

Inference Suboptimality in Variational Autoencoders

Chris Cremer¹ Xuechen Li¹ David Duvenaud¹

Abstract

Amortized inference allows latent-variable models trained via variational learning to scale to large datasets. The quality of approximate inference is determined by two factors: a) the capacity of the variational distribution to match the true posterior and b) the ability of the recognition network to produce good variational parameters for each datapoint. We examine approximate inference in variational autoencoders in terms of these factors. We find that divergence from the true posterior is often due to imperfect recognition networks, rather than the limited complexity of the approximating distribution. We show that this is due partly to the generator learning to accommodate the choice of approximation. Furthermore, we show that the parameters used to increase the expressiveness of the approximation play a role in generalizing inference rather than simply improving the complexity of the approximation.

1. Introduction

In this paper, we analyze inference suboptimality: the mismatch between the true and approximate posterior. More specifically, we are interested in understanding what factors cause the gap between the marginal log-likelihood and the evidence lower bound (ELBO) in variational autoencoders (VAEs, Kingma & Welling (2014); Rezende et al. (2014)). We refer to this as the *inference gap*. Moreover, we break down the inference gap into two components: the *approximation gap* and the *amortization gap*. The approximation gap comes from the inability of the variational distribution family to exactly match the true posterior. The amortization gap refers to the difference caused by amortizing the variational parameters over the entire training set, instead of optimizing for each training example individually. We refer the reader to Table 1 for the definitions of the gaps and to

Fig. 1 for a simple illustration of the gaps. In Fig. 1, $\mathcal{L}[q]$ refers to the ELBO evaluated using an amortized distribution q , as is typical of VAE training. In contrast, $\mathcal{L}[q^*]$ is the ELBO evaluated using the optimal approximation within its variational family.

There has been significant work on improving variational inference in VAEs through the development of expressive approximate posteriors (Rezende & Mohamed, 2015; Kingma et al., 2016; Ranganath et al., 2016; Tomczak & Welling, 2016; 2017). These works have shown that with more expressive approximate posteriors, the model learns a better distribution over the data. Our study aims to gain a better understanding of the relationship between expressive approximations and improved generative models.

Our experiments investigate how the choice of encoder, posterior approximation, decoder, and optimization affect the approximation and amortization gaps. We train VAE models in a number of settings on the MNIST (LeCun et al., 1998), Fashion-MNIST (Xiao et al., 2017), and CIFAR-10 (Krizhevsky & Hinton, 2009) datasets.

Our contributions are: a) we investigate inference suboptimality in terms of the approximation and amortization gaps, providing insight to guide future improvements in VAE inference, b) we quantitatively demonstrate that the learned generative model accommodates the choice of approximation, and c) we demonstrate that parameterized functions that improve the expressiveness of the approximation play a significant role in reducing amortization error.

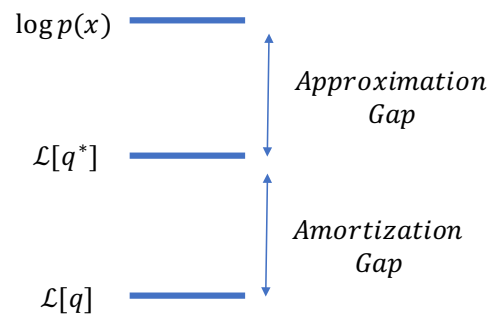


Figure 1. Gaps in Inference

¹Department of Computer Science, University of Toronto, Toronto, Canada. Correspondence to: Chris Cremer <crcemer@cs.toronto.edu>.

Term	Definition	VAE Formulation
Inference	$\log p(x) - \mathcal{L}[q]$	$\text{KL}(q(z x) p(z x))$
Approximation	$\log p(x) - \mathcal{L}[q^*]$	$\text{KL}(q^*(z x) p(z x))$
Amortization	$\mathcal{L}[q^*] - \mathcal{L}[q]$	$\text{KL}(q(z x) p(z x)) - \text{KL}(q^*(z x) p(z x))$

Table 1. Summary of Gap Terms. The middle column refers to the general case where our variational objective is a lower bound on the marginal log-likelihood. The right most column demonstrates the specific case in VAEs. $q^*(z|x)$ refers to the optimal approximation within a family \mathcal{Q} , i.e. $q^*(z|x) = \arg \min_{q \in \mathcal{Q}} \text{KL}(q(z|x)||p(z|x))$.

2. Background

2.1. Inference in Variational Autoencoders

Let x be the observed variable, z the latent variable, and $p(x, z)$ be their joint distribution. Given a dataset $X = \{x_1, x_2, \dots, x_N\}$, we would like to maximize the marginal log-likelihood with respect to the model parameters θ :

$$\log p_\theta(X) = \sum_{i=1}^N \log p_\theta(x_i) = \sum_{i=1}^N \log \int p_\theta(x_i, z_i) dz_i.$$

In practice, the marginal log-likelihood is computationally intractable due to the integration over the latent variable z . Instead, VAEs introduce an inference network $q_\phi(z|x)$ to approximate the true posterior $p(z|x)$ and optimize the ELBO with respect to model parameters θ and inference network parameters ϕ (parameterization subscripts omitted for brevity):

$$\log p(x) = \mathbb{E}_{q(z|x)} \left[\log \left(\frac{p(x, z)}{q(z|x)} \right) \right] + \text{KL}(q(z|x)||p(z|x)) \quad (1)$$

$$\geq \mathbb{E}_{q(z|x)} \left[\log \left(\frac{p(x, z)}{q(z|x)} \right) \right] = \mathcal{L}_{\text{VAE}}[q]. \quad (2)$$

From the above equation, we see that the ELBO is tight when $q(z|x) = p(z|x)$. The choice of $q(z|x)$ is often a factorized Gaussian distribution for its simplicity and efficiency. By utilizing the inference network (also referred to as encoder or recognition network), VAEs amortize inference over the entire dataset. Furthermore, the overall model is trained by stochastically optimizing the ELBO using the reparametrization trick (Kingma & Welling, 2014).

2.2. Expressive Approximate Posteriors

There are a number of strategies for increasing the expressiveness of approximate posteriors, going beyond the original factorized-Gaussian. We briefly summarize normalizing flows and auxiliary variables.

2.2.1. NORMALIZING FLOWS

Normalizing flow (Rezende & Mohamed, 2015) is a change of variables procedure for constructing complex distributions by transforming probability densities through a series of invertible mappings. Specifically, if we transform

a random variable z_0 with distribution $q_0(z)$, the resulting random variable $z_T = T(z_0)$ has a distribution:

$$q_T(z_T) = q_0(z_0) \left| \det \frac{\partial z_T}{\partial z_0} \right|^{-1}. \quad (3)$$

By successively applying these transformations, we can build arbitrarily complex distributions. Stacking these transformations remains tractable due to the determinant being decomposable: $\det(AB) = \det(A)\det(B)$. An important property of these transformations is that we can take expectations with respect to the transformed density $q_T(z_T)$ without explicitly knowing its formula due to the law of the unconscious statistician (LOTUS):

$$\mathbb{E}_{q_T}[h(z_T)] = \mathbb{E}_{q_0}[h(f_T(f_{T-1}(\dots f_1(z_0))))]. \quad (4)$$

Using equations (3) and (4), the lower bound with the transformed approximation can be written as:

$$\mathbb{E}_{z_0 \sim q_0(z|x)} \left[\log \left(\frac{p(x, z_T)}{q_0(z_0|x) \prod_{t=1}^T \left| \det \frac{\partial z_t}{\partial z_{t-1}} \right|^{-1}} \right) \right]. \quad (5)$$

The main constraint on these transformations is that the determinant of their Jacobian needs to be easily computable.

2.2.2. AUXILIARY VARIABLES

Deep generative models can be extended with auxiliary variables which leave the generative model unchanged but make the variational distribution more expressive. Just as hierarchical Bayesian models induce dependencies between data, hierarchical variational models can induce dependencies between latent variables. The addition of the auxiliary variable changes the lower bound to:

$$\begin{aligned} & \mathbb{E}_{z, v \sim q(z, v|x)} \left[\log \left(\frac{p(x, z) r(v|x, z)}{q(z, v|x)} \right) \right] \\ &= \mathbb{E}_{q(z|x)} \left[\log \left(\frac{p(x, z)}{q(z|x)} \right) - \text{KL}(q(v|z, x)||r(v|x, z)) \right] \end{aligned} \quad (6)$$

where $r(v|x, z)$ is called the reverse model. From Eqn. 7, we see that this bound is looser than the regular ELBO,

however the extra flexibility provided by the auxiliary variable can result in a higher lower bound. This idea has been employed in works such as auxiliary deep generative models (ADGM, (Maaløe et al., 2016)), hierarchical variational models (HVM, (Ranganath et al., 2016)) and Hamiltonian variational inference (HVI, (Salimans et al., 2015)).

3. Methods

3.1. Approximation and Amortization Gaps

The inference gap \mathcal{G} is the difference between the marginal log-likelihood $\log p(x)$ and a lower bound $\mathcal{L}[q]$. Given the distribution in the family that maximizes the bound, $q^*(z|x) = \arg \max_{q \in \mathcal{Q}} \mathcal{L}[q]$, the inference gap decomposes as the sum of approximation and amortization gaps:

$$\mathcal{G} = \log p(x) - \mathcal{L}[q] = \underbrace{\log p(x) - \mathcal{L}[q^*]}_{\text{Approximation}} + \underbrace{\mathcal{L}[q^*] - \mathcal{L}[q]}_{\text{Amortization}}.$$

For VAEs, we can translate the gaps to KL divergences by rearranging Eqn. (1):

$$\begin{aligned} \mathcal{G}_{\text{VAE}} = & \underbrace{\text{KL}(q^*(z|x) || p(z|x))}_{\text{Approximation}} \\ & + \underbrace{\text{KL}(q(z|x) || p(z|x)) - \text{KL}(q^*(z|x) || p(z|x))}_{\text{Amortization}}. \end{aligned} \quad (8)$$

3.2. Flexible Approximate Posteriors

Our experiments involve expressive approximations which use flow transformations and auxiliary variables. The flow transformation that we employ is of the same type as the transformations of Real NVP (Dinh et al., 2017). We partition the latent variable z into two, z_1 and z_2 , then perform the following transformations:

$$z'_1 = z_1 \circ \sigma_1(z_2) + \mu_1(z_2) \quad (9)$$

$$z'_2 = z_2 \circ \sigma_2(z'_1) + \mu_2(z'_1) \quad (10)$$

where $\sigma_1, \sigma_2, \mu_1, \mu_2 : \mathbb{R}^n \rightarrow \mathbb{R}^n$ are differentiable mappings parameterized by neural nets and \circ takes the Hadamard or element-wise product. We partition the latent variable by simply indexing the elements of the first half and the second half. The determinant of the combined transformation’s Jacobian, $\left| \det \left(\frac{\partial z'}{\partial z} \right) \right|$, can be easily evaluated. See section 6.4 of the Supplementary material for a derivation. The lower bound of this approximation is the same as Eqn. (5). We refer to this approximation as q_{Flow} .

We also experiment with an approximation that combines flow transformations and auxiliary variables. Let $z \in \mathbb{R}^n$ be the variable of interest and $v \in \mathbb{R}^n$ the auxiliary variable. The flow is the same as equations (9) and (10), where z_1 is

replaced with z and z_2 with v . We refer to this approximate distribution as q_{AF} , where AF stands for auxiliary flow. We train this model by optimizing the following bound:

$$\begin{aligned} \mathbb{E}_{q_0(z,v|x)} \left[\log \left(\frac{p(x, z_T) r(v_T|x, z_T)}{q_T(z_T, v_T|x) \left| \det \left(\frac{\partial z_t v_t}{\partial z_{t-1} v_{t-1}} \right) \right|^{-1}} \right) \right] \\ = \mathcal{L}[q_{\text{AF}}]. \end{aligned} \quad (11)$$

Note that this lower bound is looser as explained in Section 2.2.2. We refer readers to Section 6.1.2 in the Supplementary material for specific details of the flow configuration adopted in the experiments.

3.3. Marginal Log-Likelihood Estimation and Evidence Lower Bounds

In this section, we describe the estimates we use to compute the bounds of the inference gaps: $\log p(x)$, $\mathcal{L}[q^*]$, and $\mathcal{L}[q]$. We use two bounds to estimate the marginal log-likelihood, $\log p(x)$: IWAE (Burda et al., 2016) and AIS (Neal, 2001).

The IWAE bound takes multiple importance weighted samples from the variational q distribution resulting in a tighter lower bound than the VAE bound. The IWAE bound is computed as:

$$\begin{aligned} \log p(x) \geq \mathbb{E}_{z_1 \dots z_k \sim q(z|x)} \left[\log \left(\frac{1}{k} \sum_{i=1}^k \frac{p(x, z_i)}{q(z_i|x)} \right) \right] \quad (12) \\ = \mathcal{L}_{\text{IWAE}}[q]. \end{aligned}$$

As the number of importance samples approaches infinity, the bound approaches the marginal log-likelihood. It is often used as an evaluation metric for generative models (Burda et al., 2016; Kingma et al., 2016). AIS is potentially an even tighter lower bound. AIS weights samples from distributions which are sequentially annealed from an initial proposal distribution to the true posterior. See Section 6.5 in the Supplementary material for further details regarding AIS. To compute the AIS bound, we use 100 chains, each with 10000 intermediate distributions, where each transition consists of one HMC trajectory with 10 leapfrog steps. The initial distribution for AIS is the prior, so that it is encoder-independent.

We estimate the marginal log-likelihood by independently computing our tightest lower bounds then take the maximum of the two:

$$\log \hat{p}(x) = \max(\mathcal{L}_{\text{AIS}}, \mathcal{L}_{\text{IWAE}}).$$

The $\mathcal{L}[q^*]$ and $\mathcal{L}[q]$ bounds are the standard ELBOs, \mathcal{L}_{VAE} , from Eqn. (2), computed with either the amortized q or the optimal q^* (see below). When computing \mathcal{L}_{VAE} and $\mathcal{L}_{\text{IWAE}}$, we use 5000 samples.

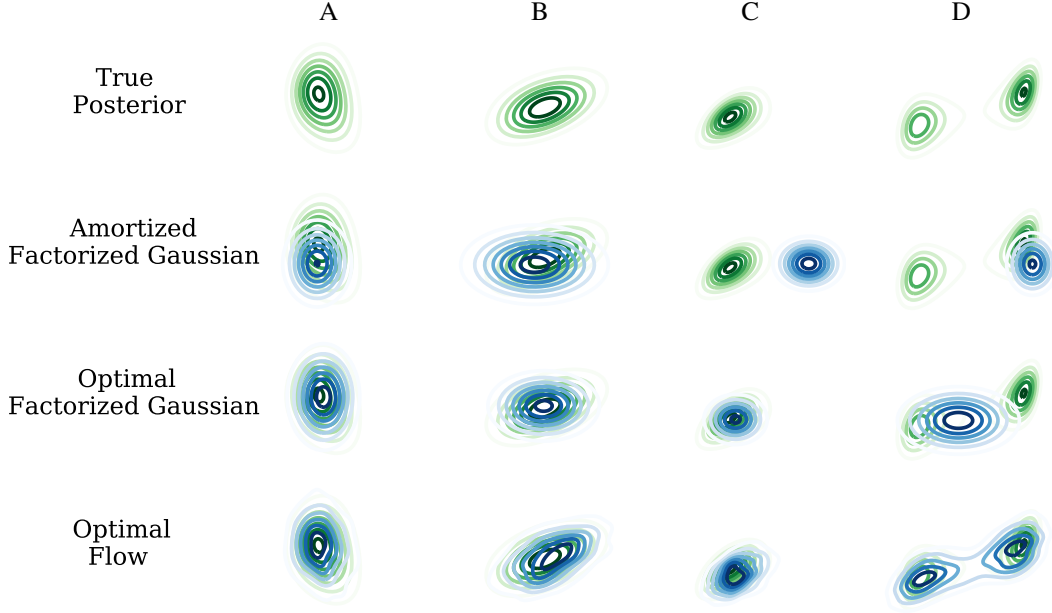


Figure 2. True Posterior and Approximate Distributions of a VAE with 2D latent space. The columns represent four different datapoints. The green distributions are the true posterior distributions, highlighting the mismatch with the blue approximations. Amortized: Variational parameters learned over the entire dataset. Optimal: Variational parameters optimized for each individual datapoint. Flow: Using a flexible approximate distribution.

3.4. Local Optimization of the Approximate Distribution

To compute $\mathcal{L}_{\text{VAE}}[q^*]$, we optimize the parameters of the variational distribution for every datapoint. For the local optimization of q_{FFG} , we initialize the mean and variance as the prior, i.e. $\mathcal{N}(0, I)$. We optimize the mean and variance using the Adam optimizer with a learning rate of 10^{-3} . To determine convergence, after every 100 optimization steps, we compute the average of the previous 100 ELBO values and compare it to the best achieved average. If it does not improve for 10 consecutive iterations then the optimization is terminated. For q_{Flow} and q_{AF} , the same process is used to optimize all of its parameters. All neural nets for the flow were initialized with a variant of the Xavier initialization (Glorot & Bengio, 2010). We use 100 Monte Carlo samples to compute the ELBO to reduce variance.

3.5. Validation of Bounds

The soundness of our empirical analysis depends on the reliability of the marginal log-likelihood estimator. For general importance sampling based estimators, the sample variance of the normalized importance weights can serve as an indicator of accuracy (Geweke, 1989; Neal, 2001). This quantitative measure, however, can also be unreliable, e.g. when the proposal misses an important mode of the target distribution (Neal, 2001).

In this work, we follow (Wu et al., 2017) to empirically validate our AIS estimates with Bidirectional Monte Carlo (BDMC, Grosse et al. (2015; 2016)). In addition to a lower bound provided by AIS, BDMC runs AIS chains backward from exact posterior samples to obtain an upper bound on the marginal log-likelihood. It should be noted that BDMC relies on the assumption that the distribution of the simulated data from the model roughly matches that of the real data. This is due to the backward chain initializes from exact posterior samples (Grosse et al., 2015).

For the MNIST and Fashion datasets, BDMC gives a gap within 0.1 nat for a linear schedule AIS with 10^4 intermediate distributions and 100 importance samples on 10^3 simulated datapoints. For 3-BIT CIFAR, the same AIS setting gives a gap within 1 nat with the sigmoidal annealing schedule (Grosse et al., 2015) on 100 simulated datapoints. Loosely speaking, this should give us confidence in how well our AIS lower bounds reflect the marginal log-likelihood computed on the real data.

4. Related Work

Much of the earlier work on variational inference focused on optimizing the variational parameters locally for each datapoint, e.g. the original Stochastic Variational Inference scheme (SVI, Hoffman et al. (2013)). To scale inference to large datasets, most related works utilize inference networks to amortize the cost of inference over the entire dataset.

	MNIST		Fashion-MNIST		3-BIT CIFAR	
	q_{FFG}	q_{AF}	q_{FFG}	q_{AF}	q_{FFG}	q_{AF}
$\log \hat{p}(x)$	-89.80	-88.94	-97.47	-97.41	-816.9	-820.56
$\mathcal{L}_{VAE}[q_{AF}^*]$	-90.80	-90.38	-98.92	-99.10	-820.19	-822.16
$\mathcal{L}_{VAE}[q_{FFG}^*]$	-91.23	-113.54	-100.53	-132.46	-831.65	-861.62
$\mathcal{L}_{VAE}[q]$	-92.57	-91.79	-104.75	-103.76	-869.12	-864.28
Approximation	1.43	1.44	3.06	1.69	14.75	1.60
Amortization	1.34	1.41	4.22	4.66	37.47	42.12
Inference	2.77	2.85	7.28	6.35	52.22	43.72

Table 2. Inference Gaps. The columns q_{FFG} and q_{AF} refer to the variational distribution used for training the model. These lower bounds are computed on the training set and are in units of nats.

Our work analyses the error that these inference networks introduce.

Most relevant to our work is the recent work of [Krishnan et al. \(2017\)](#), which explicitly remarks on two sources of error in variational learning with inference networks, and proposes to optimize approximate inference locally from an initialization output by the inference network. They show improved training on high-dimensional, sparse data with the hybrid method, claiming that local optimization reduces the negative effects of random initialization in the inference network early on in training. Thus their work focuses on reducing the amortization gap early on in training. Similar to this idea, [Hoffman \(2017\)](#) proposes to perform approximate inference during model training with MCMC at an initialization given by a variational distribution. Our work provides a means of explaining these improvements in terms of the sources of inference suboptimality that they reduce.

5. Experimental Results

5.1. Intuition through Visualization

To begin, we would like to gain an intuitive visualization of the gaps presented in Section 3.1. To this end, we trained a VAE with a two-dimensional latent space on MNIST and in Fig. 2 we show contour plots of various distributions in the latent space. The first row contains contour plots of the true posteriors $p(z|x)$ for four different training datapoints (columns). We have selected these four examples to highlight different inference phenomena. The amortized fully-factorized Gaussian (FFG) row refers to the output of the recognition net, in this case, a FFG approximation. Optimal FFG is the FFG that best fits the posterior of the datapoint. Optimal Flow is the optimal fit of a flexible distribution to the same posterior, where the flexible distribution we use is described in Section 3.2.

Posterior A is an example of a distribution where a FFG can fit relatively well. Posterior B is an example of a posterior with dependence between dimensions, demonstrating the

limitation of having a factorized approximation. Posterior C highlights a shortcoming of performing amortization with a limited-capacity recognition network, where the amortized FFG shares little support with the true posterior. Posterior D is a bi-modal distribution which demonstrates the ability of the flexible approximation to fit to complex distributions, in contrast to the simple FFG approximation. These observations raise the following question: in more typical VAEs, is the amortization of inference the leading cause of the distribution mismatch, or is it the limited expressiveness of the approximation?

5.2. Amortization vs Approximation Gap

In this section, we compare how much the approximation and amortization gaps each contribute to the total inference gap. Table 2 are results of inference on the training set of MNIST, Fashion-MNIST and 3-BIT CIFAR (a binarized version of CIFAR-10, see Section 6.1.3 for details). For each dataset, we trained models with two different approximate posterior distributions: a fully-factorized Gaussian, q_{FFG} , and the flexible distribution, q_{AF} . Due to the computational cost of optimizing the local parameters for each datapoint, our evaluation is performed on a subset of 1000 datapoints for MNIST and Fashion-MNIST and a subset of 100 datapoints for 3-BIT CIFAR.

For MNIST, we see that the amortization and approximation gaps each account for nearly half of the inference gap. On the more difficult Fashion-MNIST dataset, the amortization gap is larger than the approximation gap. For CIFAR, we see that the amortization gap is much more significant compared to the approximation gap. Thus, for the three datasets and model architectures that we consider, the amortization gap is likely to be the more prominent cause of inference suboptimality, especially when the dataset becomes more challenging to model. This indicates that improvements in inference will likely be a result of reducing amortization error, rather than approximation errors.

With these results in mind, would simply increasing the capacity of the encoder improve the amortization gap? We

Inference Suboptimality in Variational Autoencoders

	MNIST		Fashion-MNIST		MNIST		Fashion-MNIST	
	q_{FFG}	q_{AF}	q_{FFG}	q_{AF}	q_{FFG}	q_{AF}	q_{FFG}	q_{AF}
$\log \hat{p}(x)$	-89.61	-88.99	-95.99	-96.18	-89.82	-89.52	-102.56	-102.88
$\mathcal{L}_{VAE}[q_{AF}^*]$	-90.65	-90.44	-97.40	-97.91	-90.96	-90.45	-103.73	-104.02
$\mathcal{L}_{VAE}[q_{FFG}^*]$	-91.07	-108.71	-99.64	-129.70	-90.84	-92.25	-103.85	-105.80
$\mathcal{L}_{VAE}[q]$	-92.18	-91.19	-102.73	-101.67	-92.33	-91.75	-106.90	-107.01
Approximation	1.46	1.45	3.65	1.73	1.02	0.93	1.29	1.14
Amortization	1.11	0.75	3.09	3.76	1.49	1.30	3.05	2.29
Inference	2.56	2.20	6.74	5.49	2.51	2.23	4.34	4.13

Table 3. Left: Larger Encoder. Right: Models trained without entropy annealing. The columns q_{FFG} and q_{AF} refer to the variational distribution used for training the model. The lower bounds are computed on the training set and are in units of nats.

examined this by training the MNIST and Fashion-MNIST models from above but with larger encoders. See Section 6.1.2 for implementation details. Table 3 (left) are the results of this experiment. Comparing to Table 2, we see that, for both datasets and both variational distributions, using a larger encoder results in the inference gap decreasing and the decrease is mainly due to a reduction in the amortization gap.

5.3. Influence of Flows on the Amortization Gap

The common reasoning for increasing the expressiveness of the approximate posterior is to minimize the difference between the true and approximate distributions, i.e. reduce the approximation gap. However, given that the expressive approximation is often accompanied by many additional parameters, we would like to know how much influence it has on the amortization error.

To investigate this, we trained a VAE on MNIST, discarded the encoder, then retrained encoders with different approximate distributions on the fixed decoder. We fixed the decoder so that the true posterior is constant for all the retrained encoders. The initial encoder was a two-layer MLP with a factorized Gaussian distribution. In order to emphasize a large amortization gap, the retrained encoders had no hidden layers (ie. just linear transformations). For the retrained encoders, we tested three approximate distributions: fully factorized Gaussian (q_{FFG}), auxiliary flow (q_{AF}), and Flow (q_{Flow}). See Section 3.2 for the details of these distributions.

The inference gaps of the retrained encoders on the training set are shown in Table 4. As expected, we observe that the small encoder with q_{FFG} has a very large amortization gap. However, when we use q_{AF} or q_{Flow} as the approximate distribution, we see the approximation gap decrease, but more importantly, there is a significant decrease in the amortization gap. This indicates that the parameters used for increasing the complexity of the approximation also play a large role in diminishing the amortization error.

Variational Family	q_{FFG}	q_{AF}	q_{Flow}
$\log \hat{p}(x)$	-84.70	-84.70	-84.70
$\mathcal{L}_{VAE}[q^*]$	-86.61	-85.48	-85.13
$\mathcal{L}_{VAE}[q]$	-129.83	-98.58	-97.99
Approximation	1.91	0.78	0.43
Amortization	43.22	13.10	12.86
Inference	45.13	13.88	13.29

Table 4. Influence of expressive approximations on the amortization gap. The parameters used to increase the flexibility of the approximate distribution also reduce the amortization gap.

These results are expected given that the parameterization of the Flow distribution can be interpreted as an instance of the RevNet (Gomez et al., 2017) which has demonstrated that Real-NVP transformations (Dinh et al., 2017) can model complex functions similar to typical MLPs. Thus the flow transformations we employ should also be expected to increase the expressiveness while also increasing the capacity of the encoder. The implication of this observation is that models which improve the flexibility of their variational approximation, and attribute their improved results to the increased expressiveness, may have actually been due to the reduction in amortization error.

5.4. Influence of Approximate Posterior on True Posterior

To what extent does the posterior approximation affect the learned model? Turner & Sahani (2011) studied the biases in parameter learning induced by the variational approximation when learning via variational Expectation-Maximization. Similarly, we ask whether a factorized Gaussian approximation causes the true posterior to be more like a factorized Gaussian? Burda et al. (2016) visually demonstrate that when trained with an importance-weighted approximate posterior, the resulting true posterior is more complex than those trained with factorized Gaussian approximations. Just as it is hard to evaluate a generative model by visually inspecting samples, it is hard to judge how “Gaussian” the true posterior is by visual inspection. We can quantitatively determine



Figure 3. Inference gaps over epochs trained on binarized Fashion-MNIST. Blue is the approximation gap. Orange is the amortization gap. Standard is a VAE with FFG approximation. Flow is a VAE with a Flow approximation.

how close the posterior is to a fully-factorized Gaussian (FFG) by comparing the marginal log-likelihood estimate $\log \hat{p}(x)$ and the Optimal FFG bound $\mathcal{L}_{\text{VAE}}[q_{\text{FFG}}^*]$. This is equivalent to estimating the KL divergence between the optimal Gaussian and the true posterior, $\text{KL}(q^*(z|x)||p(z|x))$.

In Table 2 on MNIST, for the FFG trained model, $\text{KL}(q^*(z|x)||p(z|x))$ is nearly the same for both q_{FFG}^* and q_{AF}^* . In contrast, on the model trained with q_{AF}^* , $\text{KL}(q^*(z|x)||p(z|x))$ is much larger for q_{FFG}^* than q_{AF}^* . This suggests that the true posterior of a FFG-trained model is closer to FFG than the true posterior of the Flow-trained model. The same observation can be made on the Fashion-MNIST dataset. This implies that the decoder can learn to have a true posterior that fits better to the approximation.

These observations justify our results of Section 5.2, which showed that the amortization error is often the main cause of inference suboptimality. One reason for this is that the generator accommodates the choice of approximation, thus reducing the approximation error.

Generator Hidden Layers	0	2	4
$\log \hat{p}(x)$	-100.52	-84.78	-82.19
$\mathcal{L}_{\text{VAE}}[q_{\text{FFG}}^*]$	-104.42	-86.61	-83.82
Approximation Gap	3.90	1.83	1.63

Table 5. Increased decoder capacity reduces the approximation gap.

Given that we have seen that the generator can accommodate the choice of approximation, our next question is whether a generator with more capacity increases its ability to fit to the approximation. To this end, we trained VAEs with decoders of different sizes and measured the approximation gaps on

the training set. Specifically, we trained decoders with 0, 2, and 4 hidden layers on MNIST. See Table 5 for the results. We see that as the capacity of the decoder increases, the approximation gap decreases. This result implies that the more flexible the generator is, the less flexible the approximate distribution needs to be to ensure accurate inference.

5.5. Inference Generalization

How well does amortized inference generalize at test time? We address this question by visualizing the gaps on training and validation datapoints across the training epochs. In Fig. 3, the models are trained on 50000 binarized Fashion-MNIST datapoints and the gaps are computed on a subset of a 100 training and validation datapoints. The top and bottom boundaries of the blue region represent $\log \hat{p}(x)$ and $\mathcal{L}[q^*]$. The bottom boundary of the orange region represents $\mathcal{L}[q]$. In other words, the blue region is the approximation gap and the orange is the amortization gap.

In Fig. 3, the Standard model (top left) refers to a VAE of latent size 20 trained with a factorized Gaussian approximate posterior. In this case, the encoder and decoder both have two hidden layers each consisting of 200 hidden units. The Flow model (top right) augments the Standard model with a q_{Flow} variational distribution. Larger Decoder and Larger Encoder models have factorized Gaussian distributions and increase the number of hidden layers to three and the number of units in each layer to 500.

Firstly, we observe that for all models, the approximation gap on the training and validation sets are roughly equivalent. This indicates that the true posteriors of the held-out data are similar to that of the training data. Secondly, we note that for all models, the encoder overfits more than the decoder.

These observations resonate with the encoder overfitting findings by Wu et al. (2017).

How does increasing decoder capacity affect inference on held-out data? We know from Section 5.4 that increasing generator capacity results in a posterior that better fits the approximation making posterior inference easier. Furthermore, the Larger Decoder plot of Fig. 3 shows that increasing generator capacity causes the model to be more prone to overfitting. Thus, there is a tradeoff between ease of inference and decoder overfitting.

5.5.1. ENCODER CAPACITY AND APPROXIMATION EXPRESSIVENESS

We have seen in Sections 5.2 and 5.3 that expressive approximations as well as increasing encoder capacity can lead to a reduction in the amortization gap. This leads us to the following question: when should we increase encoder capacity versus increasing the expressiveness of the approximation? We answer this question in terms of how well each model can generalize its efficient inference (recognition network and variational distribution) to held-out data.

In Fig. 3, we see that the Flow model and the Larger Encoder model achieve similar $\log \hat{p}(x)$ on the validation set at the end of training. However, we see that the $\mathcal{L}[q]$ bound of the Larger Encoder model is significantly lower than the $\mathcal{L}[q]$ bound of the Flow model due to the encoder overfitting to the training data. Although they both model the data nearly equally well, the recognition net of the Larger Encoder model is no longer suitable to perform inference on the held-out data due to overfitting. Thus a potential rationale for utilizing expressive approximations is that they improve generalization to held-out data in comparison to increasing the encoder capacity.

We highlight that, in many scenarios, efficient test time inference is not required and consequently, encoder overfitting is not an issue, since we can use non-efficient encoder-independent methods to estimate $\log p(x)$, such as AIS, IWAE with local optimization, or potentially retraining the encoder on the held-out data. In contrast, when efficient test time inference is required, encoder generalization is important and expressive approximations are likely advantageous.

5.6. Annealing the Entropy

Typical warm-up (Bowman et al., 2015; Sønderby et al., 2016) refers to annealing the KL ($q(z|x)||p(z)$) term during training. This can also be interpreted as performing maximum likelihood estimation (MLE) early on during training. This optimization technique is known to help prevent the latent variable from degrading to the prior (Burda et al., 2016; Sønderby et al., 2016). We employ a similar annealing scheme during training by annealing the entropy of the

approximate distribution:

$$\mathbb{E}_{z \sim q(z|x)} [\log p(x, z) - \lambda \log q(z|x)],$$

where λ is annealed from 0 to 1 over training. This can be interpreted as *maximum a posteriori* (MAP) in the initial phase of training.

We find that warm-up techniques, such as annealing the entropy, are important for allowing the true posterior to be more complex. Table 3 (right) are results from a model trained without the entropy annealing schedule. Comparing these results to Table 2, we observe that the difference between $\mathcal{L}_{\text{VAE}}[q_{FFG}^*]$ and $\mathcal{L}_{\text{VAE}}[q_{AF}^*]$ is significantly smaller without entropy annealing. This indicates that the true posterior is more Gaussian when entropy annealing is not used. This suggests that, in addition to preventing the latent variable from degrading to the prior, entropy annealing allows the true posterior to better utilize the flexibility of the expressive approximation.

6. Conclusion

In this paper, we investigated how encoder capacity, approximation choice, decoder capacity, and model optimization influence inference suboptimality in terms of the approximation and amortization gaps. We discovered that the amortization gap can be a leading source to inference suboptimality and that the generator can reduce the approximation gap by learning a true posterior that fits to the choice of approximation. We showed that the parameters used to increase the expressiveness of the approximation play a role in generalizing inference rather than simply improving the complexity of the approximation. We confirmed that increasing the capacity of the encoder reduces the amortization error. Additionally, we demonstrated that optimization techniques, such as entropy annealing, help the generative model to better utilize the flexibility of expressive variational distributions. Analyzing these gaps can be useful for guiding improvements in VAEs. Future work includes evaluating other types of expressive approximations, more complex likelihood functions, and datasets.

References

- Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. Generating Sentences from a Continuous Space. *ArXiv e-prints*, November 2015.
- Burda, Y., Grosse, R., and Salakhutdinov, R. Importance weighted autoencoders. *In ICLR*, 2016.
- Clevert, Djork-Arné, Unterthiner, Thomas, and Hochreiter, Sepp. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

- Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using Real NVP. *ICLR*, 2017.
- Geweke, John. Bayesian inference in econometric models using monte carlo integration. *Econometrica: Journal of the Econometric Society*, pp. 1317–1339, 1989.
- Glorot, Xavier and Bengio, Yoshua. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256, 2010.
- Gomez, Aidan N, Ren, Mengye, Urtasun, Raquel, and Grosse, Roger B. The reversible residual network: Back-propagation without storing activations. In *Advances in Neural Information Processing Systems*, pp. 2211–2221, 2017.
- Grosse, R., Ghahramani, Z., and Adams, R. P. Sandwiching the marginal likelihood using bidirectional monte carlo. *arXiv preprint arXiv:1511.02543*, 2015.
- Grosse, Roger B, Ancha, Siddharth, and Roy, Daniel M. Measuring the reliability of mcmc inference with bidirectional monte carlo. In *Advances in Neural Information Processing Systems*, pp. 2451–2459, 2016.
- Hoffman, Matthew D. Learning deep latent gaussian models with markov chain monte carlo. In *International Conference on Machine Learning*, pp. 1510–1519, 2017.
- Hoffman, Matthew D, Blei, David M, Wang, Chong, and Paisley, John. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.
- Jarzynski, C. Nonequilibrium equality for free energy differences. *Physical Review Letters*, 78(14):2690, 1997.
- Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Kingma, D.P. and Welling, M. Auto-Encoding Variational Bayes. In *ICLR*, 2014.
- Kingma, D.P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improving Variational Inference with Inverse Autoregressive Flow. *NIPS*, 2016.
- Krishnan, R. G., Liang, D., and Hoffman, M. On the challenges of learning with inference networks on sparse, high-dimensional data. *ArXiv e-prints*, October 2017.
- Krizhevsky, Alex and Hinton, Geoffrey. Learning multiple layers of features from tiny images. *University of Toronto*, 2009.
- Larochelle, Hugo and Bengio, Yoshua. Classification using discriminative restricted boltzmann machines. In *Proceedings of the 25th international conference on Machine learning*, pp. 536–543. ACM, 2008.
- LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.
- Maaløe, L., Sønderby, CK., Sønderby, SK., and Winther, O. Auxiliary Deep Generative Models. *ICML*, 2016.
- Neal, R.M. Annealed importance sampling. *Statistics and Computing*, 2001.
- Ranganath, R., Tran, D., and Blei, D. M. Hierarchical Variational Models. *ICML*, 2016.
- Rezende, D.J. and Mohamed, S. Variational Inference with Normalizing Flows. In *ICML*, 2015.
- Rezende, D.J., Mohamed, S., and Wierstra, D. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *ICML*, 2014.
- Salakhutdinov, R. and Murray, I. On the quantitative analysis of deep belief networks. In *Proceedings of the 25th international conference on Machine learning*, pp. 872–879. ACM, 2008.
- Salimans, T., Kingma, D.P., and Welling, M. Markov chain monte carlo and variational inference: Bridging the gap. In *ICML*, 2015.
- Sønderby, Casper Kaae, Raiko, Tapani, Maaløe, Lars, Sønderby, Søren Kaae, and Winther, Ole. Ladder variational autoencoders. In *Advances in Neural Information Processing Systems*, pp. 3738–3746, 2016.
- Tomczak, J. M. and Welling, M. Improving Variational Auto-Encoders using Householder Flow. *ArXiv e-prints*, November 2016.
- Tomczak, J. M. and Welling, M. Improving Variational Auto-Encoders using convex combination linear Inverse Autoregressive Flow. *ArXiv e-prints*, June 2017.
- Turner, R and Sahani, M. Two problems with variational expectation maximisation for time-series models. inference and learning in dynamic models. *Cambridge University Press*, pp. 104–123, 2011.
- Wu, Y., Burda, Y., Salakhutdinov, R., and Grosse, R. On the Quantitative Analysis of Decoder-Based Generative Models. *ICLR*, 2017.
- Xiao, Han, Rasul, Kashif, and Vollgraf, Roland. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. github.com/zalando-research/fashion-mnist, 2017.

Supplementary

Code for the experiments in this paper can be found at: <https://github.com/chriscremer/Inference-Suboptimality> and <https://github.com/lxuechen/inference-suboptimality>.

6.1. Model Architectures and Training Hyperparameters

6.1.1. 2D VISUALIZATION

The VAE model of Fig. 2 uses a decoder $p(x|z)$ with architecture: $2 - 100 - 784$, and an encoder $q(z|x)$ with architecture: $784 - 100 - 4$. We use tanh activations and a batch size of 50. The model is trained for 3000 epochs with a learning rate of 10^{-4} using the ADAM optimizer (Kingma & Ba, 2014).

6.1.2. MNIST & FASHION-MNIST

Both MNIST and Fashion-MNIST consist of a training and test set with 60000 and 10000 datapoints respectively, where each datapoint is a 28×28 grey-scale image. We rescale the original images so that pixel values are within the range $[0, 1]$. For MNIST, We use the statically binarized version described by (Larochelle & Bengio, 2008). We also binarize Fashion-MNIST *statically*. For both datasets, we adopt the Bernoulli likelihood for the generator.

The VAE models for MNIST and Fashion-MNIST experiments have the same architecture. The encoder has two hidden layers with 200 units each. The activation function is chosen to be the exponential linear unit (ELU, Clevert et al. (2015)), as we observe improved performance compared to tanh. The latent space has 50 dimensions. The generator is the reverse of the encoder. We follow the same learning rate schedule and train for the same amount of epochs as described by (Burda et al., 2016). All models are trained with the a batch-size of 100 with ADAM.

In the large encoder setting, we change the number of hidden units for the inference network to be 500, instead of 200. The warm-up models are trained with a linear schedule over the first 400 epochs according to Section 5.6.

The auxiliary variable of requires a couple distributions: $q(v_0|z_0)$ and $r(v_T|z_T)$. These distributions are both factorized Gaussians which are parameterized by MLP's with two hidden layers, 100 units each, with ELU activations.

The flow transformation $q(z_{t+1}, v_{t+1}|z_t, v_t)$ involves functions σ_1 , σ_2 , μ_1 , and μ_2 from Eqn. 9 and 10. These also have two hidden layers with 100 units each and ELU units.

6.1.3. 3-BIT CIFAR

CIFAR-10 consists of a training and test dataset with 50000 and 10000 datapoints respectively, where each datapoint is a 32×32 RGB image. We rescale individual pixel values to be in the range $[0, 1]$. We then statically binarize the scaled pixel values by setting individual pixel values of channels to 1 if the rescaled value is greater than 0.5 and 0 otherwise. In this manner, we can model the observation with a factorized Bernoulli likelihood. We call this binarized CIFAR-10 dataset as *3-BIT CIFAR*, since 3 bits are required to encode each pixel, where 1 bit is needed for each of the channels. We acknowledge that such binarization scheme may reduce the complexity of the original problem, since originally 24 bits were required to encode a single pixel. Nevertheless, the 3-bit CIFAR dataset is still much more challenging compared MNIST and Fashion. This is because 784 bits are required to encode one MNIST/Fashion image, whereas for one 3-bit CIFAR image, 3072 bits are required. Most notably, we were able to validate our AIS estimates using BDMC with the simplified dataset. This, however, was not achievable in any reasonable amount of time with the original CIFAR-10 dataset.

For the latent variable, we use a 50-dimensional factorized Gaussian for $q(z|x)$. For all neural networks, ELU is chosen to be the activation function. The inference network consists of three 4 by 4 convolution layers with stride 2, batch-norm, and 64, 128, 256 channels respectively. Then a fully-connected layer outputs the 50-dimensional mean and log-variance of the latent variable. Similarly, the generator consists of a fully-connected layer outputting 256 by 2 by 2 tensors. Then three deconvolutional layers each with 4 by 4 filters, stride 2, batch-norm, and 128, 64, and 3 channels respectively. For the model with expressive inference, we use three normalizing flow steps, where the parametric functions in the flow and auxiliary variable distribution also take in a hidden layer of the encoder.

We use a learning rate of 10^{-3} . Warm-up is applied with a linear schedule over the first 50 epochs. All models are trained with a batch-size of 100 with ADAM. Early-stopping is applied based on the performance computed with the IWAE bound ($k=1000$) on the held-out set of 5000 examples from the original training set.

6.2. Inference Generalization

These models are trained with batch size 50 and latent dimension size of 20. The rest of the hyperparameters are equivalent to Section 6.1.2.

Architecture of q_{Flow} : The flow transformation involves functions σ_1 , σ_2 , μ_1 , and μ_2 from Eqn. 9 and 10. Each function is an MLP with a 50 unit hidden layer and ELU activations. We apply this flow transformation twice.

Fig. 4 are the plots for the q_{AF} model. The transformations are the same as q_{Flow} , but rather than partitioning the latent variable, we introduce an auxiliary variable. The auxiliary variable also requires a reverse model $r(v|z)$ which is a factorized Gaussian parameterized by an MLP with a 50 unit hidden layer and ELU activations.

Comparing AF in Fig. 4 to Flow in Fig. 3, we see that the AF has a larger approximation gap. This increase is likely due to the KL ($q(v|z, x) || r(v|x, z)$) term of the auxiliary variable lower bound from 2.2.2. This motivates also using expressive approximations for the reverse model $r(v|z)$.

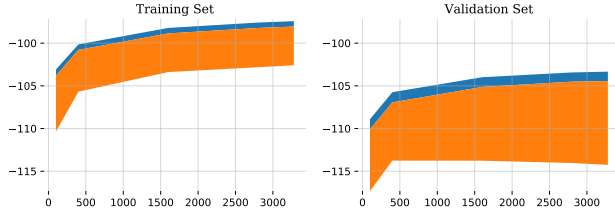


Figure 4. Gaps over epochs of the AF (auxiliary flow) model.

6.3. Influence of Flows On Amortization Gap Experiment

The aim of this experiment is to show that the parameters used for increasing the expressiveness of the approximation also contribute to reducing the amortization error. To show this, we train a VAE on MNIST, discard the encoder, then retrain two encoders on the fixed decoder: one with a factorized Gaussian distribution and the other with a parameterized 'flow' distribution. We use fixed decoder so that the true posterior is constant for both encoders. See 5.3 for the results and below for the architecture details.

The architecture of the decoder is: $D_Z - 200 - 200 - D_X$. The architecture of the encoder used to train the decoder is $D_X - 200 - 200 - 2D_Z$. The approximate distribution $q(z|x)$ is a factorized Gaussian.

Next, we describe the encoders which were trained on the fixed trained decoder. In order to highlight a large amortization gap, we employed a very small encoder architecture: $D_X - 2D_Z$. This encoder has no hidden layers, which

greatly impoverishes its ability and results in a large amortization gap.

We compare two approximate distributions $q(z|x)$. Firstly, we experiment with the typical fully factorized Gaussian (FFG). The second is what we call a flow distribution. Specifically, we use the transformations of (Dinh et al., 2017). We also include an auxiliary variable so we don't need to select how to divide the latent space for the transformations. The approximate distribution over the latent z and auxiliary variable v factorizes as: $q(z, v|x) = q(z|x)q(v)$. The $q(v)$ distribution is simply a $N(0,1)$ distribution. Since we're using an auxiliary variable, we also require the $r(v|z)$ distribution which we parameterize as $r(v|z): [D_Z] - 50 - 50 - 2D_Z$. The flow transformation is the same as in Section 3.2, which we apply twice.

6.4. Computation of the Determinant for Flow

The overall mapping f that performs $(z, v) \mapsto (z', v')$ is the composition of two sheer mappings f_1 and f_2 that respectively perform $(z, v) \mapsto (z, v')$ and $(z, v') \mapsto (z', v')$. Since the Jacobian of either one of the sheer mappings is diagonal, the determinant of the composed transformation's Jacobian Df can be easily computed:

$$\begin{aligned} \det(Df) &= \det(Df_1)\det(Df_2) \\ &= \left(\prod_{i=1}^n \sigma_1(z)_i\right) \left(\prod_{j=1}^n \sigma_2(v')_j\right). \end{aligned}$$

6.5. Annealed Importance Sampling

Annealed importance sampling (AIS, Neal (2001); Jarzynski (1997)) is a means of computing a lower bound to the marginal log-likelihood. Similarly to the importance weighted bound, AIS must sample a proposal distribution $f_1(z)$ and compute the density of these samples, however, AIS then transforms the samples through a sequence of reversible transitions $\mathcal{T}_t(z'|z)$. The transitions anneal the proposal distribution to the desired distribution $f_T(z)$.

Specifically, AIS samples an initial state $z_1 \sim f_1(z)$ and sets an initial weight $w_1 = 1$. For the following annealing steps, z_t is sampled from $\mathcal{T}_t(z'|z)$ and the weight is updated according to:

$$w_t = w_{t-1} \frac{f_t(z_{t-1})}{f_{t-1}(z_{t-1})}.$$

This procedure produces weight w_T such that $\mathbb{E}[w_T] = Z_T/Z_1$, where Z_T and Z_1 are the normalizing constants of $f_T(z)$ and $f_1(z)$ respectively. This pertains to estimating the marginal likelihood when the target distribution is $p(x, z)$ when we integrate with respect to z .

Typically, the intermediate distributions are simply defined to be geometric averages: $f_t(z) = f_1(z)^{1-\beta_t} f_T(z)^{\beta_t}$,

where β_t is monotonically increasing with $\beta_1 = 0$ and $\beta_T = 1$. When $f_1(z) = p(z)$ and $f_T(z) = p(x, z)$, the intermediate distributions are: $f_i(x) = p(z)p(x|z)^{\beta_i}$.

Model evaluation with AIS appears early on in the setting of deep belief networks (Salakhutdinov & Murray, 2008). AIS for decoder-based models was also used by Wu et al. (2017).

6.6. Extra MNIST Inference Gaps

To demonstrate that a very small inference gap can be achieved, even with a limited approximation such as a factorized Gaussian, we train the model on a small dataset. In this experiment, our training set consists of 1000 datapoints randomly chosen from the original MNIST training set. The training curves on this small dataset are shown in Fig. 5. Even with a factorized Gaussian distribution, the inference gap is very small: the AIS and IWAE bounds are overlapping and the VAE is just slightly below. Yet, the model is overfitting as seen by the decreasing test set bounds.

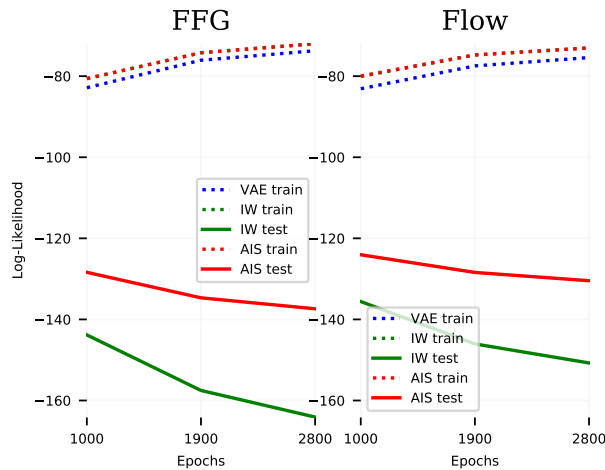


Figure 5. Training curves for a FFG and a Flow inference model on MNIST. AIS provides the tightest lower bound and is independent of encoder overfitting. There is little difference between FFG and Flow models trained on the 1000 datapoints since inference is nearly equivalent.