Improving Variational Inference with Inverse Autoregressive Flow

Diederik P. Kingma dpkingma@openai.com
Tim Salimans tim@openai.com
Rafal Jozefowicz rafal@openai.com
Xi Chen peter@openai.com

Ilya Sutskever ilya@openai.com
Max Welling* M.Welling@uva.nl

Abstract

We propose a simple and scalable method for improving the flexibility of variational inference through a transformation with autoregressive neural networks. Autoregressive neural networks, such as RNNs or the PixelCNN (van den Oord et al., 2016b), are very powerful models and potentially interesting for use as variational posterior approximation. However, ancestral sampling in such networks is a long sequential operation, and therefore typically very slow on modern parallel hardware, such as GPUs. We show that by *inverting* autoregressive neural networks we can obtain equally powerful posterior models from which we can sample efficiently on modern GPUs. We show that such data transformations, *inverse autoregressive flows* (IAF), can be used to transform a simple distribution over the latent variables into a much more flexible distribution, while still allowing us to compute the resulting variables' probability density function. The method is simple to implement, can be made arbitrarily flexible and, in contrast with previous work, is well applicable to models with high-dimensional latent spaces, such as convolutional generative models.

The method is applied to a novel deep architecture of variational auto-encoders. In experiments with natural images, we demonstrate that autoregressive flow leads to significant performance gains.

1 Introduction

Stochastic variational inference (Blei et al., 2012; Hoffman et al., 2013) is a method for scalable posterior inference with large datasets using stochastic gradient ascent. It can be made especially efficient for continuous latent variables through latent-variable reparameterization and inference networks, amortizing the cost, resulting in a highly scalable learning procedure (Kingma and Welling, 2013; Rezende et al., 2014; Salimans et al., 2014). When using neural networks for both the inference network and generative model, this results in class of models called variational auto-encoders (VAEs). In this paper we propose *inverse autoregressive flow* (IAF), a method for improving the flexibility of inference networks.

At the core of our proposed method lie autoregressive functions that are normally used for density estimation: functions that take as input a variable with some specified ordering such as multidimensional tensors, and output a mean and standard deviation for each element of the input variable conditioned on the previous elements. Examples of such functions are autoregressive neural density estimators such as RNNs, MADE (Germain et al., 2015), PixelCNN (van den Oord et al., 2016b)

*University of Amsterdam, University of California Irvine, and the Canadian Institute for Advanced Research (CIFAR)

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

or WaveNet (van den Oord et al., 2016a) models. We show that such functions can often be easily turned into invertible nonlinear transformations of the input, with a simple Jacobian determinant. Since the transformation is flexible and the determinant known, it can be used to transform a tensor with relatively simple known density, into a new tensor with more complicated density that is still cheaply computable. In contrast with most previous work on improving inference models including previously used *normalizing flows* (Rezende and Mohamed, 2015), this transformation is well suited to high-dimensional tensor variables, such as spatio-temporally organized variables.

We demonstrate this method by improving inference networks of deep variational auto-encoders. In particular, we train deep variational auto-encoders with latent variables at multiple levels of the hierarchy, where each stochastic variable is a 3D tensor (a stack of featuremaps), and demonstrate improved performance.

2 Variational Inference and Learning

Let $\mathbf{x}$ be a (set of) observed variable(s), $\mathbf{z}$ a (set of) latent variable(s) and let $p(\mathbf{x}, \mathbf{z})$ be the parametric model of their joint distribution, called the *generative model* defined over the variables. Given a dataset $\mathbf{X} = \{\mathbf{x}^1, ..., \mathbf{x}^N\}$ we typically wish to perform maximum marginal likelihood learning of its parameters, i.e. to maximize

$$\log p(\mathbf{X}) = \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)}), \tag{1}$$

but in general this marginal likelihood is intractable to compute or differentiate directly for flexible generative models, e.g. when components of the generative model are parameterized by neural networks. A solution is to introduce $q(\mathbf{z}|\mathbf{x})$, a parametric *inference model* defined over the latent variables, and optimize the *variational lower bound* on the marginal log-likelihood of each observation $\mathbf{x}$:

$$\log p(\mathbf{x}) \leq \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} \left[\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})\right] = \mathcal{L}(\mathbf{x}; \theta) \tag{2}$$

where θ indicates the parameters of p and q models. Keeping in mind that Kullback-Leibler divergences $D_{KL}(.)$ are non-negative, it's clear that $\mathcal{L}(\mathbf{x}; \theta)$ is a lower bound on $\log p(\mathbf{x})$ since it can be written as follows):

$$\mathcal{L}(\mathbf{x}; \theta) = \log p(\mathbf{x}) - D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) \tag{3}$$

There are various ways to optimize the lower bound $\mathcal{L}(\mathbf{x}; \theta)$; for continuous $\mathbf{z}$ it can be done efficiently through a re-parameterization of $q(\mathbf{z}|\mathbf{x})$, see e.g. (Kingma and Welling, 2013; Rezende et al., 2014).

As can be seen from equation (3), maximizing $\mathcal{L}(\mathbf{x}; \theta)$ w.r.t. θ will concurrently maximize $\log p(\mathbf{x})$ and minimize $D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x}))$. The closer $D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x}))$ is to 0, the closer $\mathcal{L}(\mathbf{x}; \theta)$ will be to $\log p(\mathbf{x})$, and the better an approximation our optimization objective $\mathcal{L}(\mathbf{x}; \theta)$ is to our true objective $\log p(\mathbf{x})$. Also, minimization of $D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x}))$ can be a goal in itself, if we're interested in using $q(\mathbf{z}|\mathbf{x})$ for inference after optimization. In any case, the divergence $D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x}))$ is a function of our parameters through both the inference model and the generative model, and increasing the flexibility of either is generally helpful towards our objective.

Note that in models with multiple latent variables, the inference model is typically factorized into partial inference models with some ordering; e.g. $q(\mathbf{z}^a, \mathbf{z}^b|\mathbf{x}) = q(\mathbf{z}^a|\mathbf{x})q(\mathbf{z}^b|\mathbf{z}^a, \mathbf{x})$. We'll write $q(\mathbf{z}|\mathbf{x}, \mathbf{c})$ to denote such partial inference models, conditioned on both the data $\mathbf{x}$ and a further context $\mathbf{c}$ which includes the previous latent variables according to the ordering.

2.1 Requirements for Computational Tractability

Requirements for the inference model, in order to be able to efficiently optimize the bound, are that it is computationally cheap to (1) compute and differentiate its probability density $q(\mathbf{z}|\mathbf{x})$, and (2) to sample from, since these operations need to be performed for each datapoint in a minibatch at every iteration of optimization. If $\mathbf{z}$ is high-dimensional and we want to make efficient use of parallel computational resources like GPUs, then (3) parallelizability of these operations across dimensions of $\mathbf{z}$ is a large factor towards efficiency. These three requirements restrict the class of approximate posteriors $q(\mathbf{z}|\mathbf{x})$ that are practical to use. In practice this often leads to the use of diagonal posteriors,

e.g. $q(\mathbf{z}|\mathbf{x}) \sim \mathcal{N}(\mu(\mathbf{x}), \sigma^2(\mathbf{x}))$, where $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ are often nonlinear functions parameterized by neural networks. However, as explained above, we also need the density $q(\mathbf{z}|\mathbf{x})$ to be sufficiently flexible to match the true posterior $p(\mathbf{z}|\mathbf{x})$.

2.2 Normalizing Flow

Normalizing Flow (NF), investigated by (Rezende and Mohamed, 2015) in the context of stochastic gradient variational inference, is a powerful framework for building flexible posterior distributions through an iterative procedure. The general idea is to start off with an initial random variable with a simple distribution, and then apply a series of invertible parameterized transformations $\mathbf{f}^t$, such that the last iterate $\mathbf{z}^T$ has a more flexible distribution:

$$\mathbf{z}^0 \sim q(\mathbf{z}^0|\mathbf{x}, \mathbf{c}), \quad \mathbf{z}^t = \mathbf{f}^t(\mathbf{z}^{t-1}, \mathbf{x}, \mathbf{c}) \quad \forall t = 1...T \tag{4}$$

As long as the Jacobian determinant of each of the transformations $\mathbf{f}^t$ can be computed, we can still compute the probability density function of the last iterate:

$$\log q(\mathbf{z}^T|\mathbf{x}, \mathbf{c}) = \log q(\mathbf{z}^0|\mathbf{x}, \mathbf{c}) - \sum_{t=1}^{T} \log \det \left| \frac{d\mathbf{f}^t(\mathbf{z}^{t-1}, \mathbf{x}, \mathbf{c})}{d\mathbf{z}^{t-1}} \right| \tag{5}$$

However, (Rezende and Mohamed, 2015) experiment with only a very limited family of such invertible transformation with known Jacobian determinant, namely:

$$\mathbf{f}^t(\mathbf{z}^{t-1}) = \mathbf{z}^{t-1} + \mathbf{u}h(\mathbf{w}^T\mathbf{z}^{t-1} + b) \tag{6}$$

where $\mathbf{u}$ and $\mathbf{w}$ are vectors, $\mathbf{w}^T$ is $\mathbf{w}$ transposed, b is a scalar and $h(.)$ is a nonlinearity, such that $\mathbf{u}h(\mathbf{w}^T\mathbf{z}_{t-1} + b)$ can be interpreted as a MLP with a bottleneck hidden layer with a single unit. Since information goes through the single bottleneck, a series of many such transformations is required to capture high-dimensional dependencies.

Another approach, taken by (Dinh et al., 2014), is to transform the data using shearing transformations, which have a fixed Jacobian matrix determinant of one. Typically, this type of transformations updates half of the latent variables $\mathbf{z}^{1,...,D/2}$ per step by adding a vector $\delta_(\mathbf{z}^{D/2+1,...,D})$ which is a parameterized function of the remaining latent variables $\mathbf{z}^{D/2+1,...,D}$ which are kept fixed. Such a transformation admits completely general transformations $\delta(.)$, but the requirement to partition the set of latent variables is still very limiting. Indeed, (Rezende and Mohamed, 2015) find that this type of transformation is generally less powerful than the normalizing flow presented above.

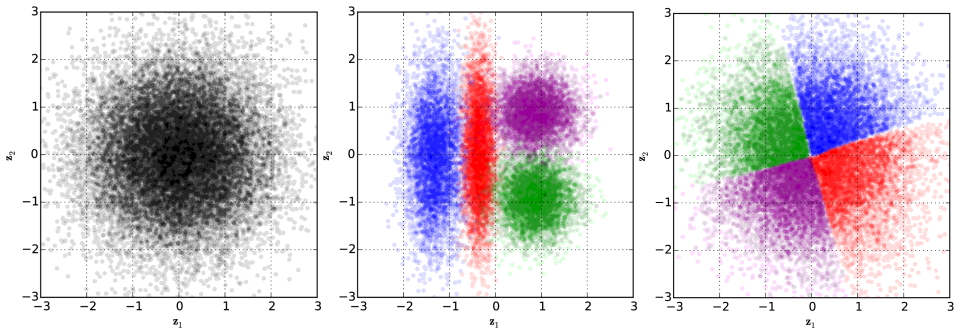
A potentially more powerful transformation is the *Hamiltonian flow* used in Hamiltonian Variational Inference (Salimans et al., 2014). Here, a transformation is generated by simulating the flow of a Hamiltonian system consisting of the latent variables $\mathbf{z}$, and a set of auxiliary variables. This type of transformation has the additional benefit that it is guided by the exact posterior distribution, and that it leaves this distribution invariant for small step sizes. Such as transformation could thus take us arbitrarily close to the exact posterior distribution if we can apply it for a sufficient number of times. In practice, however, Hamiltonian Variational Inference is very demanding computationally. Also, it requires an auxiliary variational bound to account for the auxiliary variables, which can impede progress if the bound is not sufficiently tight.

So far, no single method has been proposed that is both powerful and computationally cheap, and that satisfies all three of the computational tractability criteria of section 2.1.

3 Autoregressive Whitening of Data

In order to find a type of normalizing flow that is both powerful and computationally cheap, we consider a conceptually simple family of autoregressive Gaussian generative models which we will now briefly introduce. Let $\mathbf{y} = \{y_i\}_{i=1}^D$ be some random vector (or tensor) with some ordering on its elements. On this vector we define an autoregressive Gaussian generative model:

$$\begin{aligned} y_0 &= \mu_0 + \sigma_0 \cdot z_0, \\ y_i &= \mu_i(\mathbf{y}_{1:i-1}) + \sigma_i(\mathbf{y}_{1:i-1}) \cdot z_i, \\ z_i &\sim \mathcal{N}(0, 1) \quad \forall i, \end{aligned} \tag{7}$$



(a) Prior distribution (b) Posteriors in standard VAE (c) Posteriors in VAE with IAF

Figure 1: Best viewed in color. We fitted a variational auto-encoder (VAE) with a spherical Gaussian prior, and with factorized Gaussian posteriors **(b)** or inverse autoregressive flow (IAF) posteriors **(c)** to a toy dataset with four datapoints. Each colored cluster corresponds to the posterior distribution of one datapoint. IAF greatly improves the flexibility of the posterior distributions, and allows for a much better fit between the posteriors and the prior.

where $\mu_i(\mathbf{y}_{1:i-1})$ and $\sigma_i(\mathbf{y}_{1:i-1})$ are general functions, e.g. neural networks with parameters θ, that take the previous elements of $\mathbf{y}$ as input and map them to a predicted mean and standard deviation for each element of $\mathbf{y}$. Models of this type include LSTMs (Hochreiter and Schmidhuber, 1997) predicting a mean and/or variance over input data, and (locally connected or fully connected) Gaussian MADE models (Germain et al., 2015). This is a rich class of very powerful models, but the disadvantage of using autoregressive models for variational inference is that the elements y_i have to be generated sequentially, which is slow when implemented on a GPU.

The autoregressive model effectively transforms the vector $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ to a vector $\mathbf{y}$ with a more complicated distribution. As long as we have $\sigma_i > 0 \, \forall i$, this type of transformation is one-to-one, and can be inverted using

$$z_i = \frac{y_i - \mu_i(\mathbf{y}_{1:i-1})}{\sigma_i(\mathbf{y}_{1:i-1})}. \tag{8}$$

This inverse transformation *whitens* the data, turning the vector $\mathbf{y}$ with complicated distribution back into a vector $\mathbf{z}$ where each element is independently identically distributed according to the standard normal distribution. This inverse transformation is equally powerful as the autoregressive transformation above, but it has the crucial advantage that the individual elements z_i can now be calculated in parallel, as the z_i do not depend on each other given $\mathbf{y}$. The transformation $\mathbf{y} \to \mathbf{z}$ can thus be completely vectorized:

$$\mathbf{z} = (\mathbf{y} - \mu(\mathbf{y}))/\sigma(\mathbf{y}), \tag{9}$$

where the subtraction and division are elementwise. Unlike when using the autoregressive models naively, the *inverse* autoregressive transformation is thus both powerful *and* computationally cheap making this type of transformation appropriate for use with variational inference.

A key observation, important for variational inference, is that the autoregressive whitening operation explained above has a lower triangular Jacobian matrix, whose diagonal elements are the elements of $\sigma(\mathbf{y})$. Therefore, the log-determinant of the Jacobian of the transformation is simply minus the sum of the log-standard deviations used by the autoregressive Gaussian generative model:

$$\log \det \left| \frac{d\mathbf{z}}{d\mathbf{y}} \right| = -\sum_{i=1}^{D} \log \sigma_i(\mathbf{y}) \tag{10}$$

which is computationally cheap to compute, and allows us to evaluate the variational lower bound.

4 Inverse Autoregressive Flow (IAF)

As a result, we can use inverse autoregressive transformations (eq. (9)) as a type of normalizing flow (eq. (5)) for variational inference, which we call *inverse autoregressive flow* (IAF). If this

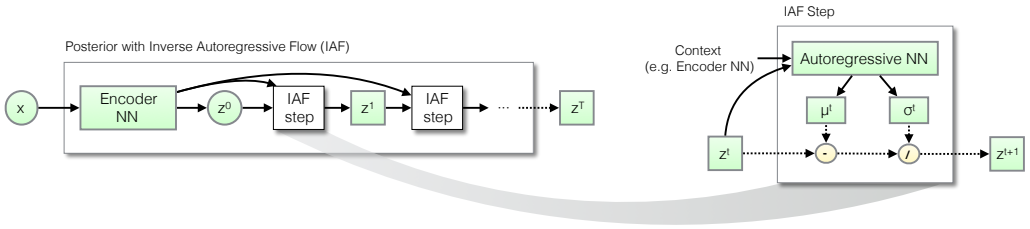


Figure 2: Like other normalizing flows. our posterior with Inverse Autoregressive Flow (IAF) consists of a iterative parameterized transformation of an initial sample $\mathbf{z}^0$, drawn from a standard Gaussian with diagonal covariance, into a final iterate $\mathbf{z}^T$ whose distribution is much more flexible. Each step in IAF performs a flexible inverse autoregressive transformation.

transformation is used to match samples from an approximate posterior to a prior $p(\mathbf{z})$ that is standard normal, the transformation will indeed whiten, but otherwise the transformation can produce very general output distributions.

When using inverse autoregressive flow in our posterior approximation, we use a factorized Gaussian distribution for $\mathbf{z}^0 \sim q(\mathbf{z}^0|\mathbf{x}, \mathbf{c})$, where $\mathbf{c}$ is some optional context. We then perform T steps op IAF:

$$\text{for } t = 1...T: \quad \mathbf{z}^t = \mathbf{f}^t(\mathbf{z}^{t-1}, \mathbf{x}, \mathbf{c}) = (\mathbf{z}^{t-1} - \mu^t(\mathbf{z}^{t-1}, \mathbf{x}, \mathbf{c}))/\sigma^t(\mathbf{z}^{t-1}, \mathbf{x}, \mathbf{c}), \tag{11}$$

where at every step we use a differently parameterized autoregressive model $\mathcal{N}(\mu^t, \sigma^t)$. If these models are sufficiently powerful, the final iterate $\mathbf{z}^T$ will have a flexible distribution that can be closely fitted to the true posterior. See also figure 2. Some examples are given below.

Perhaps the simplest special case of IAF is the transformation of a Gaussian variable with diagonal covariance to one with linear dependencies. See appendix A for an explanation.

4.1 IAF through Masked Autoencoders (MADE)

For introducing nonlinear dependencies between the elements of $\mathbf{z}$, we specify the autoregressive Gaussian model $\mu_\theta, \sigma_\theta$ using the family of deep masked autoencoders (Germain et al., 2015). These models are arbitrarily flexible neural networks, where masks are applied to the weight matrices in such a way that the output $\mu(\mathbf{y}), \sigma(\mathbf{y})$ is autoregressive, i.e. $\partial\mu_i(\mathbf{y})/\partial y_j = 0, \partial\sigma_i(\mathbf{y})/\partial y_j = 0$ for $j \geq i$. Such models can implement very flexible densities, while still being computionally efficient, and they can be specified in either a fully connected (Germain et al., 2015), or convolutional way (van den Oord et al., 2016b,c).

In experiments, we use one or two iterations of IAF with masked autoencoders, with reverse variable ordering in case of two iterations. The initial iterate, $\mathbf{z}^0$, is the sample from a diagonal Gaussian posterior (like in (Kingma and Welling, 2013)). In each iteration of IAF, we then compute the mean and standard deviation through its masked autoencoder, and apply the transformation of equation (11). The final variable will have a highly flexible distribution, which we use as our variational posterior. For CIFAR-10, we use masked convolutional autoencoders (Germain et al., 2015; van den Oord et al., 2016b).

4.2 IAF through Recurrent Neural Networks

Another method of parameterizing the $\mu_\theta, \sigma_\theta$ that define our inverse autoregressive flow are LSTMs and other recurrent neural networks (RNNs). These are generally more powerful than the masked autoencoder models, as they have an unbounded context window in computing the conditional means and variances. The downside of RNNs is that computation of $(\mu_\theta, \sigma_\theta)$ cannot generally be parallized well, relative to masked autoencoder models.

5 Related work

Inverse autoregressive flow (IAF) is a member of the family of normalizing flows, first discussed in (Rezende and Mohamed, 2015) in the context of stochastic variational inference. In (Rezende and

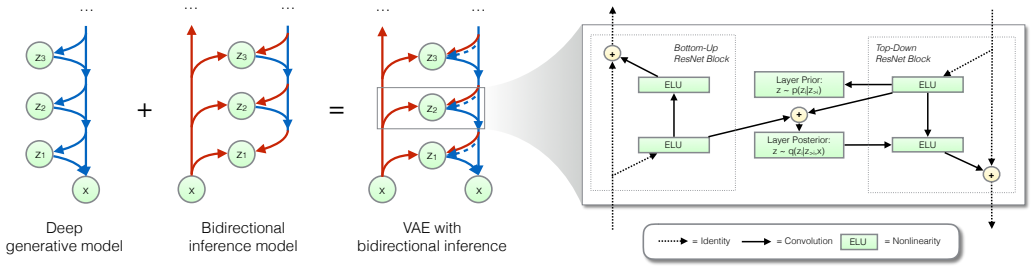


Figure 3: Overview of our ResNet VAE with bidirectional inference. The posterior of each layer is parameterized by its own IAF.

Mohamed, 2015) two specific types of flows are introduced: planar flows and radial flows. These flows are shown to be effective to problems relatively low-dimensional latent space (at most a few hundred dimensions). It is not clear, however, how to scale such flows to much higher-dimensional latent spaces, such as latent spaces of generative models of /larger images, and how planar and radial flows can leverage the topology of latent space, as is possible with IAF. Volume-conserving neural architectures were first presented in in (Deco and Brauer, 1995), as a form of nonlinear independent component analysis.

An alternative method for increasing the flexiblity of the variational inference, is the introduction of auxiliary latent variables (Salimans et al., 2014; Ranganath et al., 2015; Tran et al., 2015) and corresponding auxiliary inference models. Two such methods are the variational Gaussian process (Tran et al., 2015) (VGP) and Hamiltonian variational inference (Salimans et al., 2014) (HVI). These methods were introduced in the context of vector latent spaces, and it is not clear how to extend them to high-dimensional latent spaces with spatio-temporal topological structure, like IAF. Also, latent variable models with multiple layers of stochastic variables, such as the one used in our experiments, are often equivalent to such auxiliary-variable methods. We combine deep latent variable models with IAF in our experiments, benefiting from both techniques.

6 Experiments

We empirically evaluate inverse autoregressive flow for inference and learning of generative models on the MNIST and CIFAR-10 datasets.

Please see appendix C for details on the architectures of the generative model and inference models. Code for reproducing key empirical results is available online[2].

6.1 MNIST

In this expermiment we follow a similar implementation of the convolutional VAE as in (Salimans et al., 2014) with ResNet (He et al., 2015) blocks. A single layer of Gaussian stochastic units of dimension 32 is used. To investigate how the expressiveness of approximate posterior affects performance, we report results of different IAF posteriors with varying degrees of expressiveness. We use a 2-layer MADE (Germain et al., 2015) to implement one IAF transformation, and we stack multiple IAF transformations with ordering reversed between every other transformation. We call an IAF posterior d-deep and w-wide if it has d IAF transformations and w hidden units for each MADE. Obviously, an IAF posterior becomes more expressive when it's deeper and wider.

Results: Table 1 shows results on MNIST for these types of posteriors. Results indicate that as approximate posterior becomes more expressive, generative modeling performance becomes better. Also worth noting is that an expressive approximate posterior also tightens variational lower bounds as expected, making the gap between variational lower bounds and marginal likelihoods smaller. By making IAF deep and wide enough, we can achieve best published log-likelihood on dynamically binarized MNIST: **-79.10**. On Hugo Larochelle's statically binarized MNIST, our VAE with deep

[2] `https://github.com/openai/iaf`

Table 1: Generative modeling results on the dynamically sampled binarized MNIST version used in previous publications (Burda et al., 2015). Shown are averages; the number between brackets are standard deviations across 5 optimization runs. The right column shows an importance sampled estimate of the marginal likelihood for each model with 128 samples. Best previous results are reproduced in the first segment: [1]: (Salimans et al., 2014) [2]: (Burda et al., 2015) [3]: (Kaae Sønderby et al., 2016) [4]: (Tran et al., 2015)

Model	VLB	$\log p(\mathbf{x}) \approx$
Convolutional VAE + HVI [1]	-83.49	-81.94
DLGM 2hl + IWAE [2]		-82.90
LVAE [3]		-81.74
DRAW + VGP [4]	-79.88	
Diagonal covariance	-84.08 ($\pm$ 0.10)	-81.08 ($\pm$ 0.08)
IAF ($\mathrm{Depth} = 2, \mathrm{Width} = 320$)	-82.02 ($\pm$ 0.08)	-79.77 ($\pm$ 0.06)
IAF ($\mathrm{Depth} = 2, \mathrm{Width} = 1920$)	-81.17 ($\pm$ 0.08)	-79.30 ($\pm$ 0.08)
IAF ($\mathrm{Depth} = 4, \mathrm{Width} = 1920$)	-80.93 ($\pm$ 0.09)	-79.17 ($\pm$ 0.08)
IAF ($\mathrm{Depth} = 8, \mathrm{Width} = 1920$)	-80.80 ($\pm$ 0.07)	**-79.10** ($\pm$ 0.07)

IAF achieves a log-likelihood of **-79.88**, which is slightly worse than the best reported result, **-79.2**, using the PixelCNN (van den Oord et al., 2016b).

6.2 CIFAR-10

We also evaluated IAF on the CIFAR-10 dataset of natural images. Natural images contain a much greater variety of patterns and structure than MNIST images; in order to capture this structure well, we experiment with a novel architecture, ResNet VAE, with many layers of stochastic variables, and based on residual convolutional networks (ResNets) (He et al., 2015, 2016). For details of our ResNet VAE architecture, please see figure 3 and appendix C and our code. The main benefits of this architecture is that it forms a flexible autoregressive prior latent space, while still being straightforward to sample from.

Log-likelihood. See table 2 for a comparison to previously reported results, and the appendix for more detailed experimental results. Our architecture with IAF outperforms all earlier latent-variable architectures. We suspect that the results can be further improved with more steps of flow, which we leave to future work.

Synthesis speed. While achieving comparable log-likelihoods as the PixelCNN and PixelRNN, the generative model in a VAE is faster to sample from, and doesn't require custom code. Sampling from the PixelCNN naïvely by sequentially generating a pixel at a time, using the full generative model at each iteration, took about **52.0 seconds/image** on a NVIDIA Titan X GPU. With custom code that only evaluates the relevant part of the network, PixelCNN sampling could be sped up, but due to its sequential nature, would probably still be far from efficient on a modern GPU. Efficient sampling from the ResNet VAE does not require custom code, and took about **0.05 seconds/image**.

7 Conclusion

We presented *inverse autoregressive flow* (IAF), a method to make the posterior approximation for variational inference more flexible, thereby improving the variational lower bound. By inverting the sequential data generating process of an autoregressive Gaussian model, inverse autoregressive flow gives us a data transformation that is both very powerful and computationally efficient: the transformation has a tractable Jacobian determinant, and it can be vectorized for implementation on a GPU. By applying this transformation to the samples from our approximate posterior we can move their distribution closer to the exact posterior.

Figure 4: Random samples from generative ResNet trained on the CIFAR-10 dataset of natural image patches.

Table 2: Our results with ResNet VAEs on CIFAR-10 images, compared to earlier results, in *average number of bits per data dimension* on the test set. The number for convolutional DRAW is an upper bound, while the ResNet VAE log-likelihood was estimated using importance sampling.

Method	bits/dim
Results with tractable likelihood models:	
Uniform distribution (van den Oord et al., 2016b)	8.00
Multivariate Gaussian (van den Oord et al., 2016b)	4.70
NICE (Dinh et al., 2014)	4.48
Deep GMMs (van den Oord and Schrauwen, 2014)	4.00
Real NVP (Dinh et al., 2016)	3.49
PixelRNN (van den Oord et al., 2016b)	**3.00**
Gated PixelCNN (van den Oord et al., 2016c)	**3.03**
Results with variationally trained latent-variable models:	
Deep Diffusion (Sohl-Dickstein et al., 2015)	5.4
Convolutional DRAW (Gregor et al., 2016)	3.58 (Var. Bound)
Ours (ResNet VAE with IAF)	**3.11**

We empirically demonstrated the usefulness of inverse autoregressive flow for variational inference by training a novel deep architecture of variational auto-encoders. In experiments we demonstrated that autoregressive flow leads to significant performance gains compared to similar models with factorized Gaussian approximate posteriors, and we report the best results on CIFAR-10 for latent-variable models so far.

Acknowledgements

We thank Jascha Sohl-Dickstein, Karol Gregor, and many others at Google Deepmind for interesting discussions. We thank Harri Valpola for referring us to Gustavo Deco's very relevant pioneering work on a form of inverse autoregressive flow applied to nonlinear independent component analysis.

References

Blei, D. M., Jordan, M. I., and Paisley, J. W. (2012). Variational Bayesian inference with Stochastic Search. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 1367–1374.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. (2015). Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Burda, Y., Grosse, R., and Salakhutdinov, R. (2015). Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2015). Fast and accurate deep network learning by Exponential Linear Units (ELUs). *arXiv preprint arXiv:1511.07289*.

Deco, G. and Brauer, W. (1995). Higher order statistical decorrelation without information loss. *Advances in Neural Information Processing Systems*, pages 247–254.

Dinh, L., Krueger, D., and Bengio, Y. (2014). Nice: non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2016). Density estimation using Real NVP. *arXiv preprint arXiv:1605.08803*.

Germain, M., Gregor, K., Murray, I., and Larochelle, H. (2015). Made: Masked autoencoder for distribution estimation. *arXiv preprint arXiv:1502.03509*.

Gregor, K., Besse, F., Rezende, D. J., Danihelka, I., and Wierstra, D. (2016). Towards conceptual compression. *arXiv preprint arXiv:1604.08772*.

Gregor, K., Mnih, A., and Wierstra, D. (2013). Deep AutoRegressive Networks. *arXiv preprint arXiv:1310.8499*.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Identity mappings in deep residual networks. *arXiv preprint arXiv:1603.05027*.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural computation*, 9(8):1735–1780.

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

Kaae Sønderby, C., Raiko, T., Maaløe, L., Kaae Sønderby, S., and Winther, O. (2016). How to train deep variational autoencoders and probabilistic ladder networks. *arXiv preprint arXiv:1602.02282*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational Bayes. *Proceedings of the 2nd International Conference on Learning Representations*.

Ranganath, R., Tran, D., and Blei, D. M. (2015). Hierarchical variational models. *arXiv preprint arXiv:1511.02386*.

Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 1530–1538.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1278–1286.

Salimans, T. (2016). A structured variational auto-encoder for learning deep hierarchies of sparse features. *arXiv preprint arXiv:1602.08734*.

Salimans, T. and Kingma, D. P. (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *arXiv preprint arXiv:1602.07868*.

Salimans, T., Kingma, D. P., and Welling, M. (2014). Markov chain Monte Carlo and variational inference: Bridging the gap. *arXiv preprint arXiv:1410.6460*.

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. *arXiv preprint arXiv:1503.03585*.

Tran, D., Ranganath, R., and Blei, D. M. (2015). Variational gaussian process. *arXiv preprint arXiv:1511.06499*.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016a). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

van den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. (2016b). Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*.

van den Oord, A., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., and Kavukcuoglu, K. (2016c). Conditional image generation with pixelcnn decoders. *arXiv preprint arXiv:1606.05328*.

van den Oord, A. and Schrauwen, B. (2014). Factoring variations in natural images with deep gaussian mixture models. In *Advances in Neural Information Processing Systems*, pages 3518–3526.

Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. *arXiv preprint arXiv:1605.07146*.

Zeiler, M. D., Krishnan, D., Taylor, G. W., and Fergus, R. (2010). Deconvolutional networks. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2528–2535. IEEE.