Sylvester Normalizing Flows for Variational Inference

Rianne van den Berg* University of Amsterdam

Leonard Hasenclever* University of Oxford

Jakub M. Tomczak University of Amsterdam

Max Welling University of Amsterdam

Abstract

Variational inference relies on flexible approximate posterior distributions. Normalizing flows provide a general recipe to construct flexible variational posteriors. We introduce Sylvester normalizing flows, which can be seen as a generalization of planar flows. Sylvester normalizing flows remove the well-known single-unit bottleneck from planar flows, making a single transformation much more flexible. We compare the performance of Sylvester normalizing flows against planar flows and inverse autoregressive flows and demonstrate that they compare favorably on several datasets.

1 INTRODUCTION

Stochastic variational inference (Hoffman et al., 2013) allows for posterior inference in increasingly large and complex problems using stochastic gradient ascent. In continuous latent variable models, variational inference can be made particularly efficient through the amortized inference, in which inference networks amortize the cost of calculating the variational posterior for a data point (Gershman and Goodman, 2014). A particularly successful class of models is the variational autoencoder (VAE) in which both the generative model and the inference network are given by neural networks, and sampling from the variational posterior is efficient through the non-centered parameterization (Kingma and Welling, 2014), also known as the reparameterization trick (Kingma and Welling, 2013; Rezende et al., 2014).

Despite its success, variational inference has drawbacks compared to other inference methods such as MCMC. Variational inference searches for the best posterior approximation within a parametric family of distributions. Hence, the true posterior distribution can only be recovered exactly if it happens to be in the chosen family. In particular, with widely used simple variational families such as diagonal covariance Gaussian distributions, the variational approximation is likely to be insufficient. More complex variational families enable better posterior approximations, resulting in improved model performance. Therefore, designing tractable and more expressive variational families is an important problem in variational inference (Nalisnick et al., 2016; Salimans et al., 2015; Tran et al., 2015).

Rezende and Mohamed (2015) introduced a general framework for constructing more flexible variational distributions, called normalizing flows. Normalizing flows transform a base density through a number of invertible parametric transformations with tractable Jacobians into more complicated distributions. They proposed two classes of normalizing flows: planar flows and radial flows. While effective for small problems, these can be hard to train and often many transformations are required to get good performance. For planar flows, Kingma et al. (2016) argue that this is due to the fact that the transformation used acts as a bottleneck, warping one direction at a time. Having a large number of flows makes the inference network very deep and harder to train, empirically resulting in suboptimal performance. Kingma et al. (2016) proposed inverse auto-regressive flows (IAF), achieving state of the art results on dynamically binarized MNIST at the time of publication. While very successful, IAFs require a very large number of parameters. Due to the large number of parameters IAFs cannot amortize all flow parameters. Instead amortization is achieved through an additional context vector that is fed into each flow step.

Paper contribution In this paper, we use Sylvester's determinant identity to introduce Sylvester normalizing flows (SNFs). This family of flows is a generalization

[*] Equal contribution.

of planar flows, removing the bottleneck. We compare a number of different variants of SNFs and show that they compare favorably against planar flows and IAFs. We show that one specific variant of SNFs is related to IAFs, while requiring many fewer parameters due to direct amortization of all flow parameters.

2 VARIATIONAL INFERENCE

Consider a probabilistic model with observations $\mathbf{x}$ and continuous latent variables $\mathbf{z}$ and model parameters θ. In generative modeling we are often interested in performing maximum (marginal) likelihood learning of the parameters θ of the latent-variable model $p_\theta(\mathbf{x}, \mathbf{z})$. This requires marginalization over the unobserved latent variables $\mathbf{z}$. Unfortunately, this integration is generally intractable. Variational inference (Jordan et al., 1999) instead introduces a variational approximation $q_\phi(\mathbf{z}|\mathbf{x})$ to the posterior with learnable parameters ϕ, to construct a lower bound on the log marginal likelihood:

$$\begin{aligned}\log p_\theta(\mathbf{x}) &\geq \log p_\theta(\mathbf{x}) - \mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \,||\, p(\mathbf{z}|\mathbf{x})) \quad (1)\\ &= \mathbb{E}_{q_\phi}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \,||\, p(\mathbf{z})) \quad (2)\\ &=: -\mathcal{F}(\theta, \phi) \quad (3)\end{aligned}$$

This bound is known as the evidence lower bound (ELBO) and $\mathcal{F}$ is referred to as the variational free energy. In equation (2), the first term represents the reconstruction error, and the second term is the Kullback-Leibler (KL) divergence from the approximate posterior to the prior distribution, which acts as a regularizer. In this paper we consider variational autoencoders (VAEs), where both $p_\theta(\mathbf{x}|\mathbf{z})$ and $q_\phi(\mathbf{z}|\mathbf{x})$ are distributions whose parameters are given by neural networks. The parameters θ and ϕ of the generative model and inference model, respectively, are trained jointly through stochastic minimisation of $\mathcal{F}$ which can be made efficient through the reparameterization trick (Kingma and Welling, 2013; Rezende et al., 2014).

From equation (1) we see that the better the variational approximation to the posterior the tighter the ELBO. The simplest, but probably most widely used choice of variation distribution $q_\phi(\mathbf{z}|\mathbf{x})$ is diagonal-covariance Gaussians of the form $\mathcal{N}(\boldsymbol{\mu}(\mathbf{x}),\, \boldsymbol{\sigma}^2(\mathbf{x}))$

However, with such simple variational distributions the ELBO will be fairly loose, resulting in biased maximum likelihood estimates of the model parameters θ (see Fig. 2) and harming generative performance. Thus, for variational inference to work well, more flexible approximate posterior distributions are needed.

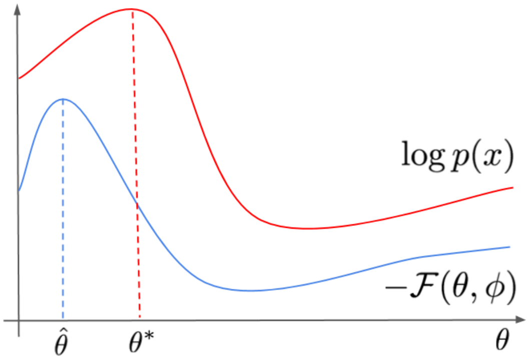

Figure 1: Since the ELBO is only a lower bound on the log marginal likelihood, they do not share the same local maxima. The looser the ELBO is the more this can bias maximum likelihood estimates of the model parameters.

2.1 NORMALIZING FLOWS

Rezende and Mohamed (2015) propose a way to construct more flexible posteriors by transforming a simple base distribution with a series of invertible transformations (known as normalizing flows) with easily computable Jacobians. The resulting transformed density after one such transformation f is as follows (Tabak and Turner, 2013; Tabak and Vanden-Eijnden, 2010):

$$p_1(\mathbf{z}') = p_0(\mathbf{z}) \left| \det \left(\frac{\partial f(\mathbf{z})}{\partial \mathbf{z}} \right) \right|^{-1}, \qquad (4)$$

where $\mathbf{z}' = f(\mathbf{z})$, $\mathbf{z}, \mathbf{z}' \in \mathbb{R}^D$ and $f : \mathbb{R}^D \mapsto \mathbb{R}^D$ is an invertible function. In general the cost of computing the Jacobian will be $\mathcal{O}(D^3)$. However, it is possible to design transformations with more efficiently computable Jacobians.

This strategy is used in variational inference as follows: first, a stochastic variable is drawn from a simple base posterior distribution such as a diagonal Gaussian $\mathcal{N}(\mathbf{z}_0|\boldsymbol{\mu}(\mathbf{x}), \boldsymbol{\sigma}^2(\mathbf{x}))$. The sample is then transformed with a number of flows. After applying K flows, the final latent stochastic variables are given by $\mathbf{z}_K = f_K \circ \ldots f_2 \circ f_1(\mathbf{z}_0)$. The corresponding log-density is then given by:

$$\begin{aligned}\log q_K(\mathbf{z}_K|\mathbf{x}) &= \log q_0(\mathbf{z}_0|\mathbf{x})\\ &\quad - \sum_{k=1}^{K} \log \left| \det \left(\frac{\partial f_k(\mathbf{z}_{k-1}; \lambda_k(\mathbf{x}))}{\partial \mathbf{z}_{k-1}} \right) \right|, \qquad (5)\end{aligned}$$

where λ_k are the parameters of the k-th transformation. Given variational posterior $q_\phi(\mathbf{z}|\mathbf{x}) = q_K(\mathbf{z}|\mathbf{x})$ parametrized by a normalizing flow of length K, the variational objective can be rewritten as:

$$\mathcal{F}(\theta, \phi) = \mathbb{E}_{q_\phi}[\log q_\phi(\mathbf{z}|\mathbf{x}) - \log p_\theta(\mathbf{x}, \mathbf{z})] \tag{6}$$

$$= \mathbb{E}_{q_0}[\log q_0(\mathbf{z}_0|\mathbf{x}) - \log p_\theta(\mathbf{x}, \mathbf{z})] - \mathbb{E}_{q_0}\left[\sum_{k=1}^{K} \log \left| \det \left(\frac{\partial f_k(\mathbf{z}_{k-1}; \lambda_k(\mathbf{x}))}{\partial \mathbf{z}_{k-1}} \right) \right| \right]. \tag{7}$$

Normalizing flows are normally used with amortized variational inference. Instead of learning variational parameters for each data point, both $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, as well as all the flow parameters are outputs of a deep neural network conditioned on $\mathbf{x}$. This is referred to as the inference network.

Rezende and Mohamed (2015) introduced a normalizing flow, called planar flow, for which the Jacobian determinant could be computed efficiently. A single transformation of the planar flow is given by:

$$\mathbf{z}' = \mathbf{z} + \mathbf{u}h(\mathbf{w}^T\mathbf{z} + b). \tag{8}$$

Here, $\mathbf{u}, \mathbf{w} \in \mathbb{R}^D$, $b \in \mathbb{R}$ and h is a suitable smooth activation function. Rezende and Mohamed (2015) show that for $h = \tanh$, transformations of this kind are invertible as long as $\mathbf{u}^T\mathbf{w} \geq -1$.

By the *Matrix determinant lemma* the Jacobian of this transformation is given by:

$$\det \frac{\partial \mathbf{z}'}{\partial \mathbf{z}} = \det \left(\mathbf{I} + \mathbf{u}h'(\mathbf{w}^T\mathbf{z} + b)\mathbf{w}^T \right) = 1 + \mathbf{u}^T h'(\mathbf{w}^T\mathbf{z} + b)\mathbf{w}, \tag{9}$$

where h' denotes the derivative of h and which can be computed in $O(D)$ time.

In practice, many planar flow transformations are required to transform a simple base distribution into a flexible distribution, especially for high dimensional latent spaces. Kingma et al. (2016) argue that this is related to the term $\mathbf{u}h(\mathbf{w}^T\mathbf{z} + b)$ in Eq. (8), which effectively acts as a single-neuron MLP. In the next section we will derive a generalization of planar flows, which does not have a single-neuron bottleneck, while still maintaining the property of an efficiently computable Jacobian determinant.

3 SYLVESTER NORMALIZING FLOWS

Consider the following more general transformation similar to a single layer MLP with M hidden units and a residual connection:

$$\mathbf{z}' = \mathbf{z} + \mathbf{A}h(\mathbf{B}\mathbf{z} + \mathbf{b}), \tag{10}$$

with $\mathbf{A} \in \mathbb{R}^{D \times M}, \mathbf{B} \in \mathbb{R}^{M \times D}$, $\mathbf{b} \in \mathbb{R}^M$, and $M \leq D$. The Jacobian determinant of this transformation can be obtained using Sylvester's determinant identity, which is a generalization of the matrix determinant lemma.

Theorem 1 (Sylvester's determinant identity)**.** *For all* $\mathbf{A} \in \mathbb{R}^{D \times M}, \mathbf{B} \in \mathbb{R}^{M \times D}$,

$$\det (\mathbf{I}_D + \mathbf{A}\mathbf{B}) = \det (\mathbf{I}_M + \mathbf{B}\mathbf{A}), \tag{11}$$

where $\mathbf{I}_M$ *and* $\mathbf{I}_D$ *are* M *and* D*-dimensional identity matrices, respectively.*

When $M < D$, the computation of the determinant of a $D \times D$ matrix is thus reduced to the computation of the determinant of an $M \times M$ matrix.

Using Sylvester's determinant identity, the Jacobian determinant of the transformation in Eq. (10) is given by:

$$\det \left(\frac{\partial \mathbf{z}'}{\partial \mathbf{z}} \right) = \det (\mathbf{I}_M + \operatorname{diag}(h'(\mathbf{B}\mathbf{z} + \mathbf{b})) \mathbf{B}\mathbf{A}). \tag{12}$$

Since Sylvester's determinant identity plays a crucial role in the proposed family of normalizing flows, we will refer to them as *Sylvester normalizing flows*.

3.1 PARAMETERIZATION OF A AND B

In general, the transformation in (10) will not be invertible. Therefore, we propose the following special case of the above transformation:

$$\mathbf{z}' = \mathbf{z} + \mathbf{Q}\mathbf{R}h(\tilde{\mathbf{R}}\mathbf{Q}^T\mathbf{z} + \mathbf{b}) = \phi(\mathbf{z}), \tag{13}$$

where $\mathbf{R}$ and $\tilde{\mathbf{R}}$ are upper triangular $M \times M$ matrices, and

$$\mathbf{Q} = (\mathbf{q}_1 \ldots \mathbf{q}_M)$$

with the columns $\mathbf{q}_m \in \mathbb{R}^D$ forming an orthonormal set of vectors. By theorem 1, the determinant of the Jacobian $\mathbf{J}$ of this transformation reduces to:

$$\det \mathbf{J} = \det \left(\mathbf{I}_M + \operatorname{diag}\left(h'(\tilde{\mathbf{R}}\mathbf{Q}^T\mathbf{z} + \mathbf{b}) \right) \tilde{\mathbf{R}}\mathbf{Q}^T\mathbf{Q}\mathbf{R} \right) = \det \left(\mathbf{I}_M + \operatorname{diag}\left(h'(\tilde{\mathbf{R}}\mathbf{Q}^T\mathbf{z} + \mathbf{b}) \right) \tilde{\mathbf{R}}\mathbf{R} \right), \tag{14}$$

which can be computed in $O(M)$, since $\tilde{\mathbf{R}}\mathbf{R}$ is also upper triangular. The following theorem gives a sufficient condition for this transformation to be invertible.

Theorem 2. *Let* $\mathbf{R}$ *and* $\tilde{\mathbf{R}}$ *be upper triangular matrices. Let* $h : \mathbb{R} \longrightarrow \mathbb{R}$ *be a smooth function with bounded, positive derivative. Then, if the diagonal entries of* $\mathbf{R}$ *and* $\tilde{\mathbf{R}}$ *satisfy* $r_{ii}\tilde{r}_{ii} > -1/\|h'\|_\infty$ *and* $\tilde{\mathbf{R}}$ *is invertible, the transformation given by* (13) *is invertible.*

Proof. **Case 1: R and $\tilde{\mathbf{R}}$ diagonal**

Recall that one-dimensional real functions with strictly positive derivatives are invertible. The columns of $\mathbf{Q}$ are orthonormal and span a subspace $\mathcal{W} = \text{span}\{\mathbf{q}_1, \ldots, \mathbf{q}_M\}$ of $\mathbb{R}^D$. Let $\mathcal{W}^\perp$ denote its orthogonal complement. We can decompose $\mathbf{z} = \mathbf{z}_{||} + \mathbf{z}_\perp$, where $\mathbf{z}_{||} \in \mathcal{W}$ and $\mathbf{z}_\perp \in \mathcal{W}^\perp$. Similarly we can decompose $\mathbf{z}' = \mathbf{z}'_{||} + \mathbf{z}'_\perp$. Clearly, $\mathbf{QR}h(\tilde{\mathbf{R}}\mathbf{Q}^T\mathbf{z} + \mathbf{b}) \in \mathcal{W}$. Hence ϕ only acts on $\mathbf{z}_{||}$ and $\mathbf{z}_\perp = \phi(\mathbf{z})_\perp = \mathbf{z}'_\perp$. Thus, it suffices to consider the effect of ϕ on $\mathbf{z}_{||}$. Multiplying (13) by $\mathbf{Q}^T$ from the left gives:

$$\begin{aligned}\underbrace{\mathbf{Q}^T\mathbf{z}'}_{\mathbf{v}'} &= \underbrace{\mathbf{Q}^T\mathbf{z}}_{\mathbf{v}} + \mathbf{R}h(\tilde{\mathbf{R}}\underbrace{\mathbf{Q}^T\mathbf{z}}_{\mathbf{v}} + \mathbf{b}) \\ &= (f_1(v_1), \ldots, f_M(v_M))^T, \end{aligned} \tag{15}$$

where the vectors $\mathbf{v}$ and $\mathbf{v}'$ are the respective coordinates of $\mathbf{z}_{||}$ and $\mathbf{z}'_{||}$ w.r.t. $\mathbf{q}_1, \ldots, \mathbf{q}_M$. The dimensions in (15) are completely independent and each dimension is transformed by a real function $f_i(v) = v + r_{ii}h(\tilde{r}_{ii}v + b_i)$. Consider a single dimension i of (15). Since $\|h'\|_\infty r_{ii}\tilde{r}_{ii} > -1$, we have $f_i'(v) > 0$ and thus f_i is invertible. Since all dimensions are independent and the transformation is invertible in each dimension we can find $f^{-1} : \mathcal{W} \to \mathcal{W}$ such that $\mathbf{z}_{||} = f^{-1}(\mathbf{z}'_{||})$. Hence we can write the inverse of ϕ as:

$$\phi^{-1}(\mathbf{z}') = \underbrace{\mathbf{z}'_\perp}_{\mathbf{z}_\perp} + \underbrace{f^{-1}(\mathbf{z}'_{||})}_{\mathbf{z}_{||}} = \mathbf{z}, \tag{16}$$

Case 2: R triangular, $\tilde{\mathbf{R}}$ diagonal

Let us now consider the case when $\mathbf{R}$ is an upper triangular matrix. By the argument for the diagonal case above, it suffices to consider the effect of the transformation in $\mathcal{W}$. Multiplying (13) by $\mathbf{Q}^T$ from the left gives:

$$\underbrace{\mathbf{Q}^T\mathbf{z}'}_{\mathbf{v}'} = \underbrace{\mathbf{Q}^T\mathbf{z}}_{\mathbf{v}} + \mathbf{R}h(\tilde{\mathbf{R}}\underbrace{\mathbf{Q}^T\mathbf{z}}_{\mathbf{v}} + \mathbf{b}) \tag{17}$$

where the vectors $\mathbf{v}$ and $\mathbf{v}'$ contain the respective coordinates of $\mathbf{z}_{||}$ and $\mathbf{z}'_{||}$ w.r.t. $\mathbf{q}_1, \ldots, \mathbf{q}_M$. As in the diagonal case consider the functions $f_i(v) = v + r_{ii}h(\tilde{r}_{ii}v + b_i)$. Since $\|h'\|_\infty r_{ii}\tilde{r}_{ii} > -1$, we have $f_i'(v) > 0$ and thus f_i is invertible. Let us rewrite (17) in terms of f_i:

$$v'_1 = f_1(v_1) + \sum_{j=2}^{M} r_{1j}h(\tilde{r}_{jj}v_j + b_j) \tag{18}$$

$$\cdots$$

$$v'_k = f_k(v_k) + \sum_{j=k+1}^{M} r_{kj}h(\tilde{r}_{jj}v_j + b_j) \tag{19}$$

$$\cdots$$

$$v'_M = f_M(v_M) \tag{20}$$

Since f_M is invertible we can write $v_M = f_M^{-1}(v'_M)$. Now suppose we have expressed $\{v_j, \forall j > k\}$ in terms of $\{v'_j, \forall j > k\}$. Then

$$\begin{aligned} f_k(v_k) &= \underbrace{v'_k - \sum_{j=k+1}^{M} r_{kj}h(\tilde{r}_{jj}v_j + b_j)}_{\text{some function of } \{v'_j, \forall j > k\}} \\ &=: g_k(v'_k, v'_{k+1}, \ldots, v'_M) \\ v_k &= f_k^{-1}(g_k(v'_k, v'_{k+1}, \ldots, v'_M)). \end{aligned} \tag{21}$$

Thus we have expressed $\{v_j, \forall j \geq k\}$ in terms of $\{v'_j, \forall j \geq k\}$. By induction, we can express $\{v_j, \forall j\}$ in terms of $\{v'_j, \forall j\}$ and hence the transformation is invertible.

Case 3: R and $\tilde{\mathbf{R}}$ triangular

Now consider the general case when $\tilde{\mathbf{R}}$ is triangular. As before we only need to consider the effect of the transformation in $\mathcal{W}$.

$$\underbrace{\mathbf{Q}^T\mathbf{z}'}_{\mathbf{v}'} = \underbrace{\mathbf{Q}^T\mathbf{z}}_{\mathbf{v}} + \mathbf{R}h(\tilde{\mathbf{R}}\underbrace{\mathbf{Q}^T\mathbf{z}}_{\mathbf{v}} + \mathbf{b}) \tag{22}$$

Let g be the function $g(\mathbf{v}) = \tilde{\mathbf{R}}\mathbf{v}$. By assumption, g is invertible with inverse g^{-1}. Multiplying (22) by $\tilde{\mathbf{R}}$ gives:

$$g(\mathbf{v}') = \underbrace{g(\mathbf{v}) + \tilde{\mathbf{R}}\mathbf{R}h(g(\mathbf{v}) + \mathbf{b})}_{=: f(g(\mathbf{v}))} \tag{23}$$

Since $\tilde{\mathbf{R}}\mathbf{R}$ is upper triangular with diagonal entries $\tilde{r}_{jj}r_{jj}$, f is covered by case 2 considered before and is invertible. Thus, $\mathbf{v}$ can be written as:

$$\mathbf{v} = g^{-1}(f^{-1}(g(\mathbf{v}'))). \tag{24}$$

Hence the transformation in (22) is invertible. □

3.2 PRESERVING ORTHOGONALITY OF Q

Orthogonality is a convenient property, mathematically, but hard to achieve in practice. In this paper we consider three different flows based on the theorem above and various ways to preserve the orthogonality of $\mathbf{Q}$. The first two use explicit differentiable constructions of orthogonal matrices, while the third variant assumes a specific fixed permutation matrix as the orthogonal matrix.

Orthogonal Sylvester flows. First, we consider a Sylvester flow using matrices with M orthogonal columns (O-SNF). In this flow we can choose $M < D$, and thus introduce a flexible bottleneck. Similar to (Hasenclever et al., 2017), we ensure orthogonality of

$\mathbf{Q}$ by applying the following differentiable iterative procedure proposed by (Björck and Bowie, 1971; Kovarik, 1970):

$$\mathbf{Q}^{(k+1)} = \mathbf{Q}^{(k)} \left(\mathbf{I} + \frac{1}{2} \left(\mathbf{I} - \mathbf{Q}^{(k)\top} \mathbf{Q}^{(k)} \right) \right). \quad (25)$$

with a sufficient condition for convergence given by $\|\mathbf{Q}^{(0)\top}\mathbf{Q}^{(0)} - \mathbf{I}\|_2 < 1$. Here, the 2-norm of a matrix $\mathbf{X}$ refers to $\|\mathbf{X}\|_2 = \lambda_{\max}(\mathbf{X})$, with $\lambda_{\max}(\mathbf{X})$ representing the largest singular value of $\mathbf{X}$. In our experimental evaluations we ran the iterative procedure until $\|\mathbf{Q}^{(k)\top}\mathbf{Q}^{(k)} - \mathbf{I}\|_F \leq \epsilon$, with $\|\mathbf{X}\|_F$ the Frobenius norm, and ϵ a small convergence threshold. We observed that running this procedure up to 30 steps was sufficient to ensure convergence with respect to this threshold. To minimize the computational overhead introduced by orthogonalization we perform this orthogonalization in parallel for all flows.

Since this orthogonalization procedure is differentiable, it allows for the calculation of gradients with respect to $\mathbf{Q}^{(0)}$ by backpropagation, allowing for any standard optimization scheme such as stochastic gradient descent to be used for updating the flow parameters.

Householder Sylvester flows. Second, we study Householder Sylvester flows (H-SNF) where the orthogonal matrices are constructed by products of Householder reflections. Householder transformations are reflections about hyperplanes. Let $\mathbf{v} \in \mathbb{R}^D$, then the reflection about the hyperplane orthogonal to $\mathbf{v}$ is given by:

$$H(\mathbf{z}) = \mathbf{z} - \frac{\mathbf{v}\mathbf{v}^T}{\|\mathbf{v}\|^2}\mathbf{z} \quad (26)$$

It is worth noting that performing a single Householder transformation is very cheap to compute, as it only requires D parameters. Chaining together several Householder transformations results in more general orthogonal matrices, and it can be shown (Bischof and Sun, 1997; Sun and Bischof, 1995) that any $M \times M$ orthogonal matrix can be written as the product of $M-1$ Householder transformations. In our Householder Sylvester flow, the number of Householder transformations H is a hyperparameter that trades off the number of parameters and the generality of the orthogonal transformation. Note that the use of Householder transformations forces us to use $M = D$, since Householder transformation result in square matrices.

Triangular Sylvester flows. Third, we consider a triangular Sylvester flow (T-SNF), in which all orthogonal matrices $\mathbf{Q}$ alternate per transformation between the identity matrix and the permutation matrix corresponding to reversing the order of $\mathbf{z}$. This is equivalent to alternating between lower and upper triangular $\tilde{\mathbf{R}}$ and $\mathbf{R}$ for each flow.

3.3 AMORTIZING FLOW PARAMETERS

When using normalizing flows in an amortized inference setting, the parameters of the base distribution as well as the flow parameters can be functions of the data point $\mathbf{x}$ (Rezende and Mohamed, 2015). Figure 2 (left) shows a diagram of one SNF step and the amortization procedure. The inference network takes datapoints $\mathbf{x}$ as input, and provides as an output the mean and variance of $\mathbf{z}^0$ such that $\mathbf{z}^0 \sim \mathcal{N}(\mathbf{z}|\mu^0, \sigma^0)$. Several SNF transformations are then applied to $\mathbf{z}^0 \to \mathbf{z}^1 \to \ldots \mathbf{z}^K$, producing a flexible posterior distribution for $\mathbf{z}^K$. All of the flow parameters ($\mathbf{R}$, $\tilde{\mathbf{R}}$ and $\mathbf{Q}$ for each transformation) are produced as an output by the inference network, and are thus fully amortized.

4 RELATED WORK

4.1 NORMALIZING FLOWS FOR VARIATIONAL INFERENCE

A number of invertible transformations with tractable Jacobians have been proposed in recent years. Rezende and Mohamed (2015) first discussed such transformations in the context of stochastic variation inference, coining the term normalizing flows.

Rezende and Mohamed (2015) proposed two different parametric families of transformations with tractable Jacobians: planar and radial flows. While effective for small problems, these transformations are hard to scale to large latent spaces and often require a large number of transformations. The transformation corresponding to planar flows is given in Eq. (8).

More recently, a successful class of flows called Inverse Autoregressive Flows was introduced in (Kingma et al., 2016). As the name suggests, one IAF transformation can be seen as the inverse of an autoregressive transformation. Consider the following autoregressive transformation:

$$\begin{aligned} z_0 &= \bar{\mu}_0 + \bar{\sigma}_0 \cdot \epsilon_0 \\ z_i &= \bar{\mu}_i(\mathbf{z}_{1:i-1}) + \bar{\sigma}_i(\mathbf{z}_{1:i-1}) \cdot \epsilon_i, \quad i = 1, \ldots, D \end{aligned} \quad (27)$$

with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. This transformation models the distribution over the variable $\mathbf{z}$ with an autoregressive factorization $p(\mathbf{z}) = p(z_0) \prod_{i=1}^{D} p(z_i|z_{i-1}, \ldots, z_0)$. Since the parameters of transformation for z_i are dependent on $\mathbf{z}_{1:i-1}$, this procedure requires D sequential steps to sam-

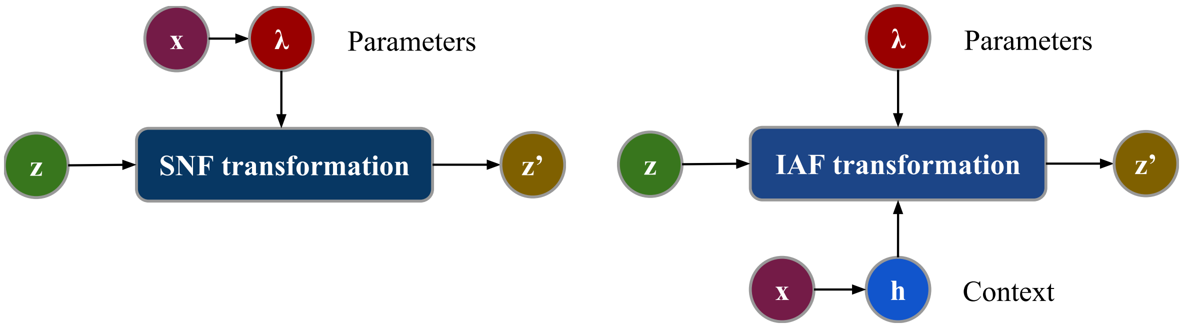

Figure 2: Different amortization strategies for Sylvester normalizing flows and Inverse Autoregressive Flows. Left: our inference network produces amortized flow parameters. This strategy is also employed by planar flows. Right: IAF has a large number of parameters, and introduces a measure of $\mathbf{x}$ dependence through a context $\mathbf{h}(\mathbf{x})$. This context acts as an additional input for each transformation. The flow parameters themselves are independent of $\mathbf{x}$.

ple a single vector $\mathbf{z}$. This is undesirable for variational inference, where sampling occurs for every forward pass.

However, the inverse transformation (which exists if $\bar{\sigma}_i > 0 \; \forall i$) is easy to sample from:

$$\epsilon_i = \frac{z_i - \bar{\mu}_i(\mathbf{z}_{1:i-1})}{\bar{\sigma}_i(\mathbf{z}_{1:i-1})}. \tag{28}$$

For this inverse transformation, ϵ_i is no longer dependent on the transformation of ϵ_j for $j \neq i$. Hence, this transformation can be computed in parallel: $\boldsymbol{\epsilon} = (\mathbf{z} - \bar{\boldsymbol{\mu}}(\mathbf{z}))/\bar{\boldsymbol{\sigma}}(\mathbf{z})$. Rewriting $\sigma_i(z_{1:i-1}) = 1/\bar{\sigma}_i(z_{1:i-1})$ and $\mu_i(z_{1:i-1}) = -\bar{\mu}(z_{1:i-1})/\bar{\sigma}_i(z_{1:i-1})$, yields the IAF transformation:

$$z_i^t = \mu_i^t(\mathbf{z}_{1:i-1}^{t-1}) + \sigma_i^t(\mathbf{z}_{1:i-1}^{t-1}) \cdot z_i^{t-1}, \quad i = 1, ..., D. \tag{29}$$

Starting from $\mathbf{z}^0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, multiple IAF transformations can be stacked on top of each other to produce flexible probability distributions.

If $\boldsymbol{\mu}^t$ and $\boldsymbol{\sigma}^t$ depend on $\mathbf{z}^{t-1}$ linearly, IAF can model full covariance Gaussian distributions. In order to move away from Gaussian distributions to more flexible distributions, it is important that $\boldsymbol{\mu}^t$ and $\boldsymbol{\sigma}^t$ are nonlinear functions of $\mathbf{z}^{t-1}$.

In practice, wide MADEs (Germain et al., 2015) or deep PixelCNN layers (van den Oord et al., 2016) are needed to increase the flexibility of IAF transformations. This results in transformations with a large number of parameters. As shown in Figure 2 (right), amortization is achieved through a context $\mathbf{h}(\mathbf{x})$ that is fed into the autoregressive networks as an additional input at every IAF step.

Our Triangular Sylvester flows are strongly related to mean-only IAF transformations ($\boldsymbol{\sigma}^t = 1$). As mentioned in Kingma et al. (2016), between every IAF transformation the order of $\mathbf{z}$ is reversed, in order to ensure that on average all dimensions get warped equally. In T-SNF, the same effect is achieved by using the permutation matrix that reverses the order of $\mathbf{z}$ in every other transformation as the orthogonal matrix. However, mean-only IAF is a volume-preserving transformation, i.e. the determinant of the Jacobian has absolute value one. T-SNF is not volume preserving due to the nonzero elements on the diagonals of $\mathbf{R}$ and $\tilde{\mathbf{R}}$. Note, that in Kingma et al. (2016) it was shown that the difference in performance between mean-only IAF and the general IAF transformation was negligible.

The most important difference between IAF and T-SNF is the way parameters are amortized. In T-SNF, $\mathbf{R}$ and $\tilde{\mathbf{R}}$ are directly amortized functions of the input $\mathbf{x}$ (see Fig. 2). This is equivalent to amortizing the MADE parameters in mean-only IAF. Having input dependent MADE parameters allows for flexible transformations with fewer parameters.

Householder Sylvesters flows can also be seen as a non-linear extension of Householder flows (Tomczak and Welling, 2016). Householder flows are volume-preserving flows, which transform the variational posterior with a diagonal covariance matrix to a full-covariance posterior. Householder flows are a special case of H-SNF if $h(\mathbf{z}) = \mathbf{z}$, $\mathbf{R}$ is the identity matrix, and the residual connection in Eq. (13) is left out.

4.2 NORMALIZING FLOWS FOR DENSITY ESTIMATION

A number of invertible transformations have been proposed in the context of density estimation. Note that density estimation requires the inverse of the flow to be tractable. Having a provably invertible transformation is not the same as being able to compute the inverse.

For density estimation with normalizing flows, we are interested maximizing the log-likelihood of the data:

$$\log p(\mathbf{x}) = \log p_0(f^{-1}(\mathbf{x})) + \log \left| \det \left(\frac{\partial f^{-1}(\mathbf{x})}{\partial \mathbf{x}} \right) \right| . \quad (30)$$

Thus, the goal is to transform a complicated data distribution back to a simple distribution. In general, both directions of an invertible transformations need not be tractable. Hence, methods developed for density estimation are generally not directly applicable to variational inference.

Non-linear independent component estimation (NICE, Dinh et al. (2014)) and the related Real NVP (Dinh et al., 2016), and Masked Autoregressive Flow (MAF, Papamakarios et al. (2017)) are recent examples of normalizing flows for density estimation.

In NICE, each transformation splits the variables into two disjoint subsets $\mathbf{z}_A, \mathbf{z}_B$. One of the subsets is transformed as $\mathbf{z}'_A = \mathbf{z}_A + f(\mathbf{z}_B)$, while $\mathbf{z}_B$ is left unchanged. In the next transformation a different subset of variables is transformed. This results in a transformation which is trivially invertible and has a tractable Jacobian. Real NVP uses the same fundamental idea. Appealingly, because of the tractable inverse, NICE and real NVP can generate data and estimate density with one forward pass. However due to fact that only a subset of variables is updated in each transformation many transformations are needed in practice. Rezende and Mohamed (2015) compared NICE to planar flows in the context of variational inference and found that planar flows empirically perform better.

Finally, Papamakarios et al. (2017) showed that fitting an MAF can be seen as fitting an implicit IAF from the data distribution to the base distribution. However, generating data from an MAF density model requires D passes, making it unappealing for variational inference.

5 NUMBER OF PARAMETERS

Here, we briefly compare the number of parameters needed by planar flows, IAF and the three Sylvester normalizing flows. We denote the size of the stochastic variables z with D, and the number of output units of the inference network with E.

Planar flows use amortized parameters $\mathbf{u}, \mathbf{w} \in \mathbb{R}^D$ and $b \in \mathbb{R}$ for each flow transformation. Therefore, the number of parameters related to K flow transformations is equal to $2EDK + EK$.

For the implementation of IAF as described in Section 6, the inference network needs to produce a context of size C, where C denotes the width of the MADE layers. The total number of flow related learnable parameters then comes down to $EC + K \times (C^2 + 3CD)$.

In the case of Orthogonal Sylvester flows with a bottleneck of size M, we require $KE \times (MD + 2M^2 + M)$ parameters. For Householder Sylvester flows with H Householder reflections per flow transformation, $KE \times (HD + 2D^2 + D)$ parameters are needed. Finally, for triangular Sylvester flows $KE \times (2D^2 + D)$ parameters require optimization.

Planar flows require the smallest number of parameters but generally result in worse results. IAFs on the other hand require a number of parameters that is quadratic in the width of the MADE layers. For good results this has to be quite large. In contrast, for SNFs the number of parameters is quadratic in the dimension of the latent space and while large, this can still be amortized.

6 EXPERIMENTS

We perform empirical studies of the performance of Sylvester flows on four datasets: statically binarized MNIST, Freyfaces, Omniglot and Caltech 101 Silhouettes. The baseline model is a plain VAE with a fully factorized Gaussian distribution. We furthermore compare against planar flows and Inverse Autoregressive Flows of different sizes.

We use annealing to optimize the lower bound, where the prefactor of the KL divergence is linearly increased from 0 to 1 during 100 epochs as suggested by Bowman et al. (2015) and Sønderby et al. (2016). A learning rate of 0.0005 was used in all experiments. In order to obtain estimates for the negative log likelihood we used importance sampling (as proposed in (Rezende et al., 2014)). Unless otherwise stated, 5000 importance samples were used.

In order to assess the performance of the different flows properly, we use the same base encoder and decoder architecture for all models. We use gated convolutions and transposed convolutions as base layers for the encoder and decoder architecture respectively. The inference network consists of several gated convolution layers that produce a hidden unit vector. After being flattened, these hidden units act as an input to two fully connected layers that predict the mean and variance of $\mathbf{z}^0$.

For planar and Sylvester flows, the flattened hidden units are passed to a separate linear layer that output the amortized flow parameters. For IAF, the flattened hidden units are also passed to a linear layer to produce the context vector $\mathbf{h}_{\text{context}}(\mathbf{x})$. For details of the architecture see Section A of the appendix. In all models the latent space is of dimension 64.

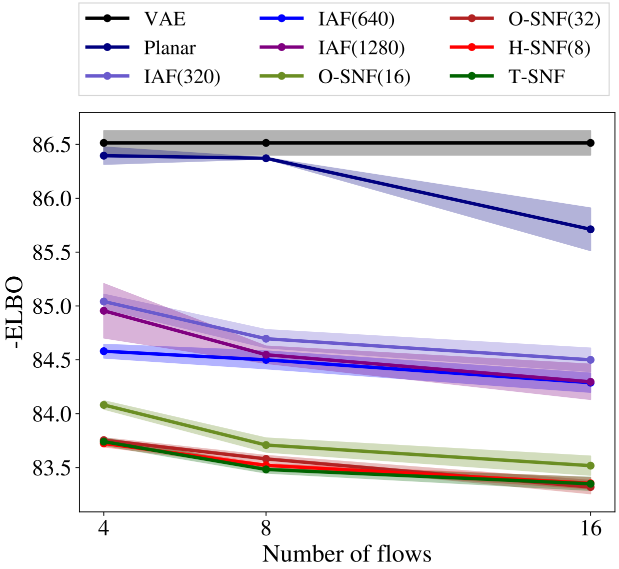

Figure 3: The negative evidence lower bound for static MNIST. The results for H-SNF with 4 reflections per orthogonal matrix are left out for clarity, as they are very similar to the results with 8 reflections. Each model is evaluated 3 times. The shaded areas indicate ± one standard deviation.

We use the following implementation for each IAF transformation[1]: one IAF transformation first applies one MADE Layer (denoted as MaskedLinear) followed by a nonlinearity to the input z, upscaling it to a hidden variable of size M. At this point the context vector $\mathbf{h}_{\text{context}}(\mathbf{x})$ is added to the hidden units, after which two more masked layers are applied to produce the mean and scale of the IAF transformation:

$$\begin{aligned}
&\mathbf{h}_z \leftarrow \text{ELU}(\text{MaskedLinear}(\mathbf{z})) \\
&\mathbf{h} \leftarrow \mathbf{h}_z + \mathbf{h}_{\text{context}}(\mathbf{x}) \\
&\mathbf{h} \leftarrow \text{ELU}(\text{MaskedLinear}(\mathbf{h})) \\
&\boldsymbol{\mu} \leftarrow \text{MaskedLinear}(\mathbf{h}), \quad \mathbf{s} \leftarrow \text{MaskedLinear}(\mathbf{h}) \\
&\mathbf{z}' \leftarrow \sigma(\mathbf{s}) \odot \mathbf{z} + (1 - \sigma(\mathbf{s})) \odot \boldsymbol{\mu}. \qquad (31)
\end{aligned}$$

Here, $\sigma(\ )$ denotes the sigmoid activation function. In Kingma et al. (2016) it was mentioned that the gated form of IAF in Eq. (31) is more stable than the form of Eq. (29). Note that the size of $\mathbf{h}_{\text{context}}(\mathbf{x})$ scales with the width of the MADE layers C.

Table 1: Negative log-likelihood and free energy (negative evidence lower bound) for static MNIST. Numbers are produced with 3 runs per model with different random initializations. Standard deviations over the 3 different runs are also shown.

Model	**-ELBO**	**NLL**
VAE	86.55 ± 0.06	82.14 ± 0.07
Planar	86.06 ± 0.31	81.91 ± 0.22
IAF	84.20 ± 0.17	80.79 ± 0.12
O-SNF	$\mathbf{83.32 \pm 0.06}$	$\mathbf{80.22 \pm 0.03}$
H-SNF	83.40 ± 0.01	80.29 ± 0.02
T-SNF	83.40 ± 0.10	80.28 ± 0.06

6.1 MNIST

Figure 3 shows the dependence of the negative evidence lower bound (or free energy) on the number of flows and the type of flow for static MNIST. The exact numbers corresponding to the figure are shown in Section B in the appendix.

For all models the performance improves as a functions of the number of flows. For 4 flows the difference between the baseline VAE and planar flows is very small. However, planar flows clearly benefit from more flow transformations.

For IAF three different widths of the MADE layers were used: $C = 320$, 640 and 1280. Surprisingly, for 4 flows the widest IAF with 1280 hidden units is outperformed by an IAF with 640 hidden units in the MADE layers. We expect this to be due to the fact that this model has more parameters and can therefore be harder to train, as indicated by the larger standard deviation for this model.

All three Sylvester flows outperform IAF and planar flows. For Orthogonal Sylvester flows, we show results for $M = 16$ and $M = 32$ orthogonal vectors per orthogonal matrix, thus corresponding to bottlenecks of size 16 and 32 respectively for a latent space of size $D = 64$. Clearly, a larger bottleneck improves performance. For Householder Sylvester flows we experimented with $H = 4$ and $H = 8$ Householder reflections per orthogonal matrix. Since the results were nearly indistinguishable between these two variants, we have left out the curve for $H = 4$ to avoid clutter. O-SNF with $M = 32$, H-SNF and T-SNF seem to perform on par.

In Table 1, the negative evidence lower bound and the estimated negative log-likelihood are shown for the baseline VAE, together with all flow models for 16 flows. The reported result for IAF is for a MADE width of 1280. The O-SNF model has a bottleneck of $M = 32$, and

[1] This implementation is based on the open source code for IAF available at `https://github.com/openai/iaf`

Table 2: Results for Freyfaces, Omniglot and Caltech 101 Silhouettes datasets. For the Freyfaces dataset the results are reported in bits per dim. For the other datasets the results are reported in nats. For each flow model 16 flows are used. For IAF a MADE width of 1280 was used, and for O-SNF flow a bottleneck of $M = 32$ was used. For H-SNF 8 householder reflections were used to construct orthogonal matrices. For all datasets 3 runs per model were performed.

Model	**Freyfaces**		**Omniglot**		**Caltech 101**	
	-ELBO	NLL	-ELBO	NLL	-ELBO	NLL
VAE	4.53 ± 0.02	4.40 ± 0.03	104.28 ± 0.39	97.25 ± 0.23	110.80 ± 0.46	99.62 ± 0.74
Planar	$\mathbf{4.40 \pm 0.06}$	$\mathbf{4.31 \pm 0.06}$	102.65 ± 0.42	96.04 ± 0.28	109.66 ± 0.42	98.53 ± 0.68
IAF	4.47 ± 0.05	4.38 ± 0.04	102.41 ± 0.04	96.08 ± 0.16	111.58 ± 0.38	99.92 ± 0.30
O-SNF	4.51 ± 0.04	4.39 ± 0.05	99.00 ± 0.29	93.82 ± 0.21	106.08 ± 0.39	94.61 ± 0.83
H-SNF	4.46 ± 0.05	4.35 ± 0.05	$\mathbf{99.00 \pm 0.04}$	$\mathbf{93.77 \pm 0.03}$	$\mathbf{104.62 \pm 0.29}$	$\mathbf{93.82 \pm 0.62}$
T-SNF	4.45 ± 0.04	4.35 ± 0.04	99.33 ± 0.23	93.97 ± 0.13	105.29 ± 0.64	94.92 ± 0.73

H-SNF contains 8 Householder reflections per orthogonal matrix. Again, all Sylvester flows outperform planar flows and IAF, both in terms of the free energy and the negative log-likelihood.

As discussed in Section 4, T-SNF is closely related to mean-only IAF, but with the MADE parameters directly amortized. The fact that T-SNF outperforms IAF indicates that amortizing the parameters directly leads to a more flexible transformation compared to taking a very wide MADE with a data dependent context as an additional input.

6.2 FREYFACES, OMNIGLOT AND CALTECH 101 SILHOUETTES

We further assess the performance of the different models on Freyfaces, Omniglot and Caltech 101 Silhouettes. The results are shown in Table 2. The model settings are the same[2] as those used for Table 1.

Freyfaces is a very small dataset of around 2000 faces. All normalizing flows increase the performance, with planar flows yielding the best result, closely followed by Triangular and Householder Sylvester flows. We expect planar flows to perform the best in this case since it is the least sensitive to overfitting.

For Omniglot and Caltech 101 Silhouettes the results are clearer, with the Sylvester normalizing flows family resulting in the best performance. Both H-SNF and T-SNF perform better than O-SNF. This could be attributed to the fact that O-SNF has a bottleneck of $M = 32$ for a latent space size of $D = 64$. The IAF scores for Caltech 101 are surprisingly bad. We expect this could be the case due to the large number of parameters that need to be trained for IAF(1280). Therefore we also evaluated the result for MADEs of width 320 for 16 flows. The resulting free energy and estimated negative log-likelihood are 111.23 ± 0.45 and 99.74 ± 0.28 respectively, only slightly improving on the results of 1280 wide IAFs.

7 CONCLUSION

We present a new family of normalizing flows: Sylvester normalizing flows. These flows generalize planar flows, while maintaining an efficiently computable Jacobian determinant through the use of Sylvester's determinant identity. We ensure invertibility of the flows through the use of orthogonal and triangular parameter matrices. Three variants of Sylvester flows are investigated. First, orthogonal Sylvester flows use an iterative procedure to maintain orthogonality of parameter matrices. Second, Householder Sylvester flows use Householder reflections to construct orthogonal matrices. Third, triangular Sylvester flows alternate between fixed permutation and identity matrices for the orthogonal matrices. We show that the triangular Sylvester flows are closely related to mean-only IAF, with directly amortized MADE parameters. While performing comparably with planar flows and IAF for the Freyfaces dataset, our proposed family of flows improve significantly upon planar flows and IAF on the three other datasets.

Acknowledgements

We would like to thank Christos Louizos for useful discussions and helping with the implementation of inverse autoregressive flows. LH is funded by the UK EPSRC OxWaSP CDT through grant EP/L016710/1. JMT is funded by the European Commission within the MSC-IF (Grant No. 702666). RvdB is funded by SAP SE.

[2] For Caltech 101 Silhouettes we used 2000 importance samples for the estimation of the negative log-likelihood.

References

Christian Bischof and Xiaobai Sun. On orthogonal block elimination. Technical Report MCS-P450-0794, Argonne National Laboratory, Argonne, IL, 10 1997.

Åke Björck and Clazett Bowie. An iterative algorithm for computing the best estimate of an orthogonal matrix. *SIAM Journal on Numerical Analysis*, 8(2):358–364, 1971.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. Generating Sentences from a Continuous Space. nov 2015. URL `http://arxiv.org/abs/1511.06349`.

Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: non-linear independent components estimation. abs/1410.8516, 2014.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. MADE: Masked Autoencoder for Distribution Estimation. *ICML*, pages 881–889, 2015.

Samuel Gershman and Noah Goodman. Amortized inference in probabilistic reasoning. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 36, 2014.

Leonard Hasenclever, Jakub Tomczak, Rianne van den Berg, and Max Welling. Variational inference with orthogonal normalizing flows. 2017.

Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347, 2013.

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

Diederik Kingma and Max Welling. Efficient gradient-based inference through transformations between bayes nets and neural nets. *ICML*, pages 1782–1790, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved Variational Inference with Inverse Autoregressive Flow. *NIPS*, pages 4743–4751, 2016.

Zdislav Kovarik. Some iterative methods for improving orthonormality. *SIAM Journal on Numerical Analysis*, 7(3):386–389, 1970.

Eric Nalisnick, Lars Hertel, and Padhraic Smyth. Approximate inference for deep latent gaussian mixtures. In *NIPS Workshop on Bayesian Deep Learning*, 2016.

George Papamakarios, Iain Murray, and Theo Pavlakou. Masked Autoregressive Flow for Density Estimation. *NIPS*, pages 2335–2344, 2017.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. *ICML*, pages 1530–1538, 2015.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Tim Salimans, Diederik Kingma, and Max Welling. Markov Chain Monte Carlo and variational inference: Bridging the gap. *ICML*, pages 1218–1226, 2015.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder Variational Autoencoders. feb 2016. URL `http://arxiv.org/abs/1602.02282`.

Xiaobai Sun and Christian Bischof. A basis-kernel representation of orthogonal matrices. *SIAM Journal on Matrix Analysis and Applications*, 16(4):1184–1196, 1995.

EG Tabak and Cristina V Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013.

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.

Jakub M Tomczak and Max Welling. Improving Variational Auto-encoders using Householder Flow. *arXiv preprint arXiv:1611.09630*, 2016.

Dustin Tran, Rajesh Ranganath, and David M Blei. The variational Gaussian process. *arXiv preprint arXiv:1511.06499*, 2015.

Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. *NIPS*, pages 4790–4798, 2016.

A Architecture

In the experiments we used convolutional layers for both the encoder and the decoder. Moreover, we used the gated activation function for convolutional layers:

$$\mathbf{h}_l = (\mathbf{W}_l * \mathbf{h}_{l-1} + \mathbf{b}_l) \odot \sigma(\mathbf{V}_l * \mathbf{h}_{l-1} + \mathbf{c}_l),$$

where $\mathbf{h}_{l-1}$ and $\mathbf{h}_l$ are inputs and outputs of the l-th layer, respectively, $\mathbf{W}_l, \mathbf{V}_l$ are weights of the l-th layer, $\mathbf{b}_l, \mathbf{c}_l$ denote biases, $*$ is the convolution operator, and $\sigma(\cdot)$ is the sigmoid activation function.

We used the following architecture of the encoder (k is a kernel size, p is a padding size, and s is a stride size):[3]

$$\mathrm{Conv}(\mathrm{in}=1, \mathrm{out}=32, \mathrm{k}=5, \mathrm{p}=2, \mathrm{s}=1)$$
$$\mathrm{Conv}(\mathrm{in}=32, \mathrm{out}=32, \mathrm{k}=5, \mathrm{p}=2, \mathrm{s}=2)$$
$$\mathrm{Conv}(\mathrm{in}=32, \mathrm{out}=64, \mathrm{k}=5, \mathrm{p}=2, \mathrm{s}=1)$$
$$\mathrm{Conv}(\mathrm{in}=64, \mathrm{out}=64, \mathrm{k}=5, \mathrm{p}=2, \mathrm{s}=2)$$
$$\mathrm{Conv}(\mathrm{in}=64, \mathrm{out}=64, \mathrm{k}=5, \mathrm{p}=2, \mathrm{s}=1)$$
$$\mathrm{Conv}(\mathrm{in}=64, \mathrm{out}=64, \mathrm{k}=5, \mathrm{p}=2, \mathrm{s}=1)$$
$$\mathrm{Conv}(\mathrm{in}=64, \mathrm{out}=256, \mathrm{k}=7, \mathrm{p}=0, \mathrm{s}=1)$$

Notice the last layer acts as a fully-connected layer. Eventually, fully-connected linear layers were used to parameterized diagonal Gaussian distribution and amortized parameters of a flow.

The decoder mirrors the structure of the encoder with transposed convolutional layers (op is an outer padding):

$$\mathrm{ConvT}(\mathrm{in}=64, \mathrm{out}=64, \mathrm{k}=7, \mathrm{p}=0, \mathrm{s}=1)$$
$$\mathrm{ConvT}(\mathrm{in}=64, \mathrm{out}=64, \mathrm{k}=5, \mathrm{p}=2, \mathrm{s}=1)$$
$$\mathrm{ConvT}(\mathrm{in}=64, \mathrm{out}=32, \mathrm{k}=5, \mathrm{p}=2, \mathrm{s}=2, \mathrm{op}=1)$$
$$\mathrm{ConvT}(\mathrm{in}=32, \mathrm{out}=32, \mathrm{k}=5, \mathrm{p}=2, \mathrm{s}=1)$$
$$\mathrm{ConvT}(\mathrm{in}=32, \mathrm{out}=32, \mathrm{k}=5, \mathrm{p}=2, \mathrm{s}=2, \mathrm{op}=1)$$
$$\mathrm{ConvT}(\mathrm{in}=32, \mathrm{out}=32, \mathrm{k}=5, \mathrm{p}=2, \mathrm{s}=1)$$
$$\mathrm{ConvT}(\mathrm{in}=32, \mathrm{out}=1, \mathrm{k}=1, \mathrm{p}=0, \mathrm{s}=1)$$

A.1 Description of datasets

In the experimetns we used the following four image datasets: static MNIST[4], OMNIGLOT[5], Caltech 101 Silhouettes[6], and Frey Faces[7]. Frey Faces contains images of size 28 × 20 and all other datasets contain 28 × 28 images.

MNIST consists of hand-written digits split into 60,000 training datapoints and 10,000 test sample points. In order to perform model selection we put aside 10,000 images from the training set.

OMNIGLOT is a dataset containing 1,623 hand-written characters from 50 various alphabets. Each character is represented by about 20 images that makes the problem very challenging. The dataset is split into 24,345 training datapoints and 8,070 test images. We randomly pick 1,345 training examples for validation. During training we applied dynamic binarization of data similarly to dynamic MNIST.

Caltech 101 Silhouettes contains images representing silhouettes of 101 object classes. Each image is a filled, black polygon of an object on a white background. There are 4,100 training images, 2,264 validation datapoints and 2,307 test examples. The dataset is characterized by a small training sample size and many classes that makes the learning problem ambitious.

Frey Faces is a dataset of faces of a one person with different emotional expressions. The dataset consists of nearly 2,000 gray-scaled images. We randomly split them into 1,565 training images, 200 validation images and 200 test images. We repeated the experiment 3 times.

B MNIST experiments

The exact numbers for the evidence lower bound as shown in Fig. 3 are listed in Table 3.

[3] We use a PyTorch convention of defining convolutional layers.

[4] http://yann.lecun.com/exdb/mnist/

[5] https://github.com/yburda/iwae/blob/master/datasets/OMNIGLOT/chardata.mat.

[6] https://people.cs.umass.edu/~marlin/data/caltech101_silhouettes_28_split1.mat.

[7] http://www.cs.nyu.edu/~roweis/data/frey_rawface.mat

Table 3: Negative evidence lower bounds for the test set on MNIST. All results are obtained with stochastic hidden units of size 64.

Model	**-ELBO**
VAE	86.51 ± 0.11
Planar ($K = 4$)	86.40 ± 0.08
Planar ($K = 8$)	86.37 ± 0.006
Planar ($K = 16$)	85.71 ± 0.20
IAF ($W = 320$, $K = 4$)	85.04 ± 0.07
IAF ($W = 320$, $K = 8$)	84.70 ± 0.08
IAF ($W = 320$, $K = 16$)	84.50 ± 0.11
IAF ($W = 640$, $K = 4$)	84.58 ± 0.06
IAF ($W = 640$, $K = 8$)	84.50 ± 0.08
IAF ($W = 640$, $K = 16$)	84.29 ± 0.09
IAF ($W = 1280$, $K = 4$)	84.96 ± 0.25
IAF ($W = 1280$, $K = 8$)	84.55 ± 0.08
IAF ($W = 1280$, $K = 16$)	84.30 ± 0.16
O-SNF ($M = 16$, $K = 4$)	84.08 ± 0.04
O-SNF ($M = 16$, $K = 8$)	83.71 ± 0.07
O-SNF ($M = 16$, $K = 16$)	83.52 ± 0.09
O-SNF ($M = 32$, $K = 4$)	83.76 ± 0.02
O-SNF ($M = 32$, $K = 8$)	83.58 ± 0.03
O-SNF ($M = 32$, $K = 16$)	83.32 ± 0.06
H-SNF ($K = 4$, $H = 4$)	83.73 ± 0.05
H-SNF ($K = 8$, $H = 4$)	83.49 ± 0.08
H-SNF ($K = 16$, $H = 4$)	83.36 ± 0.04
H-SNF ($K = 4$, $H = 8$)	83.72 ± 0.03
H-SNF ($K = 8$, $H = 8$)	83.52 ± 0.01
H-SNF ($K = 16$, $H = 8$)	83.35 ± 0.05
T-SNF ($K = 4$)	83.74 ± 0.04
T-SNF ($K = 8$)	83.48 ± 0.03
T-SNF ($K = 16$)	83.35 ± 0.06