Generalization Gap in Amortized Inference

Anonymous Author(s)
Affiliation
Address
email

Abstract

The ability of likelihood-based probabilistic models to generalize to unseen data is central to many machine learning applications such as lossless compression. In this work, we study the generalizations of a popular class of probabilistic models - the Variational Auto-Encoder (VAE). We point out the two generalization gaps that can affect the generalization ability of VAEs and show that the over-fitting phenomenon is usually dominated by the amortized inference network. Based on this observation we propose a new training objective, inspired by the classic wake-sleep algorithm, to improve the generalizations properties of amortized inference. We also demonstrate how it can improve generalization performance in the context of image modeling and lossless compression.

1 Introduction

Probabilistic models have achieved great success in modelling real-life data. Given a set of training data that are sampled from an underlying data distribution $\mathcal{X}_{train} = \{x_1, \ldots, x_N\} \sim p_d(x)$, the goal of probabilistic modelling is to approximate $p_d(x)$ with a model $p_\theta(x)$. The parameter θ is usually learned by Maximum Likelihood Estimation (MLE): $\theta^* = \arg\max_\theta \frac{1}{N}\sum_{n=1}^{N} \log p_\theta(x_n)$. As is a common concern in supervised learning, a flexible probabilistic model may also overfit to a finite training dataset $\mathcal{X}_{train}$ degrading generalization performance. The generalization performance of probabilistic models can be measured by the test likelihood [36]: $\frac{1}{M}\sum_{n=1}^{M} \log p_\theta(x'_m)$, where $\mathcal{X}_{test} = \{x'_1, \ldots, x'_M\} \sim p_d(x)$ is the test dataset.

Although the test likelihood is a common evaluation criterion for probabilistic models [29], the factors that affect the generalization of deep generative models are less studied. We posit that this is because for common tasks, like sample generation or representation learning, good generalization in terms of the test likelihood is not a sufficient measure of performance. For example implicit models can generate sharp samples without having a likelihood function [10, 2, 35] and representations learned by latent variable models can be arbitrarily transformed without changing the likelihood [19]. However, in recent applications that use deep generative models for lossless compression [30, 31, 16], generalization in terms of the test likelihood directly indicates higher compression rate [36]. Specifically, given a probabilistic model $p_\theta(x)$, a lossless compressor can be constructed to compress a test data x' to a bit string with length approximately equal to $-\log_2 p_\theta(x')$ (see [21] for a detailed introduction). For a set of test data $\mathcal{X}_{test} = \{x'_1, \ldots, x'_M\} \sim p_d(x)$ with $M \to \infty$, when $p_\theta(x) \to p_d(x)$, the average compression length attains data entropy $-\frac{1}{M}\sum_{m=1}^{M} \log_2 p_\theta(x'_m) \to \mathrm{H}(p_d(x))$, which is *optimal* under Shannon's source coding theorem [27]. Therefore, a better test likelihood can lead to a greater saving in bits and so understanding and improving generalization of deep generative models is an important challenge.

A popular type of probabilistic model is the latent variable model $p_\theta(x) = \int p_\theta(x|z)p(z)dz$, where z is an unobserved latent variable [15, 23]. For a non-linear parameterization of $p_\theta(x|z)$, the evaluation of $\log p_\theta(x)$ is usually intractable and variational inference technique are used to form the evidence

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

lower bound (ELBO) to the log-likelihood $\log p_\theta(x)$. The non-linear latent variable model with amortized inference is also referred to as the Variational Auto-Encoder (VAE) [15, 23]. In this case, the generalization performance of a VAE not only depends on the generative model $p_\theta(x|z)$ but is also affected by the variational inference procedure. We are thus interested in studying the factors that affect the generalization of VAEs and how to improve the generalization in practice.

The contributions of our paper are summarized as follows:

- We show the generalization of VAEs is affected by both the generative model (decoder) and the amortized inference network (encoder); and the overfiting of the VAE is mainly dominated by the overfiting of the amortized inference network.
- We propose a new plug-in amortized inference scheme which can improve the generalization without changing the network structure. We further demonstrate how the proposed method can improve the compression rate in the practical lossless compression system.

In the following sections, we first introduce the VAE model and then discuss the factors that affect its generalization.

2 Generalizations of VAEs

VAE [15, 23] assumes a latent variable model $p_\theta(x) = \int p_\theta(x|z)p(z)dz$, where the integration over z is intractable for non-linear $p_\theta(x|z)$. The evidence lower bound (ELBO) can be used

$$\langle \log p_\theta(x) \rangle_{p_d(x)} \geq \langle \log p_\theta(x,z) - \log q_\phi(z|x) \rangle_{q_\phi(z|x)p_d(x)} \equiv \left\langle \mathrm{ELBO}(x,\theta,\phi) \right\rangle_{p_d(x)}, \quad (1)$$

where $q_\phi(z|x)$ is a variational posterior parameterized by a neural network with parameter ϕ. The use of an approximate posterior of the form $q_\phi(z|x)$ is called *amortized inference*. We denote the posterior family of $q_\phi(z|x)$ as $\mathcal{Q}$: $\mathcal{Q} = \{q : q_\phi(z|x)\}$ for a given x. If $\mathcal{Q}$ is flexible enough such that $p_\theta(z|x) \in \mathcal{Q}$ then at the optimum of Equation 1 $q_\phi(z|x) = p_\theta(z|x)$ for every $x \in p_d(x)$ and the inequality in Equation 1 becomes an equality. Many methods have been developed to increase the flexibility of $\mathcal{Q}$. For example, adding auxiliary variables [1, 20], or flow-based methods [5, 22].

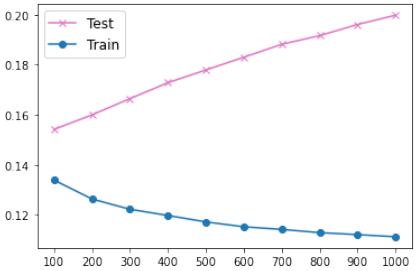


Figure 1: Overfitting in VAE. The VAE is trained for 1k epochs and we plot the training and testing BPD (y-axis) every 100 epoch. Model structures and training details can be found in Section 4.

In practice we only access a finite training dataset $\mathcal{X}_{train}$, the training objective is $\frac{1}{N}\sum_{n=1}^{N} \mathrm{ELBO}(x_n,\theta,\phi)$, which can potentially cause the VAE to over-fit to the training dataset. For example, Figure 1 shows the training and testing curve of a VAE that is trained on the Binary MNIST dataset. We plot the Bits-per-dimension (BPD), which is the negative $\log_2$ likelihood normalized by the data dimension (lower BPD indicates higher likelihood). We found the model has lower BPD (higher likelihood) on the training dataset but higher BPD on the test dataset. This indicates the model is overfitting to the training data.

Recent work [30, 31, 16] has successfully applied VAE style models to lossless compression realizing impressive performance. In this setting the average compression length on the test data set is approximately equal to $-\frac{1}{M}\sum_{m=1}^{M} \mathrm{ELBO}(x'_m,\theta,\phi)$. Hence the better the test ELBO, the better the compression performance. This motivates finding practical ways to improve the generalization of VAEs. In the following sections we discuss two factors that will potentially affect the generalizations of VAEs and discuss how to alleviate this overfitting phenomenon.

2.1 Two Factors Affect the Generalization of VAEs

To better understand what affects the generalization of VAEs we re-write the expected ELBO as

$$\left\langle \mathrm{ELBO}(x,\theta,\phi) \right\rangle_{p_d(x)} = -\mathrm{H}(p_d) - \mathrm{KL}(p_d(x)||p_\theta(x)) - \left\langle \mathrm{KL}(q_\phi(z|x)||p_\theta(z|x)) \right\rangle_{p_d(x)} \quad (2)$$

where $p_\theta(z|x) \propto p_\theta(x|z)p(z)$ is the true model posterior and the data entropy $\mathrm{H}(p_d) = -\langle \log p_d(x) \rangle_{p_d(x)}$ is a constant. Given finite training dataset $\mathcal{X}_{train}$, the training objective 2 will contain 1) a *model empirical approximation*:

$$\mathrm{KL}(p_d(x))||p_\theta(x)) \approx \frac{1}{N}\sum_{n=1}^{N} \log p_\theta(x_n) + const., \tag{3}$$

which will potentially make a flexible model $p_\theta(x)$ over-fit to the training data; and 2) an *amortized inference empirical approximation*:

$$\left\langle \mathrm{KL}(q_\phi(z|x)||p_\theta(z|x)) \right\rangle_{p_d(x)} \approx \frac{1}{N}\sum_{n=1}^{N} \mathrm{KL}(q_\phi(z|x_n)||p_\theta(z|x_n)), \tag{4}$$

where similarly a flexible $q_\phi(z|x)$ can also over-fit to the training data. More specifically, we let $\hat{\phi}$ be the optimal parameter of the empirical variational inference objective

$$\hat{\phi} = \arg\min_\phi \frac{1}{N}\sum_{n=1}^{N} \mathrm{KL}(q_\phi(z|x_n)||p_\theta(z|x_n) \tag{5}$$

and we assume for any training data point $x_n \in \mathcal{X}_{train}$

$$q_{\hat{\phi}}(z|x_n) = \arg\min_{q\in\mathcal{Q}} \mathrm{KL}(q_\phi(z|x_n)||p_\theta(z|x_n)) \equiv q^*(z|x_n),$$

where $q^*(z|x_n)$ is the realizable optimal posterior (in the $\mathcal{Q}$ family) for x_n[1]. However, for $x'_m \in \mathcal{X}_{test}$, there is no guarantee that $q_{\hat{\phi}}(z|x'_m)$ is a good approximation to the true posterior $p_\theta(z|x'_m)$. We thus refer to the difference between the ELBOs evaluated using $q_{\hat{\phi}}(z|x)$ and $q^*(z|x)$ as the *amortized inference generalization gap*, which is formally defined as

$$\left\langle \mathrm{KL}(q_{\hat{\phi}}(z|x)||p_\theta(z|x)) - \mathrm{KL}(q^*(z|x)||p_\theta(z|x)) \right\rangle_{p_d(x)}. \tag{6}$$

Equivalently, this gap can be written as the difference between two ELBOs with two different q

$$\Big\langle \underbrace{\langle \log p_\theta(x,z) - \log q^*(z|x) \rangle_{q^*(z|x)}}_{\text{ELBO with optimal inference}} - \underbrace{\langle \log p_\theta(x,z) - \log q_{\hat{\phi}}(z|x) \rangle_{q_{\hat{\phi}}(z|x)}}_{\text{ELBO with amortized inference}} \Big\rangle_{p_d(x)}. \tag{7}$$

The inference neural network introduced with amortization is the cause of this inference generalization gap. It is important to emphasize that this gap cannot be reduced by simply using a more flexible $\mathcal{Q}$. This would only make $\mathrm{KL}(q_\phi(z|x_n)||p_\theta(z|x_n))$ smaller for the training data $x_n \in \mathcal{X}_{train}$ but would not explicitly encourage better generalization performance on test data [28].

To summarize, the generalization performance of VAE models depends on two factors:

- **Generative model generalization gap:** defined as $\mathrm{KL}(p_d(x)||p_\theta(x))$ and is caused by the overfitting of the generative model (decoder).
- **Amortized inference generalization gap:** defined in Equation 7 and is caused by the overfitting of the amortized inference (encoder).

We study the contributions of both gaps to the overfitting phenomenon of VAE models.

2.2 Impact of the Generalization Gaps

The *generative model generalization gap* that is estimated by the test dataset $\mathrm{KL}(p_d(x)||p_\theta(x)) \approx \frac{1}{M}\sum_{n=1}^{M} \log p_\theta(x'_m) + const.$, cannot be calculated explicitly since $\log p_\theta(x'_m)$ is intractable in the VAE case. However, the *amortized inference generalization gap* (Equation 7) can be estimated by knowing the test dataset $\mathcal{X}_{test}$. Specifically, we approximate Equation 7 using test dataset

$$\frac{1}{M}\sum_{m=1}^{M} \langle \log p_\theta(x'_m,z) - \log q^*(z|x'_m) \rangle_{q^*(z|x)} - \langle \log p_\theta(x'_m,z) - \log q_{\hat{\phi}}(z|x'_m) \rangle_{q_{\hat{\phi}}(z|x)}, \tag{8}$$

where $q^*(z|x'_m) \equiv \arg\min_{q\in\mathcal{Q}} \mathrm{KL}(q(z|x'_m)||p_\theta(z|x'_m))$. Therefore, we can fix θ and minimize $\mathrm{KL}(q_\phi(z|x'_m)||p_\theta(z|x'_m))$ with respect to ϕ on test data x'_m, which is equivalent to

$$\max_\phi \left\langle \log p_\theta(x'_m,z) - \log q_\phi(z|x'_m) \right\rangle_{q_\phi(z|x'_m)}. \tag{9}$$

Updating ϕ using this objective would result in an optimal inference strategy. We can use this strategy to eliminate the effect of the inference generalization gap, allowing us to isolate the degree to which both the generative model and amortized inference generalization gap are contributing to overfitting.

[1] For a powerful inference network we assume that there is no amortization gap [6], which means $q_{\hat{\phi}}(z|x)$ can provide the optimal $q^*(z|x_n)$ for any training data $x_n \in \mathcal{X}_{train}$, see Section 6 for further discussion.

To visualize the amortized generalization gap, we take the VAE that is described in Figure 2 and only train the $q_\phi(z|x)$ for 100 additional epochs (with Adam [14] and lr $= 5 \times 10^{-4}$) on the test data using Equation 9 to obtain test BPD for the optimal inference. In Figure 2 we plot the test BPD using the optimal inference and classic amortized inference. The difference between two inference curves is the *amortized inference generalization gap*. We can see that after eliminating the inference generalization gap, the test BPD is stable during training, which indicates the overfitting phenomenon of this VAE is dominated by the overfitting of the amortized inference network. Although the optimal inference strategy can help eliminate the inference generalization gap, training q_ϕ at test time is not possible in most applications of interest. Therefore, we focus on how to improve the generalization of amortized inference at training time. In the next section, we introduce a consistency criterion that allows us to further understand the impact of amortized inference on generalization.

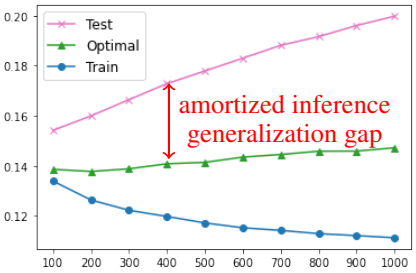


Figure 2: Visualization of amortized inference generalization gap in a VAE that is trained on MNIST.

3 Improving Generalization with Consistent Amortized Inference

We now propose an *inference consistency requirement*, which if satisfied, would result in optimal generalization for amortized variational inference, i.e. when $p_\theta \to p_d$, the amortized posterior should converge to the true posterior $q_\phi(z|x) \to p_\theta(z|x)$[2] for every $x \sim p_d(x)$. Although this requirement seems natural for variational inference, the classic amortized inference training that is widely used for VAEs [15] doesn't satisfy it. Recall the typical VAE empirical ELBO training objective

$$\tfrac{1}{N}\textstyle\sum_{n=1}^{N} \log p_\theta(x_n) - \mathrm{KL}(q_\phi(z|x_n)||p_\theta(z|x_n)). \tag{10}$$

When the model converges to the true distribution $p_{\theta^*} = p_d$ the training criterion for $q_\phi(z|x)$

$$\min_\phi -\tfrac{1}{N}\textstyle\sum_{n=1}^{N} \mathrm{KL}(q_\phi(z|x_n)||p_{\theta^*}(z|x_n)) \tag{11}$$

can still result in the amortized posterior $q_\phi(z|x)$ over-fitting to the training data. In principle, one could also limit the network capacity and/or add an explicit regularizer to the parameters [26] in an attempt to improve the generalization. However, this still cannot satisfy the consistency requirement in principle since it doesn't change the underlying training objective. Alternatively, there is another classic variational inference method, the wake-sleep training algorithm [8, 12], which satisfies the proposed consistency requirement, that we now discuss.

3.1 Wake-Sleep Training

Defining $q_\phi(x, z) = q_\phi(z|x)p_d(x)$ and $p_\theta(x, z) = p_\theta(x|z)p(z)$, the two phases of the wake-sleep training [8, 12] can be written as minimizing two different KL divergences in both x and z space.

Wake phase model learning: $p_\theta(x|z)$ is trained by minimizing the KL divergence

$$\min_\theta \mathrm{KL}(q_\phi(x, z)||p_\theta(x, z)) = \max_\theta \left\langle \mathrm{ELBO}(x, \theta, \phi) \right\rangle_{p_d(x)} + const., \tag{12}$$

where $\langle\cdot\rangle_{p_d(x)}$ is approximated using the training set. This is referred to as the *wake phase* since the model is trained on experience from the 'real environment', i.e. it uses true data samples from $p_d(x)$.

Sleep phase amortized inference: $q_\phi(z|x)$ is trained by minimizing the KL divergence

$$\min_\phi \mathrm{KL}(p_\theta(x, z)||q_\phi(x, z)) = \min_\phi \left\langle \mathrm{KL}(p_\theta(z|x)||q_\phi(z|x)) \right\rangle_{p_\theta(x,z)} + const. \tag{13}$$

Leaving out the terms that are irrelevant to ϕ, the objective can be estimated with Mon-Carlo $\langle -\log q_\phi(z|x)\rangle_{p_\theta(x,z)} \approx \frac{1}{K}\sum_{k=1}^{K} -\log q_\phi(z_k|x_k)$, where $z_k \sim p(z)$ and $x_k \sim p_\theta(x|z_k)$. This is referred to as the *sleep phase* because the samples from the model used to train q_ϕ are interpreted as dreamed experience. In contrast, the training criterion for the typical VAE amortized inference

[2] We assume the true posterior belongs to the variational family $p_\theta(z|x) \in \mathcal{Q}$.

(Equation 4) uses the true data samples to train $q_\phi(z|x)$. So following the wake-sleep nomenclature we can refer to this as *wake phase amortized inference*.

We notice that a perfect model used in the sleep phase is equivalent to minimizing

$$\left\langle \mathrm{KL}(p_\theta(z|x)||q_\phi(z|x)) \right\rangle_{p_{\theta^*}(x)} = \left\langle \mathrm{KL}(p_\theta(z|x)||q_\phi(z|x)) \right\rangle_{p_d(x)}. \tag{14}$$

Therefore, the training of the inference network satisfies the *inference consistency requirement* since we can access infinite training data from p_d by sampling from p_{θ^*}.

However, the wake-sleep algorithm presented lacks convergence guarantees [8] and minimizing $\mathrm{KL}(p_\theta(z|x)||q_\phi(z|x))$ in the sleep phase doesn't necessarily encourage an improvement to the ELBO, which directly relates to the compression rate in the lossless compression application [30]. Therefore, in the next section, we propose a new variational inference scheme: *reverse sleep amortized inference* and demonstrate how it helps improve the generalization of the inference network in practice.

3.2 Reverse Sleep Amortized Inference

We propose to use the *reverse KL divergence* in the sleep phase. We thus fix θ and train ϕ using

$$\min_\phi \left\langle \mathrm{KL}(q_\phi(z|x)||p_\theta(z|x) \right\rangle_{p_\theta(x)} = \max_\phi \left\langle \log p_\theta(x,z) - \log q_\phi(z|x) \right\rangle_{q_\phi(z|x)p_\theta(x)}, \tag{15}$$

where the integration $\langle \cdot \rangle_{p_\theta(x)}$ is approximated by Monte-Carlo using samples from the generative model $p_\theta(x)$. This reverse KL objective encourages improvements to the ELBO. When we have a perfect model $p_{\theta^*}(x) = p_d(x)$ the reverse sleep phase is equivalent to

$$\min_\phi \left\langle \mathrm{KL}(p_{\theta^*}(z|x)||q_\phi(z|x)) \right\rangle_{p_{\theta^*}(x)} = \min_\phi \left\langle \mathrm{KL}(p_{\theta^*}(z|x)||q_\phi(z|x)) \right\rangle_{p_d(x)} \tag{16}$$

which satisfies the *inference consistency requirement*.

The consistency requirement can also be validated empirically when the perfect model is known $p_{\theta^*}(x) = p_d(x)$. This can be achieved by using a pre-trained VAE as the true data generation distribution. Therefore, we first train a VAE to fit the binary MNIST problem. The VAE has the same structure as that used in Section 2 and is trained using Adam with $lr = 1 \times 10^{-3}$ for 100 epochs. After training, we treat the pre-trained decoder $p_{\theta'}(x|z)$ as the training data generator $p_d(x) \equiv \int p_{\theta'}(x|z)p(z)dz$. We then sample 10000 data samples from p_d to form a training set $\mathcal{X}_{train}$ and 1000 samples to form a test set $\mathcal{X}_{test}$. We then train a new $q_\phi(z|x)$ with: 1. wake phase inference (VAE) 2. (forward) sleep inference and 3. reverse sleep inference. The network is trained using Adam with $\mathrm{lr} = 1 \times 10^{-3}$ for 100 epochs. Figure 3 shows the test BPD calculated after every training epoch. We can see the sleep phase out-performs the wake phase and the reverse sleep inference achieves the best BPD. Intuitively, this is because both the forward and reverse sleep inference use the true model to generate additional training data whereas the wake inference only has access to the finite training dataset X_{train}.

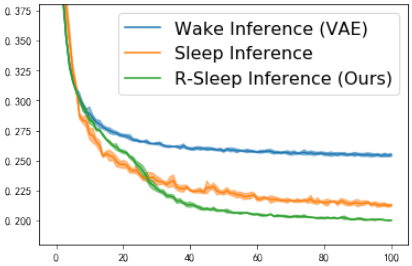


Figure 3: Amortized inference consistency. We plot the mean and std of the test BPD over three random seeds.

3.3 Reverse Half-asleep Amortized Inference with Imperfect Models

In practice our model will not be perfect $p_\theta \neq p_d$. Empirically we find that samples from even a well trained model p_θ may not always be sufficiently like the samples from the true data distribution. This can lead to a degradation in the performance of the inference network when using the reverse-sleep approach. For this reason, we propose to use a mixture distribution between the model and the empirical training data distribution as follows

$$\left\langle \mathrm{KL}\big(q_\phi(z|x)||p_\theta(z|x)\big) \right\rangle_{m(x)} \quad \text{where} \quad m(x) \equiv \alpha p_\theta(x) + (1-\alpha)\hat{p}_d. \tag{17}$$

When $\alpha = 0$, it reduces to the standard approach used in VAE training. When $\alpha = 1$, we recover the reverse sleep method (Equation 4). We find that a setting of $\alpha = 0.5$ works well in practice. This balances samples from the true underlying data distribution with samples from the model.

We thus refer to this method as *reverse half-asleep* since it uses both data and model samples to train the amortized posterior. To compare with different α, we first fit a VAE (with the same structure as that used in Figure 2) to the Binary MNIST dataset, and then train the amortized posterior using sleep inference (Equation 13) and three different α for additional 100 epochs using Adam with learning rate 3×10^{-4}. Figure 4 shows the test BPD comparison, where we find the proposed reverse half-asleep method ($\alpha = 0.5$) outperforms the reversed sleep method ($\alpha = 1$), whereas the standard amortized inference training in VAE ($\alpha = 0$) leads to overfitting of the inference network. We also plot the sleep inference as a comparison, whose BPD is less competitive since it is not directly optimizing the ELBO.

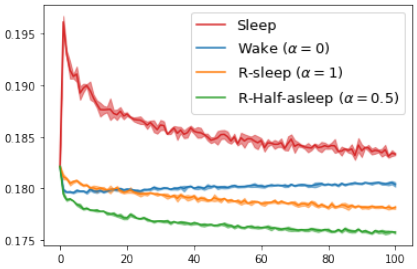


Figure 4: Test BPD comparisons of Amortized inference with different α. The mean and std are calculated with three random seeds.

4 Generalization Experiments

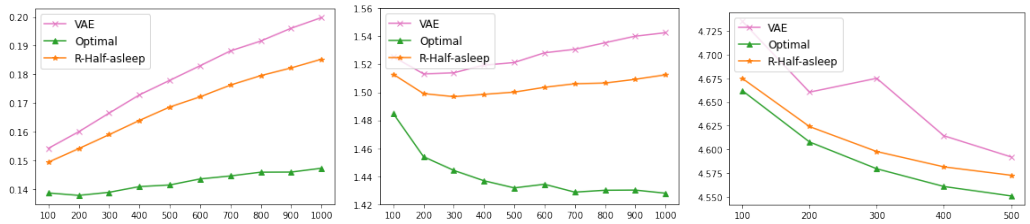
We apply the reverse half-asleep to improve the generalization of VAEs on three different datasets: binary MNIST, grey MNIST [18] and CIFAR10 [17]. For binary and grey MNIST, we use latent dimension 16/32 and neural nets with 2 layers of 500 hidden units in both the encoder and decoder. We use Bernoulli $p(x|z)$ for binary MNIST and discretized logistic distribution for grey MNIST. We train the VAE with the usual amortized inference approach using Adam with $lr = 3 \times 10^{-4}$ for 1000 epochs and save the model every 100 epochs. We then use the saved models to 1) evaluate on the test data sets, 2) conduct optimal inference by training $q_\phi(z|x)$ on the test data and 3) run reverse half-asleep method before calculating the test BPD. For the reverse half-asleep, we train the amortized posterior for 100 epochs with Adam and $lr = 5 \times 10^{-4}$. To sample from $p_\theta(x)$, we firstly sample $z' \sim p(z)$ and sample $x' \sim p(x|z = z')$. For optimal inference, we train the amortized posterior with the same optimization scheme on the test data set for additional 500 epochs to ensure the same gradient steps are conducted (since training dataset is 5 times as big as the test set).

For CIFAR10, we use the convolutional ResNet [11, 32] with 2 residual blocks and latent size 128. The observational distribution is a discretized logistic distribution with linear autoregressive parameterization within channels. We train the VAE for 500 epochs with Adam and $lr = 5 \times 10^{-4}$ and save the model every 100 epoch. The pre-trained VAE achieves 4.592 BPD on the CIFAR10, which is comparable with other single latent VAE models reported in [32]: 4.51 BPD with a VAE with latent dimension 256 and 4.67 BPD with a discrete latent VAE (VQVAE). Ideally, when the VAE model converges to the true distribution $p_\theta \to p_d$, the aggregate posterior $q_\phi(z) = \int q_\phi(z|x)p_d(x)dx$ will match the prior $p(z)$. However, for a complex distribution like CIFAR10, a significant mismatch between $q_\phi(z)$ and $p(z)$ is usually observed in practice [38, 7]. In this case, the sample x' that is generated using a latent sample from the prior $x' \sim p(x|z')$ where $z' \sim p(z)$ may be blurry or invalid. A common solution is to train another model, e.g. a VAE [7] or a PixelCNN [33, 32] to approximate $q_\phi(x)$. In our case, we instead directly sample from $q_\phi(z)$ instead of $p(z)$ to generate samples in Equation . This scheme still results in a consistent training objective since $q_{\phi^*}(z) = p(z)$ for optimal posterior $q_{\phi^*}(z|x)$. We use Adam with $lr = 1 \times 10^{-5}$ and train the reverse half-asleep inference for 100 epochs on the training data and train the optimal inference strategy for 500 epochs on the test data. Figure 5 show the test BPD comparison for three datasets. We find that the reverse half-asleep training approach consistently improves the generalization performance of our models as measured by the negative test ELBO. In contrast to the optimal inference strategy, our approach does not require further training on test data to improve generalization performance.

Paper [28] proposed to alleviate the overfitting of the amortized inference by optimizing a linear combination between the traditional amortized inference (Equation 4) and a denoising objective

$$\alpha \left\langle \mathrm{KL}(q_\phi(z|x+\epsilon)||p_\theta(z|x)) \right\rangle_{p(\epsilon)} + (1-\alpha)\mathrm{KL}(q_\phi(z|x)||p_\theta(z|x)), \tag{18}$$

where $p(\epsilon) = \mathrm{N}(0, \sigma^2 I)$. We compare this regularizer to our method by training the amortized posterior of VAEs for additional 100/300 and 100 epochs on Binary/Grey MNSIT and CIFAR respectively. For denoising regularizer, we use the same linear combination weight $\alpha = 0.5$ as that used in Equation 17 and vary $\sigma \in \{0.1, 0.2, 0.4, 0.6, 0.8, 1.0\}$. See Table 1 for the comparison results. We find for MNIST, $\sigma \in \{0.1, 0.2, 0.4\}$ can improve the generalization but larger level noise hurts



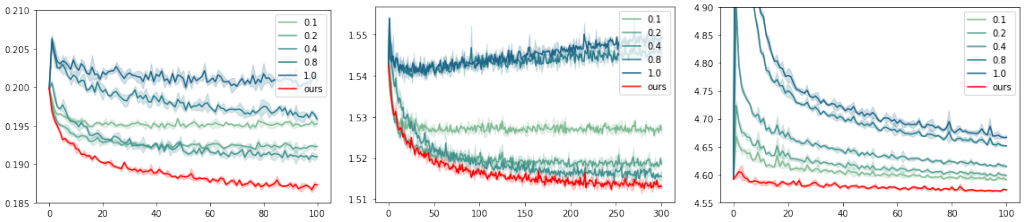
(a) Binary MNIST (b) Grey MNIST (c) CIFAR10

Figure 5: Test BPD comparisons among amortized inference (VAE), optimal inference and the reverse half-asleep inference on three datasets. The x-axis represents the training epochs.

the inference performance. For CIFAR10, only $\sigma = 0.1$ can slightly improve the generalization by 0.001 BPD. In contrast, our method can achieve better generalization performance without tuning any hyper-parameters. Our method requires sampling from the model, which does require additional computations, but is negligible comparing to the whole training procedure. Figure 6 reports the test BPD curves evaluated after every training epoch (the mean and std are calculated over 3 random seeds). We find our method consistently improves the model generalization during training.

Table 1: Average test BPD comparisons with Denoising Regularizer [28].

Methods	VAE	$\sigma = 0.1$	$\sigma = 0.2$	$\sigma = 0.4$	$\sigma = 0.8$	$\sigma = 1.0$	Ours
Binary MNIST	0.200	0.195	0.192	0.191	0.196	0.201	**0.187**
Grey MNIST	1.543	1.527	1.519	1.515	1.545	1.550	**1.513**
CIFAR10	4.592	4.591	4.598	4.614	4.651	4.667	**4.572**



(a) Binary MNIST (b) Grey MNIST (c) CIFAR 10

Figure 6: Test BPD evaluated after every training epoch.

5 Application of Lossless Compression

Lossless compression is an important application of VAEs where generalization plays a key role in the compression rate. Given a VAE with decoder $p_\theta(x|z)$, amortized posterior $q_\phi(z|x)$ and prior $p(z)$, a practical compressor can be efficiently implemented using the Bits Back algorithm [13, 30] with the ANS coder [9]. In Algorithm 1, we describe the Bits Back procedure with amortized inference to compress/decompress a test data point x'. The resulting code length for data x' is approximately equal to the negative one-data ELBO

$$\mathsf{code_len}(x') \approx -\log_2 p_\theta(x'|z') - \log_2 p(z') + \log_2 q_\phi(z'|x'). \tag{19}$$

We have shown that $q_\phi(z|x)$ may over-fit to the training data, degrading compression performance. To improve the compression BPD, the optimal inference method can also be applied in the Bits Back algorithm. In the compression stage, we can train ϕ by

$$\phi^* = \arg\max_\phi \mathrm{ELBO}(x', \theta, \phi). \tag{20}$$

When the $q_\phi(z|x')$ is parameterized to be a Gaussian, we can just take ϕ to be the mean

Algorithm 1 Bits Back with Amortized Inference.

Comp. and decomp. stages share $\{p_\theta(x|z), q_\phi(z|x)\}$.

Compression

Draw sample $z' \sim q_\phi(z|x')$ from the stack.
Encode $x' \sim p_\theta(x|z')$, $z' \sim p(z)$ onto the stack.

Decompression

Decode $z' \sim p(z)$, $x' \sim p_\theta(x|z')$ from the stack.
Encode $z' \sim q_\phi(z|x')$ onto the stack.

and standard deviation $\mathrm{N}(\phi_\mu, \phi_\sigma^2)$, which only contains two training parameters. In the decompression stage, we observe that the compressed data x' is recovered before the $q_\phi(z|x')$ is used to encode z'. Therefore, we can also train the $q_\phi(z|x')$ using the recovered x' to maximize the test ELBO. If the optimization procedure is the same as that used in the compression stage, we will get the same $q_{\phi^*}(z|x')$. In practice, we need to pre-specify the number of gradient descent steps K. When K is large, we recover the optimal inference scheme and the code length is approximately

$$\mathsf{code_len}(x') \approx -\log_2 p_\theta(x'|z') - \log_2 p(z') + \log_2 q_{\phi^*}(z'|x'). \tag{21}$$

This observation was first proposed in [34] in the context of lossy compression and then applied to lossless compression with Bits Back coding in [24]. Furthermore, by varying the optimization steps K in the optimal inference, we can also control a trade-off between the speed and the compression rate. This is valuable for practical applications with different speed/rate requirements. See Algorithm 2 for a summary of the Bits Back algorithm with K-step optimal inference.

Although the optimal inference can be used in the lossless compression, it requires extra run-time for training at the compression stages. On the other hand, the proposed reverse half-asleep inference scheme can improve the compression rate *without* scarifying any speed. Additionally, our method can also provide a better initialization for the optimal-inference to allow a better trade-off between compression rate and speed. Therefore, we implement[3] Bits Back with ANS [9] and compare the compression among four inference methods:

Algorithm 2 Bits Back with K-step Optimal Inference

Comp./decomp. stages share $\{p_\theta(x|z), q_\phi(z|x)\}$ and the optimization procedure of Equation 20.

Compression

Take K gradient steps $\phi \to \phi^K$ with Equation 20.
Draw sample $z' \sim q_{\phi^K}(z|x')$ from the stack.
Encode $x' \sim p_\theta(x|z')$, $z' \sim p(z)$ onto the stack.

Decompression

Decode $z' \sim p(z)$, $x' \sim p_\theta(x|z')$ from the stack.
Take K gradient steps $\phi \to \phi^K$ with Equation 20.
Encode $z' \sim q_{\phi^K}(z|x')$ onto the stack.

1. Baseline: This is the classic VAE-based compression introduced by [30]. For binary and grey MNIST, both the encoder and decoder contain 2 fully connected layers with 500 hidden units and latent dimension 10. The observation distributions are Bernoulli and discretized Logistic distribution respectively. For CIFAR10, we use fully convolutional ResNets [11] with 3 residual blocks in both the encoder and decoder, latent dimension 128 and discreteized Logistic distribution with channel-wise linear autoregressive[25] as the observation distribution. We train both the amortized posterior and the decoder by maximizing the ELBO (Equation 1) using Adam with $lr = 3 \times 10^{-4}$ for 100/100/500 epochs (Binary MNIST/Grey MNIST/CIFAR10), and then apply Algorithm 1 to conduct compression.

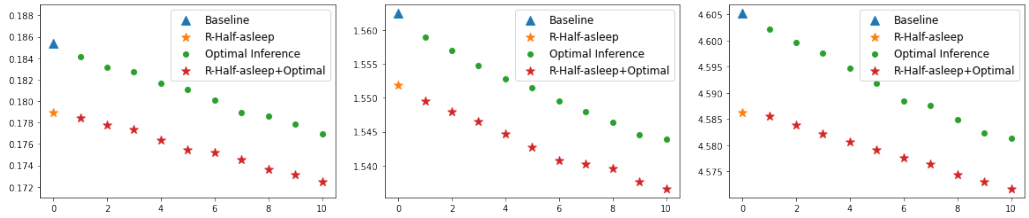
2. Reversed Half-asleep: we train the amortized posterior using the proposed training criterion (Equation 17) for 100/300 epochs using the Adam optimizer with $lr = 3 \times 10^{-4}$ for binary and grey MNIST, and $lr = 1 \times 10^{-5}$ for 100 epochs for CIFAR10. All other training details are the same as the baseline method.

3. Optimal Inference: we take the amortized posterior (encoder) and decoder from the baseline and apply the K-step optimal inference described in Algorithm 2 to do compression. We use Adam optimizer and vary the K from 1 to 10 to achieve a trade-off curve between compression rate and speed. We actively choose the highest learning rate that can make the BPD consistently improve with the increment of K: $lr = 5 \times 10^{-3}$ for binary and grey MNIST and $lr = 1 \times 10^{-3}$ for CIFAR10.

4. Reversed Half-asleep + Optimal Inference: we take the encoder in method 2 and decoder from the baseline and conduct K-step optimal inference. All other training details are as per method 3.

In Figure 7, we plot test BPD comparisons for different methods. We can see if optimization is not allowed at compression time, the use of our reverse-half-asleep method achieves better compression rate with no additional computational cost. If we allow K-step optimization during compression, for a given computational budget, the amortized posterior initialized using our reverse-half-asleep method also achieves lower BPD, which leads to a better trade-off between the time and compression rate. See Table 8 for the average time improvements of our method to compress a single MNIST and CIFAR10 image respectively.

[3]Implementation can be found in the anonymous repo: `https://anonymous.4open.science/r/GeneralizationGapInAmortizedInference-1E44/`. All experiments are run on a NVIDIA V100 GPU.



(a) Binary MNIST (b) Grey MNIST (c) CIFAR10

Figure 7: We plot the comparisons for different methods. The y-axis is the BPD and x-axis represents the K gradient steps in the optimal inference. The baseline and our R-Half-sleep can be seen as special cases of optimal inference with $K = 0$. We find given a fixed computational budget, our method achieves a lower BPD than one using traditional amortized inference training.

	Baseline	Ours	K=7
BPD	0.185	0.179	0.179
Com. Time	0.006	0.006	0.013
Dec. Time	0.006	0.006	0.013
Time Cost	-	0%	116.7%

	Baseline	Ours	K=8
BPD	4.602	4.585	4.585
Com. Time	0.27	0.27	0.38
Dec. Time	0.26	0.26	0.38
Time Cost	-	0%	46.2%

Figure 8: Compression (Com.) and decompression (Dec.) time comparison. We show that to achieve the same BPD as our method, the K-step optimal inference that initializes the amortized posterior needs $K = 7$ (binary MNIST, left) and $K = 8$ (CIFAR10, right) steps for each test datapoint, which will cost an additional 116.7% and 46.2% of time respectively during compression.

6 Related Work

A different perspective on generative models' generalization is proposed in paper [37] where the generalization is evaluated by testing if the model can generate novel combinations of features. However, the generalization defined in our paper is purely measured by the test likelihood, which is a different perspective and more relevant for the application of lossless compression.

Recent work [36] first studies the likelihood-based generalization for lossless compression. They focus on the test and train data that are from different distributions whereas we assume they follow the same distribution. Additionally, their model has a tractable likelihood and relates to the *generative model related generalization*, whereas we focus on the *inference related generalization* in VAEs.

Previous work [6] studied the *amortization gap* in amortized inference, which is caused by using $q_{\phi^*}(z|x_n)$ to generate posteriors for each input x_n rather than learning a posterior $q_n^*(z)$ for x_n individually. This gap can be alleviated using a larger capacity encoder network. This amortization gap is fundamentally different from the *inference generalization gap* we discuss in this work since the latter focuses solely on test time generalization but the former problem also exists at training time.

Paper [4] also uses samples from the model to train the encoder, which aims to make the learned representations more robust to adversarial attacks, whereas the motivation of our method is to improve generalization performance in terms of the test likelihood. Although the task in their paper is different, our work can provide a different view to explain their method: good generalization of $q_\phi(z|x)$ can benefit downstream tasks like representation learning performance.

Recent work [24] proposes a compression scheme based on the IWAE [3] bound, which is tighter than the ELBO and thus improves the compression rate. However, this method has to compress/decompress multiple latent samples, which requires extra time cost. On the other hand, we focus on improving the ELBO-based compression that only needs to compress one single latent sample. Nevertheless, similar to the K-step optimal inference method, our amortized training objective can also be used in the IWAE-based method, which gives a better proposal distribution for importance sampling.

7 Conclusion

We have shown how the generalization of VAEs is largely affected by the amortized inference network and proposed a new variational inference scheme that provides better generalization as demonstrated in the application of lossless compression.

References

[1] F. V. Agakov and D. Barber. An auxiliary variational method. In *International Conference on Neural Information Processing*, pages 561–566. Springer, 2004.

[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

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

[4] A. T. Cemgil, S. Ghaisas, K. Dvijotham, S. Gowal, and P. Kohli. Autoencoding variational autoencoder. *Neural Information Processing Systems*, 2020.

[5] E. Challis and D. Barber. Affine independent variational inference. *Advances in Neural Information Processing Systems*, 25:2186–2194, 2012.

[6] C. Cremer, X. Li, and D. Duvenaud. Inference suboptimality in variational autoencoders. In *International Conference on Machine Learning*, pages 1078–1086. PMLR, 2018.

[7] B. Dai and D. Wipf. Diagnosing and enhancing vae models. *arXiv preprint arXiv:1903.05789*, 2019.

[8] P. Dayan, G. E. Hinton, R. M. Neal, and R. S. Zemel. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

[9] J. Duda. Asymmetric numeral systems: entropy coding combining speed of huffman coding with compression rate of arithmetic coding. *arXiv preprint arXiv:1311.2540*, 2013.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[12] G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.

[13] 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.

[14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *International conference on machine learning*, 2015.

[15] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *International Conference on Learning Representations*, 2013.

[16] F. Kingma, P. Abbeel, and J. 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.

[17] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[18] Y. LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[19] F. Locatello, S. Bauer, M. Lucic, G. Raetsch, S. Gelly, B. Schölkopf, and O. Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, pages 4114–4124. PMLR, 2019.

[20] L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther. Auxiliary deep generative models. In *International conference on machine learning*, pages 1445–1453. PMLR, 2016.

[21] D. J. MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[22] D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[23] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and variational inference in deep latent gaussian models. In *International Conference on Machine Learning*, volume 2, page 2. Citeseer, 2014.

[24] Y. Ruan, K. Ullrich, D. Severo, J. Townsend, A. Khisti, A. Doucet, A. Makhzani, and C. J. Maddison. Improving lossless compression rates via monte carlo bits-back coding. *arXiv preprint arXiv:2102.11086*, 2021.

[25] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

[26] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[27] C. E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review*, 5(1):3–55, 2001.

[28] R. Shu, H. H. Bui, S. Zhao, M. J. Kochenderfer, and S. Ermon. Amortized inference regularization. *Neural Information Processing Systems*, 2018.

[29] L. Theis, A. v. d. Oord, and M. Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

[30] J. Townsend, T. Bird, and D. Barber. Practical lossless compression with latent variables using bits back coding. *International Conference on Learning Representations*, 2019.

[31] J. Townsend, T. Bird, J. Kunze, and D. Barber. Hilloc: Lossless image compression with hierarchical latent variable models. *International Conference on Learning Representations*, 2020.

[32] A. Van Den Oord, O. Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[33] A. Van Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pages 1747–1756. PMLR, 2016.

[34] Y. Yang, R. Bamler, and S. Mandt. Improving inference for neural image compression. *arXiv preprint arXiv:2006.04240*, 2020.

[35] M. Zhang, P. Hayes, T. Bird, R. Habib, and D. Barber. Spread divergence. In *International Conference on Machine Learning*, pages 11106–11116. PMLR, 2020.

[36] M. Zhang, A. Zhang, and S. McDonagh. On the out-of-distribution generalization of probabilistic image modelling. In *Neural Information Processing Systems*, 2021.

[37] S. Zhao, H. Ren, A. Yuan, J. Song, N. Goodman, and S. Ermon. Bias and generalization in deep generative models: An empirical study. *arXiv preprint arXiv:1811.03259*, 2018.

[38] S. Zhao, J. Song, and S. Ermon. Infovae: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262*, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] In the end of section 4.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See the footnote in page 8.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See section 4 and 5.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Figure 6.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See the footnote in page 8.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cite the datasets in section 4.
   (b) Did you mention the license of the assets? [N/A] We didn't find licences for MNIST and CIFAR10.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]