Inference Suboptimality in Variational Autoencoders

Chris Cremer
Department of Computer Science
University of Toronto
ccremer@cs.toronto.edu

Xuechen Li
Department of Computer Science
University of Toronto
lxuechen@cs.toronto.edu

David Duvenaud
Department of Computer Science
University of Toronto
duvenaud@cs.toronto.edu

Abstract

Amortized inference has led to efficient approximate inference for large datasets. The quality of posterior inference is largely determined by the ability of the variational distribution to model the true posterior as well as the capacity of the recognition network to generalize inference over all datapoints. We analyze inference in variational autoencoders by separating these factors. This analysis can be useful for guiding improvements to inference. Our experiments also illustrate how the choice of the approximate posterior influences the trained model.

1 Introduction

There has been significant work on improving inference in variational autoencoders (VAEs) [12, 19] through the development of expressive approximate posteriors [18, 11, 17, 23, 24]. These works have shown that with more expressive approximations, the model learns better distributions over the data.

In this paper, we examine the sources that contribute to the mismatch between the true and approximate posterior. We refer to this mismatch as the *inference gap*. Moreover, we break down the gap into two components: the *approximation gap* and the *amortization gap*. The approximation gap comes from the inability of the approximate 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 datapoint independently.

To understand what roles these gaps play in inference, we train and evaluate the gaps of VAE models in a number of settings including different approximate posterior families, different encoder sizes, and on the MNIST and Fashion-MNIST [26] datasets.

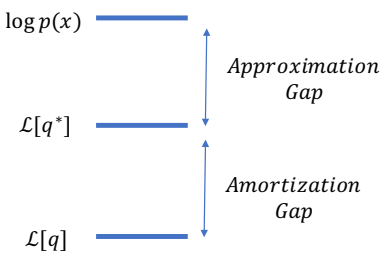


Figure 1: Gaps in Inference

Our contributions are: a) we introduce the amortization and approximation gaps and study how they contribute to the inference gap, providing a tool to guide future improvements in VAE inference, and b) we quantitatively demonstrate the influence that the expressiveness of the approximate posterior has on the trained true posterior as well as the role that warm-up plays in the optimization.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

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

Table 1: Summary of Gap Terms. The middle column is the general case where the variational objective is a lower bound on the marginal log-likelihood. The right most is the specific case for VAEs. $q^*(z|x)$ refers to the optimal approximation within a family of distributions (e.g. Gaussian Family) $\mathcal{Q}$.

2 The Inference Gap

2.1 Variational Autoencoders

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

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

In practice, the marginal log-likelihood is computationally intractable due to the integration over the latent variable z. Instead, VAEs optimize the *evidence lower bound* (ELBO) [12, 19]:

$$\log p(x) = \mathbb{E}_{z\sim q(z|x)}\left[\log\left(\frac{p(x,z)}{q(z|x)}\right)\right] + \mathrm{KL}\big(q(z|x)||p(z|x)\big) \tag{2}$$

$$\geq \mathbb{E}_{z\sim q(z|x)}\left[\log\left(\frac{p(x,z)}{q(z|x)}\right)\right] = \mathcal{L}_{\text{VAE}}[q]. \tag{3}$$

2.2 The 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 variational family that maximizes the lower bound, $q^*(z|x) = \arg\max_{q\in\mathcal{Q}} \mathcal{L}[q]$, the inference gap can be decomposed as the sum of the approximation and amortization gaps:

$$\mathcal{G} = \log p(x) - \mathcal{L}[q] = \underbrace{\log p(x) - \mathcal{L}[q^*]}_{\text{Approximation}} + \underbrace{\mathcal{L}[q^*] - \mathcal{L}[q]}_{\text{Amortization}}. \tag{4}$$

For VAEs, we can translate the gaps to KL divergences by rearranging (2):

$$\mathcal{G}_{\text{VAE}} = \underbrace{\mathrm{KL}\big(q^*(z|x)||p(z|x)\big)}_{\text{Approximation}} + \underbrace{\mathrm{KL}\big(q(z|x)||p(z|x)\big) - \mathrm{KL}\big(q^*(z|x)||p(z|x)\big)}_{\text{Amortization}}. \tag{5}$$

See Table 1 for a summary of these gaps. To obtain q^*, we optimize the parameters of the variational distribution for each datapoint independently.

3 Methods

3.1 Flexible Approximate Posterior

Our experiments compare two families of approximate posteriors: the fully-factorized Gaussian (FFG) and a flexible flow (Flow). Our choice of flow is a combination of the Real NVP [6] and auxiliary variables [17, 14]. Let $z \in \mathbb{R}^n$ be the variable of interest and $v \in \mathbb{R}^n$ the auxiliary variable. Each flow step involves:

$$v' = v \circ \sigma_1(z) + \mu_1(z)\,,\;\; z' = z \circ \sigma_2(v') + \mu_2(v') \tag{6}$$

where $\sigma_1, \sigma_2, \mu_1, \mu_2 : \mathbb{R}^n \to \mathbb{R}^n$ are differentiable mappings parameterized by neural nets and $\circ$ takes the Hadamard or element-wise product. Further details are given in the Appendix.

Dataset	MNIST				Fashion MNIST			
Variational Family	q_{FFG}	q_{FFG}	q_{Flow}	q_{Flow}	q_{FFG}	q_{FFG}	q_{Flow}	q_{Flow}
Encoder Capacity	Regular	Large	Regular	Large	Regular	Large	Regular	Large
$\log \hat{p}(x)$	-89.85	-89.09	-88.82	-88.59	-97.78	-94.55	-97.35	-96.16
$\mathcal{L}_{\text{VAE}}[q^*_{Flow}]$	-90.83	-90.05	-90.40	-90.26	-98.03	-95.93	-97.74	-96.87
$\mathcal{L}_{\text{VAE}}[q^*_{FFG}]$	-91.19	-90.34	-102.88	-103.53	-99.03	-98.09	-130.90	-129.24
$\mathcal{L}_{\text{VAE}}[q]$	-92.76	-91.12	-91.42	-91.25	-103.20	-101.28	-102.19	-100.60
Approximation	1.34	1.25	1.58	1.67	1.25	3.54	0.39	0.71
Amortization	1.57	0.78	1.02	0.99	4.17	3.19	4.45	3.73
Inference Gap	2.91	2.03	2.60	2.66	5.42	6.73	4.84	4.44

Table 2: Inference Gaps. The variational family specifies the approximate distribution used for training the model. Increasing the encoder capacity reduces the amortization gap. All numbers are in nats.

3.2 Evaluation Bounds

We test two different variational distributions q: the fully-factorized Gaussian q_{FFG} and the flexible approximation q_{Flow} described above. $\mathcal{L}_{\text{VAE}}[q]$ refers to the bound of Eqn. 3, with q being either q_{FFG} or q_{Flow}.

Similarly, for the locally optimized bound $\mathcal{L}_{\text{VAE}}[q^*]$, q^* is either q^*_{FFG} or q^*_{Flow}, where we optimize the variational parameters for each distribution. Specifically, for the Optimal-FFG variational distribution (q^*_{FFG}), we optimize the mean and variance for each datapoint. For the Optimal Flow (q^*_{Flow}), we optimize the parameters of the flow and auxiliary variable networks for each datapoint.

We approximate the marginal log-likelihood $\log p(x)$, denoted $\log \hat{p}(x)$, by $\max(\mathcal{L}_{\text{AIS}}, \mathcal{L}_{\text{IWAE}}[q^*])$. For $\mathcal{L}_{\text{IWAE}}[q^*]$, we use 5000 importance samples. For $\mathcal{L}_{\text{AIS}}$, we use Hamiltonian Monte Carlo [16] as our transition operator $\mathcal{T}_t(z'|z)$. We use 100 samples, 10 HMC steps, and 500 intermediate distributions. The initial distribution is the prior, so that it is encoder-independent.

4 Experimental Results

4.1 Amortization vs Approximation Gap

We compare the effect on inference that amortization has compared to the posterior approximation. The evaluation is performed on a subset of 100 datapoints from the training dataset. Table 2 are results from training on MNIST and Fashion MNIST. For MNIST, both the amortization and approximation gaps contribute roughly equally to the inference gap. For the Flow-trained model, although the inference gap is slightly smaller than the FFG model, a larger portion of its inference gap comes from the approximation gap. On Fashion MNIST, the amortization gap is often much larger than the approximation gap for our model setting, indicating that inference in this model could benefit from a encoder with larger capacity.

How does increasing the capacity of the encoder affect the gaps? We train the same models with encoders that have more modeling capacity (referred to as large encoder). The columns of Table 2 with *Large* encoder capacity are the results of this experiment. We see that the inference gap of the FFG-trained model decreases mainly due to a reduction in the amortization gap. For the Flow-trained model, increasing the size of the encoder is less beneficial, however there is still a decrease in the amortization gap.

4.2 Influence of Approximate Posterior on True Posterior

To what extent does the approximation affect the true posterior? Will a fully-factorized Gaussian approximation cause the true posterior to be more FFG? Just as it is hard to evaluate a generative model by visually inspecting samples from the model, it is hard to say how Gaussian the true posterior is by visual inspection. We can quantitatively determine how close the posterior is to an FFG distribution by comparing the Optimal FFG bound $\mathcal{L}_{\text{VAE}}[q^*_{FFG}]$, and the Optimal Flow bound $\mathcal{L}_{\text{VAE}}[q^*_{Flow}]$.

In Table 2 on MNIST, the Optimal Flow $\mathcal{L}_{\text{VAE}}[q^*_{Flow}]$ improves upon the Optimal FFG $\mathcal{L}_{\text{VAE}}[q^*_{FFG}]$ for the FFG trained model by 0.36 nats. In contrast, on the Flow-trained model, the difference increases to 12.48 nats. This suggests that the true posterior of an FFG-trained model is more Gaussian than the true posterior of the Flow-trained model. The same observation can be made on the Fashion MNIST dataset.

These observations imply that the decoder learns to have a true posterior that fits better to the approximation. Although the generative model can learn to have a posterior that fits to the approximation, it seems that not having this constraint, i.e. using a flexible approximation, results in better generative models.

4.2.1 Annealing the Entropy

Typical warm-up [2, 22] refers to annealing $\text{KL}(q(z|x)||p(z))$ 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 [3, 22]. We employ a similar annealing during training. Rather than annealing the KL, we anneal the entropy of the approximate distribution q:

$$\mathbb{E}_{z\sim q(z|x)}\left[\log p(x,z) - \lambda \log q(z|x)\right], \tag{7}$$

where λ is annealed from 0 to 1 over training. This can be interpreted as *maximum a posteriori* (MAP) in the initial phase. Due to its similarity, we will also refer to this technique as warm-up.

We find that warm-up is very important for allowing the true posterior to be more complex. Table 3 are results from a model trained without the warm-up schedule. The difference between the Optimal Flow and Optimal FFG for the Flow-trained model is 1.56 nats on MNIST. In contrast, when trained with warm-up in Table 2, the difference is 12.48 nats. Thus, the difference between the Optimal Flow and Optimal FFG is significantly smaller without warmup than with warmup. This suggests that, in addition to preventing the latent variable from degrading to the prior, warmup allows the true posterior to better utilize the flexibility of the expressive approximation, resulting in a better trained model.

Dataset	MNIST		Fashion MNIST	
Variational Family	q_{FFG}	q_{Flow}	q_{FFG}	q_{Flow}
$\log \hat{p}(x)$	-89.84	-89.41	-100.56	-100.33
$\mathcal{L}_{\text{VAE}}[q^*_{Flow}]$	-90.89	-90.78	-100.74	-100.61
$\mathcal{L}_{\text{VAE}}[q^*_{FFG}]$	-91.02	-92.34	-101.26	-102.80
$\mathcal{L}_{\text{VAE}}[q]$	-94.31	-94.29	-103.63	-104.05
Approximation Gap	1.18	1.37	0.70	0.28
Amortization Gap	3.29	3.51	2.37	3.44
Inference Gap	4.47	4.88	3.07	3.72

Table 3: Without warm-up. On the model trained with q_{Flow}, $\mathcal{L}_{\text{VAE}}[q^*_{FFG}]$ and $\mathcal{L}_{\text{VAE}}[q^*_{Flow}]$ are relatively close compared to the setting with warm-up in Fig. 2. All numbers are in nats.

5 Concluding Remarks

In this paper, we broke down the inference gap into two parts: the amortization and approximation gaps. We explored how different posterior approximations and different encoder capacities affect these gaps. We confirmed that increasing the capacity of the encoder reduced the gap in amortization. These gaps could serve for diagnosing inference in VAEs. We also demonstrated the effect that the posterior approximation had on the true posterior by observing how well approximations fit to models trained with different variational distributions. We also improved our understanding of how warm-up helps optimization, by demonstrating that warm-up allows the generative model to better utilize the flexibility of the variational distribution. Future work includes evaluating other types of expressive approximations and more complex likelihood functions.

Acknowledgments

We would like to thank Roger Grosse for useful and insightful discussions.

References

[1] P. Bachman and D. Precup. Training Deep Generative Models: Variations on a Theme. *NIPS Approximate Inference Workshop*, 2015.

[2] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio. Generating Sentences from a Continuous Space. *ArXiv e-prints*, Nov. 2015.

[3] Y. Burda, R. Grosse, and R. Salakhutdinov. Importance weighted autoencoders. *In ICLR*, 2016.

[4] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[5] C. Cremer, Q. Morris, and D. Duvenaud. Reinterpreting Importance-Weighted Autoencoders. *ICLR Workshop*, 2017.

[6] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using Real NVP. *ICLR*, 2017.

[7] R. Grosse, Z. Ghahramani, and R. P. Adams. Sandwiching the marginal likelihood using bidirectional monte carlo. *arXiv preprint arXiv:1511.02543*, 2015.

[8] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

[9] C. Jarzynski. Nonequilibrium equality for free energy differences. *Physical Review Letters*, 78(14):2690, 1997.

[10] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[11] D. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improving Variational Inference with Inverse Autoregressive Flow. *NIPS*, 2016.

[12] D. Kingma and M. Welling. Auto-Encoding Variational Bayes. *In ICLR*, 2014.

[13] R. G. Krishnan, D. Liang, and M. Hoffman. On the challenges of learning with inference networks on sparse, high-dimensional data. *ArXiv e-prints*, Oct. 2017.

[14] L. Maaløe, C. Sønderby, S. Sønderby, and O. Winther. Auxiliary Deep Generative Models. *ICML*, 2016.

[15] R. Neal. Annealed importance sampling. *Statistics and Computing*, 2001.

[16] R. Neal. MCMC using hamiltonian dynamics. *Hand- book of Markov Chain Monte Carlo*, 2011.

[17] R. Ranganath, D. Tran, and D. M. Blei. Hierarchical Variational Models. *ICML*, 2016.

[18] D. Rezende and S. Mohamed. Variational Inference with Normalizing Flows. *In ICML*, 2015.

[19] D. Rezende, S. Mohamed, and D. Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *ICML*, 2014.

[20] R. Salakhutdinov and H. Larochelle. Efficient learning of deep boltzmann machines. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 693–700, 2010.

[21] T. Salimans, D. Kingma, and M. Welling. Markov chain monte carlo and variational inference: Bridging the gap. *In ICML*, 2015.

[22] C. Sønderby, T. Raiko, L. Maaløe, S. Kaae Sønderby, and O. Winther. Ladder Variational Autoencoders. *CONF*, 2016.

[23] J. M. Tomczak and M. Welling. Improving Variational Auto-Encoders using Householder Flow. *ArXiv e-prints*, Nov. 2016.

[24] J. M. Tomczak and M. Welling. Improving Variational Auto-Encoders using convex combination linear Inverse Autoregressive Flow. *ArXiv e-prints*, June 2017.

[25] Y. Wu, Y. Burda, R. Salakhutdinov, and R. Grosse. On the Quantitative Analysis of Decoder-Based Generative Models. *ICLR*, 2017.

[26] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *github.com/zalandoresearch/fashion-mnist*, 2017.

Appendix

6 Intuition through Visualization

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

Posterior A is an example of a distribution where FFG can fit well. Posterior B is an example of 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 the approximate distribution the leading cause of the distribution mismatch, or is it the approximation?

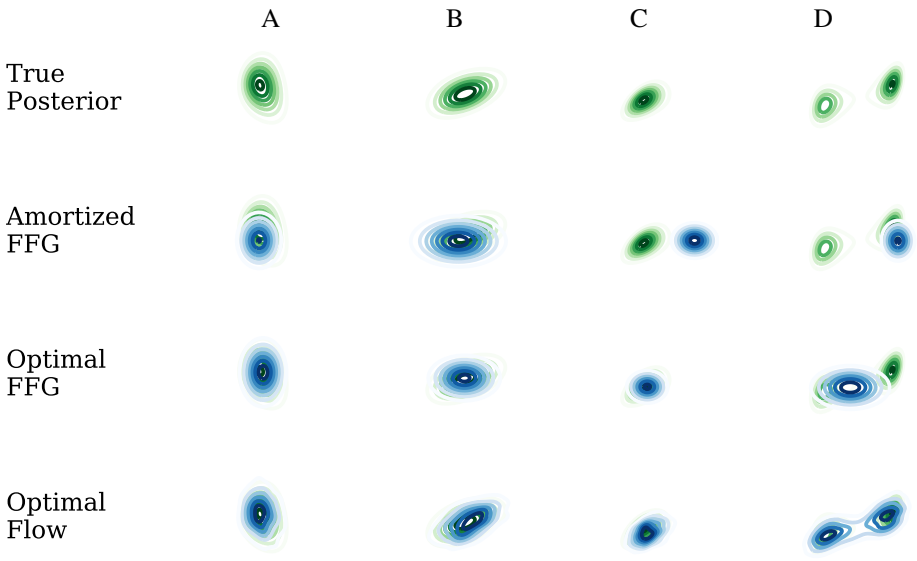


Figure 2: True Posterior and Approximate Distributions of a VAE with 2D latent space. Columns: 4 different datapoints. FFG: Fully-factorized Gaussian. Flow: Using a flexible approximate distribution. Amortized: Using amortized parameters. Optimal: Parameters optimized for individual datapoints. The green distributions are the true posterior distributions, highlighting the mismatch with the approximation.

7 Background

7.1 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.

7.1.1 Normalizing Flows

Normalizing flow [18] 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:

$$q_T(z_T) = q_0(z_0) \left| \det \frac{\partial z_T}{\partial z_0} \right|^{-1} \tag{8}$$

By successively applying these transformations, we can build arbitrarily complex distributions. Stacking these transformations remains tractable due to the determinant being decomposable: $\det(A \cdot B) = \det(A)\det(B)$. 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:

$$\mathbb{E}_{q_T}[h(z_T)] = \mathbb{E}_{q_0}[h(f_T(f_{T-1}(...f_1(z_0))))] \tag{9}$$

This is known as the law of the unconscious statistician (LOTUS). Using the change of variable and LOTUS, the lower bound can be written as:

$$\log p(x) \geq \mathbb{E}_{z_0 \sim q_0(z|x)} \left[\log \left(\frac{p(x, z_T)}{q_0(z_0|x) \prod_{t=1}^{T} \left| \det \frac{\partial z_t}{\partial z_{t-1}} \right|^{-1}} \right) \right]. \tag{10}$$

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

7.1.2 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:

$$\log p(x) \geq \mathbb{E}_{z,v \sim q(z,v|x)} \left[\log \left(\frac{p(x, z) r(v|x, z)}{q(z|x, v) q(v|x)} \right) \right] \tag{11}$$

$$= \mathbb{E}_{q(z|x)} \left[\log \left(\frac{p(x, z)}{q(z|x)} \right) - \mathrm{KL}\Big(q(v|z, x) \| r(v|x, z) \Big) \right] \tag{12}$$

where $r(v|x, z)$ is called the reverse model. From Eqn. 12, we see that this bound is looser than the regular ELBO, however the extra flexibility provided by the auxiliary variable can compensate and result in a higher lower bound. This idea has been employed in works such as auxiliary deep generative models (ADGM, [14]), hierarchical variational models (HVM, [17]) and Hamiltonian variational inference (HVI, [21]).

7.2 Evaluation Metrics

Here we describe two options for estimating the marginal log-likelihood of a model: IWAE and AIS.

7.2.1 IWAE Bound

The bound used in [3] is a tighter lower bound than the VAE bound. 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 [3, 11]. More specifically, if we take multiple samples from the q distribution, we can compute a tighter lower bound to the marginal log likelihood:

$$\log p(x) \geq \mathbb{E}_{z_1 \ldots z_k \sim q(z|x)} \left[\log \left(\frac{1}{k} \sum_{i=1}^{k} \frac{p(x, z_i)}{q(z_i|x)} \right) \right] = \mathcal{L}_{\text{IWAE}}[q]. \tag{13}$$

This importance weighted bound was introduced along with the Importance Weighted Autoencoder [3], so we will refer to it as the IWAE bound. As shown by [1] and [5], the IWAE bound can be seen as using the VAE bound but with an importance weighted q distribution.

7.2.2 Annealed Importance Sampling

Annealed importance sampling (AIS, [15, 9]) is another 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 $\mathbb{E}[w_T] = \mathcal{Z}_T/\mathcal{Z}_1$, 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 $p(x, z)$ and we are integrating with respect to z.

Typically, the intermediate distributions are simply defined to be geometric averages: $f_t(x) = f_1(x)^{1-\beta_t} f_T(x)^{\beta_t}$, where β_t is monotonically increasing with $\beta_1 = 0$ and $\beta_T = 1$. When $f_1(x) = p(z)$ and $f_T(x) = p(x, z)$, the intermediate distributions are: $f_i(x) = f_0(x) f_n(x)^{1-\beta_i}$.

7.3 Model Architectures and Training Hyperparameters

D_X refers to the dimensionality of the input and D_Z refers to the dimensionality of the latent space. For MNIST and Fashion MNIST, $D_X = 784$.

7.3.1 2D Visualization

The VAE model of Fig. 2 used a decoder $p(x|z)$ with architecture: $D_Z - 100 - D_X$. The encoder $q(z|x)$ was $D_X - 100 - 2D_Z$, where the latent space has dimensionality of 2. We used tanh activation functions and a batch size of 50. The model was trained for 3000 epochs with a learning rate of 0.0001.

7.3.2 MNIST

Here we describe the network architectures for the MNIST experiments. The decoder $p(x|z)$ is $D_Z - 200 - 200 - D_X$. The encoder $q(z|x)$ is $D_X - 200 - 200 - 2D_Z$, where the latent space has dimensionality of 50, so it outputs a mean and variance. For the large encoder experiment of Table 2, the encoder architecture is $D_X - 500 - 200 - 200 - 2D_Z$.

For flexible model, we use two flow steps where each step contains a number of other networks:
$q(v_0|x)$: $D_X - 200 - 2D_Z$
$q(z_0|v_0, x)$: $[D_X + D_Z] - 200 - 2D_Z$
$r(v_T|x)$: $[D_X + D_Z] - 200 - 2D_Z$

We use tanh for the activation functions and batch size of 100. We use the same learning rate schedule as [3]. We use the warm-up schedule described in 4.2.1, where the annealing is over the first 100 epochs.

7.3.3 Fashion-MNIST

Fashion-MNIST consists of a training and test dataset with 60k and 10k datapoints respectively. We rescale the original dataset so that pixel values are within the greyscale range, i.e. $[0, 1]$. We then binarize the whole dataset statically, so that our Bernoulli likelihood model $p(x|z)$ is valid.

The VAE model used to trained on Fashion-MNIST has the same architecture as that used to train on MNIST. All neural networks chosen to parameterize the flow have hidden layers consisting of 100 units as opposed to 200 units as in the MNIST setting:

$q(v_0|x)$: $D_X - 100 - 2D_Z$
$q(z_0|v_0, x)$: $[D_X + D_Z] - 100 - 2D_Z$
$r(v_T|x)$: $[D_X + D_Z] - 100 - 2D_Z$

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 were trained with a linearly increasing annealing constant over the first 400 epochs of training.

The activation function were chosen to be the exponential linear unit (ELU, [4]), as we observed improved performance with it over tanh. We follow the same learning rate schedule and train for the same amount of epochs as described by [3]. All models were trained with the ADAM optimizer [10].

8 Flexible Approximate Posterior

Our choice of flow is a combination of the Real NVP [6] and auxiliary variables [17, 14]. Let $z \in \mathbb{R}^n$ be the variable of interest and $v \in \mathbb{R}^n$ the auxiliary variable. Each flow step involves:

$$v' = v \circ \sigma_1(z) + \mu_1(z) \tag{14}$$

$$z' = z \circ \sigma_2(v') + \mu_2(v') \tag{15}$$

where $\sigma_1, \sigma_2, \mu_1, \mu_2 : \mathbb{R}^n \to \mathbb{R}^n$ are differentiable mappings parameterized by neural nets and $\circ$ takes the Hadamard or element-wise product. The determinant of the combined transformation's Jacobian, $\left|\det \frac{\partial z' v'}{\partial z v}\right|$, can be easily evaluated.

Thus, we can jointly train the generative and flow-based inference model by optimizing the bound:

$$\log p(x) \geq \mathbb{E}_{z_0, v_0 \sim q(z,v|x)} \left[\log \left(\frac{p(x, z_T) r(v_T|x, z_T)}{q(z_0|x, v_0) q(v_0|x) \prod_{t=1}^{T} \left|\det \frac{\partial z_t v_t}{\partial z_{t-1} v_{t-1}}\right|^{-1}} \right) \right] = \mathcal{L}_{flow}[q]. \tag{16}$$

Additionally, multiple such type of transformations can be stacked to improve expressiveness. Stacking these transformations remains tractable due to the fact that: $det(A \cdot B) = det(A)det(B)$

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 therefore be easily computed:

$$\det(Df) = \det(Df_1) \cdot \det(Df_2) = \Big(\prod_{i=1}^{n} \sigma_1(z)_i\Big)\Big(\prod_{j=1}^{n} \sigma_2(v')_j\Big).$$

9 Related Work

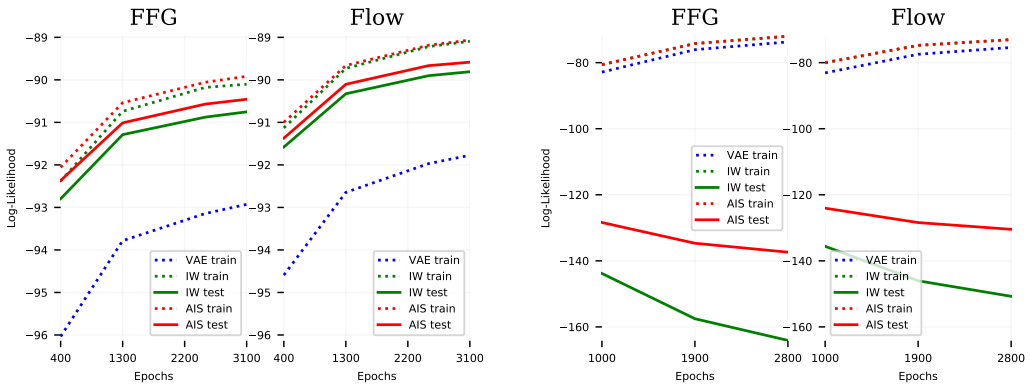
Model evaluation with AIS for decoder-based models was proposed by [25]. They validated the accuracy of the approach with Bidirectional Monte Carlo (BDMC, [7]) and demonstrated the advantage of using AIS over the IWAE bound for evaluation when the inference network overfits to the training data.

Much of the earlier work on variational inference focused on optimizing the variational parameters locally for each datapoint. [20] perform such local optimization when learning deep Boltzmann

machines. The original Stochastic Variational Inference scheme (SVI, [8]) also specifies the variational parameters to be optimized locally in the inner loop. [13] remark on two sources of error in variational learning with inference networks, and propose to optimize locally with 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. Yet, their work only dwells on reducing the amortization gap during training and does not touch on analyzing the approximation gap caused by using the Gaussian family.

Even though it is clear that failed inference would lead to a failed generative model, little quantitative assessment has been done showing the effect of the approximate posterior on the true posterior. [3] visually demonstrate that trained with an importance-weighted approximate posterior, the resulting true posterior is more complex than those trained with fully-factorized Gaussian approximations. We extend this observation quantitatively in the setting of flow-based approximate inference in VAEs.

10 VAE Inference



(a) Full dataset (b) 1000 datapoints

Figure 3: 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.

10.1 The Inference Gap

How well is inference done in VAEs during training? Are we close to doing the optimal or is there much room for improvement? To answer this question, we quantitatively measure the inference gap: the gap between the true marginal log-likelihood and the lower bound. This amounts to measuring how well inference is being done during training. Since we cannot compute the exact marginal log-likelihood, we estimate it using the maximum of any of its lower bounds, described in 3.2.

Fig. 3a shows training curves for a FFG and Flow inference network as measured by the VAE, IWAE, and AIS bounds on the training and test set. The inference gap on the training set with the FFG model is 3.01 nats, whereas the Flow-trained model is 2.71 nats. Accordingly, the plot shows that the training IWAE bound is much tighter for the Flow model compared to the FFG. Due to this lower inference gap during training, the Flow model achieves a higher AIS bound on the test set than the FFG model.

To demonstrate that strong inference can be achieved, even with FFG, we train a 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 datatset are show in Fig. 3b. Even with FFG, inference is so strong to the point that the AIS and IWAE bounds are overlapping. Yet, the model is overfitting as seen by the decreasing test set bounds.

10.1.1 Encoder and Decoder Overfitting

Decoder overfitting is the same as the regular supervised learning scenario, where we compare the train and test error. To measure decoder overfitting independently from encoder overfitting, we use the AIS bound since it is encoder-independent. Thus we can observe decoder overfitting through the AIS test training curve. In contrast, the encoder can only overfit in the sense that the recognition network becomes unsuitable for computing the marginal likelihood on the test set. Thus, encoder overfitting is computed by: $\mathcal{L}_{\text{AIS}} - \mathcal{L}_{\text{IW}}$ on the test set.

For the small dataset of Fig. 3b, it clear that there is significant encoder and decoder overfitting. A model trained in this setting would benefit from regularization. For Fig. 3a, the model is not overfit and would benefit from more training. However, there is some encoder overfitting due to the gap between the AIS and IWAE bounds on the test set. Comparing the FFG and Flow models, it appears that the Flow does not have a large effect on encoder or decoder overfitting.