A Ablation Study

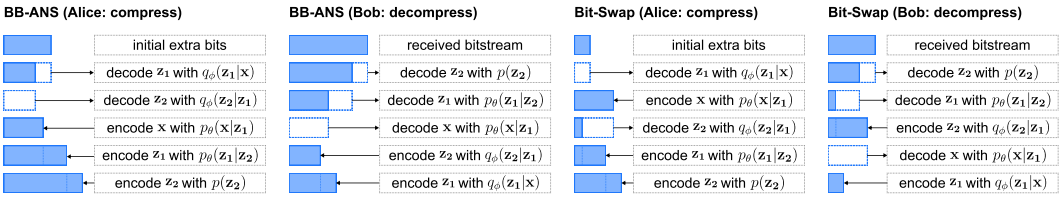

Figure 1: BB-ANS V.S. Bit-Swap

Number of Latent Variables: The difference between how BB-ANS [1] and Bit-Swap [2] compress is shown in Figure 1. We can see these two compressors are only different when there is more than one latent variable, and by utilizing encoded bitstreams from previous latent variables, Bit-Swap can reduce the cost of initial bits significantly. We carry out experiments on CIFAR-10 and keep the aggregation method constant. We can see in Table 1, Bit-Swap performs better with more latent variables. But Bit-Swap with eight latent variables is still worse than BB-ANS with two.

Aggregation Methods: We also evaluate how different aggregation methods affect the classification accuracy, shown in the right hand side of Table 1. We aggregate two images by "average", "'minimum", "maximum", "concatenation", and "greyscale+average". For "concat", the actual operation is that we compress one image after another in a way that the compressed bitstream of the first image could be used as the "extra" bits for the second image. "gs+avg" means we use greyscale of the image and then average pixel values. Obviously, "greyscale" is not a way of aggregation, but we notate it in the table this way for simplicity and comparison, and also to stress that this operation is only used during compression. That is, the generative model is trained on the original images instead of on greyscale images. By applying simple image processing methods during compression, we find that combining aggregation with image manipulation can be effective as no compressor needs to be changed, and thus no retraining for generative models is needed. We also plot their bitrate vs. classification accuracy for various aggregation methods. Figure 2 shows "concat" for images is an outlier - lower bitrate with low accuracy. We will show in Appendix B that "concat" disqualifies BB-ANS and Bit-Swap from being a *normal* compressor. Please note the special case of "concat" doesn't apply to other compressors like gzip which treat images as bytes in the first place.

Bit-Swap [2] achieves the new state of the art bitrate but why doesn't it surpass BB-ANS for classification? One of the reasons that Bit-Swap achieves a better compression rate than BB-ANS is because Bit-Swap requires fewer initial bits, details shown in Appendix F. However, initial bits can only be amortized when multiple data points need to be compressed. But in our application, at most two data points need to be compressed sequentially. This leads us to choose the net bitrate, which excludes the length of the initial bits, and makes Bit-Swap less advantegeous. Theoretically, net bitrate should be the same for both methods. However, empirically we can see in Figure 2, Bit-Swap uses slightly more net bits than BB-ANS.

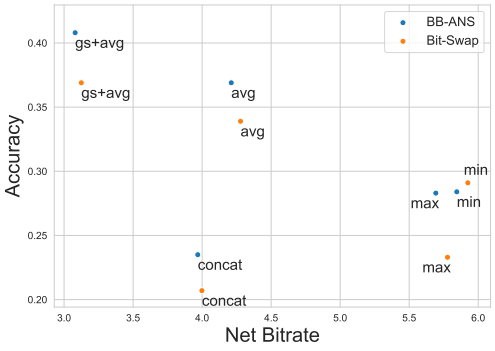

Figure 2: Bitrate versus Accuracy on CIFAR-10 with various aggregation methods.

Although we cannot draw a definite conclusion, it appears that the more a compressor can compress, the more accurate of classification NPC can obtain with the compressor. We can see that given a compressor, there is a correlation between bitrate and accuracy even with different aggregation methods, except for aggregation methods that alter a compressor to be not *normal* (e.g., "concat').

Other Alternatives: In this work, we don't go thoroughly through all the state-of-the-art compressors, but only focus on the VAE-based lossless compressors. For this specific category, there are already

# latent variables z	1	2	8
Bit-Swap	0.226	0.339	0.348
BB-ANS	0.226	0.369	0.356

aggregation	gs+avg	avg	min	max	concat
Bit-Swap	0.369	0.339	0.291	0.233	0.207
BB-ANS	0.408	0.369	0.284	0.283	0.235

Table 1: Effects of the number of latent variables and aggregation method.

numerous important factors: the choice of architectures, the choice of the number of latent variables, the choice of the hierarchy topology (e.g., asymmetrical tree structure or symmetrical one) and the choice of discretization method. Beyond this line of compressors, deep learning based compressors discussed in Section 3.3 can also be used under our framework.

Beyond compressors, aggregation methods also cause diverging differences in the final classification accuracy. We cover a few basic ones but there are other non-training-required aggregation methods like *linear blend operator* and even completely different aggregation strategies (e.g., using conditional VAE). On colored real-world images like CIFAR-10, image manipulation can be another easy and effective way to improve the accuracy. In effect, "greyscale" can be viewed as "lossy compression" and this opens up the question of how lossy compressors perform under NPC framework.

B Normal Compressor

Definition 1 (Normal Compressor). *A compressor is normal if it satisfies, up to an additive $O(\log n)$ term, where n means the maximal binary length of an element of Ω:*

1. *Idempotency: $C(xx) = C(x)$ and $C(\epsilon) = 0$ where ϵ is the empty string*
2. *Symmetry: $C(xy) = C(yx)$*
3. *Monotonicity: $C(xy) \geq C(x)$*
4. *Distributivity: $C(xy) + C(z) \leq C(xz) + C(yz)$*

Definition 2 (Metric). *A distance function $D : \Omega \times \Omega \to \mathbb{R}^+$ is a metric if it satisfies the following three criteria for any $x, y, z \in \Omega$, where Ω is a non-empty set, $\mathbb{R}^+$ represents the set of non-negative real number:*

1. *Identity: $D(x, y) = 0$ iff $x = y$*
2. *Symmetry: $D(x, y) = D(y, x)$*
3. *Triangle Inequality: $D(x, y) \leq D(x, z) + D(z, y)$*

Definition 3 (Admissible Distance). *A function $D : \Omega \times \Omega \to \mathbb{R}^+$ is an admissible distance if for every pair of objects $x, y \in \Omega$, the distance $D(x, y)$ is computable, symmetric and satisfies the density condition $\sum_y 2^{-D(x,y)} \leq 1$.*

Cilibrasi and Vitányi [3] formally prove that if the compressor is *normal*, NCD is a normalized *admissible* distance satisfying the metric inequalities, which is shown in Definition 2. Cebrián et al. [4] systematically evaluate how far real world compressors like gzip, bz2, PPMZ can satisfy the idempotency axiom. Here we empirically evalute all 4 axioms on MNIST with the BB-ANS compressor. We randomly take 100 samples and plot $C(\cdot)$ on LHS as x-axis, $C(\cdot)$ on RHS as y-axis. For simplicity, we use one latent variable, under which BB-ANS equals Bit-Swap. As shown in Figure 3, BB-ANS satisfies monotonicity, symmetry and distributivity. However, it fails on the identity axiom, with $C(xx) \approx 1.988C(x)$. Unlike gzip, bz2, or lzma, BB-ANS doesn't treat the concatenation of two images as a sequence of bytes but images, similar to PNG and WebP, making it hard to satisfy identity axiom. By default, the aggregation method refers to "concatenation". But simple concatenation does not perform well. Practically, BB-ANS fails the identity test when using "concatenation". Without changing the compressor, is it still possible to satisfy the above conditions so that NCD can be used as a normalized admissible distance metric?

We can change the aggregation method. We investigate whether BB-ANS with average can satisfy the above conditions. Obviously, identity and symmetry axioms can hold, as $C(\text{avg}(x, x)) = C(\frac{x+x}{2}) = C(x)$. We empirically evaluate monotonicity and distributivity, and find that they are both

satisfied with "average". Figure 4 illustrates that $C(\text{avg}(x,y)) \geq C(x)$ and $C(\text{avg}(x,y)) + C(z) \leq C(\text{avg}(x,z)) + C(\text{avg}(y,z))$ always hold. Given the compressor is normal under "average" function, we now prove NCD is an admissible distance metric.

Definition 4. *Let D be an admissible distance. $D^+(x)$ is defined as $D^+(x) = \max\{D(x,z) : C(z) \leq C(x)\}$, and $D^+(x,y)$ is defined as $D^+(x,y) = \max\{D^+(x), D^+(y)\}$*

Lemma 1. *If C is a normal compressor, then $E_c(x,y) + O(1)$ is an admissible distance, where $E_c(x,y) = C(xy) - \min\{C(x) - C(y)\}$ is the compression distance.*

Lemma 2. *If C is a normal compressor, then $E_c^+(x,y) = \max\{C(x), C(y)\}$*

Theorem 1. *If the compressor is normal, then the NCD is a normalized admissible distance satifying the metric (in)equalities.*

Proof. Lemma 1 and Lemma 2 show that NCD is a normalized admissible distance. We now show how NCD satisfies the metric (in)equalities.

1. For identity axiom,
$$\text{NCD}(x,x) = \frac{C(\text{avg}(x,x)) - C(x)}{C(x)} = 0. \tag{1}$$

2. For symmetry axiom,
$$\text{NCD}(x,y) = \frac{C(\text{avg}(x,y)) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}} = \text{NCD}(y,x). \tag{2}$$

3. For triangle inequality, without loss of generality, we assume $C(x) \leq C(y) \leq C(z)$. As NCD is symmetrical, there are three triangle inequalities that can be expressed by $\text{NCD}(x,y), \text{NCD}(y,z), \text{NCD}(x,z)$. For simplicity, we prove one of them, $\text{NCD}(x,y) \leq \text{NCD}(x,z) + \text{NCD}(z,y)$ as the procedure for the other two is similar. Since BB-ANS is a *normal* compressor under "avg", we have distributivity: $C(\text{avg}(x,y)) + C(z) \leq C(\text{avg}(x,z)) + C(\text{avg}(z,y))$. Subtracting $C(x)$ from both sides and rearraging results in $C(\text{avg}(x,y)) - C(x) \leq C(\text{avg}(x,z)) - C(x) + C(\text{avg}(z,y)) - C(z)$. Dividing by $C(y)$ on both sides, we have
$$\frac{C(\text{avg}(x,y)) - C(x)}{C(y)} \leq \frac{C(\text{avg}(x,z)) - C(x) + C(\text{avg}(z,y)) - C(z)}{C(y)}. \tag{3}$$
We know LHS≤ 1 and RHS can be ≤ 1 or > 1.

   (a) RHS≤ 1: Let $C(z) = C(y) + \Delta$; adding Δ to both the numerator and denominator of RHS increases RHS and makes it closer to 1.
$$\frac{C(\text{avg}(x,y)) - C(x)}{C(y)} \leq \frac{C(\text{avg}(x,z)) - C(x)}{C(y) + \Delta} + \frac{C(\text{avg}(z,y)) - C(z) + \Delta}{C(y) + \Delta} \tag{4}$$
$$= \frac{C(\text{avg}(x,z)) - C(x)}{C(z)} + \frac{C(\text{avg}(z,y)) - C(y)}{C(z)}. \tag{5}$$

   (b) RHS> 1: The procedure is similar to the case when RHS≤ 1. The difference is that adding Δ to both numerator and denominator makes RHS decrease instead of increase. But RHS cannot decrease less than 1. Thus, we still have
$$\frac{C(\text{avg}(x,y)) - C(x)}{C(y)} \leq \frac{C(\text{avg}(x,z)) - C(x)}{C(z)} + \frac{C(\text{avg}(z,y)) - C(y)}{C(z)}. \tag{6}$$

□

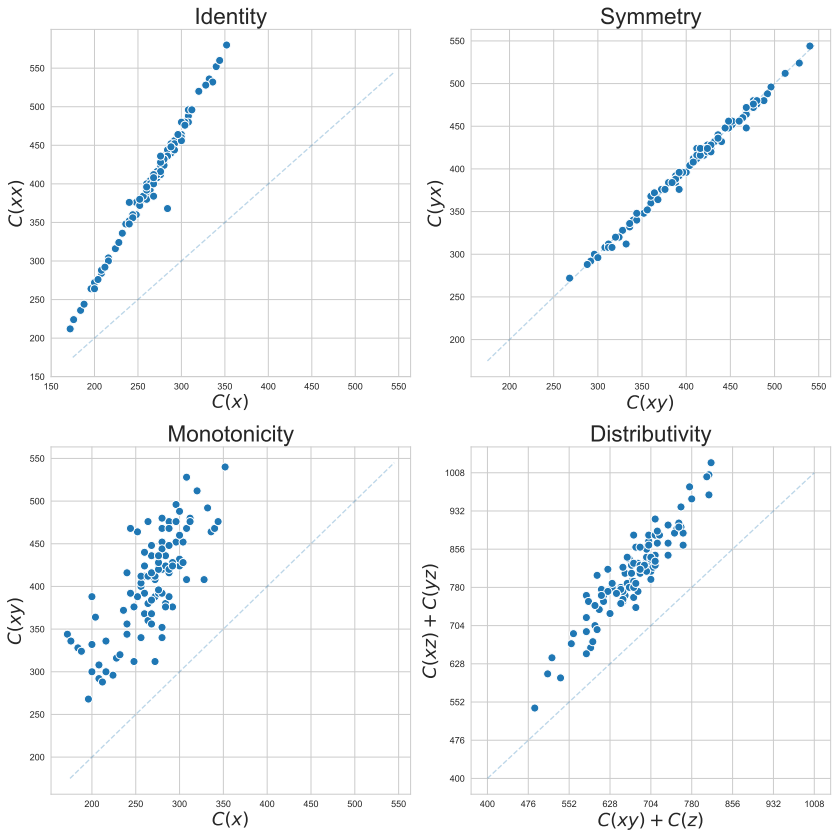

Figure 3: BB-ANS Normal Compressor Test for "concatenation"

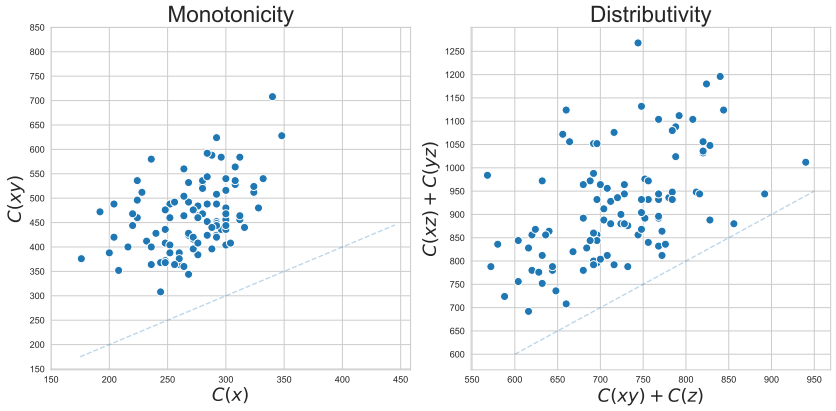

Figure 4: BB-ANS Normal Compressor Test for "average"

C Training Details

For both CNN and VGG, we tune hyperparameters on a validation set to obtain the optimized performance. The reason that we use VGG11 instead of VGG16 or VGG19 is because VGG11 performs the best in the low data regime in our experiments. For CNN, we first normalize MNIST, FashionMNIST and CIFAR-10. We use batch size $= 4$, epoch number $= 14$, Adadelta [5] as the optimizer with learning rate $= 1$, decaying learning rate by $\gamma = 0.7$ every step for MNIST and FashionMNIST. Some of hyperparameters are from PyTorch's official MNIST example. We use the same hyperparameters except we increase the number of epochs to 20. For VGG11 on

MNIST and FashionMNIST, we use epoch number $= 20$ when given 50 samples per class, and use epoch number $= 40$ when given less. We use Adam [6] with learning rate $= 0.0001$ as the optimizer, and batch size $= 4$. For CIFAR-10, we use learning rate $= 0.00001$, epoch number $= 80$.

For MeanTeacher and VAT, we follow Zhang et al. [7]'s implementation and hyperparameter settings. We use WideResNet [8] with depth $= 28$ and widen factor $= 2$ as the architecture for all three datasets and iterate 60,000 for MNIST and FashionMNIST, and iterate 200,000 for CIFAR-10. SGD with momentum is used for all three datasets, with learning rate $= 0.03$, momentum $= 0.9$.

For latent variable models used in this paper, we follow the training procedure and hyperparameters in Kingma et al. [2] — four 'Processing' Residual blocks at the beginning of the inference model and the end of the generative model; eight 'Ordinary' Residual block in total for all latent layers in both inference model and generative model. The Dropout [9] rate is 0.2 for MNIST and FashionMNIST, 0.3 for CIFAR-10. The learning rate for all datasets is 0.002 with Adam optimizer. Dimension of latent variables for MNIST and FashionMNIST is $1 \times 16 \times 16$ while dimension of latent variables for CIFAR-10 is $8 \times 16 \times 16$. During kNN, we use $k = 2$ for MNIST and FashionMNIST and $k = 3$ for CIFAR-10.

We use one NVIDIA Tesla P40 GPU for training and compression. For pairwise computation in 50-shot setting, it takes roughly ten hours to calculate distance matrix on MNIST and FashionMNIST; it takes about thirty hours for CIFAR10. But once the distance matrix is calculated, evaluation on 5-shot or 10-shot just takes seconds. For both CNN and VGG, it takes about half an hour to train on 50-shot for one experiment. For VAT and MT, we need to re-train for every shot setting. For MNIST and FashionMNIST it takes about three hours to run one experiment in a single shot setting and for CIFAR10 it takes about twelve hours. The time that VAT and MT take positively relate to the number of iterations, which makes them slower than our method in the 5- and 10-shot setting but faster in the 50-shot setting.

D Details of ANS

We briefly introduce ANS and show the proof of the optimal code length obtained from ANS. The essence of ANS is to encode one or more data points into a single natural number, called state $s \in \mathbb{N}$. Depending on different vocabularies and manipulations, there are different variations of ANS (details can be seen in Duda et al. [10]). We introduce one of them - rANS (range ANS), which is the variant we use in this paper. The notation we use here is unconventional in order to be consistent with the main part of the paper.

Let's notate our state at timestamp t as $s_t \in \mathbb{N}$, and notate our symbol/message at t as x_t, $x_t \in V$, where $V = \{0, 1\}$ is the vocabulary set. We have two simple methods to encode a binary sequence into a natural number bit by bit — $s_t = 2s_{t-1} + x_t$ or $s_t = s_{t-1} + 2^m x_t$. The former means appending information to the least significant position while the latter is adding information to the most significant position. It's obvious that encoding a new symbol into the most significant position requires remembering m while encoding in the least significant position only needs the previous state s_{t-1} and new information x_t. It's also easy to decode: depending on whether the current state s_t is even or odd, we not only know if the last encoded symbol x_t is 0 or 1, but we can also decode the state following $s_{t-1} = \frac{s_t}{2}$ or $s_{t-1} = \frac{s_t - 1}{2}$.

The above example illustrates encoding and decoding methods when there are two elements in the vocabulary with uniform distribution $p(0) = p(1) = \frac{1}{2}$. In this case, it's optimal to scale up s_t to two for both 0 or 1 as we essentially only spend 1 bit per encoded symbol. However, when the probability is not uniformly distributed, the entropy is smaller, and scaling up by 2 for both symbols will not be optimal anymore. rANS generalizes the process to any discrete probability distribution and any size of vocabulary.

Intuitively, scaling up by a smaller factor for a more probable symbol and scaling up by a larger factor for a less probable symbol will provide us with a more efficient representation. Concretely, we have a sequence of messages $\mathbf{x} = (x_1, x_2, x_3, ..., x_n)$, and a vocabulary $V = \{v_1, v_2, v_3, ..., v_k\}$, with size k, $x_i \in V$. We also have probability mass distribution of V: $P = \{p_{v_1}, p_{v_2}, p_{v_3}, ..., p_{v_k}\}$. Correspondingly, let's define frequency counts $F = \{f_{v_1}, f_{v_2}, f_{v_3}, ..., f_{v_k}\}$, $f_{v_i} = p_{v_i} \times M$ where $M = \sum_{i=1}^{k} f_{v_i}$. M can be viewed as a multiplier, demonstrating the precision of ANS, which is a predefined variable in the implementation. We can also get cumulative frequency counts from F

as follows $B = \{b_{v_1}, b_{v_2}, b_{v_3}, ..., b_{v_k}\}$ where $b_{v_i} = \sum_{j=1}^{i-1} f_{v_j}$. Now we get everything we need to define the encoding function G:

$$\begin{aligned} &s_t = G(s_{t-1}, x_t), \\ &G(s_{t-1}, x_t) = \left\lfloor \frac{s_{t-1}}{f_{x_t}} \right\rfloor \times M + b_{x_t} + s_{t-1} \bmod f_{x_t}. \end{aligned} \tag{7}$$

The procedure can be interpreted as follows: We have various M-sized blocks partitioning natural number $\mathbb{N}$. Encoding can be viewed as finding the exact location of the natural number that represents the state, by first finding the corresponding block ($\left\lfloor \frac{s_{t-1}}{f_{x_t}} \right\rfloor \times M$) followed by finding the sub-range representing that symbol within M (b_{x_t}) and finding the exact location within that sub-range ($s_{t-1} \bmod f_{x_t}$). The decoding function $H(s_t)$ is the reverse of the encoding:

$$\begin{aligned} &s_{t-1}, x_t = H(s_t), \\ &x_t = \text{argmax}\ \{b_{x_t} < (s_t \bmod M)\}, \\ &s_{t-1} = f_{x_t} \left\lfloor \frac{s_t}{M} \right\rfloor + s_t \bmod M - b_{x_t}. \end{aligned} \tag{8}$$

We first find the precise location within the sub-range using inverse function of cumulative counts (argmax $\{b_{x_t} < (s_t \bmod M)\}$). With x_t we can reverse steps in Equation (7) to get the previous state. As we can see, ANS decodes in the reverse order of encoding (i.e., last in first out), which makes it compatible with bits-back argument.

From encoding function, we know that:

$$\frac{s_t}{s_{t-1}} \approx \frac{M}{f_{x_t}} = \frac{1}{p_{x_t}}. \tag{9}$$

Encoding a sequence of symbols $\mathbf{x}$ results in:

$$s_n \approx \frac{s_0}{p_{x_1} p_{x_2} ... p_{x_n}}. \tag{10}$$

Thus, the total coding length is:

$$\log s_n \approx \log s_0 + \sum_{i=1}^{n} \log \frac{1}{p_{x_i}}, \tag{11}$$

where s_0 refers to the initial state. Dividing by n we will get the average coding length that approximates the entropy of the data.

E Discretization

ANS is defined for symbols in a finite alphabet; bits-back coding works for discrete latent variables. However, continuous latent variables have proven to be powerful in many latent variable models. In order to use those latent variable models for lossless compression, discretizing continuous variables into discrete ones is a necessary step. Townsend et al. [1] derives, based on MacKay [11], that using bits-back coding, continuous latent variables can be discretized to arbitrary precision without affecting the compression rate. Suppose a probability density function p is approximated using a number of "buckets" of equal width $\sigma\mathbf{z}$. For any given bucket j, we can know its probability mass $p(\mathbf{z}^{(j)})\sigma\mathbf{z}$ where $\mathbf{z}^{(j)}$ is some point in the bucket j. Let's notate the discrete distribution as P and Q for both prior and posterior distribution. Then for any given bucket j, $P(j) \approx p(\mathbf{z}^{(j)})\sigma\mathbf{z}$. The expected message length with a discretized latent variable is:

$$-\mathbb{E}_{Q(j|\mathbf{x})} \log \frac{p(\mathbf{x}|\mathbf{z}^{(j)}) p(\mathbf{z}^{(j)}) \sigma\mathbf{z}}{q(\mathbf{z}^{(j)}|\mathbf{x}) \sigma\mathbf{z}}. \tag{12}$$

The width of buckets $\sigma\mathbf{z}$ is cancelled. Therefore, as long as the bins for inference models match the generative models, continuous latent variables can be discretized up to an arbitrary precision.

In this paper we only consider basic discretization techniques like dividing continuous distribution into bins with *equal width* or *equal mass*. We discretize the prior (top layer) with equal mass and all subsequent latent layers with equal width. As Equation (12) shows, ideally we want the discretization to align between inference models and generative models. However, discretization of $\mathbf{z}_i \sim p_\theta(\mathbf{z}_i|\mathbf{z}_{i+1})$ relying on $\mathbf{z}_i \sim q_\phi(\mathbf{z}_i|\mathbf{z}_{i-1})$ is not possible without sampling. In the compression stage, when decoding $\mathbf{z}_i$, $\mathbf{z}_{i+1}$ is not available and so is $p_\theta(\mathbf{z}_i|\mathbf{z}_{i+1})$. In the decompression stage, similarly, $q_\phi(\mathbf{z}_i|\mathbf{z}_{i-1})$ is not available for $p_\theta(\mathbf{z}_i|\mathbf{z}_{i+1})$ to match with. Therefore, we need to sample from training dataset beforehand to get unbiased estimates of the statistics of the marginal distribution [2]. This process only needs to be done once and can be saved for the future use.

F Initial Bits of BB-ANS and Bit-Swap

The main difference between BB-ANS and Bit-Swap is that BB-ANS requires the sender *Alice* to decode $\mathbf{z}_{i+1}$ with $q_\phi(\mathbf{z}_{i+1}|\mathbf{z}_i)$ for i from 1 to $L-1$ first, and then encode $\mathbf{z}_i$ with $p_\theta(\mathbf{z}_i|\mathbf{z}_{i+1})$ for i from 1 to $L-1$. While Bit-Swap interleaves this encoding and decoding procedure and applies it recursively for latent variables, as illustrated in Figure 1. The advantage of Bit-Swap's procedure is that, after decoding $\mathbf{z}_1$, the bits encoded from x can be used in decoding $\mathbf{z}_2$; then bits encoded from $\mathbf{z}_1$ can be used for decoding $\mathbf{z}_3$. As a result, the initial bits required for Bit-Swap is much less than BB-ANS. Concretely, for BB-ANS, the minimum initial bits required:

$$-\log q_\phi(\mathbf{z}_1|\mathbf{x}) - \sum_{i=1}^{L-1} \log q_\phi(\mathbf{z}_{i+1}|\mathbf{z}_i). \tag{13}$$

For Bit-Swap, the minimum initial bits required:

$$\begin{aligned} &-\max\ (0, \log q_\phi(\mathbf{z}_1|\mathbf{x})) + \sum_{i=1}^{L-1} \max\ \left(0, \log \frac{p_\theta(\mathbf{z}_{i-1}|\mathbf{z}_i)}{q_\phi(\mathbf{z}_{i+1}|\mathbf{z}_i)}\right) \\ &\leq -\log q_\phi(\mathbf{z}_1|\mathbf{x}) - \sum_{i=1}^{L-1} \log q_\phi(\mathbf{z}_{i+1}|\mathbf{z}_i). \end{aligned} \tag{14}$$

The initial bits Bit-Swap required is less than BB-ANS, making Bit-Swap reach the optimal compression rate.

G Hierarchical Latent Variable Models

The hierarchical autoencoder in the paper uses deep latent Gaussian models (DLGM) [12] following the sampling process based on Markov chains, whose marginal distributions are:

$$\begin{aligned} p_\theta(\mathbf{x}) &= \int p_\theta(\mathbf{x}|\mathbf{z}_1) p_\theta(\mathbf{z}_1) d\mathbf{z}_1, \\ p_\theta(\mathbf{z}_1) &= \int p_\theta(\mathbf{z}_1|\mathbf{z}_2) p_\theta(\mathbf{z}_2) d\mathbf{z}_2, \\ &... \\ p_\theta(\mathbf{z}_{L-1}) &= \int p_\theta(\mathbf{z}_{L-1}|\mathbf{z}_L) p_\theta(\mathbf{z}_L) d\mathbf{z}_L. \end{aligned} \tag{15}$$

Combining the above equations, the marginal distribution of $\mathbf{x}$ is:

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z}_1) p_\theta(\mathbf{z}_1|\mathbf{z}_2) ... p_\theta(\mathbf{z}_{L-1})|\mathbf{z}_L) p_\theta(\mathbf{z}_L) d\mathbf{z}_{1:L}. \tag{16}$$

Accordingly, inference models $q_\phi(\mathbf{z}_{i+1}|\mathbf{z}_i)$ need to be defined for every latent layer. ELBO that includes multiple latent variables then becomes:

$$\mathbb{E}_{q_\phi(\cdot|\mathbf{x})}[\log p_\theta(\mathbf{x}, \mathbf{z}_{1:L}) - \log q_\phi(\mathbf{z}_{1:L}|\mathbf{x})]. \tag{17}$$

In this paper, we use Logistic distribution ($\mu = 0, \sigma = 1$) as the prior $p(\mathbf{z}_L)$, and use conditional Logistic distribution for both inference models $q_\phi(\mathbf{z}_{i+1}|\mathbf{z}_i), q_\phi(\mathbf{z}_1|\mathbf{x})$ and generative models $p_\theta(\mathbf{z}_i|\mathbf{z}_{i+1})$. These distributions are modeled by neural networks, which is stacked by Residual blocks [13] as hidden layers. More architecture details can be referred to Kingma et al. [2], where they also discusses other possible topologies regarding to hierarchical latent variable models.

H A Brief Proof of Universality of Information Distance

Information Distance $E(x, y)$ refers to the length of the shortest binary program generated by universal prefix Turing machine, that with input x computes y, and with input y computes x. It's shown that $E(x, y) = \max\{K(x|y), K(y|x)\}$. We now prove Theorem 1, based on Lemma 3 [14].

Lemma 3. *For every upper-semicomputable function $f(x, y)$, satisfying $\sum_y 2^{-f(x,y)} \leq 1$, we have $K(y|x) < f(x, y)$.*

To prove that $E(x, y)$ is a metric, we show it satisfies metric (in)equalities. We can infer the non-negativity and symmetry directly from the definition $E(x, y) = \max\{K(x|y), K(y|x)\}$. For triangle inequality, given x, y, z, without loss of generality, let $E(x, z) = K(z|x)$. By the self-limiting property, we have

$$\begin{aligned} E(x, z) = K(z|x) &< K(y, z|x) < K(y|x) + K(z|x, y) \\ &< K(y|x) + K(z|y) \leq E(x, y) + E(y, z). \end{aligned} \tag{18}$$

To prove $E(x, y)$ is *admissible*, we show it satisfies density requirement:

$$\sum_{y:y\neq x} 2^{-E(x,y)} \leq \sum_{y:y\neq x} 2^{-K(y|x)} \leq 1. \tag{19}$$

The second inequality is due to Kraft's inequality for prefix codes.
To prove the minimality, as for every admissible distance metric $D(x, y)$, it satisfies $\sum_{y:y\neq x} 2^{D(x,y)} \leq 1$. According to Lemma 3, we have $K(y|x) < D(x, y)$ and $K(x|y) < D(y, x)$.

I Potential Negative Social Impact

We propose a general framework for classification and clustering tasks with minimum assumption about the dataset. We don't foresee any negative social impact itself. However, our framework uses generative models for their explicit density estimation and the development of generative models may be used by people with ulterior motives to generate fake data.

References

[1] James Townsend, Thomas Bird, and David Barber. Practical lossless compression with latent variables using bits back coding. In *International Conference on Learning Representations*, 2019.

[2] Friso Kingma, Pieter Abbeel, and Jonathan Ho. Bit-swap: Recursive bits-back coding for lossless compression with hierarchical latent variables. In *International Conference on Machine Learning*, pages 3408–3417. PMLR, 2019.

[3] Rudi Cilibrasi and Paul MB Vitányi. Clustering by compression. *IEEE Transactions on Information theory*, 51(4):1523–1545, 2005.

[4] Manuel Cebrián, Manuel Alfonseca, and Alfonso Ortega. Common pitfalls using the normalized compression distance: What to watch out for in a compressor. *Communications in Information & Systems*, 5(4):367–384, 2005.

[5] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

[7] Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. *Advances in Neural Information Processing Systems*, 34, 2021.

[8] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[9] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[10] Jarek Duda, Khalid Tahboub, Neeraj J Gadgil, and Edward J Delp. The use of asymmetric numeral systems as an accurate replacement for huffman coding. In *2015 Picture Coding Symposium (PCS)*, pages 65–69. IEEE, 2015.

[11] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[12] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] Charles H Bennett, Péter Gács, Ming Li, Paul MB Vitányi, and Wojciech H Zurek. Information distance. *IEEE Transactions on information theory*, 44(4):1407–1423, 1998.