# Richer posteriors with flow-based variational autoencoders

**Isaac Henrion** **Sean Welleck**

## Abstract

Variational autoencoders (VAEs) are a class of latent variable models widely used for large-scale stochastic variational inference. A class of methods known as normalizing flows has been proposed to enrich the form of the approximate posterior in a standard VAE. We examine three families of normalizing flows, and investigate their properties through experimentation. We find that one family, inverse autoregressive flow, consistently outperforms the standard VAE. Another, unconstrained residual flow, suffers from training instability. The third, Householder flow, is stable but does not improve upon the VAE baseline. We discuss possible reasons for these behaviours and study the changes to the posterior caused by the flow transformations.

## 1 Introduction

Variational autoencoders (VAEs) are a class of latent variable models widely used for large-scale stochastic variational inference. They have found success in many domains, including generative image models [3], semi-supervised classification [8], topic models [5], and sentence-level language models [1].

Like other variational methods, VAEs have both a generative model of the observed variables given the latent variables, and an inference model of the posterior over latent variables. The inference model is constrained to come from a chosen family of distributions, yielding an approximate posterior. This approximation gives a lower bound for the log-evidence. Training consists of maximizing this lower bound with respect to the parameters of both the generative model and the inference model. Two points distinguish VAEs from other variational methods. First, they employ a neural network for the approximate posterior, which allows the cost of inference to be shared across data points through the weights. Second, by reparameterizing the latent stochastic variables in terms of nonparametric noise and a deterministic transformation, VAEs can be trained end-to-end with stochastic backpropagation [12].

The approximate posterior is potentially a grave disadvantage for VAEs, as for other variational methods. For many applications, the posterior will be extremely complex. Thus it is difficult to approximate it with a function from a simple parametric family, such as a diagonal Gaussian. Since the lower bound to the log-evidence is not necessarily tight, one cannot hope to recover a perfect model of the data, even asymptotically. This situation stands in contrast to Markov chain Monte Carlo (MCMC), which recovers the exact posterior (and hence the exact data distribution) in the infinite-sample limit [13]. On the other hand, VAEs are faster than MCMC due to amortized inference through the neural network. Therefore it appears that there is a trade-off between computational efficiency and statistical exactness in the choice of approximate posterior.

In view of this problem, it is natural to seek to improve VAEs by enriching the family of posterior distributions beyond diagonal Gaussians. One approach that has found traction in recent years is the concept of *normalizing flows* [13]. A normalizing flow is a sequence of parametric mappings for transforming a simple probability distribution into a more complex one. Each mapping must satisfy two crucial properties. First, it must be smooth and invertible, so that backpropagation can be used

to train the parameters. Second, the log-det-Jacobian term must be simple to compute, so that we can easily and precisely evaluate the transformed density. To use a normalizing flow in a VAE, we sample from an "initial posterior" and then apply the flow to the sample. The parameters of the flow can be constant, or depend on the data in various ways.

One plausible objection to the normalizing flows is that they simply push the choice of approximating family upstream. Now we must select both the type of flow and the number of steps to maximize the flexibility of the posterior under the constraint of trainability. However, the success of deep neural networks suggests that given a parameter budget, a long sequence of simple transformations is more expressive than a short sequence of complex transformations [11]. Therefore it is natural to presume that the individual flow transformations could be very simple, and produce a complex posterior through composition.

In this work, we survey a number of flow-based VAEs and compare their performance on the MNIST data set. Specifically, we examine an unconstrained residual flow [13], inverse autoregressive flow [9], and Householder flow [17]. We are only able to demonstrate the superior performance of inverse autoregressive flow relative to a standard VAE baseline. This stands in contrast to the results of the aforementioned papers, which we attribute to ambiguity in training techniques.

As we study the behaviour of the flow-based models, we find that the unconstrained residual flow is too expressive and underdetermined to be effectively trainable. The complexity of the approximate posterior causes high instability in the KL term, dominating the error signal and preventing learning. The other two flows are special cases of unconstrained residual flow, and their restricted nature permits easier training.

Unfortunately, even in the case of our best-performing model, inverse autoregressive flow, we observe that the flow wastes much of its computation. The model learns to encode a simple and broad initial posterior that is subsequently contracted to the final approximate posterior. Increasing the flow length only serves to broaden the initial distribution but does not have a very significant impact on the quality of the final posterior.

## 2 Variational autoencoders

The problem of how to perform inference and learning in large-scale probabilistic models of data has occupied researchers for a long time [4, 14, 10]. We wish to model the data $\mathbf{x}$ using latent variables $\mathbf{z}$: we assume that $\mathbf{z}$ is generated from a parametric prior $p_{\boldsymbol{\theta}}(\mathbf{z})$, and then $\mathbf{x}$ is drawn from $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$. Under this generative model, we are often interested in the posterior over latent variables $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$, as well as the marginal distribution $p_{\boldsymbol{\theta}}(\mathbf{x})$. Unfortunately, for complex parametric models $p_{\boldsymbol{\theta}}$ these quantities are typically intractable. Moreover, if the amount of data is large, it is infeasible to use sampling-intensive approaches [10].

The variational auto-encoder (VAE) was proposed simultaneously in 2014 by Kingma and Welling [10] and Rezende et al [12] as a solution to the problem of learning complex generative models with large datasets, and *performing inference in them efficiently*. VAEs employ a *recognition model* $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ to approximate the posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. Using Bayes' Rule, we can manipulate the form of the KL-divergence of $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ from $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ to obtain the mean-field approximation of the log evidence $\log p_{\boldsymbol{\theta}}(\mathbf{x})$:

$$\begin{aligned}
D_{KL}[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})] &= \mathbb{E}_{\mathbf{z}\sim q}\left[\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) - \log p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})\right] \\
&= \mathbb{E}_{\mathbf{z}\sim q}\left[\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) - \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) - \log p_{\boldsymbol{\theta}}(\mathbf{z}) + \log p_{\boldsymbol{\theta}}(\mathbf{x})\right] \\
&= \mathbb{E}_{\mathbf{z}\sim q}\left[\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) - \log p_{\boldsymbol{\theta}}(\mathbf{z})\right] - \mathbb{E}_{\mathbf{z}\sim q}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right] + \mathbb{E}_{\mathbf{z}\sim q}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x})\right] \\
&= D_{KL}\left[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z})\right] - \mathbb{E}_{\mathbf{z}\sim q}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right] + \log p_{\boldsymbol{\theta}}(\mathbf{x})
\end{aligned}$$

and rearranging terms we obtain

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) - D_{KL}[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})] = -D_{KL}\left[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z})\right] + \mathbb{E}_{\mathbf{z}\sim q}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right]. \quad (1)$$

The KL-divergence is non-negative, and 0 if and only if the distributions match. Hence the right-hand side is a lower bound for the evidence, which is tight just when $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. This lower bound can be considered as an objective function $\mathcal{L}$ to be optimized using gradient descent on both $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$. However, given a function $f$ it is difficult to compute $\nabla_{\boldsymbol{\phi}}\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}} f(\mathbf{z})$ because the distribution of $\mathbf{z}$ itself depends on $\boldsymbol{\phi}$, leading to high-variance gradient estimates if conventional sampling methods are used.

The insight of [10] is that if $\mathbf{z}$ can be suitably reparameterized, then the intrinsic non-differentiability of sampling can be sidestepped. Specifically, let $\mathbf{z} \sim q_{\boldsymbol{\phi}}$ be reparameterized as $\mathbf{z} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x})$, where $g_{\boldsymbol{\phi}}$ is a smooth deterministic function and $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$ is a noise variable whose distribution has fixed parameters (e.g. Gaussian white noise). Then expectations with respect to $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ are equivalent to expectations with respect to $p(\boldsymbol{\epsilon})$. The latter can be easily approximated using a Monte Carlo integral, yielding an unbiased estimator for $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{x})$:

$$\tilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{x}) = \frac{1}{L} \sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}, \mathbf{z}^{(i,l)}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(i,l)}|\mathbf{x}^{(i)})$$

We can take exact gradients of this estimator with respect to the variational parameters, and then use gradient-based optimizers such as SGD or ADAM [7]. Thus the variational autoencoder provides a way to perform simple and fast inference in an expressive generative model.

The variational autoencoder is an instance of the above framework in which the recognition model $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ and generative model $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ are parameterized by neural networks. Then $q$ and $p$ can be interpreted as a probabilistic encoder and decoder, respectively, and the objective function $\mathcal{L}$ can be viewed as the expected reconstruction error $-\mathbb{E}_{\mathbf{z}\sim q}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right]$ plus a regularization term $D_{KL}\left[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z})\right]$.

## 3 Flow-based methods

The standard VAE model uses a simple mapping for the stochastic variable $\mathbf{z}$, namely $g_{\phi}(\boldsymbol{\epsilon}, \mathbf{x}) = \boldsymbol{\mu}(\mathbf{x}) + \boldsymbol{\sigma}(\mathbf{x}) \odot \boldsymbol{\epsilon}$ where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are neural networks. This is a diagonal Gaussian, and therefore cannot capture information about the covariance structure of the latent variable. The crudeness of this posterior could be a disadvantage in complex domains where there is significant off-diagonal covariance. For example, if we interpret the possible latent variables in image generation to be classes like "dog", "car" and so on, there is clearly a lot of dependence structure among them. This raises a fairly obvious question: can we improve VAEs by enriching the approximate posterior beyond a diagonal Gaussian?

One way to enrich the posterior is by using a more complex model than a Gaussian. For example, we could use a mixture of diagonal Gaussians. However, this will not scale well because the complexity of this model is linear in the number of components. In high dimensions we would require large numbers of components, hence this approach is not particularly suitable. We do not pursue such methods in this work.

Alternatively, we can consider flow-based methods. Normalizing flows were originally introduced in [16] and further explored in [15] in the context of non-parametric density estimation. The idea is to transform the data $x \sim p(x)$ with a function $y(x)$ such that $y(x)$ is distributed according to a simple distribution $q_y(y)$. If we know the map $y$, then we can recover the original density $p(x)$ since $p(x) = J^y(x) q_y(y(x))$, where $J^y$ is the Jacobian. Thus the density estimation problem becomes one of finding the normalizing map $y$.

In the context of enriching posterior approximations in VAEs, an opposite view of [16] is adopted; instead of transforming arbitrarily distributed data such that it is eventually distributed according to a simple distribution, flows are used to transform data from a simple Gaussian distribution to an arbitrary distribution.

The normalizing flow technique works by passing a simple initial density through a sequence of smooth invertible mappings in order to obtain a more complex density. Each mapping is a change of variables $\mathbf{z}' = f(\mathbf{z})$ and therefore the new density can be expressed in terms of the old one:

$$p(\mathbf{z}') = p(\mathbf{z}) \left| \det \frac{\partial f}{\partial \mathbf{z}} \right|^{-1}$$

Evidently, this technique will only be useful if we can compute the change-of-variables formula efficiently. More precisely, we need to do it in the log-domain since we are minimizing the log-loss. Therefore the log-det-Jacobian $\log \left| \det \frac{\partial f}{\partial \mathbf{z}} \right|$ must either have a simple form or be easy to compute. If we can satisfy this condition, and the mapping $f$ is sufficiently expressive, then we can hope to learn richer posteriors.

The functions $f$ will be parametric, and these parameters can themselves be learned functions of the observed data $\mathbf{x}$ and/or some context $\mathbf{c}$. In this case, the parameter functions are themselves parameterized by a neural network, which is trained by backpropagation. Alternatively we can fix the flow parameters as weights in a neural network and make them independent of the input $\mathbf{x}$.

Inspired by the success of deep neural networks, it is intuitive that the flow should be a long sequence of simple mappings rather than a short sequence of complex mappings. Recently it has been shown theoretically that network depth increases the expressivity of the set of possible functions in several senses [11]. There is a consensus that depth is more important than width in a neural network, given a fixed parameter budget.

### 3.1 Free energy bound

Let $f_0, \ldots, f_K$ be a normalizing flow, so that $\mathbf{z}_K = f_K \circ \cdots \circ f_0(\mathbf{z}_0)$. We can rewrite the free energy bound as an expectation with respect to the initial density $q_0(\mathbf{z}_0)$:

$$\begin{aligned}\mathcal{F}(\mathbf{x}) &= \mathbb{E}_{q_K(\mathbf{z}_K)}\left[\log q_K(\mathbf{z}_K) - p(\mathbf{z}_K) - \log p(\mathbf{x}|\mathbf{z}_K)\right] \\ &= \mathbb{E}_{q_0(\mathbf{z}_0)}\left[\log q_0(\mathbf{z}_0) - p(\mathbf{z}_K) - \log p(\mathbf{x}|\mathbf{z}_K) - \sum_{k=0}^{K} \log\det J_k\right]\end{aligned}$$

where $J_k$ is the Jacobian term from the mapping $f_k$:

$$J_k = \frac{\partial f_k}{\partial \mathbf{z}_k}.$$

This follows from the standard rules for changing variables in probability distributions. Each of these terms can be computed with a Monte Carlo estimate, by sampling from the initial posterior distribution $q_0$.

### 3.2 Unconstrained residual flow

The flow proposed in [13] is of the form

$$f(\mathbf{z}) = \mathbf{z} + \mathbf{u}h\left(\mathbf{w}^\top \mathbf{z} + b\right)$$

where $h$ is a nonlinear transformation such as tanh or sigmoid. Here the parameters of the flow are $\mathbf{u}$, $\mathbf{w}$ and $b$, and the Jacobian is

$$J(\mathbf{z}) = \mathbf{I} + \mathbf{w}^\top h'(\mathbf{w}^\top \mathbf{z} + b)\mathbf{u}.$$

The determinant of this Jacobian is simple to compute. Sylvester's determinant lemma allows us to reorder the multiplication, since $\det(\mathbf{I} + \mathbf{AB}) = \det(\mathbf{I} + \mathbf{BA})$ whenever the dimensions agree. Hence we need only compute a dot-product rather than the full matrix multiplication:

$$\det J(\mathbf{z}) = |1 + \mathbf{u}^\top h'(\mathbf{w}^\top \mathbf{z} + b)\mathbf{w}|.$$

This flow is a residual mapping: starting from an initial $\mathbf{z}$, we compute the residual and add it to $\mathbf{z}$, iterating until $\mathbf{z}$ sufficiently resembles a sample from the true posterior. The residual mapping potentially has an advantage over a nonresidual mapping, because it can easily learn the identity if necessary. If the posterior really were a diagonal Gaussian, then the flow would be able to drive the residual mappings to zero, even if the flow is long.

However, in this type of flow the log-det-Jacobian is unbounded either below or above. The lack of constraint on the volume of each transformation naturally leads to redundancy in the model, because there are many ways to attain the same overall Jacobian. We speculate that this unconstrained residual flow will be harder to train than volume-preserving flows, because it is underdetermined.

### 3.3 Inverse autoregressive flow

Inverse autoregressive flow (IAF) was proposed in [9], and is based on the standard autoregressive Gaussian model:

$$\begin{aligned}y_0 &= \mu_0 + \sigma_0 z_0 \\ y_i &= \mu_i(\mathbf{y}_{0:i-1}) + \sigma_i(\mathbf{y}_{0:i-1}) z_i \quad \text{for } i = 1 \ldots D \\ z_i &\sim \mathcal{N}(0, 1).\end{aligned}$$

where $\mu_i$ and $\sigma_i$ are arbitrary functions, such as neural networks. The key insight is that the inverse model is equally expressive, and can be computed elementwise in parallel because given $\mathbf{y}$ the variables $z_i$ are independent of one another:

$$\mathbf{z} = \frac{\mathbf{y} - \boldsymbol{\mu}(\mathbf{y})}{\boldsymbol{\sigma}(\mathbf{y})}.$$

The only requirement is that the functions $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are autoregressive, i.e. each component is constant with respect to the earlier components. This gives rise to the following flow model:

$$f(\mathbf{z}) = \frac{\mathbf{z} - \boldsymbol{\mu}(\mathbf{z})}{\boldsymbol{\sigma}(\mathbf{z})}$$
$$\log\det J(\mathbf{z}) = -\sum_{i}^{D} \log \sigma_i(\mathbf{z})$$

with the last equation following from the fact that the Jacobian matrix is lower triangular with $\sigma_i^{-1}$ on the diagonal.

In practice, one must choose a particular autoregressive function. One possibility is the Masked Autoencoder, or MADE [2]. MADE consists of an autoencoder with binary weight masks that enforce an autoregressive constraint. The masks are applied to the weights on either side of the autoencoder's hidden state and have a product that is lower-diagonal, causing the output to only depend on previous input dimensions. That is, when the autoencoder reconstructs an input $x$, the $d$th dimension of the output $\tilde{x}_d$ only depends on the previous input dimensions $x_1, ..., x_{d-1}$.

We use MADE as the autoregressive function in this work since it is nonlinear, relatively simple to implement, and efficient to compute. Of course, alternative autoregressive functions could be used, such as a linear autoregressive Gaussian or a recurrent neural network.

### 3.4 Householder flow

The final flow we consider is based on the Householder transformation [17]. In a real vector space, a Householder transformation is a reflection parameterized by a normal vector $\mathbf{v}$:

$$\mathbf{H} = \mathbf{I} - 2\mathbf{v}\mathbf{v}^\top.$$

The mapping $f(\mathbf{z}) = \mathbf{H}\mathbf{z}$ is then a volume-preserving linear map which can be used as a flow transformation. Since $f$ preserves volumes, the log-det-Jacobian terms are zero. The motivation for this flow is the theorem that any orthogonal matrix of rank $K$ can be decomposed as a product of at most $K$ Householder transformations. Therefore, starting from unit covariance we can generate arbitrary covariance matrices for the latent variables $\mathbf{z}$. The appeal of this method is clear: there are no log-det-Jacobian terms, and with a sufficiently long flow we can model all second-order moments of the posterior (at least, in theory). The parameters of the initial diagonal Gaussian will determine the relative scale of the hidden variables, and then the Householder transformations will recover the covariance structure.

## 4 Experiments

Implementations of the models and code for reproducing key empirical results is available online[1].

### 4.1 Data

For our experiments, we use the binarized MNIST data set. This comprises 50,000 training images, 10,000 validation images and 10,000 test images. Each image is a $28 \times 28$ binary matrix corresponding to white/black pixels.

[1] `https://github.com/wellecks/vaes`

### 4.2 Methodology

To model the probability of a given image $\mathbf{x}$, we consider each pixel $x_i$ to be an independent Bernoulli random variable with probability $p_i$. Thus

$$p(\mathbf{x}) = \prod_{i=1}^{N} p_i^{x_i} (1 - p_i)^{1 - x_i}$$

and the log-evidence is

$$\log p(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \sum_{i=1}^{N} x_i \left( \log p_i + (1 - x_i) \log \left( 1 - p_i \right) \right)$$

In order to estimate the evidence lower bound $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{x})$ we use a Monte Carlo estimate with a single sample $\mathbf{z}$ from the approximate posterior. We evaluate our models in terms of the lower bound, rather than the log-evidence.

In order to make the results comparable across methods, we fixed the number of stochastic variables at 40, and used two-layer fully connected neural networks with 300 neurons per layer for both the basic decoder and encoder architectures. We trained each model for 2000 epochs over the whole training set with batch size 100. The learning rate was slowly annealed: at epoch $t$, we set $\eta_t = 0.0002/(1 + t/100)$; we used the ADAM optimizer [7] to speed up training. We also used batch normalization [6].

It was observed in [1] and [9] that the KL divergence term can initially dominate the cost relative to the reconstruction error. At the start of training an attractive local optimum is $q(\mathbf{z}|\mathbf{x}) \approx p(\mathbf{z})$, which can be hard to escape. Therefore we experimented with an annealing schedule for the KL term, increasing its weight from 0 to 1 over the course of training. However, we were unable to find any benefits for training and so fixed the weight at 1 for our longer runs. Another approach is the "minimum information constraint", which forces the KL term to be at least $\lambda K$ where $K$ is the number of latent dimensions. This constraint means that the KL term is never improved by using less than $\lambda$ nats of information per latent dimension. This will encourage the encoder to learn more complex distributions, because using more information should always help reduce the reconstruction error. Due to time constraints we did not implement and test this new objective function.

### 4.3 Comparison of free energies

Table 1 shows the free energy lower bound on the train, validation and test sets for the various methods trained on the binarized MNIST data set. Of all the flow-based methods, only inverse autoregressive flow outperforms the simple VAE baseline. The best-performing model was IAF with a flow-length of 2.

Figures 1 and 4.3 show the training loss progression for the various methods, providing a global view and a zoomed view of epochs 800 through 900. Globally, we see that the ordering of methods remains roughly consistent over the course of training, with inverse flows outperforming the vanilla VAE, residual flows performing worse than the vanilla VAE, and Householder flows performing near the vanilla VAE. In the zoomed view we see that by 800 epochs the methods have converged, with an ordering consistent with the results in Table 1.

We were unable to replicate the results in [13, 9, 17] for the free energy bound. These authors obtained significantly better errors using the same model architectures, with a best result of 79.5 for Householder flow with length 2. We are not sure why we could not match these results. However, several of the hyperparameters in the papers were unspecified, which could mean that we were using suboptimal values for e.g. the learning rate. Given that we did not replicate the existing results, all of our discussion below is tentative and based only on what we observed during training.

| Model | Training | Validation | Test |
|---|---|---|---|
| Baseline VAE | 88.3 | 95.6 | 94.8 |
| Unconstrained residual flow ($K = 1$) | 90.7 | 97.4 | 96.6 |
| Unconstrained residual flow ($K = 2$) | 91.7 | 97.9 | 97.1 |
| Unconstrained residual flow ($K = 10$) | 99.9 | 104.5 | 103.5 |
| Inverse autoregressive flow ($K = 1$) | 87.8 | 94.7 | 94.0 |
| Inverse autoregressive flow ($K = 2$) | 87.5 | 94.2 | **93.5** |
| Inverse autoregressive flow ($K = 10$) | 88.9 | 94.2 | 93.6 |
| Householder flow ($K = 1$) | 89.1 | 95.9 | 95.1 |
| Householder flow ($K = 2$) | 89.6 | 96.0 | 95.2 |
| Householder flow ($K = 10$) | 90.3 | 96.0 | 95.3 |

Table 1: Comparison of free energy lower bound for the various methods. Inverse autoregressive flow outperforms all the other methods, and is the only flow-based method to outperform the baseline VAE.

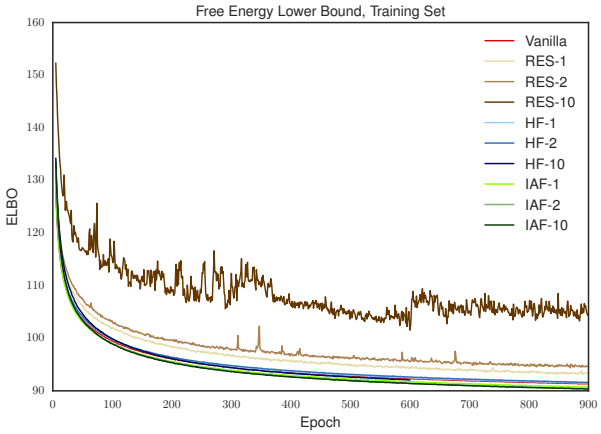


Figure 1: Comparison of training error for the different methods.

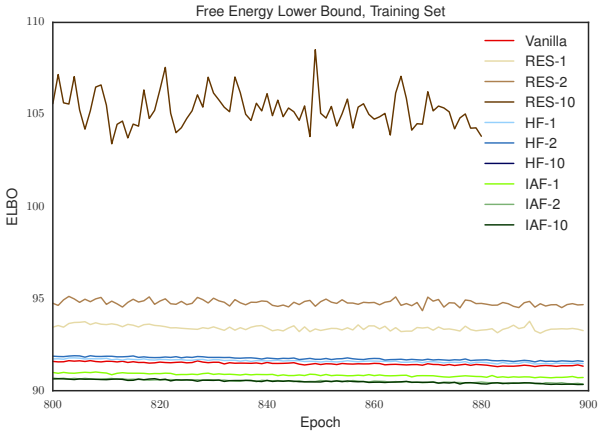


Figure 2: Comparison of training error for the different methods, only showing epochs 800-900.

### 4.4 Unstable training for unconstrained residual flows

It is noteworthy that the unconstrained residual flows substantially underperformed the baseline, in contrast to the results in [13]. We suspect that this is due to difficulty in training the unconstrained residual flow, which proved to be highly unstable. The KL term fluctuated during training because samples from the approximate posterior $q_K$ did not have a consistent probability under the prior. Indeed, the longer the flow, the more often the model received samples that were either very likely or very unlikely under the prior. This phenomenon is illustrated in Figure 3, and leads to wild oscillations in KL divergence which dominate the error signal. Aside from the instability, the KL divergence increased with flow length, in line with expectation. As the flow becomes longer, the domain of possible posterior distributions becomes more complex, leading to higher KL divergence from the prior.

One possible reason for the instability of the unconstrained residual flow was outlined above: that the unboundedness of the log-det-Jacobian makes the flow underdetermined. On the other hand, this also applies to inverse autoregressive flow, since the log-det-Jacobian terms are the log-standard deviations from the autoregressive model. The only volume-preserving flow we examined, Householder flow, was extremely stable during training. Alternatively, this could be due to the fact that the unconstrained residual flow is *too* expressive and therefore difficult to train.

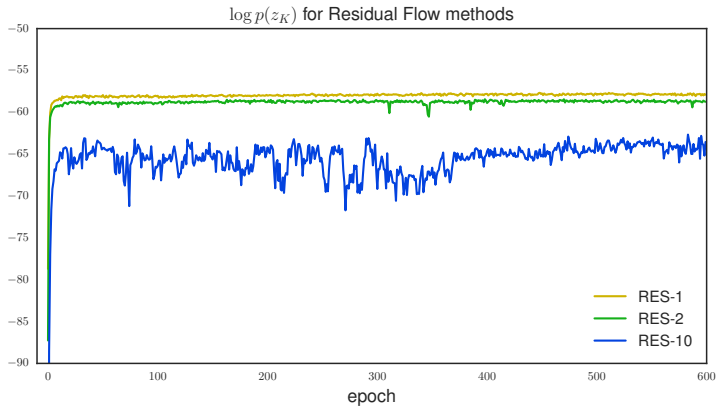


Figure 3: Unconstrained residual flow: plot of the log-density of samples from the approximate posterior, *under the prior* $p(\mathbf{z})$. The $x$-axis shows the number of parameter updates. The training patterns for unconstrained residual flow are shown for $K = 1$ (yellow), $K = 2$ (green) and $K = 10$ (blue). We observe that as flow length increases, (a) the samples become less likely under the prior and (b) the variance of the prior log-density increases. This causes training to be extremely unstable.

### 4.5 Increased flow length causes contraction

In Figure 4 we study the evolution of the inverse autoregressive flow transformations over the course of training. We see that for all three flow lengths, the transformations represent a contraction of the initial posterior density $q_0$ (due to the negative log-det-Jacobian terms). We can also see the effect of the flow $\mathbf{z}_K = f(\mathbf{z}_0)$ through the plots of $\log q_0(\mathbf{z}_0)$ and $\log q_K(\mathbf{z}_K)$ over the course of training. As training progresses, $q_0(\mathbf{z}_0)$ decreases, while $q_K(\mathbf{z}_K)$ increases and converges. The effect is more pronounced in the multi-step flows ($K = 2$ and $K = 10$) than in the one step flow ($K = 1$).

One explanation of this phenomenon is that the increased flow length introduces slack into the model, allowing it to take many steps to arrive at the same density. If only a short flow is allowed, the encoder is under more pressure to give a precise initial density $q_0$. The samples from $q_0$ will have relatively high density, because they come from a sharply peaked distribution. Conversely, as we allow more steps of flow, the encoder only needs to give a broad distribution for $q_0$ and allows the flow to successively refine it to a sharp posterior. In this case the samples from $q_0$ have low density because the probability mass is spread over more space. A sequence of contractions is required to sharpen the density, resulting in the increasingly negative log-det-Jacobian terms.

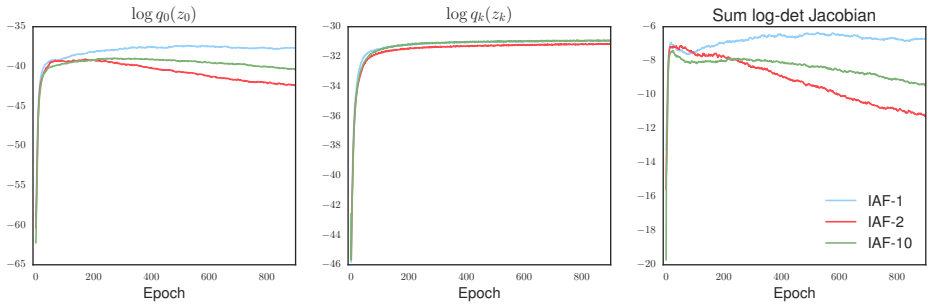


Figure 4: Left: initial log-densities for IAF with varying flow lengths. Center: final log-densities after IAF transformation. Right: sum of log-det-Jacobian terms. We observe that all three flows achieve similar densities for the approximate posterior, but the longer flows compensate lower initial densities by contracting over the course of the flow.

## 5 Conclusion

In this project, we examined a number of flow-based methods for enriching the approximate posterior in VAEs. These techniques offer a remedy to one of the main problems with VAEs: that the simplicity of the approximating family hinders them from learning complex posteriors. The fundamental principle underlying all these flows is that a fixed tractable initial density can be transformed via a sequence of simple transformations into a more flexible one. We compared three kinds of flow on the MNIST dataset: unconstrained residual flow, inverse autoregressive flow and Householder flow. Using inverse autoregressive flow, we were able to improve upon a baseline VAE that used a diagonal Gaussian for the approximate posterior. However, the other methods failed to outstrip the baseline. In particular, unconstrained residual flow proved to be difficult to train, in line with its greater expressivity. Moreover, we found that increasing the flow length did not result in much greater performance, but instead encouraged the encoder to output a much broader initial density that was subsequently refined by the flow.

Although inspecting the Jacobian terms revealed the contractive nature of the flow methods, we did not examine the changes in covariance *structure* as the flow progressed. Future work could try to characterize the sequence of distributions, and see e.g. what changes to the initial density are made in early parts of the flow versus later parts. In addition, it would be interesting to study the trade-off between flow-length and the complexity of an individual normalizing transformation. It could be that for sufficiently long flow lengths, relatively inexpressive transformations suffice, and that this is preferable to a short flow with more expressive transformations.

## References


[1] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. Generating Sentences from a Continuous Space. *International Conference on Learning Representations*, pages 1–13, 2016.

[2] Mathieu Germain, Karol Gregor, Iain Murray, Hugo Larochelle, I Murray Ed, and A C Uk. MADE: Masked Autoencoder for Distribution Estimation. *Proceedings of The 32nd International Conference on Machine Learning*, 37:881–889, 2015.

[3] Karol Gregor, Ivo Danihelka, Alex Graves, and Daan Wierstra. DRAW: A Recurrent Neural Network For Image Generation. *Proceedings of the 32nd International Conference on Machine Learning*, pages 1–16, 2015.

[4] G Hinton, P Dayan, B J Frey, and R M Neal. The Wake-Sleep Algorithm for Unsupervised Neural Networks. *Science*, 268(5214):1158–1161, 1995.

[5] Matthew D Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic Variational Inference. *Journal of Machine Learning Research*, 14:1303–1347, 2013.

[6] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *Proceedings of The 32nd International Conference on Machine Learning*, 2015.

[7] Diederik P Kingma and Lei Jimmy Ba. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*, pages 1–13, 2014.

[8] Diederik P Kingma, Danilo J Rezende, Shakir Mohamed, and Max Welling. Semi-Supervised Learning with Deep Generative Models. *Advances in Neural Information Processing Systems*, pages 1–9, 2014.

[9] Diederik P Kingma, Tim Salimans, and Max Welling. Improving Variational Inference with Inverse Autoregressive Flow. *Advances in Neural Information Processing Systems*, pages 1–8, 2016.

[10] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*, pages 1–14, 2014.

[11] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. *arXiv preprint*, 2016.

[12] Danilo J Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *Proceedings of the 31st International Conference on Machine Learning*, 32:1278–1286, 2014.

[13] Danilo Jimenez Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. *Proceedings of the 32nd International Conference on Machine Learning*, 37:1530–1538, 2015.

[14] Ruslan Salakhutdinov and Geoffrey Hinton. Deep Boltzmann Machines. *Artificial Intelligence*, 5(2):448–455, 2009.

[15] Esteban G. Tabak and Cristina V. Turner. A Family of Nonparametric Density Estimation Algorithms. *Communications on Pure and Applied Mathematics*, LXVI:145–164, 2013.

[16] Esteban G. Tabak and Eric Vanden-Eijnden. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217–233, 2010.

[17] Jakub M. Tomczak and Max Welling. Improving Variational Auto-Encoders using Householder Flow. *NIPS Workshop on Bayesian Variational Inference*, 2016.