Improving Lossless Compression Rates via Monte Carlo Bits-Back Coding

Yangjun Ruan* [1,2] Karen Ullrich* [2,3] Daniel Severo* [1,2]
James Townsend [4] Ashish Khisti [1] Arnaud Doucet [5]
Alireza Makhzani [1,2] Chris J. Maddison [1,2]

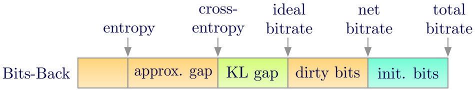
[1]University of Toronto [2]Vector Institute
[3]Facebook AI Research [4]University College London [5]University of Oxford

ICML 2021 (Long Talk)

Overview

Bits-back coding [8]...

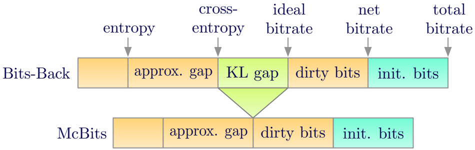
- ☺ successfully applies latent variable models to lossless compression
- ☺ achieves a bitrate equal to the negative ELBO
- ☹ suffers from a KL gap in the bitrate to the cross-entropy



Overview

We derive better bits-back schemes from tighter variational bounds..

- ✔ remove the KL gap with better bitrates
- ✔ introduce little additional cost
- ✔ better for out-of-distribution data compression



Background

Lossless Compression

Goal: find shortest binary codes for discrete i.i.d. symbols $x \sim p_d(x)$

Lossless Compression

Goal: find shortest binary codes for discrete i.i.d. symbols $x \sim p_d(x)$

- Typically a *model* distribution $p(x)$ is used to approximate $p_d(x)$

Lossless Compression

Goal: find shortest binary codes for discrete i.i.d. symbols $x \sim p_d(x)$

- Typically a *model* distribution $p(x)$ is used to approximate $p_d(x)$
- The best possible bitrate is the cross-entropy:

$$H(p_d, p) = -\sum_x p_d(x) \log p(x)$$

Lossless Compression

Goal: find shortest binary codes for discrete i.i.d. symbols $x \sim p_d(x)$

- Typically a *model* distribution $p(x)$ is used to approximate $p_d(x)$
- The best possible bitrate is the cross-entropy:

$$H(p_d, p) = -\sum_x p_d(x) \log p(x)$$

☺ Achieved by various near-optimal entropy coders

Lossless Compression

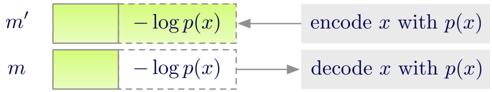
Goal: find shortest binary codes for discrete i.i.d. symbols $x \sim p_d(x)$

- Typically a *model* distribution $p(x)$ is used to approximate $p_d(x)$
- The best possible bitrate is the cross-entropy:

$$H(p_d, p) = -\sum_x p_d(x) \log p(x)$$

☺ Achieved by various near-optimal entropy coders

Key to understand: entropy coder is a store of randomness



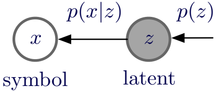
Lossless Compression with Latent Variable Models

Latent variable model is a class of highly flexible generative models

Lossless Compression with Latent Variable Models

Latent variable model is a class of highly flexible generative models

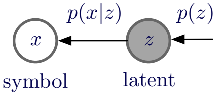
- Introduce an unobserved latent variable $z \in \mathbb{S}'$



Lossless Compression with Latent Variable Models

Latent variable model is a class of highly flexible generative models

- Introduce an unobserved latent variable $z \in \mathbb{S}'$



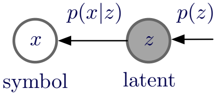
- Not directly applicable with entropy coders since evaluating $p(x)$ is often intractable:

$$p(x) = \sum_{z \in \mathbb{S}'} p(z) p(x \mid z)$$

Lossless Compression with Latent Variable Models

Latent variable model is a class of highly flexible generative models

- Introduce an unobserved latent variable $z \in \mathbb{S}'$



- Not directly applicable with entropy coders since evaluating $p(x)$ is often intractable:

$$p(x) = \sum_{z \in \mathbb{S}'} p(z) p(x \mid z)$$

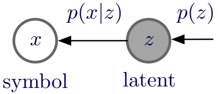
Naive approach: to encode a symbol x ...

1. pick some $z \in \mathbb{S}'$
2. encode (x, z) using $p(x, z)$

Lossless Compression with Latent Variable Models

Latent variable model is a class of highly flexible generative models

- Introduce an unobserved latent variable $z \in \mathbb{S}'$



- Not directly applicable with entropy coders since evaluating $p(x)$ is often intractable:

$$p(x) = \sum_{z \in \mathbb{S}'} p(z) p(x \mid z)$$

Naive approach: to encode a symbol x ...

1. pick some $z \in \mathbb{S}'$
2. encode (x, z) using $p(x, z)$

☹ Communicating z is redundant!

Bits-Back Compression with ANS

Bits-Back with Asymmetric Numeral Systems (BB-ANS) [11] achieves a better bitrate!

- Adopt a *stack*-like entropy coder ANS
- Compress a sequence of symbols in a chain
- Decode latents z with an *approximate posterior* $q(z \mid x)$ from the intermediate message state instead of picking z

Bits-Back Compression with ANS

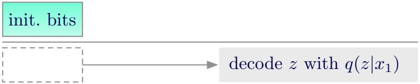
Bits-Back with Asymmetric Numeral Systems (BB-ANS) [11] achieves a better bitrate!

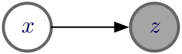
- Adopt a *stack*-like entropy coder ANS
- Compress a sequence of symbols in a chain
- Decode latents z with an *approximate posterior* $q(z \mid x)$ from the intermediate message state instead of picking z

Bits-Back Compression with ANS

Bits-Back with Asymmetric Numeral Systems (BB-ANS) [11] achieves a better bitrate!

- Adopt a *stack*-like entropy coder ANS
- Compress a sequence of symbols in a chain
- Decode latents z with an *approximate posterior* $q(z \mid x)$ from the intermediate message state instead of picking z

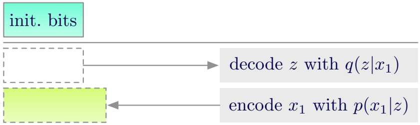


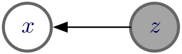


Bits-Back Compression with ANS

Bits-Back with Asymmetric Numeral Systems (BB-ANS) [11] achieves a better bitrate!

- Adopt a *stack*-like entropy coder ANS
- Compress a sequence of symbols in a chain
- Decode latents z with an *approximate posterior* $q(z \mid x)$ from the intermediate message state instead of picking z

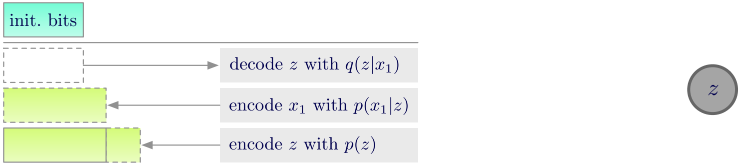




Bits-Back Compression with ANS

Bits-Back with Asymmetric Numeral Systems (BB-ANS) [11] achieves a better bitrate!

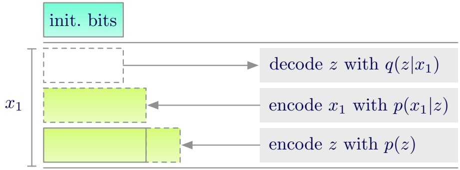
- Adopt a *stack*-like entropy coder ANS
- Compress a sequence of symbols in a chain
- Decode latents z with an *approximate posterior* $q(z \mid x)$ from the intermediate message state instead of picking z



Bits-Back Compression with ANS

Bits-Back with Asymmetric Numeral Systems (BB-ANS) [11] achieves a better bitrate!

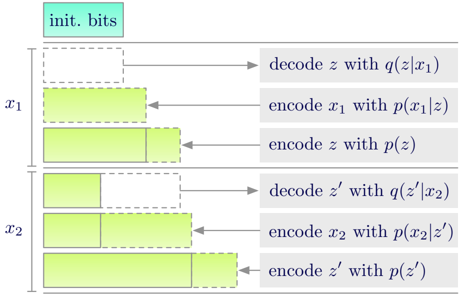
- Adopt a *stack*-like entropy coder ANS
- Compress a sequence of symbols in a chain
- Decode latents z with an *approximate posterior* $q(z \mid x)$ from the intermediate message state instead of picking z



Bits-Back Compression with ANS

Bits-Back with Asymmetric Numeral Systems (BB-ANS) [11] achieves a better bitrate!

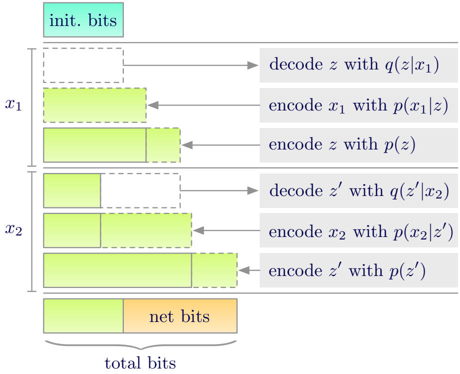
- Adopt a *stack*-like entropy coder ANS
- Compress a sequence of symbols in a chain
- Decode latents z with an *approximate posterior* $q(z \mid x)$ from the intermediate message state instead of picking z



Bits-Back Compression with ANS

Bits-Back with Asymmetric Numeral Systems (BB-ANS) [11] achieves a better bitrate!

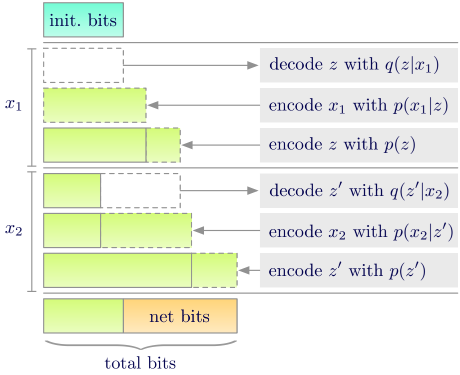
- Adopt a *stack*-like entropy coder ANS
- Compress a sequence of symbols in a chain
- Decode latents z with an *approximate posterior* $q(z|x)$ from the intermediate message state instead of picking z



Bits-Back Compression with ANS

Bits-Back with Asymmetric Numeral Systems (BB-ANS) [11] achieves a better bitrate!

- Adopt a *stack*-like entropy coder ANS
- Compress a sequence of symbols in a chain
- Decode latents z with an *approximate posterior* $q(z \mid x)$ from the intermediate message state instead of picking z



Quantities of interest

- Initial bits
- Net bitrate
- Total bitrate

Bits-Back Compression with ANS

BB-ANS...

- ☺ leads to $-\log q(z \mid x)$ *net* bitrate saving

BB-ANS...

- ☺ leads to $-\log q(z \mid x)$ *net* bitrate saving
- ☹ needs $-\log q(z \mid x)$ initial bits for the *first* symbol, causing a one-time overhead

BB-ANS...

- ☺ leads to $-\log q(z \mid x)$ *net* bitrate saving
- ☹ needs $-\log q(z \mid x)$ initial bits for the *first* symbol, causing a one-time overhead

In an ideal scenario, if we assume $z \sim q(z \mid x)$, the *net* bitrate of BB-ANS achieves the (negative) 'evidence lower bound' (ELBO):

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

We refer to vanilla BB-ANS as BB-ELBO

Motivation

Tighter variational bound is better!

☹ ELBO may be a *loose* bound on the marginal log-likelihood
$\Rightarrow$ a KL gap of BB-ELBO to the cross-entropy

Motivation

Tighter variational bound is better!

- ☹ ELBO may be a *loose* bound on the marginal log-likelihood
  $\Rightarrow$ a KL gap of BB-ELBO to the cross-entropy
- ☺ *Tighter* variational bounds bridge the gap

Motivation

Tighter variational bound is better!

- ☹ ELBO may be a *loose* bound on the marginal log-likelihood
  $\Rightarrow$ a KL gap of BB-ELBO to the cross-entropy
- ☺ *Tighter* variational bounds bridge the gap

Can we derive bits-back coders from those tighter bounds and approach the cross-entropy?

Our Method: McBits
Monte Carlo Bits-Back Coding

General Framework

McBits coders are built from elaborate Monte Carlo estimators

General Framework

McBits coders are built from elaborate Monte Carlo estimators

- Given a positive unbiased MC estimator of the marginal likelihood $\hat{p}_N(x)$ that can be simulated with $\mathcal{O}(N)$ random variables

$$\mathbb{E}[\hat{p}_N(x)] = p(x)$$

General Framework

McBits coders are built from elaborate Monte Carlo estimators

- Given a positive unbiased MC estimator of the marginal likelihood $\hat{p}_N(x)$ that can be simulated with $\mathcal{O}(N)$ random variables

$$\mathbb{E}[\hat{p}_N(x)] = p(x)$$

- A variational bound on $\log p(x)$ can be derived from $\hat{p}_N(x)$ by Jensen's inequality

$$\mathbb{E}[\log \hat{p}_N(x)] \leq \log p(x)$$

General Framework

Goal: design bits-back schemes with a net bitrate of $-\mathbb{E}[\log \hat{p}_N(x)]$

General Framework

Goal: design bits-back schemes with a net bitrate of $-\mathbb{E}[\log \hat{p}_N(x)]$

Key step: identity the extended latent space representation of $\hat{p}_N(x)$

- Extended latent variables $\mathcal{Z} \sim Q(\mathcal{Z} \mid x)$
- Proposal distribution $Q(\mathcal{Z} \mid x)$
- Target distribution $P(x, \mathcal{Z})$

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

General Framework

Goal: design bits-back schemes with a net bitrate of $-\mathbb{E}[\log \hat{p}_N(x)]$

Key step: identity the extended latent space representation of $\hat{p}_N(x)$

- Extended latent variables $\mathcal{Z} \sim Q(\mathcal{Z} \mid x)$
- Proposal distribution $Q(\mathcal{Z} \mid x)$
- Target distribution $P(x, \mathcal{Z})$

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

☺ The variational bound can be viewed as an ELBO!

$$\mathcal{Z} \leftrightarrow z$$

$$Q(\mathcal{Z} \mid x) \leftrightarrow q(z \mid x)$$

$$P(x, \mathcal{Z}) \leftrightarrow p(x, z)$$

General Framework

Derive McBits coders in a similar way to BB-ELBO

Algorithm: General Procedures of McBits Coders

Procedure *Encode(sym x, msg m)*
- decode $\mathcal{Z}$ with $Q(\mathcal{Z} \mid x)$
- encode x and $\mathcal{Z}$ with $P(x, \mathcal{Z})$
- **return** m'

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

General Framework

Derive McBits coders in a similar way to BB-ELBO

Algorithm: General Procedures of McBits Coders

Procedure *Encode(sym x, msg m)*
- decode $\mathcal{Z}$ with $Q(\mathcal{Z} \mid x)$
- encode x and $\mathcal{Z}$ with $P(x, \mathcal{Z})$
- **return** m'

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

☺ It achieves an ideal net bitrate of $-\mathbb{E}[\log \hat{p}_N(x)]$!

☺ If $-\mathbb{E}[\log \hat{p}_N(x)] \to -\log p(x)$, it approaches the cross-entropy!

Bits-Back Importance Sampling

Importance Sampling (IS)

Bits-Back Importance Sampling

Importance Sampling (IS)

- IS samples N particles $z_i \sim q(z_i \mid x)$ i.i.d. and uses the average importance weights to estimate $p(x)$

$$\hat{p}_N(x) = \frac{1}{N}\sum_{i=1}^{N}\frac{p(x, z_i)}{q(z_i \mid x)}$$

Bits-Back Importance Sampling

Importance Sampling (IS)

- IS samples N particles $z_i \sim q(z_i \mid x)$ i.i.d. and uses the average importance weights to estimate $p(x)$

$$\hat{p}_N(x) = \frac{1}{N} \sum_{i=1}^{N} \frac{p(x, z_i)}{q(z_i \mid x)}$$

- The corresponding variational bound (IWAE) [3]:

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

Bits-Back Importance Sampling

Importance Sampling (IS)

- IS samples N particles $z_i \sim q(z_i \mid x)$ i.i.d. and uses the average importance weights to estimate $p(x)$

$$\hat{p}_N(x) = \frac{1}{N} \sum_{i=1}^{N} \frac{p(x, z_i)}{q(z_i \mid x)}$$

- The corresponding variational bound (IWAE) [3]:

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

- Under mild conditions, the IWAE bound converges monotonically to $\log p(x)$ as $N \to \infty$

Bits-Back Importance Sampling

IWAE is an ELBO over an extended space! [1, 6, 7]

Bits-Back Importance Sampling

IWAE is an ELBO over an extended space! [1, 6, 7]

- The extended space variables $\mathcal{Z}$ include the configurations of the N particles $\{z_i\}_{i=1}^{N}$ and an index $j \in \{1 \mathinner{..} N\}$

Bits-Back Importance Sampling

IWAE is an ELBO over an extended space! [1, 6, 7]

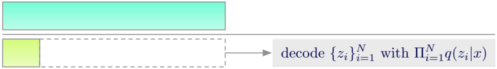
- The extended space variables $\mathcal{Z}$ include the configurations of the N particles $\{z_i\}_{i=1}^N$ and an index $j \in \{1 \,..\, N\}$
- IWAE is the ELBO between a different pair of distributions $P(x, \mathcal{Z})$ and $Q(\mathcal{Z} \mid x)$ over the extended space

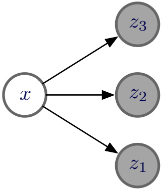
$$\frac{P(x, \mathcal{Z})}{Q(\mathcal{Z} \mid x)} = \sum_{i=1}^{N} \frac{1}{N} \frac{p(x, z_i)}{q(z_i \mid x)}$$

Bits-Back Importance Sampling

Derive BB-IS in a similar way to BB-ELBO

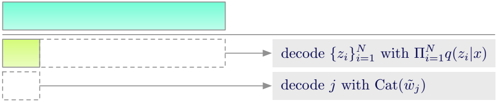
Derive BB-IS in a similar way to BB-ELBO

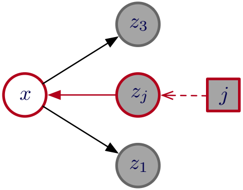




Bits-Back Importance Sampling

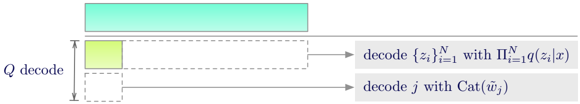
Derive BB-IS in a similar way to BB-ELBO

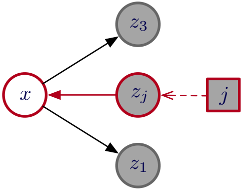




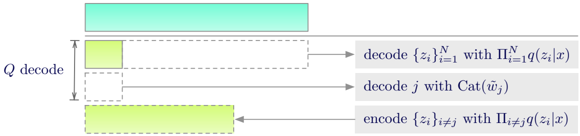
Bits-Back Importance Sampling

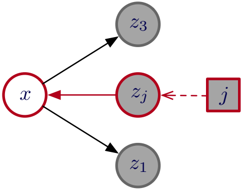
Derive BB-IS in a similar way to BB-ELBO





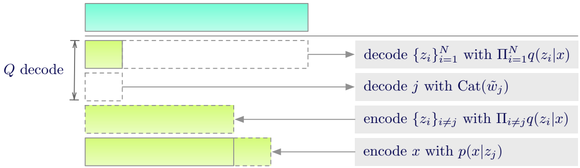
Derive BB-IS in a similar way to BB-ELBO

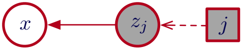




Bits-Back Importance Sampling

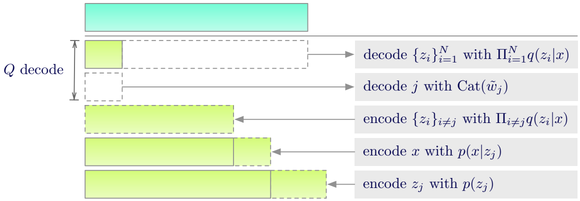
Derive BB-IS in a similar way to BB-ELBO





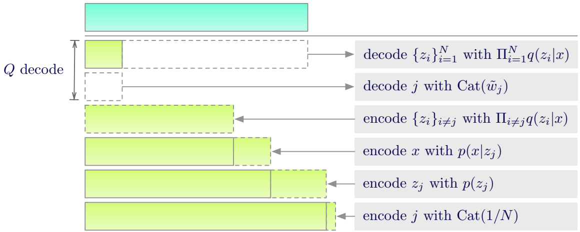
Bits-Back Importance Sampling

Derive BB-IS in a similar way to BB-ELBO

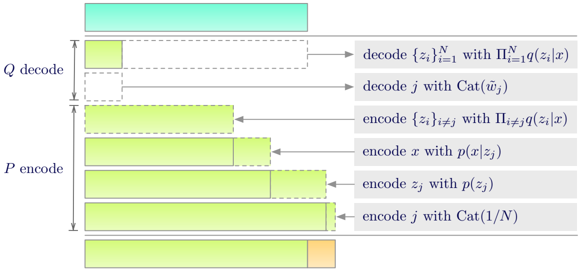


Bits-Back Importance Sampling

Derive BB-IS in a similar way to BB-ELBO



Derive BB-IS in a similar way to BB-ELBO



Bits-Back Importance Sampling

BB-IS...

- ☺ ideally achieves a net bitrate equal to the negative IWAE and asymptotically reaches the cross-entropy
- ☹ requires $\mathcal{O}(N)$ *initial bits* $\Leftarrow \mathcal{O}(N)$ decoded latent variables

Bits-Back Coupled Importance Sampling

Key idea: coupling the particles $\{z_i\}_{i=1}^N$ by a shared random number $\Rightarrow$ decoding a *single* random number is enough!

Bits-Back Coupled Importance Sampling

Key idea: coupling the particles $\{z_i\}_{i=1}^{N}$ by a shared random number $\Rightarrow$ decoding a *single* random number is enough!

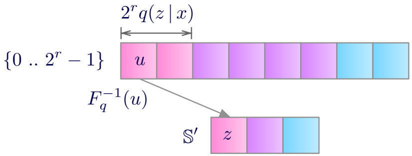
Recap (inverse CDF technique): for a *continuous* distribution, we can simulate it by applying its inverse CDF to a random uniform on $[0, 1]$

Bits-Back Coupled Importance Sampling

Key idea: coupling the particles $\{z_i\}_{i=1}^N$ by a shared random number $\Rightarrow$ decoding a *single* random number is enough!

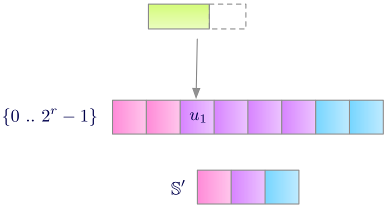
Recap (inverse CDF technique): for a *continuous* distribution, we can simulate it by applying its inverse CDF to a random uniform on $[0, 1]$

Discrete analog: suppose q is approximated to an integer precision r such that $2^r q(z \mid x)$ is an integer for all $z \in \mathbb{S}'$. The discrete analog of the inverse CDF function F_q^{-1} maps the uniform samples on $\{0 \,..\, 2^r - 1\}$ into samples from q



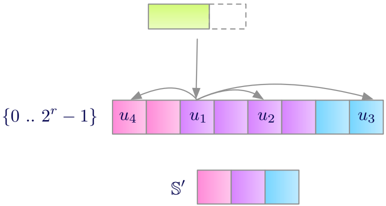
Bits-Back Coupled Importance Sampling

Reparamerization: reparamerize the particles $\{z_i\}_{i=1}^N$ by a *single* uniform random variable u_1



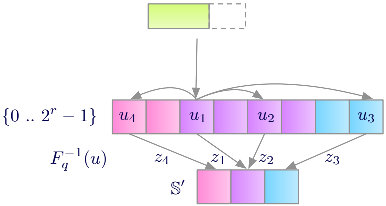
Bits-Back Coupled Importance Sampling

Reparamerization: reparamerize the particles $\{z_i\}_{i=1}^N$ by a *single* uniform random variable u_1



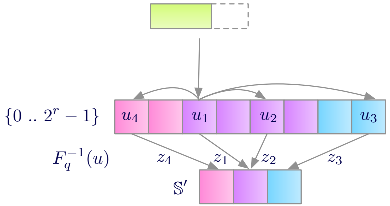
Bits-Back Coupled Importance Sampling

Reparamerization: reparamerize the particles $\{z_i\}_{i=1}^N$ by a *single* uniform random variable u_1



Bits-Back Coupled Importance Sampling

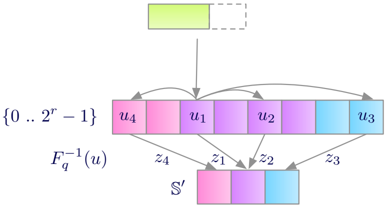
Reparamerization: reparamerize the particles $\{z_i\}_{i=1}^N$ by a *single* uniform random variable u_1



☺ Decoding a single uniform is all you need!

Bits-Back Coupled Importance Sampling

Reparamerization: reparamerize the particles $\{z_i\}_{i=1}^{N}$ by a *single* uniform random variable u_1



- ☺ Decoding a single uniform is all you need!
- ☺ The initial bit cost is reduced to

$$r - \log \tilde{w}_j \in \mathcal{O}(1) + \mathcal{O}(\log N) = \mathcal{O}(\log N)$$

In practice, the $\mathcal{O}(1)$ term dominates

Bits-Back Coupled Importance Sampling

BB-CIS...

☺ achieves a net bitrate comparable to BB-IS

$$-\mathbb{E}_{u_1}\left[\log\left(\sum_{i=1}^{N}\frac{1}{N}\frac{p(x, z_i)}{q(z_i \mid x)}\right)\right]$$

Bits-Back Coupled Importance Sampling

BB-CIS...

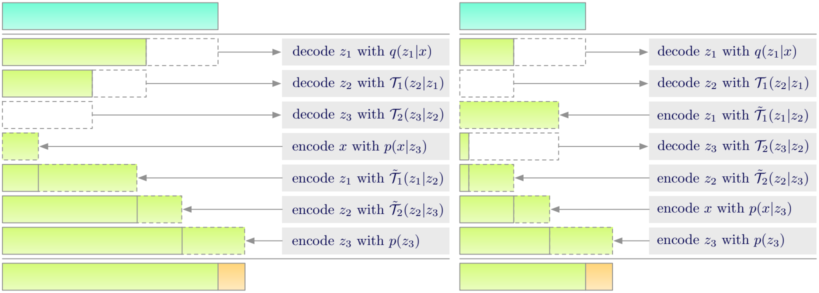
- ☺ achieves a net bitrate comparable to BB-IS

$$-\mathbb{E}_{u_1}\left[\log\left(\sum_{i=1}^{N}\frac{1}{N}\frac{p(x, z_i)}{q(z_i \mid x)}\right)\right]$$

- ☺ significantly reduces the initial bit cost of BB-IS
  $\Rightarrow$ may motivate other coupling schemes that reduce initial bits

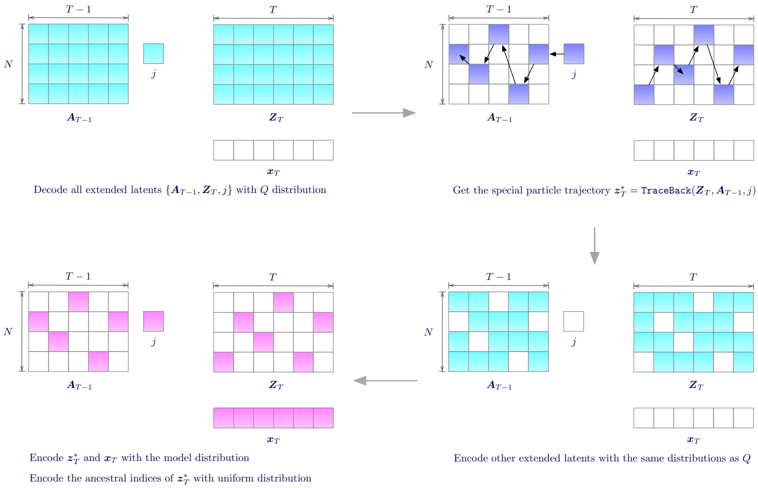
Bits-Back Annealed Importance Sampling

Bits-Back Annealed Importance Sampling (BB-AIS) and its Bit-Swap [9] variant (BB-AIS-BitSwap) for reducing initial bit cost



(a) BB-AIS **(b)** BB-AIS-BitSwap

Bits-Back Sequential Monte Carlo (BB-SMC) and its coupled variant (BB-CSMC) for reducing initial bit cost



Experiments

Computational Cost

- McBits coders scale linearly with the number of particles N

[1] Available at *https://github.com/j-towns/crayjax*

Computational Cost

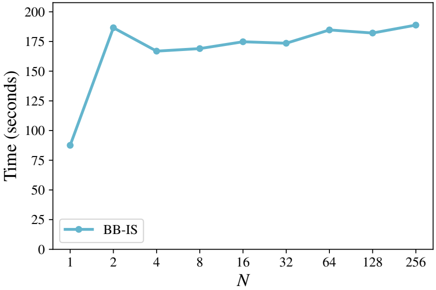
- McBits coders scale linearly with the number of particles N
- However, particles are amenable to parallelization

[1] Available at *https://github.com/j-towns/crayjax*

Computational Cost

- McBits coders scale linearly with the number of particles N
- However, particles are amenable to parallelization
- We implemented[1] vectorized rANS on the GPU, which allows McBits coders to scale sublinearly with particles

Total encode + decode times for the binarized MNIST test set



[1] Available at *https://github.com/j-towns/crayjax*

Toy Mixture Model: Net Bitrate $\rightarrow$ Entropy

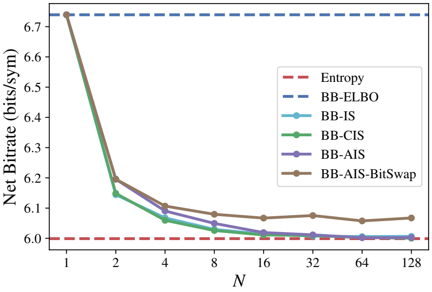
- We performed experiments on a toy mixture model, where the data generating distribution is randomly initialized and known

Toy Mixture Model: Net Bitrate $\rightarrow$ Entropy

- We performed experiments on a toy mixture model, where the data generating distribution is randomly initialized and known
- A uniform approximate posterior was used to ensure a large mismatch with the true posterior

Toy Mixture Model: Net Bitrate $\rightarrow$ Entropy

- We performed experiments on a toy mixture model, where the data generating distribution is randomly initialized and known
- A uniform approximate posterior was used to ensure a large mismatch with the true posterior
- As $N \rightarrow \infty$, the net bitrate converges to the entropy for most coders, as expected



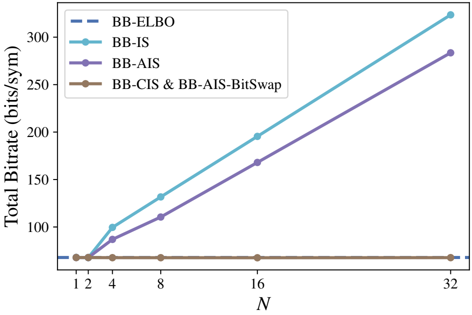
- Observation and latent alphabet sizes were 64 and 256, respectively
- Compression was performed on 5000 symbols

Toy Mixture Model: Initial Bit Cost

- We quantified the initial bit cost by computing the total bitrate (the net bitrate plus initial bits per symbol) after the first symbol

Toy Mixture Model: Initial Bit Cost

- We quantified the initial bit cost by computing the total bitrate (the net bitrate plus initial bits per symbol) after the first symbol
- The initial bit cost of naive coders scales linearly with particles, but coupled and BitSwap [9] variants significantly reduce it

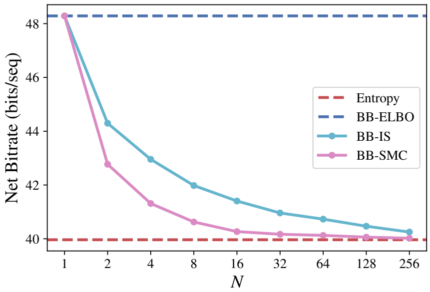


Toy Hidden Markov Model: Net Bitrate $\rightarrow$ Entropy

- We performed experiments on data from a toy Hidden Markov Model where prior, emission and transition probabilities were known, and a uniform approximate posterior was used

Toy Hidden Markov Model: Net Bitrate $\rightarrow$ Entropy

- We performed experiments on data from a toy Hidden Markov Model where prior, emission and transition probabilities were known, and a uniform approximate posterior was used
- As $N \rightarrow \infty$, the net bitrates of BB-IS and BB-SMC converge to the entropy, but BB-SMC converges much faster



- Observation and latent alphabet sizes were 16 and 32, respectively
- A uniform approximate posterior was used, and other distributions were randomly initialized
- Compression was performed on 5000 sequences with 10 time-steps each
- The entropy was estimated empirically using the forward algorithm

EMNIST: Transfer Learning Setting

- We trained a VAE, with Gaussian latents and Bernoulli observations, on the binarized EMNIST-Letters and EMNIST-MNIST datasets [5]

EMNIST: Transfer Learning Setting

- We trained a VAE, with Gaussian latents and Bernoulli observations, on the binarized EMNIST-Letters and EMNIST-MNIST datasets [5]
- Compression performance was evaluated on both tests sets

EMNIST: Transfer Learning Setting

- We trained a VAE, with Gaussian latents and Bernoulli observations, on the binarized EMNIST-Letters and EMNIST-MNIST datasets [5]
- Compression performance was evaluated on both tests sets
- BB-IS achieves greater rate savings than BB-ELBO in the out-of-distribution setting

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

Polyphonic Music Datasets: BB-SMC for Sequential Data

- We used a chunked version of 4 polyphonic music datasets from [2] to evaluate the compression performance of BB-SMC on sequential datasets

Polyphonic Music Datasets: BB-SMC for Sequential Data

- We used a chunked version of 4 polyphonic music datasets from [2] to evaluate the compression performance of BB-SMC on sequential datasets
- Models were based on the variational RNN [4], with Gaussian latents and Bernoulli observations, and trained following [10]

Polyphonic Music Datasets: BB-SMC for Sequential Data

- We used a chunked version of 4 polyphonic music datasets from [2] to evaluate the compression performance of BB-SMC on sequential datasets
- Models were based on the variational RNN [4], with Gaussian latents and Bernoulli observations, and trained following [10]
- 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 ($N = 4$)	10.66	4.86	12.03	11.38
BB-SMC ($N = 4$)	**9.58**	**4.76**	**10.92**	**11.20**
Savings	10.1%	18.9%	12.8%	2.0%

Conclusion

- McBits are bits-back coders that exploit the extended space representations of tighter variational bounds for better bitrates

Conclusion

- McBits are bits-back coders that exploit the extended space representations of tighter variational bounds for better bitrates
- The initial bit cost of McBits coders scales linearly with particles, but can be significantly reduced by coupling the extended latents

Conclusion

- McBits are bits-back coders that exploit the extended space representations of tighter variational bounds for better bitrates
- The initial bit cost of McBits coders scales linearly with particles, but can be significantly reduced by coupling the extended latents
- When parallelizing computation over particles, McBits coders can achieve better bitrates than BB-ANS with little overhead

Conclusion

- McBits are bits-back coders that exploit the extended space representations of tighter variational bounds for better bitrates
- The initial bit cost of McBits coders scales linearly with particles, but can be significantly reduced by coupling the extended latents
- When parallelizing computation over particles, McBits coders can achieve better bitrates than BB-ANS with little overhead
- Experiments indicate that BB-IS enjoys better bitrate savings in out-of-distribution compression settings

Thank you!

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

N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent.
Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription.
arXiv preprint arXiv:1206.6392, 2012.

Y. Burda, R. Grosse, and R. Salakhutdinov.
Importance weighted autoencoders.
arXiv preprint arXiv:1509.00519, 2015.

J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio.
A recurrent latent variable model for sequential data.
Advances in Neural Information Processing Systems, 28:2980–2988, 2015.

G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik.
Emnist: Extending mnist to handwritten letters.
In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

C. Cremer, Q. Morris, and D. Duvenaud.
Reinterpreting importance-weighted autoencoders.
arXiv preprint arXiv:1704.02916, 2017.

J. Domke and D. R. Sheldon.
Importance weighting and variational inference.
In *Advances in Neural Information Processing Systems*, pages 4470–4479, 2018.

G. E. Hinton and D. Van Camp.
Keeping the neural networks simple by minimizing the description length of the weights.
In *Proceedings of the sixth annual conference on Computational Learning Theory*, pages 5–13, 1993.

F. H. Kingma, P. Abbeel, and J. Ho.
Bit-swap: Recursive bits-back coding for lossless compression with hierarchical latent variables.
arXiv preprint arXiv:1905.06845, 2019.

C. J. Maddison, J. Lawson, G. Tucker, N. Heess, M. Norouzi, A. Mnih, A. Doucet, and Y. Teh.
Filtering variational objectives.
In *Advances in Neural Information Processing Systems*, pages 6573–6583, 2017.

J. Townsend, T. Bird, and D. Barber.
Practical lossless compression with latent variables using bits back coding.
ICLR, 2019.