50) | 0.227 | 0.228 | 0.230 | 0.238 | 0.239 | 0.241 |
| BB-CIS (50) | 0.227 | 0.228 | **0.228** | 0.238 | 0.239 | **0.239** |

Finally, we compared BB-IS and BB-CIS to other benchmark lossless compression schemes by measuring the *total* bitrates on EMNIST test sets (without transferring). We also compared with amortized-iterative inference, (Yang et al., 2020) that optimizes the ELBO objective over local variational parameters for each data example at the compression stage. To roughly match the computation budget, the number of optimization steps was set to 50 and this method is denoted as BB-ELBO-IF (50). Both BB-IS and BB-CIS outperformed all other baselines on both test sets, and BB-CIS was better than BB-IS in terms of total bitrate since it effectively reduces the initial bit cost.

Table 4: BB-CIS achieves the best total bitrates compared to baselines on EMNIST test sets.

| Method | MNIST | Letters |
|---|---|---|
| PNG | 0.819 | 0.900 |
| WebP | 0.464 | 0.533 |
| gzip | 0.423 | 0.413 |
| lzma | 0.383 | 0.369 |
| bz2 | 0.375 | 0.364 |
| BB-ELBO | 0.236 | 0.250 |
| BB-ELBO-IF (50) | 0.233 | 0.246 |
| BB-IS (50) | 0.230 | 0.241 |
| BB-CIS (50) | **0.228** | **0.239** |

As the amortized-iterative inference method is orthogonal to our method, the compression performance of our McBits coder (in particular BB-IS in this experiment) can be further improved by applying amortized-iterative inference at the compression stage. Similar to the original amortized-iterative inference, one can initialize the local variational parameters from the trained VAE inference model and optimize the IWAE objective with $N$ particles (as opposed to the ELBO objective) over them. In Table 5, we compared the *net* bitrate of combining the BB-IS coder and the amortized-iterative inference method. We observed that the amortized-iterative inference method could further boost the compression rate of our BB-IS coder.

Table 5: The compression performance of BB-IS can be improved by combining with amortized-iterative inference method. The net bitrates (bits/dim) are shown in the table.

| | MNIST | | Letters | |
|---|---|---|---|---|
| | w/ IF | w/o IF | w/ IF | w/o IF |
| BB-IS (1) | 0.236 | 0.233 | 0.250 | 0.246 |
| BB-IS (5) | 0.231 | 0.229 | 0.243 | 0.241 |
| BB-IS (50) | 0.228 | **0.226** | 0.239 | **0.237** |

### D.3 Lossless Compression on Sequential Data

We quantified the performance of BB-SMC on sequential data compression tasks with 4 polyphonic music datasets: Nottingham, JSB, MuseData, and Piano-midi.de (Boulanger-Lewandowski et al., 2012). Experimental details and results are presented below.

**Datasets** We used 4 polyphonic music datasets to evaluate the compression performance of the BB-SMC coder on sequential datasets: Nottingham folk tunes, the JSB chorales, the MuseData library of classical piano and orchestral music, and the Piano-midi.de MIDI archive (Boulanger-Lewandowski et al., 2012). All datasets were composed of sequences of binary 88-dimensional vectors representing active musical notes at one timestep. For all datasets, we imitated the experimental setup presented in Maddison et al. (2017). We used the same train/validation/test split and echoed their data preprocessing. The only difference was that we used a chuncked version of these datasets where each sequence was chunked to sub-sequences with maximum length of 100. We found it slightly improved the model performance and significantly reduced the initial bit cost (as the original maximum sequence length was very large).

**Models** All models were based on the variational RNN architecture (Chung et al., 2015). All distributions over latent variables were factorized Gaussians, and the output distributions were factorized Bernoullis for binary observations on 4 polyphonic music datasets. JSB models were trained with 32 hidden units, Muse-data with 256, and all other models with 64 units. For each aforementioned dataset, there was one model trained with the ELBO, IWAE, and FIVO objectives, respectively. All models were trained with 4 particles, a batch size of 4 and the Adam optimizer with learning rate $3 \times 10^{-5}$. All models were initially evaluated on the validation set, which allowed for early stopping. In Table 6, we present our models' performance in nats and bits to allow for easy comparison of generative modelling and compression literature.

Table 6: Sequential model evaluation: we trained VRNN models on the Nottingham, JSB, Musedata and piano-midi.de datasets. For each dataset, we trained 3 VRNN models with the ELBO, IWAE and FIVO objectives, respectively. All models were trained with 4 particles. Our models were trained in an identical fashion as with Table 5 in Maddison et al. (2017). For comparison, we include the estimated data log-likelihood as model evaluation metric. We estimated the log-likelihood by computing the maximum of the ELBO, IWAE and FIVO bound with 128 particles. We include this metric for better comparison to other work. However for this work, this bound is not relevant. Relevant metrics include the respective bounds in nats or bits per time step.

| Training Objective | Evaluation Metric | Unit | Notingham | | JSB | | Musedata | | Piano-midi.de | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Train | Test | Train | Test | Train | Test | Train | Test |
| ELBO | $-\log p(x)$ | nats/step | 3.49 | 4.06 | 8.05 | 8.67 | 6.50 | 7.33 | 7.30 | 7.92 |
| | $-$ELBO | nats/step | 3.50 | 4.07 | 8.07 | 8.67 | 6.53 | 7.38 | 7.31 | 7.93 |
| | $-$ELBO | bits/step | 5.05 | 5.87 | 11.64 | 12.51 | 9.41 | 10.65 | 10.55 | 11.44 |
| IWAE | $-\log p(x)$ | nats/step | 2.51 | 3.03 | 7.50 | 8.13 | 6.40 | 7.33 | 7.25 | 7.88 |
| | $-$IWAE | nats/step | 2.63 | 3.24 | 7.65 | 8.36 | 6.42 | 7.38 | 7.89 | 7.89 |
| | $-$IWAE | bits/step | 3.79 | 4.67 | 11.04 | 12.06 | 9.26 | 10.65 | 11.38 | 11.38 |
| FIVO | $-\log p(x)$ | nats/step | 2.50 | 3.02 | 6.41 | 7.24 | 5.82 | 6.46 | 6.94 | 7.70 |
| | $-$FIVO | nats/step | 2.60 | 3.20 | 6.59 | 7.50 | 5.97 | 6.64 | 6.99 | 7.76 |
| | $-$FIVO | bits/step | 3.75 | 4.62 | 9.51 | 10.82 | 8.61 | 9.58 | 10.08 | 11.20 |

**Results** We compared the *net* bitrates of all coders for compressing each test set in Table 7. BB-SMC clearly outperformed BB-ELBO and BB-IS with the same number of particles on all datasets.

Table 7: BB-SMC achieves the best net bitrates (bits/timestep) on all piano roll test sets.

| | Musedata | Nott. | JSB | Piano. |
|---|---|---|---|---|
| BB-ELBO | 10.66 | 5.87 | 12.53 | 11.43 |
| BB-IS (4) | 10.66 | 4.86 | 12.03 | 11.38 |
| BB-SMC (4) | **9.58** | **4.76** | **10.92** | **11.20** |
| Savings | 10.1% | 18.9% | 12.8% | 2.0% |

Then we compared our coders with benchmark lossless compression schemes in Table 8. Our coders were comparable with those baselines and the BB-SMC coder outperformed all the baselines on the JSB dataset. Note that the compression performance of our coders is bottlenecked by the simple VRNN architecture that we used in our experiments. And we suppose that with more powerful and better trained VRNN models, our coders could outperform those benchmark schemes. However, our main focus is to compare the compression performance of our BB-SMC coder and the BB-ELBO/BB-IS coders with the same VRNN architectures to show the effectiveness of BB-SMC for compressing sequential data, and this is clearly illustrated by the results.

Table 8: The comparison of net bitrate (bits/timestep) with benchmark lossless compression schemes on piano roll test sets.

| Method | Musedata | Nottingham | JSB | Piano-midi.de |
|---|---|---|---|---|
| gzip | 11.01 | 3.86 | 13.94 | 9.46 |
| bz2 | 11.25 | **2.95** | 11.97 | 10.67 |
| lzma | **8.44** | 3.12 | 12.78 | **7.27** |
| BB-ELBO | 10.66 | 5.87 | 12.53 | 11.43 |
| BB-IS (4) | 10.66 | 4.86 | 12.03 | 11.38 |
| BB-SMC (4) | 9.58 | 4.76 | **10.92** | 11.20 |

**Discussion of Initial Bit Cost** The initial bit cost of compressing sequential data using (Monte Carlo) bits-back algorithms scales linearly with both the sequence length and the number of particles, i.e., $\mathcal{O}(NT)$. The original four polyphonic music datasets have a special characteristic that the average and maximum sequence lengths are large but the number of sequences is small, which means that the initial bit cost is huge but cannot be sufficiently amortized. Thus, if we compress the original datasets without chunking sequences, the *total* bitrate will be much larger than the *net* bitrate. For example, there are only 124 sequences in the Musedata test set but the average length is 519 and the maximum length is 4273. As a result, the total bitrates of the BB-ELBO coder and the BB-SMC coders for compressing the original dataset are 136.81 and 544.40 bits/timestep respectively and much larger than their net ones. Therefore, we chose to chunk long sequences to short ones of a predefined maximum length (100) and compress them independently, which could effectively decrease and amortize the initial bit cost. When compressing the chunked dataset, the total bitrates of BB-ELBO and BB-SMC reduce to 12.83 and 21.39 bits/timestep respectively. As for the initial bit cost caused by the particles, we can also use the coupled variant of BB-SMC (aka BB-CSMC) for compressing sequential data.

### D.4 Lossy Compression on Images

Current state-of-the-art lossy image compressors use hierarchical latent VAEs with quantized latents that are losslessly compressed with hyperlatents (Ballé et al., 2016; 2018; Minnen et al., 2018). Yang et al. (2020) observed that the marginalization gap of jointly compressing the latent and the hyperlatent can be bridged by bits-back. Thus, our McBits coders can be used to further reduce the gap. We experimented on a simplified setting where we used the binarized EMNIST datasets and a modification of the VAE model with 2 stochastic layers in Burda et al. (2015). Experimental details and results are presented below.

**Lossy Compression Setup** We used the binarized EMNIST datasets to benchmark the lossy compression performance. We considered the lossy compression setup with hierarchical VAE models (Ballé et al., 2018; Minnen et al., 2018). Specifically, the compressing data $x$ is transformed by trained hierarchical inference models $f_l$ and $f_h$ with parameters $\phi_l$ and $\phi_h$ to produce discretized latent $y$ and hyperlatent $z$ as $y = \lfloor \mu_y^f \rceil$ and $z = \lfloor \mu_z^f \rceil$, where $\mu_y^f = f_l(x; \phi_l)$ and $\mu_z^f = f_l(y; \phi_h)$ are their continuous representations. In our experiments, the latent is rounded to the nearest integer and the hyperlatent is discretized by the maximum entropy discretization scheme introduced in D.2 for lossless compression. Then the latent $y$ and the hyperlatent $z$ are compressed with bits-back coding as in lossless compression. On the decoder side, both $y$ and $z$ can be losslessly recovered and the reconstructed data $\hat{x}$ is transformed from $y$ using the generative models $g_l$ and $g_h$ with parameters $\theta_l$ and $\theta_h$.

**Model** We used the VAE model with 2 stochastic layers in Burda et al. (2015) with several modifications based on Ballé et al. (2018) for adapting to the lossy compression setup. Specifically, the approximate posterior distribution of the latent was a uniform distribution centered at $\mu_y$, i.e., $q(y|x) = \mathrm{unif}(\mu_y^f - \frac{1}{2}, \mu_y^f + \frac{1}{2})$, which is a differentiable substitute for rounding during training. The conditional likelihood distribution of the latent should also support quantization which was a factorized Gaussian distribution convolved with a standard uniform $p(y|z) = \mathcal{N}(\mu_y^g, \sigma_y^{g2}) * \mathrm{unif}(-\frac{1}{2}, \frac{1}{2})$, where $(\mu_y^g, \sigma_y^g) = g_h(z; \theta_h)$. The convolved distribution agrees with the discretized distribution on all integers (Ballé et al., 2016; 2018). This is important because the discretization of the latent would affect the compression rate which should be taken into account during training. In contrast, the disretization of the hyperlatent does not affect the compression rate as a result of getting bits back (see the discussion in D.2), we kept the distribution over the hyperlatent unchanged as in Burda et al. (2015). Specifically, its approximate posterior distribution $q(z|y)$ was a factorized Gaussian distribution centered at $\mu_z^f$ and its prior distribution $p(z)$ was a standard Gaussian. The conditional likelihood distribution was kept as a factorized Bernoulli for binary observations.

**Training** The loss function of training the hierarchical VAE model is a relaxed rate-distortion objective:

$$\mathcal{L}_\lambda(\theta_l, \theta_h, \phi_l, \phi_h) = \mathbb{E}_{q(y|x)q(z|y)} \left[ \underbrace{-\lambda \log(x|y)}_{\text{weighted distortion}} \underbrace{-\log \frac{p(y,z)}{q(z|y)}}_{\text{rate as ELBO}} + \underbrace{\cancel{\log q(y|x)}}_{0} \right] \tag{60}$$

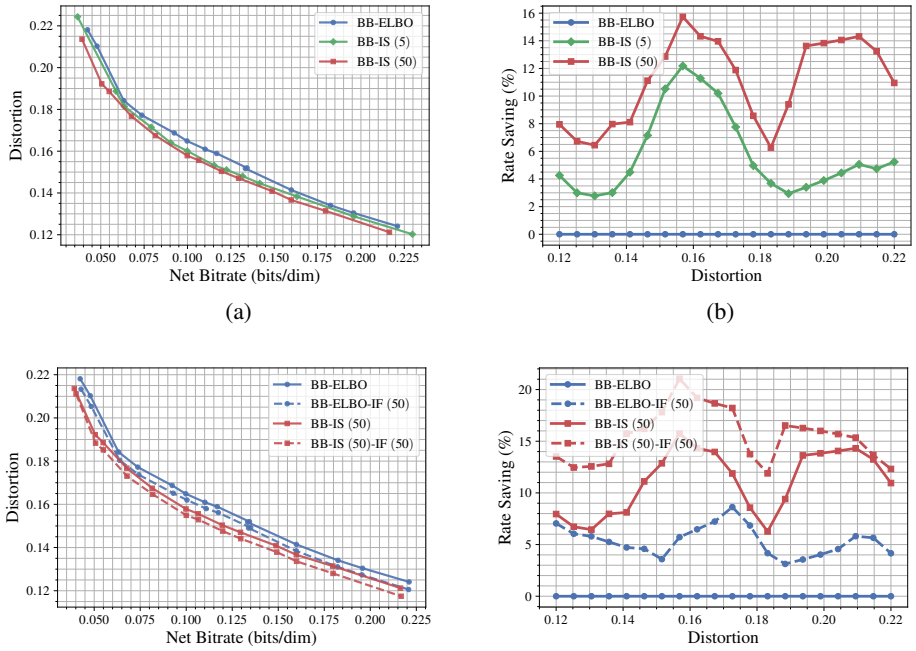
The distortion is measured as the negative log likelihood of the Bernoulli observations and the rate is measure as the negative ELBO marginalized over the hyperlatent. $\lambda$ is the hyperparameter that controls the rate-distortion trade-off. The last term is measured as 0 since $q(y|x)$ is a uniform distribution. The above rate-distortion objective is very similar to the objective function in $\beta$-VAE (Higgins et al., 2016) and can be optimized with the reparameterization trick. Note that the ELBO rate term can be changed to the IWAE objective with multiple particles.

In our experiments, we trained the models on the binarized EMNIST-MNIST dataset and evaluated on both EMNIST-MNIST and EMNIST-Letters test sets (for evaluating the performance in the transfer setting). we trained the model with the above loss function using the IWAE objective with $M \in \{1, 5, 50\}$ particles as the rate term. For each setup, we trained models with different $\lambda$ values in the range $\{1.0, 1.5, 2.0, 2.5, 3.0, 4.0, 5.0, 6.0, 7.0, 7.5, 8.0, 9.0, 10.0, 12.5, 15.0, 17.5, 20.0\}$. For training each model, we tuned the learning rate in the range $\{5 \times 10^{-3}, 2.5 \times 10^{-3}, 1 \times 10^{-3}, 7.5 \times 10^{-4}, 5 \times 10^{-4}\}$.

**Results** We evaluated *net* bitrate savings of BB-IS compared to BB-ELBO on the EMNIST-MNIST test set with different $\lambda$ values, as in Fig. 12b. We found that BB-IS achieved more than $15\%$ rate savings in some setups, see also the rate-distortion curves in Fig. 12a. We also implemented these experiments for the model in Ballé et al. (2018), but did not observe significant improvements. This may be due to the specific and complex model architecture.

We also evaluated the performance of applying amortized -iterative inference (Yang et al., 2020) in lossy compression in Fig. 12c & 12d. The main purposes were to: 1) compare the amortized-iterative inference and our BB-IS coder with similar computation budget; 2) illustrate the potential of combining amortized-iterative inference with our BB-IS coder. Specifically, we used 50 optimization steps for amortized-iterative inference to roughly match the computation budget (see discussion in

D.2) with our BB-IS coder with 50 particles (denoted as BB-IS (50)). We used a 2-stage amortized-iterative inference similar to Yang et al. (2020) and each stage contained 25 optimization steps. In the first stage, both the local variational parameters of the latent and the hyperlatent were optimized with the rate-distortion objective. In the second stage, the local variational parameters of the latent were fixed while those of the hyperlatent were optimized with the rate term (i.e., negative ELBO) of the rate-distortion objective. The learning rate was tuned in the range $\{1 \times 10^{-2}, 5 \times 10^{-3}, 1 \times 10^{-3}\}$. This method is denoted as BB-ELBO-IF (50). We observed that BB-ELBO-IF (50) improved over BB-ELBO but underperformed our BB-IS (50) in terms of rate-distortion trade-off. We also combined BB-IS (50) with 50 optimization steps of amortized-iterative inference using negative IWAE as the rate term (denoted as BB-IS (50)-IF (50)), which we found to further improve the performance and achieved more than $20\%$ rate savings in some setups.




(a) (b) (c) (d)

Figure 12: The lossy compression performance on EMNIST-MNIST test set with models trained on EMNIST-MNIST training set. (a) & (b): The rate-distortion curve and the rating saving curve for comparing BB-IS and BB-ELBO. Our BB-IS coder achieves better rate-distortion trade-off than BB-ELBO. (c) & (d): The rate-distortion curve and the rating saving curve with amortized-iterative inference. With fixed computation budget, BB-IS outperforms amortized-iterative inference and can be combined with it to further improve the performance. We measure the rate savings ($\%$) relative to BB-ELBO for fixed distortion values.

We also evaluated the lossy compression performance in a transfer setting where we used models trained on the EMNIST-MNIST to compress EMNIST-Letters test set, as in Fig. 13. We observed that the improvement of BB-IS was not as significant as on the EMNIST-MNIST test set. This might be due to that although BB-IS could improve the rate term over BB-ELBO more on the EMNIST-Letters test set (as shown in Table 1 for lossless compression), the distortion term of the model trained with IWAE might not generalize well on the dataset of a slightly different distribution.

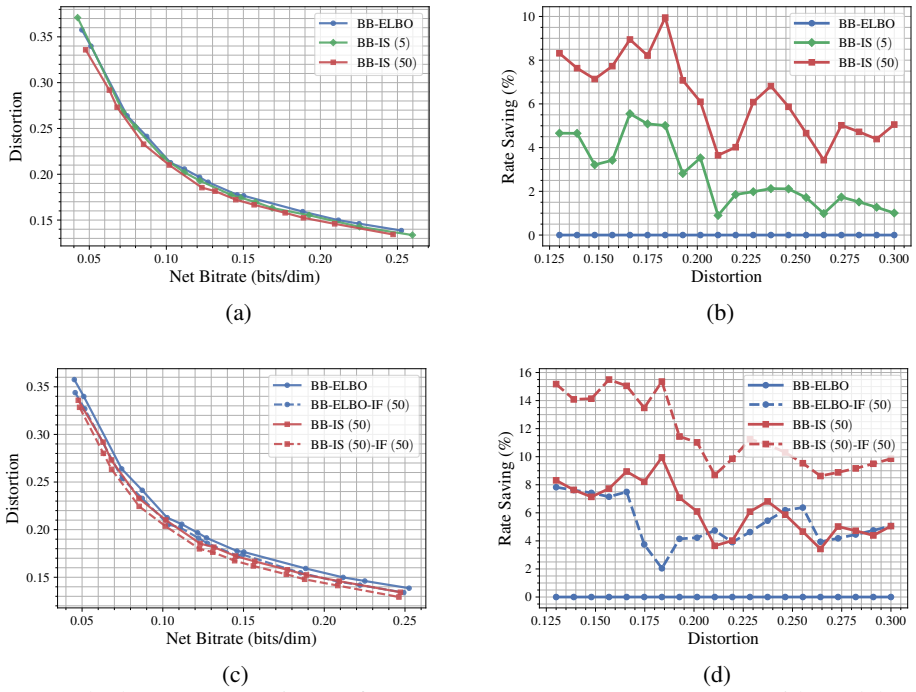



Figure 13: The lossy compression performance on EMNIST-Letters test set with models trained on EMNIST-MNIST training set. We observe similar results as Fig. 12, but the improvement of BB-IS is not as significant as Fig. 12 in this transfer setting. (a) & (b): The rate-distortion curve and the rating saving curve for comparing BB-IS and BB-ELBO. (c) & (d): The rate-distortion curve and the rating saving curve with amortized-iterative inference.