Approximate Posterior Building Blocks

Chris Cremer

June 27 2017

1 Introduction

Consider a general probabilistic model of data x, latent variables z, and model parameters θ given by $p_\theta(x, z)$. Posterior inference in this model can be made efficient through latent variable reparameterization and inference networks. When using neural networks for both the inference network and generative model, the result is a class of models called variational autoencoders (VAEs, [Kingma and Welling, 2014]). VAEs maximize a lower bound of the marginal log likelihood, often referred to as the evidence lower bound (ELBO), which is derived in the following equations:

$$
\begin{aligned}
log\, p_\theta(x) &= log \int p_\theta(x|z)p(z)dz && (1)\\
&= log \left(\mathbb{E}_{q_\phi(z|x)} \left[\frac{p_\theta(x|z)p(z)}{q_\phi(z|x)} \right] \right) && (2)\\
&= \mathbb{E}_{q_\phi(z|x)} \left[log \left(\frac{p_\theta(x|z)p(z)}{q_\phi(z|x)} \right) \right] + KL(q_\phi(z|x) \| p(z|x)) && (3)\\
&\geq \mathbb{E}_{q_\phi(z|x)} \left[log \left(\frac{p_\theta(x|z)p(z)}{q_\phi(z|x)} \right) \right] && (4)\\
&= \mathbb{E}_{q_\phi(z|x)} \left[log \left(p_\theta(x|z) \right) \right] - KL(q_\phi(z|x) \| p(z)) && (5)
\end{aligned}
$$

where the inequality follows from the removal of the non-negative KL term. From Eqn. 3, we see that the closer the approximation $q_\phi(z|x)$ is to the true posterior $p(z|x)$, the tighter the bound will be.

In a typical VAE, the approximate posterior defined by the inference network is a factorized Gaussian. That is, the approximate posterior distribution is defined as $q_\phi(z|x) = N(\mu, \Sigma)$, where the covariance matrix Σ is diagonal. This approximation places strong assumptions on the model and there has been much work to improve upon it.

In this report, I review some popular themes for improving posterior approximations in latent variable models. I also consider how these themes could be combined and how it would affect their flexibility and computational complexity.

2 Richer Approximate Posteriors

In this section, I will review some current techniques for increasing the complexity of the posterior approximation of continuous latent variables, which include, importance weighting, change of variables, and auxiliary variables. Note that this is by no means an exhaustive list. Other techniques not discussed here include include improving the covariance matrix of Gaussian approximations and mixture distributions [Miller et al., 2016]. See the appendix for a summary of all the lower bounds described in this section.

2.1 Importance Weighting

One approach to improving the approximate posterior is to use importance weighting of multiple samples. More specifically, if we take multiple samples from the q distribution, we can compute a tighter lower bound to the marginal log likelihood:

$$log(p(x)) \geq \mathbb{E}_{z_1 \dots z_k \sim q(z|x)} \left[log \left(\frac{1}{k} \sum_{i=1}^{k} \frac{p(x, z_i)}{q(z_i|x)} \right) \right]. \tag{6}$$

This importance weighted bound was introduced in the Importance Weighted Autoencoder paper [Burda et al., 2016], so we will refer to it as the IWAE bound. As shown by [Bachman and Precup, 2015] and [Cremer et al., 2017], the IWAE bound can be seen as using the VAE bound but with an importance weighted q distribution. Through the use of multiple samples, the IWAE bound has the flexibility to learn generative models whose posterior distributions do not fit the original fixed q distribution. Importantly, this importance weighted lower bound approaches $log\, p(x)$ as k goes to infinity.

Furthermore, [Burda et al., 2016] observed that the IWAE bound increased the number of active dimensions in the latent space. Reducing inactive latent dimensions is important because they cause the model to only use a fraction of its full capacity. Another technique used to support stochastic units staying active is to initialize training using the reconstruction error only, and then gradually introducing the variational regularization.

2.2 Change of Variable (Normalizing Flows)

A change of variable procedure such as normalizing flows (NFs) is a tool for constructing complex distributions by transforming probability densities through a series of invertible mappings. More 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:

$$q_T(z_T) = q_0(z_0) \left| det \frac{\partial z_T}{\partial z_0} \right|^{-1} \tag{7}$$

By successively applying these transformations, we can build arbitrarily complex distributions. Stacking these transformations remains tractable due to the fact that: $det(A \cdot B) = det(A)det(B)$. 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 $q_T(z_T)$. The expectation can be written as:

$$\mathbb{E}_{q_T}[h(z_T)] = \mathbb{E}_{q_0}[h(f_T(f_{T-1}(...f_1(z_0))))] \tag{8}$$

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:

$$log(p(x)) \geq \mathbb{E}_{z_0 \sim q_0(z|x)} \left[log \left(\frac{p(x, z_T)}{q_0(z_0|x) \prod_{t=1}^{T} \left| det \frac{\partial z_t}{\partial z_{t-1}} \right|^{-1}} \right) \right]. \tag{9}$$

The main constraint on these transformations is that the determinant of their Jacobian (det-Jacobian) needs to be easily computable. Fortunately, there are a number of flexible transformations that fit this requirement. One simple transformation is the planar flow [Jimenez Rezende and Mohamed, 2015]: $f(z) = z + uh(w^T z + b)$, where w, u, and b are free parameters and h is a smooth element-wise non-linearity. This transformation has a det-Jacobian which can be efficiently computed; however, this transformation can be seen as a MLP with a bottleneck hidden layer with a single unit.

Recently, there has been many developments in more flexible transformations that also have easily computable det-Jacobians due to the observation that the determinant of a triangular matrix can be efficiently computed as the product of its diagonal terms.

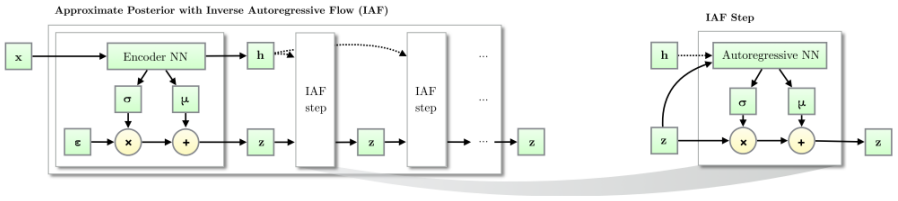


Figure 1: Illustration of IAF, from [Kingma et al., 2016].

For instance, Inverse Autoregressive Flows (IAF, [Kingma et al., 2016]) induce nonlinear dependencies between the elements of z by transforming z with deep masked autoencoders. Deep masked autoencoders, such as [Germain et al., 2015], mask their weight matrices so that they are autoregressive, meaning the Jacobian of the output is lower triangular with zeros on the diagonal. These autoregressive models are used in IAF to produce the scale σ and translation μ of z. See Fig. for an illustration from [Kingma et al., 2016] and see Section 5.2 for a short derivation of the det-Jacobian. The autoregressive models also take a vector h from the encoder and the Jacobian of z is unaffected by h.

Another powerful change of variable model is Real NVP [Dinh et al., 2017]. Although it is not a latent variable model, we can generalize its change of variable technique to the problem of posterior approximation in latent variable models. Real NVP builds its flexible distribution by stacking a sequence of simple bijections. That is, in each step of the transformation, part of the input vector is updated using a function which is simple to invert, but which depends on the remainder of the input vector in a complex way. Given a D dimensional input x, and $d < D$, the output y is given by:

$$y_{1:d} = x_{1:d} \tag{10}$$

$$y_{d+1:D} = x_{d+1:D} \odot exp(s(x_{1:d})) + t(x_{1:d}), \tag{11}$$

where s and t are complex functions (MLPs). See Fig. 2 for a visual representation of this flow. The Jacobian of this transformation is lower triangular with ones and $exp(s(x_{1:d}))$ on the diagonal, thus computing its det-Jacobian can be done efficiently. Partitioning of the input is implemented via binary masks. They use two specific partitionings that exploit the local correlation structure of images: spatial checkerboard patterns, and channel-wise masking. Comparing IAF and Real NVP, we see that IAF ensures its tractability by masking the weights of its flow whereas Real NVP masks its inputs.

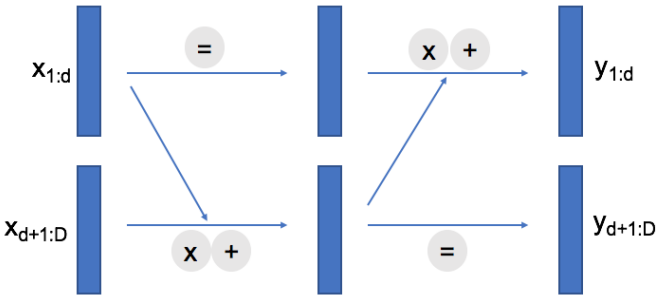


Figure 2: Diagram of two Real NVP flows

2.3 Auxiliary Variables

Deep generative models can be extended with auxiliary variables which leave the generative model unchanged but make the variational distribution more expressive. See Fig. 3 for a graphical model of an auxiliary variable model. As we can see, the generative process from z to x is unchanged, whereas the inference of z is now dependent on auxiliary variable v. Just as hierarchical Bayesian models induce dependencies between data, hierarchical variational models can induce dependencies between latent variables. They can capture structure of correlated variables because they turn the posterior into a mixture of distributions:

$$q(z|x) = \int q(z|x, v)q(v|x)dv \tag{12}$$

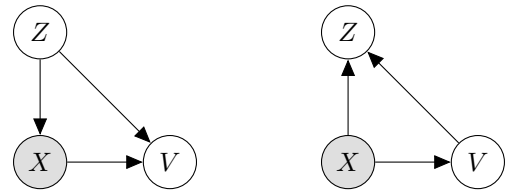
The addition of the auxiliary variable changes the lower bound to:

$$log(p(x)) \geq \mathbb{E}_{z,v\sim q(z,v|x)}\left[log\left(\frac{p(x,z)r(v|x,z)}{q(z|x,v)q(v|x)}\right)\right] \tag{13}$$

$$= \mathbb{E}_{q(z|x)}\left[log\left(\frac{p(x,z)}{q(z|x)}\right)\right] - \mathbb{E}_{q(z|x)}\left[KL(q(v|z,x)\|r(v|x,z))\right], \tag{14}$$

where $r(v|x,z)$ is called the reverse model. From Eqn. 14, 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 [Maaløe et al., 2016]) and hierarchical variational models (HVM, [Ranganath et al., 2016]). Interestingly, [Maaløe et al., 2016] remark that warm-up (annealing the KL term) is required to activate the auxiliary variables. Similarly, I have noticed that, if naively implemented, these models may not use the auxiliary variable at all (see AV in Fig. 6 in the appendix).

In order to increase the flexibility of the distributions over the auxiliary variables, these models often take advantage of normalizing flows. In [Ranganath et al., 2016], inverse flows are used to increase the complexity of the reverse model $r(v|x,v)$ and in MNF [Louizos and Welling, 2017], normalizing flows transform the auxiliary variable of a variational Bayesian neural net.



(a) Generative model (b) Inference model

Figure 3: Probabilistic graphical model of a auxiliary variable model

2.4 Hamiltonian Variational Inference

Hamiltonian Variational Inference (HVI, [Salimans et al., 2015]) is a combination of auxiliary variables and change of variable where the flow is based on Hamiltonian Monte Carlo [Neal, 2011]. In other words, HVI is a flow on an augmented space $z = (z, v)$ with Hamiltonian dynamics. The flow is given by:

$$v_{\frac{\epsilon}{2}} = v_0 - (\frac{\epsilon}{2}\frac{\partial}{\partial z} log\, p(x,z)) \tag{15}$$

$$z_T = z_0 + \epsilon v_{\frac{\epsilon}{2}} \tag{16}$$

$$v_T = v_{\frac{\epsilon}{2}} - (\frac{\epsilon}{2}\frac{\partial}{\partial z} log\, p(x,z)) \tag{17}$$

The variable v is often referred to as momentum. This flow is volume-preserving because its log det-Jacobian is zero. The lower bound that HVI maximizes is:

$$log(p(x)) \geq \mathbb{E}_{z_0, v_0 \sim q(z,v|x)} \left[log \left(\frac{p(x, z_T) r(v_T|x, z_T)}{q(z_0|x) q(v_0|x, z_0)} \right) \right]. \tag{18}$$

HVI can be seen as similar to Real NVP, where rather than partitioning the latent variable, HVI augments z with the auxiliary variable v. A disadvantage of HVI is that for every step, we need to compute the gradient of $log\, p(x, z)$ which can be computationally expensive. In addition, it requires the auxiliary bound, seen in Eqn. 14, which can impede progress if the bound is not sufficiently tight.

3 Diminishing Returns of Importance Weighting

One possible criticism of importance weighting is that as the dimensionality of the latent space z increases, the improvement gained by the multiple samples will decrease. The reasoning for this is that the volume of probability space grows exponentially in the number of dimensions, whereas increasing the number of samples k is only linear. So for models that require large latent sizes, IWAE might not be very beneficial. In [Burda et al., 2016], they test models with a latent size of 50 and they show improvements with increasing k. I'd like to test how the latent size effects the effectiveness of IWAE. I trained models with varying latent size D_z and samples k. The results are shown in Fig. 4 and in Table 3 of the appendix.

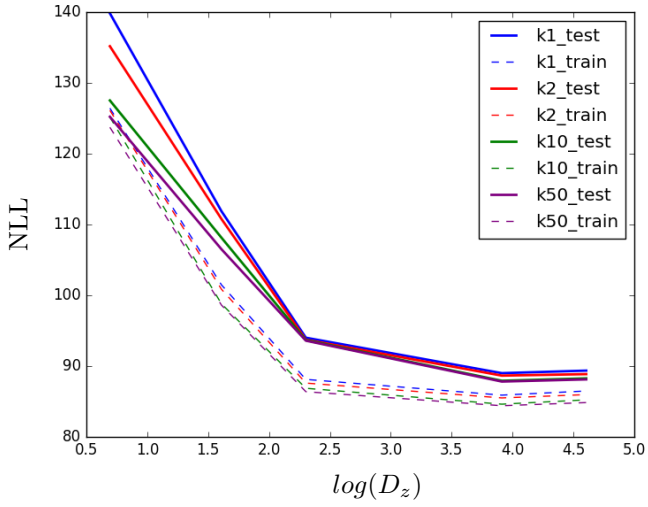


Figure 4: Negative log likelihood (NLL) of IWAE with different latent sizes (D_z). The latent sizes evaluated are 2, 5, 10, 50, and 100.

The latent sizes tested in Fig. 4 are 2, 5, 10, 50, and 100, but are shown in log space on the x-axis for visibility. We see that with $D_z = 2$ ($log2 = .69$), the difference between $k = 1$ and $k = 50$ is significant. Whereas with $D_z = 10$ ($log10 = 2.3$), the increase in k provides improvements of about 1-2 nats. Unintuitively, the variance of the NLLs grows slightly for larger latent sizes. An important factor in this experiment is how the log likelihood is approximated. In [Wu et al., 2017], they show that there is a significant gap between the IWAE estimate and the marginal likelihood. Here I used importance weighting of 5000 samples. To improve the approximation I could AIS as was done in [Wu et al., 2017]. Nevertheless, if 5000 samples is unable to approximate $logp(x)$, then this reinforces the idea of the diminishing returns of the number of samples with increased latent size.

4 Model Combinations

In many ways, the techniques described above are orthogonal, meaning they can be used together to possibly improve the approximate posterior. In this section, I will explore a few implications of model combinations. Some of the runtime complexity considerations are summarized in Table 1. See the appendix for a more complete table.

IW	NF	LF	NAME	Complexity
-	-	-	FG[1]	$\mathcal{O}(T)$
√	-	-	IWAE[2]	$\mathcal{O}(kT)$
-	√	-	NF[3]/IAF[4]/NVP[5]	$\mathcal{O}(f) + \mathcal{O}(T)$
-	-	√	HVI[6]	$\mathcal{O}(fT)$
√	√	-	-	$\mathcal{O}(f^2) + \mathcal{O}(fT)$ or $\mathcal{O}(kf) + \mathcal{O}(kT)$
√	-	√	-	$\mathcal{O}(f^2) + \mathcal{O}(fT)$ or $\mathcal{O}(kfT)$
-	√	√	-	$\mathcal{O}(fT)$
√	√	√	-	$\mathcal{O}(f^2) + \mathcal{O}(fT)$ or $\mathcal{O}(kfT)$

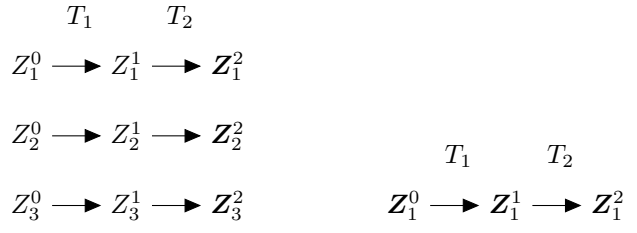
Table 1: Possible combinations of ideas and their asymptotic run time complexity. If the method already exists, its name is listed. IW: importance weighting, NF: normalizing flows, LF: leap-frog, T: time to evaluate decoder, k: number of particles (or samples), f: number of flows (or steps)
[1][Kingma and Welling, 2014], [2][Burda et al., 2016], [3][Jimenez Rezende and Mohamed, 2015], [4][Kingma et al., 2016], [5][Dinh et al., 2017], [6][Salimans et al., 2015]

4.1 Multiple particles with flow models

As discussed in section 3, using multiple samples tends to lose its effectiveness as the latent size increases. Perhaps we could increase the effectiveness of each sample by using more complex posteriors, so that each sample is in a space of

high probability. I will consider combining importance weighting with a flow model. I will discuss them in terms of time to evaluate the decoder T, number of particles (or samples) k, and number of flows (or steps) f.

One option is to perform importance weighting on the final samples of multiple transformed samples. This is represented in Fig. 5a and the lower bound is in Table 2, labeled as IW + NF (2). The runtime of this model is $O(kf)$ for the flows plus $O(kT)$ to evaluate the likelihood of the samples. This could be valuable if the cost of evaluating the decoder is much higher than the cost of the transformations.



(a) Multiple flows, use last samples

(b) Single flow, use each sample

Figure 5: Multiple particles with flow models. The bold samples are the ones used in the importance weighted lower bound.

Another option is to perform importance weighting on each of the samples of one flow (Fig. 5b). In this case, the runtime is $\mathcal{O}(f^2)+\mathcal{O}(fT)$, where we need $\mathcal{O}(fT)$ to compute the $p(x, z)$ at each step and $\mathcal{O}(f^2)$ to compute the mixture of the intermediate distributions. This could be useful for a model such as HVI where we already need to compute $p(x, z)$ at each step when computing its gradient. The q distribution is a mixture of the intermediate flow distributions given by:

$$q(z|x) = \frac{1}{T}\sum_{i=0}^{T} q_i(z|x) \tag{19}$$

$$= \frac{1}{T}\sum_{i=0}^{T} q_0(z|x) \prod_{t=1}^{i} \left| det \frac{\partial z_t}{\partial z_{t-1}} \right|^{-1}, \tag{20}$$

where $z_0 = z$. The lower bound of this model is given in Table 2, labeled as IW + NF (1). According to [Salimans et al., 2015], this method of using multiple samples can be effective at reducing the variance when working with long Markov chains.

A third option is to perform importance weighting at each step. This is similar to the recent work of [Maddison et al., 2017, Le et al., 2017, Naesseth et al., 2017], but to mitigate the effect of degenerate samples, they perform importance resampling at each step.

4.2 Combining normalizing and gradient flows

Models such as HVI can be very powerful because they have access to the gradient of $log\, p(x, z)$, which means they are guided by the exact posterior distribution. Also, for small enough step sizes, HVI will leave the posterior distribution invariant, meaning the transformation could take us arbitrarily close to the exact posterior distribution if we can apply it for a sufficient number of times. Given that computing the gradient is often an expensive operation, we'd rather take as few steps as possible. Perhaps it would be beneficial to instead combine the gradient flow with an auxiliary variable and some complex normalizing flows. One possible combination of these ideas could be:

$$v_T = v \cdot \sigma(z) + \mu(z) \cdot (-\frac{\partial}{\partial z} log\, p(x, z)) \tag{21}$$

$$z_T = z + f(v_T) \tag{22}$$

This is the transformation used for HF (Hamiltonian Flow) in Fig. 6 of the appendix. It seems to improve upon HVI and therefore could be of interest for future work.

References

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

[Burda et al., 2016] Burda, Y., Grosse, R., and Salakhutdinov, R. (2016). Importance weighted autoencoders. *In ICLR*.

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

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

[Germain et al., 2015] Germain, M., Gregor, K., Murray, I., and Larochelle, H. (2015). MADE: Masked Autoencoder for Distribution Estimation. *ArXiv e-prints*.

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

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

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

[Le et al., 2017] Le, T. A., Igl, M., Jin, T., Rainforth, T., and Wood, F. (2017). Auto-Encoding Sequential Monte Carlo. *ArXiv e-prints*.

[Louizos and Welling, 2017] Louizos, C. and Welling, M. (2017). Multiplicative Normalizing Flows for Variational Bayesian Neural Networks. *ArXiv e-prints*.

[Maaløe et al., 2016] Maaløe, L., Sønderby, C., Sønderby, S., and Winther, O. (2016). Auxiliary Deep Generative Models. *ICML*.

[Maddison et al., 2017] Maddison, C. J., Lawson, D., Tucker, G., Heess, N., Norouzi, M., Mnih, A., Doucet, A., and Whye Teh, Y. (2017). Filtering Variational Objectives. *ArXiv e-prints*.

[Miller et al., 2016] Miller, A. C., Foti, N., and Adams, R. P. (2016). Variational Boosting: Iteratively Refining Posterior Approximations. *ArXiv e-prints*.

[Naesseth et al., 2017] Naesseth, C. A., Linderman, S. W., Ranganath, R., and Blei, D. M. (2017). Variational Sequential Monte Carlo. *ArXiv e-prints*.

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

[Ranganath et al., 2016] Ranganath, R., Tran, D., and Blei, D. M. (2016). Hierarchical Variational Models. *ICML*.

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

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

5 Appendix

5.1 Summary of Lower Bounds

Model	Lower Bound
VAE	$\mathbb{E}_{z\sim q(z\|x)}\left[log\left(\frac{p(x,z)}{q(z\|x)}\right)\right]$
IWAE	$\mathbb{E}_{z_1\dots z_k\sim q(z\|x)}\left[log\left(\frac{1}{k}\sum_{i=1}^{k}\frac{p(x,z_i)}{q(z_i\|x)}\right)\right]$
NF	$\mathbb{E}_{z_0\sim q(z_0\|x)}\left[log\left(\frac{p(x,z_T)}{q(z_0\|x)\prod_{t=1}^{T}\left\|det\frac{\partial z_t}{\partial z_{t-1}}\right\|^{-1}}\right)\right]$
Aux	$\mathbb{E}_{z,v\sim q(z,v\|x)}\left[log\left(\frac{p(x,z)r(v\|x,z)}{q(z\|x,v)q(v\|x)}\right)\right]$
HVI	$\mathbb{E}_{z_0,v_0\sim q(z,v\|x)}\left[log\left(\frac{p(x,z_T)r(v_T\|x,z_T)}{q(z_0\|x,v_0)q(v_0\|x)}\right)\right]$
Aux + NF	$\mathbb{E}_{z_0,v_0\sim q(z,v\|x)}\left[log\left(\frac{p(x,z_T)r(v_T\|x,z_T)}{q(z_0\|x,v_0)q(v_0\|x)\left\|det\frac{\partial z_t v_t}{\partial z_{t-1}v_{t-1}}\right\|^{-1}}\right)\right]$
IW + Aux	$\mathbb{E}_{z_1v_1\dots z_kv_k\sim q(z,v\|x)}\left[log\left(\frac{1}{k}\sum_{i=1}^{k}\frac{p(x,z_i)r(v_i\|x,z_i)}{q(z_i\|x,v_i)q(v_i\|x)}\right)\right]$
IW + NF (1)	$\mathbb{E}_{z_0\dots z_T\sim q(z_{0:T}\|x)}\left[log\left(\frac{1}{T}\sum_{i=0}^{T}\frac{p(x,z_i)}{\frac{1}{T}\sum_{j=0}^{T}q_0(z_i\|x)\prod_{t=1}^{j}\left\|det\frac{\partial z_t}{\partial z_{t-1}}\right\|^{-1}}\right)\right]$
IW + NF (2)	$\mathbb{E}_{z_0^1\dots z_0^k\sim q_0(z\|x)}\left[log\left(\frac{1}{k}\sum_{i=1}^{k}\frac{p(x,z_T^i)}{q(z_0^i\|x)\prod_{t=1}^{T}\left\|det\frac{\partial z_t^i}{\partial z_{t-1}^i}\right\|^{-1}}\right)\right]$

Table 2: Summary of Lower Bounds

5.2 IAF derivation

$$z_T = z \odot \sigma(z) + \mu(z) \tag{23}$$

$$\frac{\partial z_T}{\partial z} = (z' \cdot diag(\sigma(z)) + diag(z) \cdot \sigma'(z)) + \mu'(z) \tag{24}$$

$$= I \cdot diag(\sigma(z)) + (diag(z) \cdot \sigma'(z)) + \mu'(z) \tag{25}$$

$$det(\frac{\partial z_T}{\partial z}) = \prod_{i=1}^{D} \sigma_i \tag{26}$$

Since $\sigma(z)$ and $\mu(z)$ are both autoregressive functions of z, their Jacobians are lower triangular with zeros on the diagonal. The determinant of a triangular matrix is the product of its diagonal, thus $det((diag(z) \cdot \sigma'(z)) + \mu'(z)) = 0$.

5.3 Table of IWAE results

k	2	5	10	50	100
1	139.83	111.88	94.01	89.00	89.36
2	135.18	110.78	93.73	88.66	88.86
10	127.52	108.14	93.67	87.93	88.27
50	125.22	106.51	93.59	87.82	88.12

Table 3: The test results of the IWAE experiment of section 3. Across: latent sizes (2-100)

5.4 Table of model combinations with auxiliary variables

IW	NF	AV	LF	NAME	Complexity
-	-	-	-	FG[1]	$\mathcal{O}(T)$
$\surd$	-	-	-	IWAE[2]	$\mathcal{O}(kT)$
-	$\surd$	-	-	NF[3]/IAF[4]/NVP[5]	$\mathcal{O}(f) + \mathcal{O}(T)$
-	-	$\surd$	-	HVM[7]	$\mathcal{O}(T)$
-	-	-	$\surd$	HVI[6]	$\mathcal{O}(fT)$
$\surd$	$\surd$	-	-	-	$\mathcal{O}(f^2) + \mathcal{O}(fT)$ or $\mathcal{O}(kf) + \mathcal{O}(kT)$
-	$\surd$	$\surd$	-	MNF[8]/HVM[7]	$\mathcal{O}(f) + \mathcal{O}(T)$
$\surd$	-	$\surd$	-	-	$\mathcal{O}(kT)$
$\surd$	$\surd$	$\surd$	-	-	$\mathcal{O}(f^2) + \mathcal{O}(fT)$ or $\mathcal{O}(kf) + \mathcal{O}(kT)$
-	-	$\surd$	$\surd$	HVI[6]	$\mathcal{O}(fT)$
-	$\surd$	-	$\surd$	-	$\mathcal{O}(fT)$
$\surd$	-	-	$\surd$	-	$\mathcal{O}(f^2) + \mathcal{O}(fT)$ or $\mathcal{O}(kfT)$
$\surd$	$\surd$	-	$\surd$	-	$\mathcal{O}(f^2) + \mathcal{O}(fT)$
$\surd$	-	$\surd$	$\surd$	-	$\mathcal{O}(f^2) + \mathcal{O}(fT)$ or $\mathcal{O}(kfT)$
-	$\surd$	$\surd$	$\surd$	-	$\mathcal{O}(f^2) + \mathcal{O}(fT)$
$\surd$	$\surd$	$\surd$	$\surd$	-	$\mathcal{O}(f^2) + \mathcal{O}(fT)$ or $\mathcal{O}(kfT)$

Table 4: Enumeration of possible combination of models. T: time to evaluate decoder. k: number of particles/samples. f: number of flows/steps
[1][Kingma and Welling, 2014], [2][Burda et al., 2016], [3][Jimenez Rezende and Mohamed, 2015], [4][Kingma et al., 2016], [5][Dinh et al., 2017], [6][Salimans et al., 2015], [7][Ranganath et al., 2016], [8][Louizos and Welling, 2017]

5.5 Example plots

The following plots are illustrative implementations of various models described in this report. The hyperparameters (number of steps, types of flows, etc.) of the models were not tuned, thus this is just a rough examination of the behaviour of the various models. The models are trying to fit the posteriors distributions on the left. The approximate posteriors are plotted using a KDE of a 1000 samples. Notice that the auxiliary variable (AV) model isn't taking advantage of its possible flexibility which results in plots similar to the factorized Gaussian (FG). The normalizing flow (NF) model seems to model only one mode. HVI (HV) does a better job at modelling multiple nodes. Combining HVI with NF seems to improve its modelling ability.

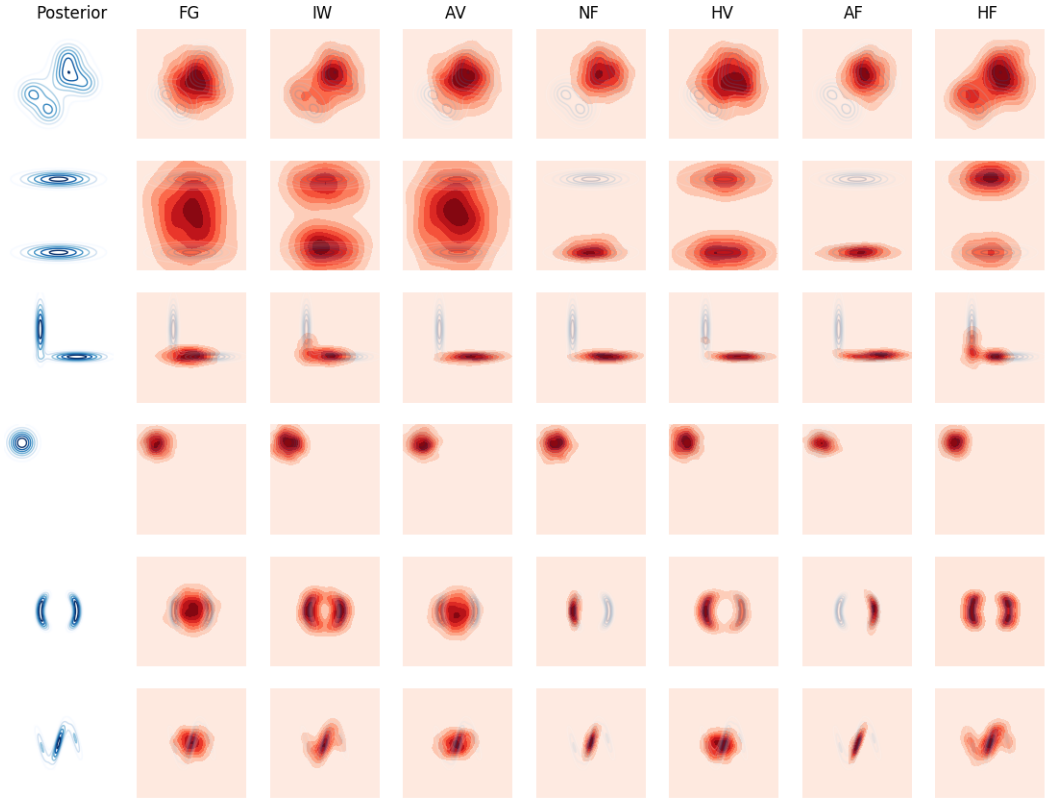


Figure 6: Example implementations of various posterior approximations on 2D problems. FG: factored gaussian, IW: importance weighted, AV: auxiliary variable, NF: normalizing flows, HV: hamiltonian variational, AF: auxiliary flows (AV+NF), HF: hamiltonian flows (HV+NF)