A. Monte Carlo Bits-Back Coders

A.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.

A.2. Assumptions

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

A.3. General Framework

Algorithm 2: Encode Procedure of McBits Coders

```
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 3: Decode Procedure of McBits Coders

```
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, given the extended space representation of an unbiased estimator of the marginal likelihood, we can derive the general procedure of McBits coders as Alg. 2. The decode procedure could be easily derived from the encode procedure by simply reverse the order and switch ‘encode’ with ‘decode’, as in Alg. 3. Therefore, we do not tediously present the decode procedure for each McBits coder in the following subsections.

A.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 (3). 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

$$\sum_{i=1}^{N} \frac{w_i}{N} = \sum_{i=1}^{N} \frac{1}{N} \frac{p(x, z_i)}{q(z_i \mid x)}. \tag{8}$$

Extended latent space representation The extended latent space representation of the importance sampling estimator is presented in Alg. 1, from which we can derive the proposal and target distribution as followed:

$$Q(\{z_i\}_{i=1}^N, j \mid x) = \tilde{w}_j \prod_{i=1}^{N} q(z_i \mid x) \tag{9}$$

$$P(x, \{z_i\}_{i=1}^N, j) = \frac{1}{N} p(x, z_j) \prod_{i \neq j} q(z_i \mid x). \tag{10}$$

Now note,

$$\frac{P(x, \{z_i\}_{i=1}^N, j)}{Q(\{z_i\}_{i=1}^N, j \mid x)} = \frac{1}{N} \frac{\sum_{i=1}^N w_i}{w_j} \frac{p(x, z_j)}{q(z_j \mid x)} = \sum_{i=1}^{N} \frac{w_i}{N}, \tag{11}$$

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. 4a. The expected net bit length for encoding a symbol x is:

$$-\mathbb{E}_{\{z_i\}_{i=1}^N, j}\left[\log \frac{P(x, \{z_i\}_{i=1}^N, j)}{Q(\{z_i\}_{i=1}^N, j \mid x)}\right] = -\mathbb{E}_{\{z_i\}_{i=1}^N}\left[\log\left(\sum_{i=1}^{N} \frac{w_i}{N}\right)\right]. \tag{12}$$

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.

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 $m'$
```

A.5. Bits-Back Coupled Importance Sampling

The basic idea behind Bits-Back Coupled Importance Sampling is to couple the randomness that generates the particles z_i. We assume that $q(z \mid x)$ has been discretized to precision r. In particular, we assume that for all $z \in \mathbb{S}'$

$$q(z \mid x) = \frac{q_z}{2^r}, \tag{13}$$

where q_z is an integer. Note that this assumption is also required for ANS, so this is not an additional assumption. Assume that $\mathbb{S}'$ is totally ordered (any ordering works) and $u \in \{0 \mathinner{..} 2^r - 1\}$. Define the unnormalized cumulative distribution function F_q of q as well as the following related objects:

$$F_q(z) = \sum_{z' \leq z} q_{z'} \tag{14}$$

$$F_q^{-1}(u) = \arg\min\{z \in \mathbb{S}' : F_q(z) > u\} \tag{15}$$

$$U(z) = \{u : F_q^{-1}(u) = z\}. \tag{16}$$

Notice that $|U(z)| = q_z$. Thus, if $u \sim \mathrm{unif}\{0 \mathinner{..} 2^r - 1\}$, 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 $\{0 \mathinner{..} 2^r - 1\}$ 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. 5.

Coupled importance sampling (CIS) samples the latent variables by coupling their underlying uniforms. Let $T_i : \{0 \mathinner{..} 2^r - 1\} \to \{0 \mathinner{..} 2^r - 1\}$ be bijective functions. If $u \sim \mathrm{unif}\{0 \mathinner{..} 2^r - 1\}$, then

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

Thus, if $u \sim \mathrm{unif}\{0 \mathinner{..} 2^r - 1\}$, 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:

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

Algorithm 5: Extended Latent Space Representation of Coupled Importance Sampling

```
Process $Q(\mathcal{Z} \mid x)$
    sample $u_1 \sim \mathrm{unif}\{0, \dots, 2^r - 1\}$
    for $i = 1, \dots, N$ 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 $j \sim \mathrm{Cat}(\tilde{w}_i)$
    return $\{z_i\}_{i=1}^N, \{u_i\}_{i=1}^N, j$

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

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 $u_1 \sim \mathrm{unif}\{0 \mathinner{..} 2^{-}1\}$. 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

$$\sum_{i=1}^N \frac{w_i}{N} = \sum_{i=1}^N \frac{1}{N} \frac{p(x, F_q^{-1}(T_i(u_1)))}{q(F_q^{-1}(T_i(u_1)) \mid x)}. \tag{17}$$

Extended latent space representation The extended latent space representations of the coupled importance sampling estimator is presented in Alg. 5. The Q and P processes have the following probability mass functions. For convenience, let $\mathbb{U} = \{0 \mathinner{..} 2^r - 1\}$ and $z_i = F_q^{-1}(T_i(u_1))$.

$$Q(\{z_i\}_{i=1}^N, \{u_i\}_{i=1}^N, j \mid x) = \frac{\mathbb{I}(u_1 \in \mathbb{U})}{2^r} \prod_{i=2}^N \mathbb{I}(T_i(u_1) = u_i) \prod_{i=1}^N \mathbb{I}\left(F_q^{-1}(u_i) = z_i\right) \frac{w_j}{\sum_{i=1}^N w_i} \tag{18}$$

$$P(x, \{z_i\}_{i=1}^N, \{u_i\}_{i=1}^N, j) = \frac{1}{N} p(x, z_j) \frac{\mathbb{I}(u_j \in U(z_j))}{2^r q(z_j \mid x)} \prod_{i \neq j} \mathbb{I}\left(T_i\left(T_j^{-1}(u_j)\right) = u_i\right) \mathbb{I}\left(F_q^{-1}(u_i) = z_i\right), \tag{19}$$

where we used the fact that $q_{z_j} = 2^r q(z_j \mid x) = |\,U(z_j)\,|$. 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,

$$\frac{\mathbb{I}(u_1 \in \mathbb{U})}{2^r} \prod_{i=2}^N \mathbb{I}(T_i(u_1) = u_i) = \frac{\mathbb{I}(u_j \in \mathbb{U})}{2^r} \prod_{i \neq j} \mathbb{I}\left(T_i\left(T_j^{-1}(u_j)\right) = u_i\right). \tag{20}$$

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

$$\frac{\mathbb{I}(u \in \mathbb{U})}{2^r} \mathbb{I}\left(F_q^{-1}(u) = z\right) = \frac{\mathbb{I}(u \in U(z))}{2^r} = q(z \mid x) \frac{\mathbb{I}(u \in U(z))}{2^r q(z \mid x)}. \tag{21}$$

Thus,

$$Q(\{z_i\}_{i=1}^N, \{u_i\}_{i=1}^N, j \mid x) = \frac{\mathbb{I}(u_1 \in \mathbb{U})}{2^r} \prod_{i=2}^N \mathbb{I}(T_i(u_1) = u_i) \prod_{i=1}^N \mathbb{I}\left(F_q^{-1}(u_i) = z_i\right) \frac{w_j}{\sum_{i=1}^N w_i} \tag{22}$$

$$= \frac{\mathbb{I}(u_j \in \mathbb{U})}{2^r} \prod_{i \neq j} \mathbb{I}\left(T_i(T_j^{-1}(u_j)) = u_i\right) \prod_{i=1}^N \mathbb{I}\left(F_q^{-1}(u_i) = z_i\right) \frac{w_j}{\sum_{i=1}^N w_i} \tag{23}$$

$$= q(z_j \mid x) \frac{\mathbb{I}(u_j \in U(z_j))}{2^r q(z_j \mid x)} \prod_{i \neq j} \mathbb{I}\left(T_i(T_j^{-1}(u_j)) = u_i\right) \mathbb{I}\left(F_q^{-1}(u_i) = z_i\right) \frac{w_j}{\sum_{i=1}^N w_i}. \tag{24}$$

Algorithm 6: Encode Procedure of BB-CIS

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

From this it directly follows that

$$\frac{P(x, \{z_i\}_{i=1}^N, \{u_i\}_{i=1}^N, j)}{Q(\{z_i\}_{i=1}^N, \{u_i\}_{i=1}^N, j \mid x)} = \frac{1}{N} \frac{\sum_{i=1}^N w_i}{w_j} \frac{p(x, z_j)}{q(z_j \mid x)} = \sum_{i=1}^N \frac{w_i}{N} = \sum_{i=1}^N \frac{1}{N} \frac{p(x, F_q^{-1}(T_i(u_1)))}{q(F_q^{-1}(T_i(u_1)) \mid x)}, \tag{25}$$

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. 4b. The expected net bit length for encoding a symbol x is:

$$-\mathbb{E}_{u_1, j}\left[\log \frac{P(x, \{z_i\}_{i=1}^N, \{u_i\}_{i=1}^N, j)}{Q(\{z_i\}_{i=1}^N, \{u_i\}_{i=1}^N, j \mid x)}\right] = -\mathbb{E}_{u_1}\left[\log\left(\sum_{i=1}^N \frac{w_i}{N}\right)\right] = -\mathbb{E}_{u_1}\left[\log\left(\sum_{i=1}^N \frac{1}{N} \frac{p(x, F_q^{-1}(T_i(u_1)))}{q(F_q^{-1}(T_i(u_1)) \mid x)}\right)\right]. \tag{26}$$

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, $N \le 2^r$. This is because we can execute numerical integration with a budget of 2^r particles. Assuming $T_i(u) = i$ for $i \in \{0 \,..\, 2^r - 1\}$:

$$\sum_{i=1}^N \frac{1}{N} \frac{p(x, F_q^{-1}(T_i(u_1)))}{q(F_q^{-1}(T_i(u_1)) \mid x)} = \sum_{u=0}^{2^r-1} \frac{1}{2^r} \frac{p(x, F_q^{-1}(u))}{q(F_q^{-1}(u) \mid x)} = \sum_{z \in \mathbb{S}'} q(z \mid x) \frac{p(x, z)}{q(z \mid x)} = p(x). \tag{27}$$

We briefly mention two possibilities:

1. Let $T_i(u) = (u + k_i) \bmod 2^r$ where $k_i \sim \text{unif}\{0 \,..\, 2^r - 1\}$ i.i.d. generated by a pseudorandom 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 $u \sim \text{unif}\{0 \,..\, 2^r - 1\}$.
2. Inspired by the idea of numerical integration, let $T_i(u) = (u + k_i) \bmod 2^r$ where k_i is the ith 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.

A.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 $q(z \mid x)$ and the unnormalized target distribution $p(x, z)$. For AIS with N steps, for $i \in \{0, \dots, N\}$, define annealing distributions

$$\pi_i(z) \propto f_i(z) = q(z \mid x)^{1-\beta_i} p(x, z)^{\beta_i} \tag{28}$$

$$\beta_i \in [0, 1], \beta_0 = 0, \beta_i < \beta_{i+1}, \beta_N = 1. \tag{29}$$

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 $i = 1, \dots, N-1$ 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 $i = N-1, \dots, 1$ do
        sample $z_i \sim \tilde{\mathcal{T}}_i(z_i \mid z_{i+1})$
    sample $x \sim p(x \mid z_N)$
    return $x, \{z_i\}_{i=1}^N$
```

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 π_i invariant and its reverse $\tilde{\mathcal{T}}_i$:

$$\int \mathcal{T}_i(z' \mid z)\pi_i(z)\, dz = \pi_i(z') \tag{30}$$

$$\tilde{\mathcal{T}}_i(z' \mid z) = \mathcal{T}_i(z \mid z')\frac{\pi_i(z')}{\pi_i(z)} = \mathcal{T}_i(z \mid z')\frac{f_i(z')}{f_i(z)}. \tag{31}$$

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.

$$\hat{p}_N(x) = \frac{f_1(z_1)f_2(z_2)\dots f_N(z_N)}{f_0(z_1)f_1(z_2)\dots f_{N-1}(z_N)}. \tag{32}$$

The corresponding variational bound is the log importance weight:

$$-\mathbb{E}_{\{z_i\}_{i=1}^N}\left[\log \frac{f_1(z_1)f_2(z_2)\dots f_N(z_N)}{f_0(z_1)f_1(z_2)\dots f_{N-1}(z_N)}\right] \geq -\log p(x). \tag{33}$$

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:

$$\begin{aligned} Q(\{z_i\}_{i=1}^N) &= q(z_1 \mid x)\mathcal{T}_1(z_2 \mid z_1)\mathcal{T}_2(z_3 \mid z_2)\dots \mathcal{T}_{N-1}(z_N \mid z_{N-1}) & (34)\\ &= f_0(z_1)\mathcal{T}_1(z_2 \mid z_1)\mathcal{T}_2(z_3 \mid z_2)\dots \mathcal{T}_{N-1}(z_N \mid z_{N-1}) & (35)\end{aligned}$$

$$\begin{aligned} P(x, \{z_i\}_{i=1}^N) &= p(x, z_N)\tilde{\mathcal{T}}_{N-1}(z_{N-1} \mid z_N)\tilde{\mathcal{T}}_{N-2}(z_{N-2} \mid z_{N-1})\dots \tilde{\mathcal{T}}_1(z_1 \mid z_2) & (36)\\ &= f_N(z_N)\tilde{\mathcal{T}}_{N-1}(z_{N-1} \mid z_N)\tilde{\mathcal{T}}_{N-2}(z_{N-2} \mid z_{N-1})\dots \tilde{\mathcal{T}}_1(z_1 \mid z_2). & (37)\end{aligned}$$

Now note

$$\begin{aligned} \frac{P(x, \{z_i\}_{i=1}^N)}{Q(\{z_i\}_{i=1}^N)} &= \frac{f_N(z_N)\tilde{\mathcal{T}}_{N-1}(z_{N-1}|z_N)\tilde{\mathcal{T}}_{N-2}(z_{N-2}|z_{N-1})\dots \tilde{\mathcal{T}}_1(z_1|z_2)}{f_0(z_1)\mathcal{T}_1(z_2|z_1)\mathcal{T}_2(z_3|z_2)\dots \mathcal{T}_{N-1}(z_N|z_{N-1})} & (38)\\ &= \frac{f_1(z_1)f_2(z_2)\dots f_N(z_N)}{f_0(z_1)f_1(z_2)\dots f_{N-1}(z_N)}, & (39)\end{aligned}$$

which exactly gives us the AIS estimator. Note that we have used equation (31) 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. 11a. The expected net bit length for encoding a symbol x is:

$$-\mathbb{E}_{\{z_i\}_{i=1}^N}\left[\frac{\log P(x, \{z_i\}_{i=1}^N)}{\log Q(\{z_i\}_{i=1}^N \mid x)}\right] = -\mathbb{E}_{\{z_i\}_{i=1}^N}\left[\log \frac{f_1(z_1)f_2(z_2)\dots f_N(z_N)}{f_0(z_1)f_1(z_2)\dots f_{N-1}(z_N)}\right]. \tag{40}$$

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.

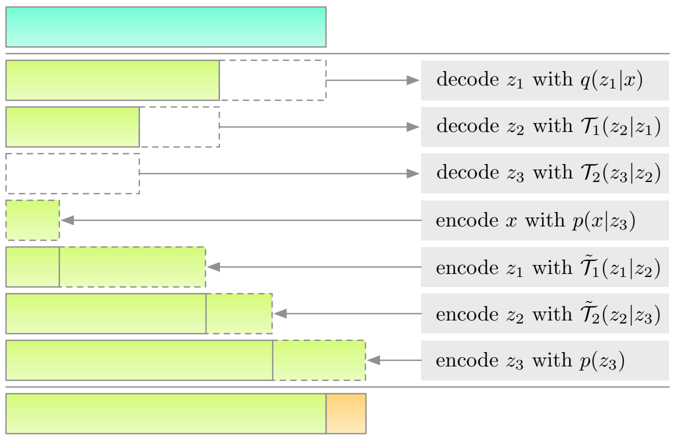
The BB-AIS coder has several practical issues. First is the computational cost. Since the intermediate distributions are usually not factorized over the latent dimensions, it is very inefficient to encode and decode the latents with entropy coders for high dimensional latents. Second is the increased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

$$-\log q(z_1 \mid x) - \sum_{i=1}^{N-1} \log \mathcal{T}_i(z_{i+1} \mid z_i), \tag{41}$$

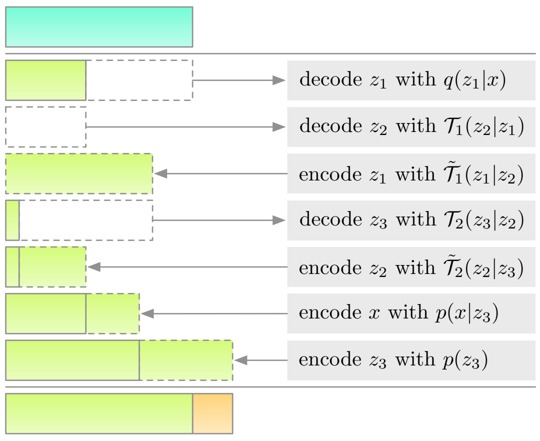
which scales like $\mathcal{O}(N)$. However, because the structure of BB-AIS is very similar to the hierarchical latent variable modelss with Markov structure in (Kingma et al., 2019), the BitSwap trick can be applied with BB-AIS to reduce the initial bit cost. The BB-AIS coder with BitSwap is derived in Alg. 9 and visualized in 11b. 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:

$$-\log q(z_1 \mid x) - \log \mathcal{T}_1(z_2 \mid z_1) + \sum_{i=1}^{N-2} \max(0, -\log \mathcal{T}_{i+1}(z_{i+2} \mid z_{i+1}) + \log \tilde{\mathcal{T}}_i(z_i \mid z_{i+1})). \tag{42}$$

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. 8b 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.


decode z_1 with $q(z_1|x)$
decode z_2 with $\mathcal{T}_1(z_2|z_1)$
decode z_3 with $\mathcal{T}_2(z_3|z_2)$
encode x with $p(x|z_3)$
encode z_1 with $\tilde{\mathcal{T}}_1(z_1|z_2)$
encode z_2 with $\tilde{\mathcal{T}}_2(z_2|z_3)$
encode z_3 with $p(z_3)$

(a) Bits-Back Annealed Importance Sampling


decode z_1 with $q(z_1|x)$
decode z_2 with $\mathcal{T}_1(z_2|z_1)$
encode z_1 with $\tilde{\mathcal{T}}_1(z_1|z_2)$
decode z_3 with $\mathcal{T}_2(z_3|z_2)$
encode z_2 with $\tilde{\mathcal{T}}_2(z_2|z_3)$
encode x with $p(x|z_3)$
encode z_3 with $p(z_3)$

(b) Bits-Back Annealed Importance Sampling with BitSwap

Figure 11. 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.

A.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 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:

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

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:

$$q(\boldsymbol{z}_T \,|\, \boldsymbol{x}_T) = \prod_{t=1}^{T} q(z_t \,|\, \boldsymbol{x}_t, \boldsymbol{z}_{t-1}). \tag{44}$$

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 ith 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 ith 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:

$$w_t^i = \frac{f(z_t^i \,|\, \boldsymbol{x}_{t-1}, \boldsymbol{\tau}_{t-1}^i) g(x_t \,|\, \boldsymbol{x}_{t-1}, (\boldsymbol{\tau}_{t-1}^i, z_t^i))}{q(z_t^i \,|\, \boldsymbol{x}_t, \boldsymbol{\tau}_{t-1}^i)} \tag{45}$$

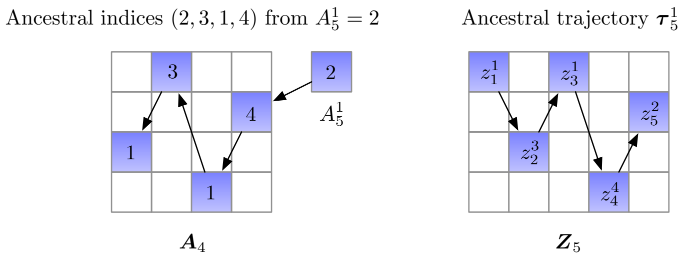
$$\tilde{w}_t^i = \frac{w_t^i}{\sum_{i=1}^N w_t^i}. \tag{46}$$

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 ith particle 'inherits' the ancestral trajectory of the A_t^ith particle. More precisely,

$$\boldsymbol{\tau}_t^i = (\boldsymbol{\tau}_{t-1}^{A_t^i}, z_t^{A_t^i}). \tag{47}$$

Algorithm 10: Tracing Back the Ancestral Trajectory $\boldsymbol{\tau}_{t-1}^i$ for the ith 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 $\boldsymbol{\tau}_{t-1}^i$
```



Ancestral indices $(2,3,1,4)$ from $A_5^1 = 2$
3
2
4
A_5^1
1
1
$\boldsymbol{A}_4$
Ancestral trajectory $\boldsymbol{\tau}_5^1$
z_1^1
z_3^1
z_5^2
z_2^3
z_4^4
$\boldsymbol{Z}_5$

Figure 12. Trace back the ancestral trajectory $\boldsymbol{\tau}_{t-1}^i$ for the particle $i = 1$ at timestep $t = 6$.

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 (47).

For brevity and consistency, we provide the pseudocode (Alg. 10) and visualization (Fig. 12) for tracing back the ancestral trajectory $\boldsymbol{\tau}_{t-1}^i$ for the ith particle at timestep t. In particular, we first obtain the ancestral index B_k^i at each previous timestep k by using the fact

$$B_k^i = A_k^{B_{k+1}^i}. \tag{48}$$

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 (48).

SMC obtains an unbiased estimate of $p(\boldsymbol{x}_T)$ using the intermediate importance weights (Del Moral, 2004, Proposition 7.4.1):

$$\hat{p}_N(\boldsymbol{x}_T) = \prod_{t=1}^{T} \left(\frac{1}{N} \sum_{i=1}^{N} w_t^i \right). \tag{49}$$

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

$$-\mathbb{E}_{\boldsymbol{Z}_T, \boldsymbol{A}_{T-1}} \left[\sum_{t=1}^{T} \log \left(\frac{1}{N} \sum_{i=1}^{N} w_t^i \right) \right] \geq -\log p(\boldsymbol{x}_T). \tag{50}$$

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 $\boldsymbol{z}_T^* = \texttt{TraceBack}(\boldsymbol{Z}_T, \boldsymbol{A}_{T-1}, j)$.

Algorithm 11: Extended Latent Space Representation of Sequential Monte Carlo

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

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

$$Q(\boldsymbol{Z}_T, \boldsymbol{A}_{T-1}, j \mid \boldsymbol{x}_T) = \tilde{w}^j_T \prod_{i=1}^{N} q(z^i_1 | x_1) \prod_{t=2}^{T} \prod_{i=1}^{N} \tilde{w}^{A^i_{t-1}}_{t-1} q(z^i_t \mid \boldsymbol{x}_t, \boldsymbol{\tau}^i_{t-1}) \tag{51}$$

$$\begin{aligned} P(\boldsymbol{x}_T, \boldsymbol{Z}_T, \boldsymbol{A}_{T-1}, j) = &\frac{1}{N^T} \mu(z^{B_1}_1) g(x_1 | z^{B_1}_1) \prod_{t=2}^{T} f(z^{B_t}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{\tau}^{B_t}_{t-1}) g(x_t \mid \boldsymbol{x}_{t-1}, (\boldsymbol{\tau}^{B_t}_{t-1}, z^{B_t}_t)) \\ &\prod_{i \neq B_1} q(z^i_1 | x_1) \prod_{t=2}^{T} \prod_{i \neq B_t} \tilde{w}^{A^i_{t-1}}_{t-1} q(z^i_t \mid \boldsymbol{x}_t, \boldsymbol{\tau}^i_{t-1}) \qquad (52) \\ = &\frac{1}{N^T} p(\boldsymbol{x}_T, \boldsymbol{z}^*_T) \prod_{i \neq B_1} q(z^i_1 | x_1) \prod_{t=2}^{T} \prod_{i \neq B_t} \tilde{w}^{A^i_{t-1}}_{t-1} q(z^i_t \mid \boldsymbol{x}_t, \boldsymbol{\tau}^i_{t-1}). \qquad (53) \end{aligned}$$

Algorithm 12: Encode Procedure of BB-SMC

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

Now note,

$$\frac{P(\boldsymbol{x}_T, \boldsymbol{Z}_T, \boldsymbol{A}_{T-1}, j)}{Q(\boldsymbol{Z}_T, \boldsymbol{A}_{T-1}, j \mid \boldsymbol{x}_T)} = \frac{1}{N^T \tilde{w}^j_T} \frac{p(\boldsymbol{x}_T, \boldsymbol{z}^*_T)}{q(z^{B_1}_1|x_1) \prod_{t=2}^T \tilde{w}^{A^{B_t}_{t-1}}_{t-1} q(z^{B_t}_t \mid \boldsymbol{x}_t, \boldsymbol{\tau}^{B_t}_{t-1})} \tag{54}$$

$$= \frac{1}{N^T \tilde{w}^j_T} \frac{p(\boldsymbol{x}_T, \boldsymbol{z}^*_T)}{q(z^{B_1}_1|x_1) \prod_{t=2}^T \tilde{w}^{B_{t-1}}_{t-1} q(z^{B_t}_t \mid \boldsymbol{x}_t, \boldsymbol{\tau}^{B_t}_{t-1})} \tag{55}$$

$$= \frac{1}{N^T \prod_{t=1}^T \tilde{w}^{B_t}_t} \frac{\mu(z^{B_1}_1) g(x_1|z^{B_1}_1) \prod_{t=2}^T f(z^{B_t}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{\tau}^{B_t}_{t-1}) g(x_t \mid \boldsymbol{x}_{t-1}, (\boldsymbol{\tau}^{B_t}_{t-1}, z^{B_t}_t))}{q(z^{B_1}_1|x_1) \prod_{t=2}^T q(z^{B_t}_t \mid \boldsymbol{x}_t, \boldsymbol{\tau}^{B_t}_{t-1})} \tag{56}$$

$$= \frac{\prod_{t=1}^T w^{B_t}_t}{N^T \prod_{t=1}^T \tilde{w}^{B_t}_t} \tag{57}$$

$$= \prod_{t=1}^T \left(\frac{1}{N} \sum_{i=1}^N w^i_t\right), \tag{58}$$

which exactly gives us the SMC estimator. Note that we have used equation (48) 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. 13. 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 order. The expected net bit length for

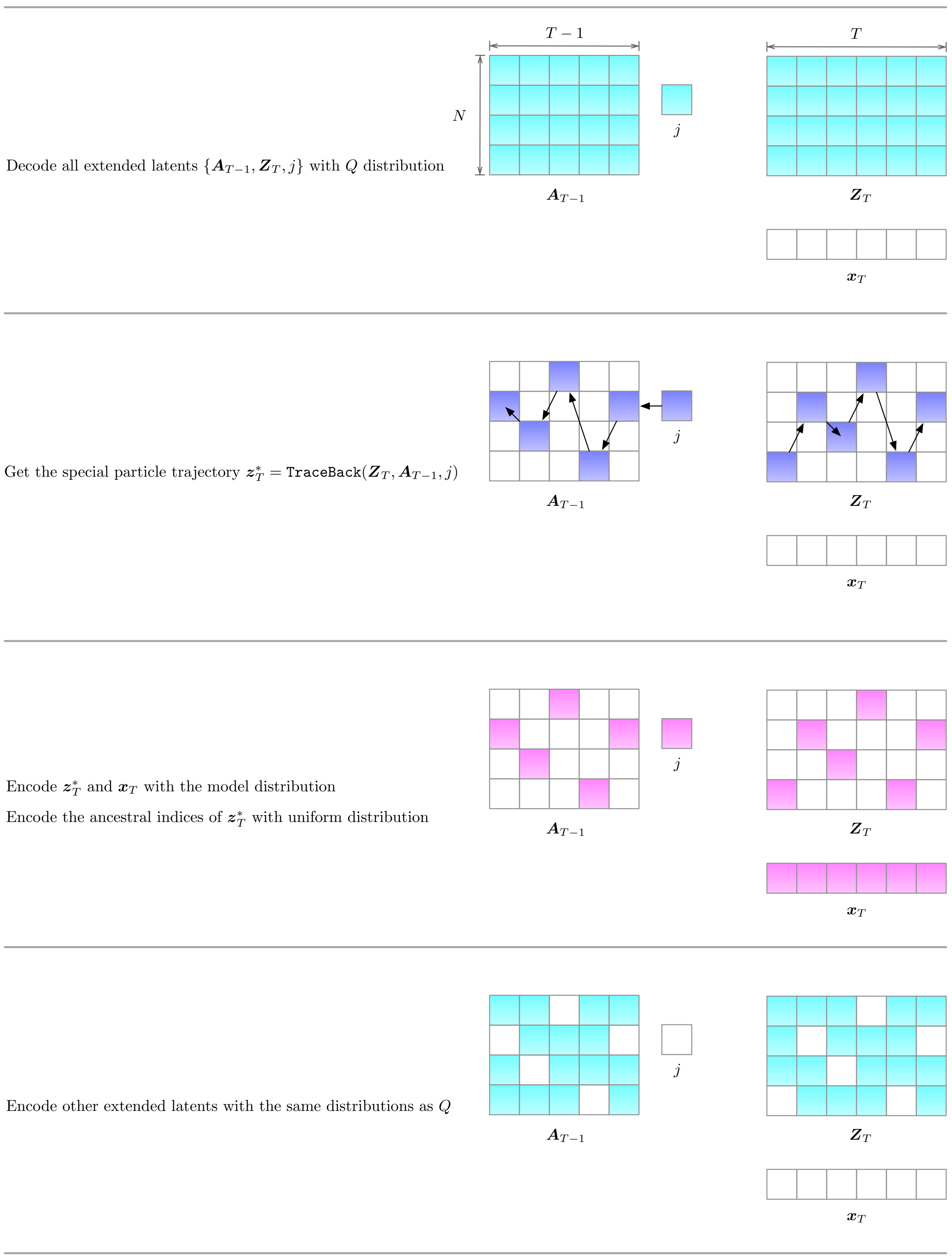

$T-1$
T
N
Decode all extended latents $\{\boldsymbol{A}_{T-1}, \boldsymbol{Z}_T, j\}$ with Q distribution
j
$\boldsymbol{A}_{T-1}$
$\boldsymbol{Z}_T$
$\boldsymbol{x}_T$
Get the special particle trajectory $\boldsymbol{z}_T^* = \texttt{TraceBack}(\boldsymbol{Z}_T, \boldsymbol{A}_{T-1}, j)$
j
$\boldsymbol{A}_{T-1}$
$\boldsymbol{Z}_T$
$\boldsymbol{x}_T$
Encode $\boldsymbol{z}_T^*$ and $\boldsymbol{x}_T$ with the model distribution
Encode the ancestral indices of $\boldsymbol{z}_T^*$ with uniform distribution
j
$\boldsymbol{A}_{T-1}$
$\boldsymbol{Z}_T$
$\boldsymbol{x}_T$
Encode other extended latents with the same distributions as Q
j
$\boldsymbol{A}_{T-1}$
$\boldsymbol{Z}_T$
$\boldsymbol{x}_T$

Figure 13. 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 $t = 1, \dots, T$ do
        if $t \neq 1$ then
            decode $v_{t-1}$ with unif$\{0, \cdots, 2^r - 1\}$
            assign $p_{t-1} = \mathrm{Cat}(\tilde{w}_{t-1})$
            for $i = 1, \dots, N$ 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 unif$\{0, \cdots, 2^r - 1\}$
        for $i = 1, \dots, N$ do
            assign $\boldsymbol{\tau}_{t-1}^i = \mathtt{TraceBack}(\boldsymbol{Z}_{t-1}, \boldsymbol{A}_{t-2}, A_{t-1}^i)$
            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 $t = T-1, \dots, 1$
    for $t = T, \dots, 1$ do
        encode $u_t^{B_t}$ with 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 unif$\{v : F_{p_{t-1}}^{-1}(v) = A_{t-1}^{B_t}\}$
        encode $B_t$ with $\mathrm{Cat}(1/N)$
    return $m'$
```

encoding a symbol $\boldsymbol{x}_T$ is:

$$-\mathbb{E}_{\boldsymbol{Z}_T, \boldsymbol{A}_{T-1}, j}\left[\frac{\log P(\boldsymbol{x}_T, \boldsymbol{Z}_T, \boldsymbol{A}_{T-1}, j)}{\log Q(\boldsymbol{Z}_T, \boldsymbol{A}_{T-1}, j \,|\, \boldsymbol{x}_T)}\right] = -\mathbb{E}_{\boldsymbol{Z}_T, \boldsymbol{A}_{T-1}}\left[\sum_{t=1}^{T} \log\left(\frac{1}{N}\sum_{i=1}^{N} w_t^i\right)\right]. \tag{59}$$

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 (4). 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:

$$-\mathbb{E}_{\{u_t\}_{t=1}^T, \{v_t\}_{t=1}^{T-1}} \left[\sum_{t=1}^{T} \log \left(\frac{1}{N} \sum_{i=1}^{N} w_t^i \right) \right] \tag{60}$$

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

B. Computational Cost

The computational requirements of our McBits coders can be distilled into two components: calculation of the distributions used (which usually involves neural network computation) and the actual encoding/decoding operations themselves. In our implementations of McBits the neural network computations tend to dominate.

For most 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-IS for example, the neural network for the approximate posterior needs to be computed once for each encoded symbol, while the neural net for the conditional likelihood is executed N times, once for each particle, in order to compute the weights for the categorical sampler.

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). We have not implemented the alias method, because the computational cost of decoding/encoding ancestral indices only accounted for a small fraction of the overall cost in our experiments.

The neural net computation and the encoding/decoding operations in BB-IS and BB-SMC can also be parallelized over particles, which in an ideal implementation would result in $\mathcal{O}(1)$ and $\mathcal{O}(T)$ time for the two methods, respectively. However, BB-AIS is not easily parallelizable due to its sequential nature. To test parallel performance, we have written an end-to-end parallelized version of BB-IS, using the JAX framework (Bradbury et al.), results are shown in Figure 7.

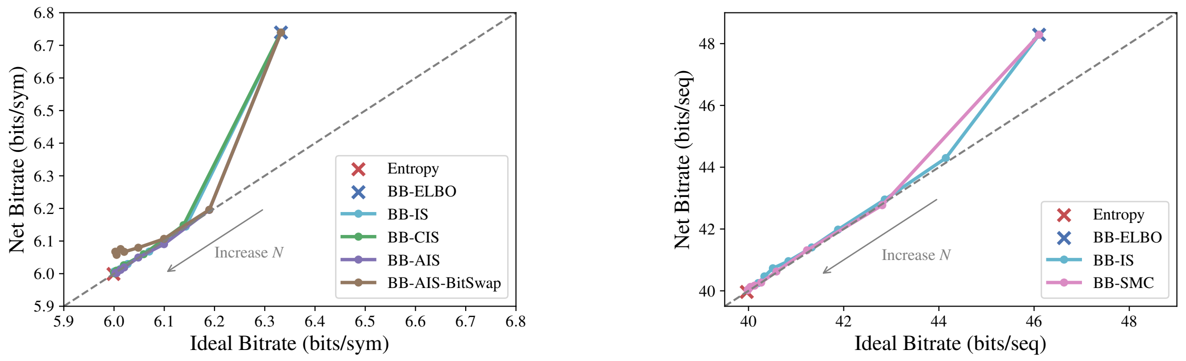
C. Experimental Details

C.1. Lossless Compression on Synthetic Data

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 include the cleanliness plots of all evaluated coders on the toy mixture and the toy HMM model in Fig. 14.



(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 14. (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. 8b. and Fig, 9, respectively.

C.2. Lossless Compression on Images

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 δ_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 include some additional results for lossless compression on images here.

Coupling does not hurt net bitrates 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 5. 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 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 5. 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**

Combine BB-IS with amortized-iterative inference The compression performance of our BB-IS coder 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 6, 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 6. 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**

Comparison with amortized-iterative inference in OOD settings We also compare the out-of-distribution (OOD) compression performance of BB-IS with amortized-iterative inference, as shown in Table 7. We observed that with roughly the same computation budget, BB-IS outperforms the amortized-iterative inference method in OOD compression settings. The most improved compression rates can be achieved when BB-IS is combined with it (denoted as BB-IS (50)-IF (50)).

Table 7. BB-IS outperforms amortized-iterative inference in OOD compression setting. The net bitrates (bits/dim) are shown in the table.

Trained on	MNIST		Letters	
Compressing	MNIST	Letters	MNIST	Letters
BB-ELBO	0.236	0.310	0.257	0.250
BB-ELBO-IF (50)	0.233	0.294	0.252	0.246
BB-IS (50)	0.228	0.280	0.244	0.239
BB-IS (50)-IF (50)	0.227	0.272	0.241	0.237

Comparison with other existing methods We include the comparison of BB-IS with existing neural lossless compression baselines on CIFAR-10, as shown in Table 8. Note that our McBits coders are not directly comparable with these methods since all of these assume different computational regimes or exploit *distinct* model classes from ours (e.g., VQ-VAE), but we still include the comparison to provide context for readers.

Table 8. The comparison of total bitrates of BB-IS with other neural compression baselines on CIFAR-10. We emphasize that these methods are not directly comparable since the model classes used are very different, and the comparison between BB-IS and BB-ELBO within the same model class (i.e., VQ-VAE) is meaningful.

Method	CIFAR-10
gzip	7.37
bzip2	6.98
lzma	6.09
PNG	5.89
JPEG	5.20
WebP	4.61
FLIF	4.37
REC (Flamich et al., 2020)	4.18
BitSwap (Kingma et al., 2019)	3.82
IDF (Hoogeboom et al., 2019)	**3.34**
BB-ELBO	4.90
BB-IS (10)	4.81

C.3. Lossless Compression on Sequential Data

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 3×10^{-5}. All models were initially evaluated on the validation set, which allowed for early stopping. In Table 9, we present our models' performance in nats and bits to allow for easy comparison of generative modelling and compression literature.

Table 9. 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	Evaluation	Unit	Notingham		JSB		Musedata		Piano-midi.de	
Objective	Metric		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

Additional Results We include some additional results for lossless compression on sequential data here.

We compared our coders with benchmark lossless compression schemes in Table 10. 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 10. The comparison of net bitrate (bits/timestep) with benchmark lossless compression schemes on sequential data benchmarks.

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.

C.4. Lossy Compression on Images

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 ϕ_l and ϕ_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 C.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 θ_l and θ_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 μ_y, i.e., $q(y|x) = \text{unif}(\mu_y^f - \frac{1}{2}, \mu_y^f + \frac{1}{2})$, 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 $p(y|z) = \mathcal{N}(\mu_y^g, {\sigma_y^g}^2) * \text{unif}(-\frac{1}{2}, \frac{1}{2})$, 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 C.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 μ_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:

$$\mathcal{L}_\lambda(\theta_l, \theta_h, \phi_l, \phi_h) = \mathbb{E}_{q(y|x)q(z|y)}\left[\underbrace{-\lambda \log(x|y)}_{\text{weighted distortion}} \underbrace{-\log\frac{p(y,z)}{q(z|y)}}_{\text{rate as ELBO}} + \underbrace{\log \cancel{q(y|x)}}_{0}\right] \tag{61}$$

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. λ 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 β-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 λ 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 $\{5 \times 10^{-3}, 2.5 \times 10^{-3}, 1 \times 10^{-3}, 7.5 \times 10^{-4}, 5 \times 10^{-4}\}$.

Additional Results We include some additional results for lossy image compression here.

We include the rate-distortion curve and the rate saving curve evaluated on the EMNIST-MNIST test set in Fig. 15a and Fig. 15b, respectively. We found that BB-IS achieved better rate-distortion trade-off than BB-IS and achieved more than 15 % rate savings in some setups.

We also evaluated the performance of applying amortized -iterative inference (Yang et al., 2020) in lossy compression in Fig. 15c & 15d. 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 C.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.

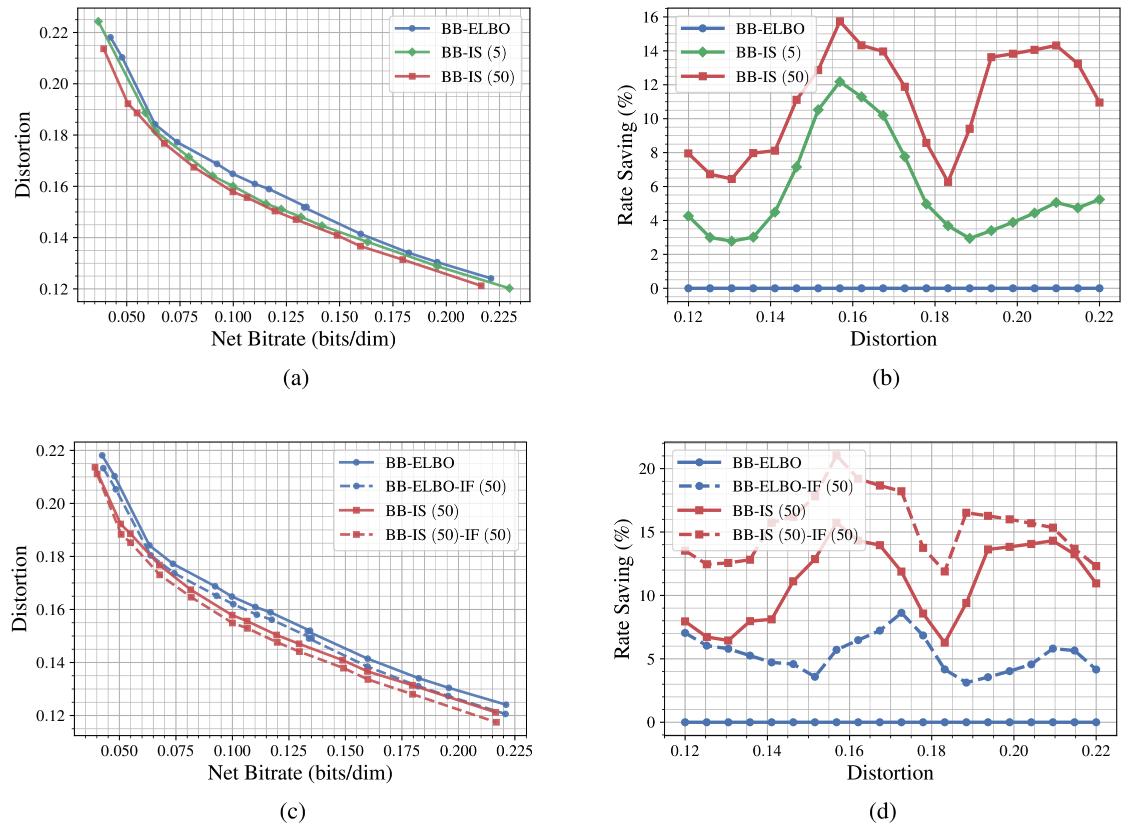
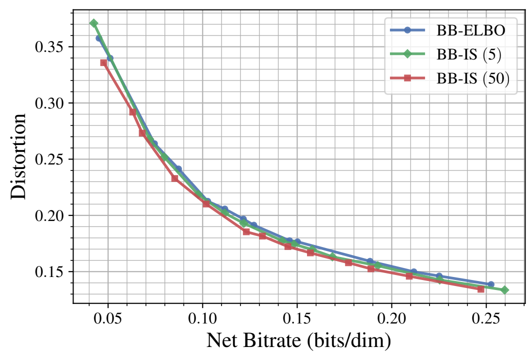
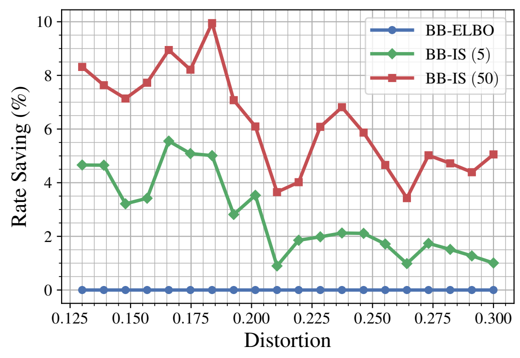


Figure 15. The lossy compression performance on EMNIST-MNIST test set with models trained on EMNIST-MNIST training set. (a) & (b): The rate-distortion curve and the rating saving curve for comparing BB-IS and BB-ELBO. Our BB-IS coder achieves better rate-distortion trade-off than BB-ELBO. (c) & (d): 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.

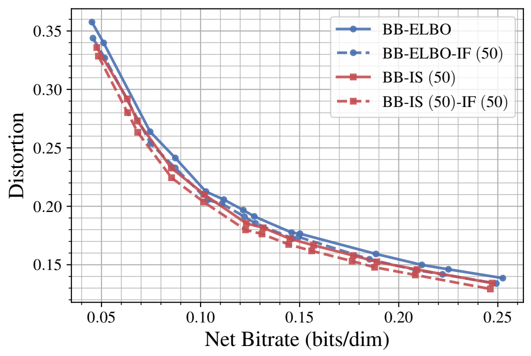
We also evaluated the lossy compression performance in a transfer setting where we used models trained on the EMNIST-MNIST to compress EMNIST-Letters test set, as in Fig. 16. We observed that the improvement of BB-IS was not as significant as on the EMNIST-MNIST test set. This might be due to that although BB-IS could improve the rate term over BB-ELBO more on the EMNIST-Letters test set (as shown in Table 2 for lossless compression), the distortion term of the model trained with IWAE might not generalize well on the dataset of a slightly different distribution.



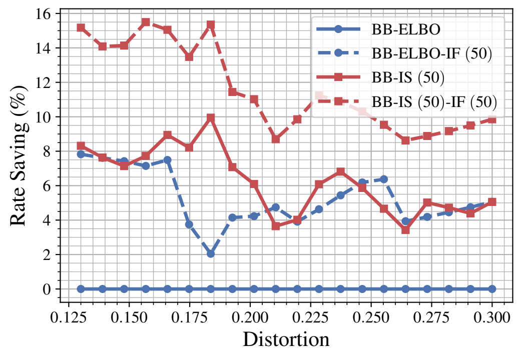
(a)



(b)



(c)



(d)

Figure 16. The lossy compression performance on EMNIST-Letters test set with models trained on EMNIST-MNIST training set. We observe similar results as Fig. 15, but the improvement of BB-IS is not as significant as Fig. 15 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.