Improving Normalizing Flows via Better Orthogonal Parameterizations

Adam Goliński [*1 2] **Mario Lezcano-Casado** [*3] **Tom Rainforth** [1]

Abstract

Normalizing flows have to be designed in a manner that permits efficient computation of the determinant of the transformation Jacobian, while ensuring that the transformation remains invertible. Many methods achieve this by using transformations based on matrix decompositions, with decompositions that produce an orthogonal matrix being a popular such approach. However, this introduces the additional difficulty of performing a constrained optimization over the space of orthogonal matrices. Current approaches for achieving this use methods which are prone to numerical instability and which may introduce additional local minima into the optimization landscape. To address this, we introduce two orthogonal matrix parameterizations stemming from Lie group theory—the exponential map and the Cayley map—which alleviate these issues. In particular, they are proven to not introduce new local minima. Focusing on the example architecture of Sylvester normalizing flows (van den Berg et al. (2018)), we show empirically that using our suggested parameterizations lead to significantly improved optimization and, in turn, more effective normalizing flows.

1. Introduction

Normalizing flows are a class of flexible probabilistic models that allow one to represent distributions through a series of invertible mappings applied to a simple base distribution (Tabak et al., 2010; Tabak & Turner, 2013; Rezende & Mohamed, 2015; Dinh et al., 2015; 2017; Kingma et al., 2016; Papamakarios et al., 2017; van den Berg et al., 2018; Huang et al., 2018; Kingma & Dhariwal, 2018; Behrmann et al., 2018; Grathwohl et al., 2019; Hoogeboom et al., 2019; De Cao et al., 2019). The fact that they are invertible allows their density to be calculated analytically, while maintaining the ability to generate samples and formulate highly complex distributions. This makes them highly suitable models for various tasks such as density estimation (Dinh et al., 2015; 2017; Papamakarios et al., 2017; Huang et al., 2018), learning generative models (Kingma & Dhariwal, 2018; Hoogeboom et al., 2019), and providing flexible distribution families in variational inference and variational autoencoders (Rezende & Mohamed, 2015; Kingma et al., 2016; van den Berg et al., 2018).

More formally, finite normalizing flows define a complex parametric family of probability distribution on $\mathbb{R}^D$ by transforming some simple distribution U_0—*e.g.*, an isotropic Gaussian—along a bijective function $f\colon \mathbb{R}^D \to \mathbb{R}^D$, *e.g.*, an invertible neural network. In general, we apply a chain of K invertible functions $f(\mathbf{u}) := (f_K \circ \ldots \circ f_1)(\mathbf{u})$ giving a log-likelihood of

$$\log p(f(\mathbf{u})) = \log p_{U_0}(\mathbf{u}_0) - \sum_{k=1}^{K} \log|\det \mathbf{J}_k| \qquad (1)$$

where $\mathbf{u}_0 := \mathbf{u}$, $\mathbf{u}_k := f_k(\mathbf{u}_{k-1}) \in \mathbb{R}^D$, and $\mathbf{J}_k = \partial \mathbf{u}_k / \partial \mathbf{u}_{k-1}$ is the Jacobian of f_k. Using this likelihood, one can construct, for example, variational objectives for training deep generative models or performing Bayesian inference, with the subsequent training typically performed using (stochastic) gradient descent methods.

Arbitrary choices of the transformations f_k result in potentially high costs for computing the determinant of the Jacobian, with naive approaches scaling as $\mathcal{O}(D^3)$ in the dimensionality of the transformed random variable. Therefore, one usually aims to design the f_k in such a way as to make this computation more efficient, typically by constructing them in a manner that ensures the Jacobian takes on a particular form (e.g. lower- or upper-triangular) for which the determinant can be calculated efficiently.

As part of achieving this, a number of methods (Kingma & Dhariwal, 2018; van den Berg et al., 2018; Hoogeboom et al., 2019) rely on using matrix decompositions in their definition of f_k that both produce convenient forms of the Jacobian and ensure that its determinant is non-zero to ensure invertibility. A downside of this approach is that it can

[*]Equal contribution [1]Department of Statistics, University of Oxford, United Kingdom [2]Department of Engineering Science, University of Oxford, United Kingdom [3]Mathematical Institute, University of Oxford, United Kingdom. Correspondence to: Adam Goliński <adamg@robots.ox.ac.uk>, Mario Lezcano-Casado <mario.lezcanocasado@maths.ox.ac.uk>.

First workshop on *Invertible Neural Networks and Normalizing Flows* (ICML 2019), Long Beach, CA, USA

impose constraints on the matrices parameterizing the flows, converting the original unconstrained optimization problem into a constrained one. In particular, Sylvester normalizing flows (SNF) (van den Berg et al., 2018) and the adaptation of Glow proposed by (Hoogeboom et al., 2019) rely on QR decompositions, imposing an *orthogonality* constraint on the resulting $\mathbf{Q}$ matrices learned when training the flow.

To solve the resulting constrained optimization problems, these approaches parameterize the $\mathbf{Q}$ matrices in an unconstrained domain that is then mapped onto the set of orthogonal matrices. For example, (van den Berg et al., 2018) suggest one method based on Householder reflections and another based on the iterative procedure of (Kovarik, 1970).

In this paper, we highlight a number of shortfalls in these common parameterizations for orthogonal matrices in the context of normalizing flows. At a high-level, these shortfalls stem from the fact that while previous methods provide effective parameterizations from the perspective of the orthogonal matrix itself, this does not necessarily transfer to effective parameterizations of its *gradients*. More specifically, we highlight the following issues that can occur.

Spurious local minima The mapping induced by the parameterizing function may introduce additional local minima, undermining the gradient-based schemes used for the optimization.

Numerical instabilities Many of the methods used in the current literature are prone to numerical instabilities due to underlying topological issues.

Poor gradient estimates Using methods like the iterative procedure of (Kovarik, 1970) implicitly relies on using automatic differentiation on the output of a numerical approximation and this can produce poor approximations of the true derivatives.

We show how these shortfalls can be corrected using two parameterizations for orthogonal matrices stemming from Lie group theory—the exponential map and the Cayley map. In particular, we show that both are numerically stable and are theoretically proven not to introduce local minima for practical purposes (*cf.*, (Lezcano-Casado & Martínez-Rubio, 2019; Falorsi et al., 2019)).

Moreover, focusing on the example architecture of SNF, we show that using our suggested parameterizations can lead to substantially improved optimization and, in turn, more effective normalizing flows. Specifically, our empirical results show that, in a variational autoencoder encoder context (Kingma & Welling, 2014; Rezende et al., 2014), our parameterizations produced improved inference networks and, in turn, improved generative networks, when compared with the original SNF approaches and other state-of-the-art normalizing flow approaches.

2. Background

2.1. Sylvester Normalizing Flows

Among many normalizing flow models, van den Berg et al. (2018) proposed Sylvester Normalizing Flows (SNF), a generalization of planar flows (Rezende & Mohamed, 2015) to remove their single-unit bottleneck. To limit the cost of computation of the determinant of the Jacobian while also ensuring invertibility, they use QR decompositions to define

$$f_k(\mathbf{u}_{k-1}) = \mathbf{u}_{k-1} + \mathbf{Q}_k \mathbf{R}_k h(\widetilde{\mathbf{R}}_k \mathbf{Q}_k^T \mathbf{u}_{k-1} + \mathbf{b}_k) \quad (2)$$

where $\mathbf{Q}_k \in \mathbb{R}^{M \times D}$ is a *orthogonal* matrix (i.e. a matrix consisting of a set of *orthonormal* column vectors), $\mathbf{R}_k$ and $\widetilde{\mathbf{R}}_k$ are upper triangular $M \times M$ matrices, $\mathbf{b}_k \in \mathbb{R}^M$, and $M \leq D$. These restrictions ensure that the transformation is invertible if $h : \mathbb{R} \longrightarrow \mathbb{R}$ is a smooth function with bounded, positive derivative, the diagonal entries of $\widetilde{\mathbf{R}}_k$ and $\mathbf{R}_k$ satisfy $r_{ii}\widetilde{r}_{ii} > -1/\left\|h'\right\|_\infty$, and $\widetilde{\mathbf{R}}_k$ is invertible (see (van den Berg et al., 2018, Theorem 2) for full details).

This parameterization further allows efficient calculation of the Jacobian determinant (needed to calculate the likelihood density as per (1)) through invoking Sylvester's identity,

$$\det \mathbf{J}_k = \det\Big(\mathbf{I}_M + \operatorname{diag}\big(h'(\widetilde{\mathbf{R}}_k \mathbf{Q}_k^T \mathbf{u}_{k-1} + \mathbf{b}_k)\big)\widetilde{\mathbf{R}}_k \mathbf{R}_k\Big)$$

where h' is the derivative of h, and $\operatorname{diag}(\mathbf{d})$ is a square matrix with $\mathbf{d}$ on the diagonal. This expression can be computed in $\mathcal{O}(MD)$: because $\widetilde{\mathbf{R}}_k \mathbf{R}_k$ is also upper triangular, the determinant calculation itself is only $\mathcal{O}(M)$ (we only need to consider the diagonal terms of $\mathbf{J}_k$) and the limiting operation becomes the matrix-vector product $\mathbf{Q}_k^T \mathbf{u}_{k-1}$.

Though this provides an efficient way to overcome the single-unit bottleneck, it comes at the cost of having to perform a constrained optimization over the space of orthogonal matrices. As we will later show, approaches taken to satisfy these constraints have a number of shortfalls.

Learning a SNF requires one to learn a set of matrices $\mathbf{Q}_{1:K}$, $\mathbf{R}_{1:K}$, and $\widetilde{\mathbf{R}}_{1:K}$, and vectors $\mathbf{b}_{1:K}$. In an amortized inference context, one can instead learn a neural network to produce a suitable set of matrices and vectors from a given datapoint. In this paper, we will focus on square $\mathbf{Q}_k$ matrices, i.e. $M = D$, leaving analysis of the non-square case for future research. We note that this same restriction is made by two of the three methods proposed by (van den Berg et al., 2018).

2.2. QR decompositions for Glow

Though our focus with be on the use of QR decompositions in SNF, we note that SNF do not represent the only use of orthogonal matrices in normalizing flows and our methods are more generally applicable. Another setting of particular note is the adaptation of Glow (Kingma & Dhariwal, 2018) proposed by (Hoogeboom et al., 2019). They show that

improvements to Glow can be achieved by replacing the PLU decomposition used for the 1×1 convolutions with a QR decomposition. They go onto to use the Householder reflection approach of (van den Berg et al., 2018) given in (5), for which our methods can provide a drop-in replacement.

2.3. The orthogonal group

The set of *orthogonal matrices* of dimension n is a compact Lie group defined as

$$\mathrm{O}(n) = \{\mathbf{Q} \in \mathbb{R}^{n \times n} \mid \mathbf{Q}^\top \mathbf{Q} = \mathbf{I}\}. \tag{3}$$

We can think of $\mathrm{O}(n)$ as a submanifold of $\mathbb{R}^{n \times n}$ with two components, namely the orthogonal matrices with determinant 1 and those with determinant -1. The set of orthogonal matrices with determinant 1 is called the *special orthogonal group*, denoted by $\mathrm{SO}(n)$. In general, the methods we suggest will produce matrices in $\mathrm{SO}(n)$, whilst previously adopted approaches generally produce matrices in $\mathrm{O}(n)$.

This topological understanding of the space is important because having multiple separate connected components may introduce stability problems related to the continuity of the algorithm—gradient descent is an inherently local algorithm and it does not work well on disconnected domains. Thus by introducing methods restricted to $\mathrm{SO}(n)$ we can ensure such problems are avoided.

The orthogonal manifold $\mathrm{O}(n)$ and the special orthogonal manifold $\mathrm{SO}(n)$ have dimension $n(n-1)/2$, which is less than half of the dimension of the ambient space $\mathbb{R}^{n \times n}$. As such, we need fewer parameters to generate a matrix in $\mathrm{SO}(n)$ than to a more general matrix in $\mathbb{R}^{n \times n}$.

3. Shortfalls of previous proposed parameterizations

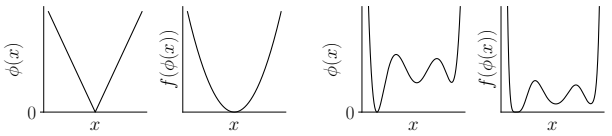
As previously explained, SNF make use of orthogonal matrices $\mathbf{Q}_k$. To address the resulting constraints, van den Berg et al. (2018) presents three different parameterizations of orthogonal matrices.

The first method, triangular Sylvester normalizing flows (T-SNF), is a restricted approach where, rather than being learned, the $\mathbf{Q}_k$ simply alternate between the identity matrix and a permutation matrix that reverses the order of the dimensions of $\mathbf{u}$.

The second method is called Orthogonal Sylvester normalizing flows (O-SNF). It consists of generating an unconstrained matrix $\mathbf{Q}^{(0)} \in \mathbb{R}^{n \times n}$ and then iterating the following procedure:

$$\mathbf{Q}^{(i+1)} := \mathbf{Q}^{(i)} \Big(\mathbf{I} + \tfrac{1}{2}(\mathbf{I} - \mathbf{Q}^{(i)\top} \mathbf{Q}^{(i)}) \Big). \tag{4}$$

This is repeated until a stopping condition criterion is satisfies, namely $\|\mathbf{Q}^{(i)\top}\mathbf{Q}^{(i)} - \mathbf{I}\|_F \leq \varepsilon$ where $\|\cdot\|_F$ is the Frobenius norm and $\varepsilon > 0$ is a small convergence threshold,



(a) $\phi(x) = |x|$ (b) Complex nonlinear mapping

Figure 1: Two different possible parameterizations of the non-negative real numbers $\phi \in \mathbb{R}_{\geq 0}$: (a) $\phi(x) = |x|$ and (b) a more complex nonlinear mapping. When used for the optimization $\min_\phi f(\phi)$ s.t.$y \geq 0$ with $f(\phi) = \phi^2$, we see that the second parameterization induces additional local minima which are likely to undermine the optimization.

with $\mathbf{Q}_k$ fixed to the resulting final $\mathbf{Q}^{(i)}$. This method is known to converge in the limit to a projection of $\mathbf{Q}^{(0)}$ onto $\mathrm{O}(n)$ (Kovarik, 1970; Björck & Bowie, 1971), with the authors reporting that 30 iterations are usually sufficient for convergence within the required threshold. Though differentiable, the method represents a numerical approximation of an orthogonal matrix. As we will later explain, convergence of this approximation does not imply convergence of its gradients, leading to potential issues when learning the $\mathbf{Q}_k$.

The final method introduced, Householder Sylvester normalizing flows (H-SNF), takes the product of H Householder reflections to generate the orthogonal matrix. Namely, they parameterize each $\mathbf{Q}_k$ as

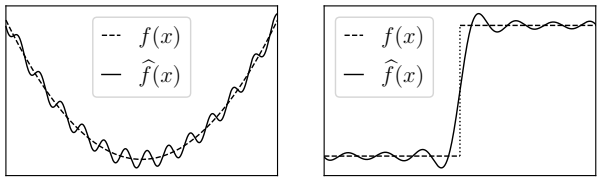
$$\mathbf{Q}_k := \Big(\mathbf{I}_n - 2\frac{\mathbf{v}_H \mathbf{v}_H^\top}{\|\mathbf{v}_H\|^2} \Big) \dots \Big(\mathbf{I}_n - 2\frac{\mathbf{v}_1 \mathbf{v}_1^\top}{\|\mathbf{v}_1\|^2} \Big) \tag{5}$$

where $\mathbf{v}_h \in \mathbb{R}^n$ and each $\mathbf{Q}_k$ has a different set of vectors parameterizing it. Any matrix in $\mathrm{O}(n)$ can be represented by a product of $n-1$ Householder reflections, but as an approximation to reduce computational cost, they use just $H \leq n-1$ trading off expressiveness for computation cost.

Unfortunately, these methods for parameterizing $\mathbf{Q}_k$ present three problems, we now consider each of these in turn.

Spurious local minima The aforementioned approaches convert a constrained optimization problem—$\phi^* = \arg\min_\phi f(\phi)$ s.t. conditions on ϕ—to an unconstrained optimization through a reparameterization—$\phi^* = \phi(\mathbf{x}^*)$ where $x^* = \arg\min_{\mathbf{x}} f(\phi(\mathbf{x}))$. Though such a reparameterization never effects the global optima (provided the reparameterization is surjective), it can induce additional local optima as, in general, $\nabla_{\mathbf{x}} f(\phi(\mathbf{x})) \neq \nabla_\phi f(\phi)$. An illustrative example of this behavior is show in Figure 1. More specifically, by the chain rule, $\nabla_{\mathbf{x}} f(\phi(\mathbf{x})) = \mathbf{0}$ whenever $\nabla_\phi f(\phi) = \mathbf{0}$ or $\nabla_{\mathbf{x}} \phi(\mathbf{x}) = \mathbf{0}$, such that the transformation can induce additional stationary points.

While the methods we introduce in Section 4 are proven not to induce additional stationary points, it is possible for them to be introduced by O-SNF and H-SNF. Remembering



(a) (b)

Figure 2: Illustrations of issues with some numerical methods. (a) Sketch of a function $f(x)$ and its approximation $\widehat{f}(x)$ which yields a good approximation of the function itself, but a poor approximation of its gradients. (b) Sketch of a discontinuous function $f(x)$ approximated by a continuous function $\widehat{f}(x)$. Note the extremely poor estimate of the gradient of the function we would obtain in the vicinity of the discontinuity if we compute it based on the behavior of the approximation.

that training of normalizing flows is generally carried out by stochastic gradient methods, which only aim to find a local optimum, this presents the potential for O-SNF and H-SNF to become stuck in suboptimal solutions that would not be local optima for other parameterizations.

Poor gradient estimates O-SNF approximates a function—a projection onto $\mathrm{O}(n)$—by an algorithm with convergence guarantees. However, this algorithm does not offer any guarantees about correctly approximating its *derivative*: a good numerical approximation of the function does not always lead to a good approximation of its derivatives. Thus if we use automatic differentiation to compute the gradients based on the computation graph of this numerical approximation, we may end up with very poor gradient estimates; an example of this phenomenon is shown in Figure 2a. As such, when using a numerical algorithm to approximate a function in a gradient-based optimization setting, one should look to directly approximate its derivative, rather than taking a derivative of the approximation. One of our methods—the exponential parameterization—allows for doing exactly this, while the other does not require a numerical approximation and so is inherently robust to this problem. H-SNF is similarly unaffected by this issue as it does not require a numerical approximation.

Numerical instabilities For both O-SNF and H-SNF, the calculations involve a multiplication of multiple, potentially ill-conditioned, matrices, which can cause numerical instabilities. In particular, when using O-SNF, nothing stops $\mathbf{Q}^{(0)}$ from being ill-conditioned and if that is the case the entire chain of matrix multiplications will be numerically unstable, as will the computation of its gradients. A prime example of this issue in the machine learning literature is the well-known problem of exploding and vanishing gradients in RNNs (Hochreiter, 1998). For H-SNF, on the other hand, one possible source of numerical instability is from the divisions by $\|\mathbf{v}_h\|^2$: if one of these are close to 0, the division will result in an exploding gradient problem.

In addition to the aforementioned computational view of these potential numerical instabilities, one can also view them from a topological viewpoint related to the discontinuity in $\mathrm{O}(n)$. In particular, O-SNF makes use of continuous functions from its input domain, which is a single connected space, onto $\mathrm{O}(n)$. However, $\mathrm{O}(n)$ is formed of two connected components which are disconnected. This implies that we will see exploding gradients in the vicinity of the discontinuity in the range of the function. A conceptual example of this phenomenon is shown in Figure 2b.

4. Proposed orthogonal parameterizations

We now propose two methods that mitigate the problems presented in the previous section. Both of our parameterizations use a strictly upper triangular matrix $\mathbf{C}$ (thereby using $n(n-1)/2$ parameters), from which we derive a *skew-symmetric* matrix

$$\mathbf{D} := \mathbf{C} - \mathbf{C}^\top, \quad \mathbf{D} \in \mathrm{Skew}(n) \tag{6}$$

This transformation is a vector space isomorphism between $\mathbb{R}^{n(n-1)/2}$ and $\mathrm{Skew}(n)$, meaning that optimizing over $\mathbb{R}^{n(n-1)/2}$ and optimizing over $\mathrm{Skew}(n)$ are equivalent.

Each method then uses this skew-symmetric matrix to produce a special orthogonal matrix $\mathbf{Q} \in \mathrm{SO}(n)$, differing only in their mappings $\mathrm{Skew}(n) \to \mathrm{SO}(n)$. We highlight that, unlike previous approaches, we are deliberately restricting $\mathbf{Q} \in \mathrm{SO}(n)$, rather than just $\mathbf{Q} \in \mathrm{O}(n)$, due to the importance of having a continuous domain in the context of gradient-based optimization, as described in the Section 3.

4.1. Cayley map parameterization

Our first method uses the Cayley map, a mapping which has been extensively studied in the optimization and differential geometry literature (see *e.g.*, (Wen & Yin, 2013)). The function $\mathrm{cay}\colon \mathrm{Skew}(n) \to \mathrm{SO}(n)$ is defined as

$$\mathbf{Q} := \mathrm{cay}(\mathbf{D}) = (\mathbf{I} + \mathbf{D})(\mathbf{I} - \mathbf{D})^{-1}. \tag{7}$$

Since this transformation can be computed exactly, we can safely rely on automatic differentiation of the computational graph to compute its gradients. When coupling this with SNF, we refer to the resulting approach as Cayley Sylvester normalizing flows (C-SNF).

The Cayley map is a *diffeomorphism* (*i.e.*, it is differentiable bijection and its inverse is also differentiable) between $\mathbf{D} \in \mathrm{Skew}(n)$ and the set $\mathrm{V}(n) = \{\mathbf{Q} \in \mathrm{SO}(n) \mid \det(\mathbf{Q}+\mathbf{I}) \neq 0\}$, i.e. all the matrices in $\mathrm{SO}(n)$ where $(\mathbf{Q}+\mathbf{I})$ is invertible. The fact that it is a diffeomorphism implies that it preserves the critical points, and hence it also does not introduce any

saddle points or local minima.

The fact that the Cayley map is a bijection onto $\mathrm{V}(n)$ implies that there is a set of matrices in $\mathrm{SO}(n)$ that cannot be expressed under this parameterization. This set corresponds to matrices $\mathbf{Q}$ such that $\mathbf{Q}+\mathbf{I}$ is non-invertible which occurs when the norm of the corresponding matrix $\mathbf{D}$ tends to infinity. To give some intuition, for $\mathrm{SO}(3)$ this arises when $\mathbf{Q}$ is a rotation through a full π radians about an arbitrary axis. The exponential map parameterization, introduced below, does not present this issue, and hence at times it might be more effective approach. On the other hand, the Cayley map has the practical advantage of being easier to implement and faster to compute.

4.2. Exponential map parameterization

The second method that we propose, exponential Sylvester normalizing flows E-SNF, builds on the work of (Lezcano-Casado & Martínez-Rubio, 2019) by using the *exponential parameterization*. The function $\exp\colon \mathrm{Skew}(n) \to \mathrm{SO}(n)$ is defined such that

$$\mathbf{Q} := \exp(\mathbf{D}) = \sum_{k=0}^{\infty} \frac{\mathbf{D}^k}{k!} = \mathbf{I} + \mathbf{D} + \tfrac{1}{2}\mathbf{D}\mathbf{D} + \cdots \quad (8)$$

where the powers denote matrix multiplications. This function is surjective, and thus, every special orthogonal matrix $\mathbf{Q} \in \mathrm{SO}(n)$ can be represented as the exponential of a skew-symmetric matrix. It also has good properties in terms of the optimization landscape: (Lezcano-Casado & Martínez-Rubio, 2019) show that exponential map is a diffeomorphism between a bounded, connected, subset of $\mathrm{Skew}(n)$ and almost all of $\mathrm{SO}(n)$, with the exception of a set for which the corresponding matrices $\mathbf{D}$ are of measure zero in $\mathrm{Skew}(n)$. Intuitively, this means that the subset of $\mathrm{SO}(n)$ affected is much smaller than for the Cayley map and hence will not cause problems in practice. More specifically, it does not experience issues with the values in matrix $\mathbf{D}$ tending to infinity to represent any matrix $\mathbf{Q}$ or introduce any spurious local minima. See (Lezcano-Casado & Martínez-Rubio, 2019) for further details.

We use the efficient numerical implementation of the exponential function provided by (Lezcano-Casado & Martínez-Rubio, 2019),[1] which is based on the approach by (Higham, 2005) and which is able to tractably approximate the value of the function to within machine precision. Though this approximation algorithm does not offer a guarantee on the convergence of the gradient to the correct gradient, their implementation corrects for this by separately computing a tailored approximation of the gradient for the exponential function itself, rather than relying on an auto-differentiation to compute the gradient of the approximation.

[1]Their code is available at https://github.com/lezcano/exprnn/.

5. Experiments

One of the most common applications for normalizing flows is to use them to parameterize a flexible posterior approximation $q_\phi(\mathbf{z}|\mathbf{x})$ in a variational autoencoder (VAE) (Kingma & Welling, 2014; Rezende et al., 2014) context. VAEs are an approach to learning a generative model $p_\theta(\mathbf{x}, \mathbf{z})$, where $\mathbf{z}$ represents a set of latent variables and $\mathbf{x}$ represents data that we are trying to model. Direct maximum likelihood estimation of the model parameters θ (usually taking the form of a neural network) is generally intractable and so a common strategy is to instead optimize a variational lower bound on $\log p_\theta(\mathbf{x})$, known as an ELBO, constructed using an *inference network* $q_\phi(\mathbf{z}|\mathbf{x})$:

$$\log p_\theta(\mathbf{x}) \geq \overbrace{\log p_\theta(\mathbf{x})}^{\mathrm{LL}} - \overbrace{\mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x}))}^{\mathrm{KL}(q_\phi\|p_\theta)} \quad (9)$$
$$\underbrace{\qquad\qquad\qquad\qquad}_{\mathrm{ELBO}}$$

where LL is the log marginal likelihood, *a.k.a.* evidence. Alternatively, one can think of a VAE in terms of a deep stochastic autoencoder, for which $q_\phi(\mathbf{z}|\mathbf{x})$ plays the role of the encoder and $p_\theta(\mathbf{z}|\mathbf{x})$ the decoder.

The success of a VAE relies on having access to powerful variational families for the inference network, which can be thought of as amortized proposal. The flexibility of normalizing flows therefore makes them an ideal candidate for this variational family. In this amortized context, a neural network is typically used the generate the parameters of the normalizing flow (e.g. $\mathbf{C}_{1:K}$, $\mathbf{R}_{1:K}$, $\widetilde{\mathbf{R}}_{1:K}$, and $\mathbf{b}_{1:K}$ for our methods) for a given datapoint $\mathbf{x}$.

The $\mathrm{KL}(q_\phi\|p_\theta)$ term indicates how well the approximate posterior, $q_\phi(\mathbf{z}|\mathbf{x})$, matches the true posterior and hence is the value of particular interest in the context of our evaluation because, as long as LL stays constant or increases,[2] reductions in $\mathrm{KL}(q_\phi\|p_\theta)$ indicate improvements in the posterior approximation. The LL conveys the fidelity of the learned model, but this is a secondary consequence of improving the performance of our posterior approximation. Hence, investigating the KL term allows us to better single out the effect of improved normalizing flow performance.

We perform the evaluation of E-SNF and C-SNF on a similar suite of experiments as (van den Berg et al., 2018), building upon their implementation.[3] Qualitative plots showing example reconstructions from the learned VAEs are presented in Figure 3 in Appendix C. For more concrete numerical assessment, we compare to the three variants of SNF de-

[2]Otherwise one could argue that such a choice of approximate posterior makes the training converge to a model $p_\theta(\mathbf{x}|\mathbf{z})$ with a simpler posterior $p_\theta(\mathbf{z}|\mathbf{x})$ and hence allowing the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ to match $p_\theta(\mathbf{z}|\mathbf{x})$ better, at the expense of the model's log marginal likelihood $p_\theta(\mathbf{x})$.

[3]Their code is available at https://github.com/riannevdberg/sylvester-flows.

Table 1: Negative evidence lower bound (-ELBO), negative log marginal likelihood (NLL), and the $\mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x}))$ term [lower is better, best score in bold] on the test set for statically binarized MNIST, OMNIGLOT, and CALTECH 101 Silhouettes datasets, reported in nats. Mean and standard deviations are estimated over 3 different runs. For models other than C-SNF and E-SNF we use the numbers reported by (van den Berg et al., 2018) and (De Cao et al., 2019). The standard deviations for the KL term is only reported for C-SNF and E-SNF because we do not have access to single run results of (van den Berg et al., 2018) or (De Cao et al., 2019).

Model	MNIST			OMNIGLOT			CALTECH 101		
	−ELBO	NLL	$\mathrm{KL}(q_\phi\|p_\theta)$	−ELBO	NLL	$\mathrm{KL}(q_\phi\|p_\theta)$	−ELBO	NLL	$\mathrm{KL}(q_\phi\|p_\theta)$
VAE	$86.55_{\pm.06}$	$82.14_{\pm.07}$	4.41	$104.28_{\pm.39}$	$97.25_{\pm.23}$	7.03	$110.80_{\pm.46}$	$99.62_{\pm.74}$	11.18
Planar	$86.06_{\pm.31}$	$81.91_{\pm.22}$	4.15	$102.65_{\pm.42}$	$96.04_{\pm.28}$	6.61	$109.66_{\pm.42}$	$98.53_{\pm.68}$	11.13
IAF	$84.20_{\pm.17}$	$80.79_{\pm.12}$	3.41	$102.41_{\pm.04}$	$96.08_{\pm.16}$	6.33	$111.58_{\pm.38}$	$99.92_{\pm.30}$	11.66
B-NAF	$83.59_{\pm.15}$	$80.71_{\pm.09}$	**2.88**	$100.08_{\pm.07}$	$94.83_{\pm.10}$	5.25	$105.42_{\pm.49}$	$94.91_{\pm.83}$	10.51
O-SNF	$83.32_{\pm.06}$	$80.22_{\pm.03}$	3.10	$99.00_{\pm.29}$	$93.82_{\pm.21}$	5.18	$106.08_{\pm.39}$	$94.61_{\pm.83}$	11.47
H-SNF	$83.40_{\pm.01}$	$80.29_{\pm.02}$	3.11	$99.00_{\pm.04}$	$93.77_{\pm.03}$	5.23	$104.62_{\pm.29}$	$93.82_{\pm.62}$	10.80
T-SNF	$83.40_{\pm.10}$	$80.28_{\pm.06}$	3.12	$99.33_{\pm.23}$	$93.97_{\pm.13}$	5.36	$105.29_{\pm.64}$	$94.92_{\pm.73}$	10.37
C-SNF	$83.08_{\pm.02}$	**80.08**$_{\pm.01}$	$3.00_{\pm.03}$	**97.14**$_{\pm.84}$	$93.48_{\pm.23}$	**3.66**$_{\pm.99}$	$102.71_{\pm.54}$	$93.92_{\pm.08}$	$8.80_{\pm.49}$
E-SNF	**83.03**$_{\pm.04}$	$80.09_{\pm.06}$	$2.94_{\pm.01}$	$98.11_{\pm.80}$	**93.47**$_{\pm.10}$	$4.63_{\pm.71}$	**101.88**$_{\pm.36}$	**93.30**$_{\pm.32}$	**8.58**$_{\pm.45}$

scribed in Section 3, along with a number of other popular parameterizations: a vanilla VAE where the approximate posterior is parameterized with an isotropic Gaussian distribution, and other normalizing flows which have been developed with variational inference in mind—namely planar flows (Rezende & Mohamed, 2015), and inverse autoregressive flows (IAF) (Kingma et al., 2016). We further add the evaluations of block neural autoregressive flows (B-NAF) from (De Cao et al., 2019) to our comparisons.

We used the same setup and parameters as (van den Berg et al., 2018) with the exception of minor changes, the experimental details are in the Appendix A.

Negative LL (NLL) on the test set was estimated using importance sampling, as proposed by (Rezende et al., 2014), taking 2000 samples for CALTECH 101, and 5000 samples for MNIST and OMNIGLOT. The KL is computed as the difference between ELBO and the NLL estimate. We also perform time benchmarking of all of the methods, the results for which are given in Appendix B.

Our quantitative experimental results are summarized in Table 1. These show that E-SNF and C-SNF outperformed all the baseline methods on all three metrics (ELBO, NLL, $\mathrm{KL}(q_\phi\|p_\theta)$) on all three datasets (statically binarized MNIST (Larochelle & Murray, 2011), OMNIGLOT (Lake et al., 2015), and CALTECH 101 Silhouettes (Marlin et al., 2010)), with the exception that B-NAF produced a lower $\mathrm{KL}(q_\phi\|p_\theta)$ for MNIST (which may simply be a reflection of learning a simpler model, given its worse NLL score). What is more, for all three dataset the reduction of KL between our methods and the other SNFs is of roughly the same magnitude as between those other SNFs and the IAF, implying that our orthogonalization methods significantly improve the training of the SNFs. In terms of the KL decrease, training using the previous methods of orthogonalization (H-SNF and O-SNF) gave little improvement over using a fixed $\mathbf{Q}$ (i.e. T-SNF).

Both E-SNF and C-SNF achieved higher ELBOs than the baselines and a big part of the reduction came from decreases in $\mathrm{KL}(q_\phi\|p_\theta)$. Both also achieve better NLL. As NLL is a metric which assesses solely the performance of the learned generative model, the inference network has only indirect influence on this score. It is possible that on the datasets we have used there is little potential left for improving the model through improving the approximate posterior beyond the gains already made by the other SNFs. This suggests that there might be more potential performance improvement for a model learning perspective on more difficult problems—more complicated datasets or more structured models (Webb et al., 2018). Further, there are many cases in which the quality of the provided inference is the main objective, such as in representation learning contexts (Bengio et al., 2013; Mathieu et al., 2019; Kingma et al., 2014; Siddharth et al., 2017) and more generally in any setting where we care about the fidelity of the approximate posterior itself (Goliński et al., 2019; Huang et al., 2018; Papamakarios et al., 2017).

6. Conclusions

We have explained how existing orthogonal matrix parameterizations in normalizing flows can suffer from issues such as spurious local minima, numerical instabilities, and poor gradient estimates. We introduced methods for correcting these shortfalls based on parameterizations stemming from Lie group through—the exponential map and the Cayley map. Combining these with Sylvester normalizing flows (van den Berg et al., 2018), we constructed two new normalizing flow approaches C-SNF and E-SNF, and showed that these provided empirical performance improvements in a VAE context over the previous parameterizations of (van den Berg et al., 2018), in addition to the normalizing flow approaches of (De Cao et al., 2019; Kingma et al., 2016; Rezende & Mohamed, 2015).

References

Behrmann, J., Grathwohl, W., Chen, R. T. Q., Duvenaud, D., and Jacobsen, J.-H. Invertible residual networks. *arxiv:1811.00995v2*, 2018.

Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35 (8):1798–1828, August 2013.

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

De Cao, N., Titov, I., and Aziz, W. Block neural autoregressive flow. *arxiv:1904.04676*, 2019.

Dinh, L., Krueger, D., and Bengio, Y. NICE: Non-linear Independent Components Estimation. *International Conference on Learning Representations (ICLR)*, 2015.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using Real NVP. *International Conference on Learning Representations (ICLR)*, 2017.

Falorsi, L., de Haan, P., Davidson, T. R., and Forré, P. Reparameterizing distributions on lie groups. *arXiv preprint arXiv:1903.02958*, 2019.

Goliński, A., Wood, F., and Rainforth, T. Amortized Monte Carlo Integration. *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.

Grathwohl, W., Chen, R. T. Q., Bettencourt, J., Sutskever, I., and Duvenaud, D. FFJORD: free-form continuous dynamics for scalable reversible generative models. *International Conference on Learning Representations (ICLR)*, 2019.

Higham, N. J. The scaling and squaring method for the matrix exponential revisited. *SIAM Journal on Matrix Analysis and Applications*, 26(4):1179–1193, 2005.

Hochreiter, S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

Hoogeboom, E., van den Berg, R., and Welling, M. Emerging convolutions for generative normalizing flows. *arXiv:1901.11137v2*, 2019.

Huang, C.-W., Krueger, D., Lacoste, A., and Courville, A. Neural autoregressive flows. *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. *Advances in Neural Information Processing Systems (NIPS)*, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. *International Conference on Learning Representations (ICLR)*, 2014.

Kingma, D. P., Rezende, D. J., Mohamed, S., and Welling, M. Semi-supervised learning with deep generative models. *Advances in Neural Information Processing Systems (NIPS)*, 2014.

Kingma, D. P., Salimans, T., and Welling, M. Improving variational inference with inverse autoregressive flow. *Advances in Neural Information Processing Systems (NIPS)*, 2016.

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

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Larochelle, H. and Murray, I. The Neural Autoregressive Distribution Estimator. *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

Lezcano-Casado, M. and Martínez-Rubio, D. Cheap orthogonal constraints in neural networks: A simple parametrization of the orthogonal and unitary group. *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.

Marlin, B., Swersky, K., Chen, B., and Freitas, N. Inductive principles for restricted boltzmann machine learning. *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

Mathieu, E., Rainforth, T., Siddharth, N., and Teh, Y. W. Disentangling disentanglement in variational auto-encoders. *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.

Papamakarios, G., Pavlakou, T., and Murray, I. Masked autoregressive flow for density estimation. *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. *Proceedings of the International Conference on Machine Learning (ICML)*, 2014.

Siddharth, N., Paige, B., , Van de Meent, J.-W., Desmaison, A., Wood, F., Goodman, N. D., Kohli, P., and Torr,

P. H. Learning disentangled representations with semi-supervised deep generative models. *Advances in Neural Information Processing Systems (NIPS)*, 2017.

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

Tabak, E. G., Vanden-Eijnden, E., et al. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217–233, 2010.

van den Berg, R., Hasenclever, L., Tomczak, J. M., and Welling, M. Sylvester normalizing flows for variational inference. *Proceedings of the conference on the Uncertainty in Artificial Intelligence (UAI)*, 2018.

Webb, S., Goliński, A., Zinkov, R., Siddharth, N., Rainforth, T., Teh, Y. W., and Wood, F. Faithful inversion of generative models for effective amortized inference. *Advances in Neural Information Processing Systems (NIPS)*, 2018.

Wen, Z. and Yin, W. A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1-2):397–434, 2013.

A. Experimental details

For each dataset the latent space was of size $D = 64$ and $K = 16$ flows were taken for each model. A bottleneck of $M = 32$ was used for O-SNF, while $M = 64$ was taken for all others. For H-SNF $H = 8$ householder reflections were used to construct orthogonal matrices. For IAF a MADE width of 1280 was used.

Lezcano-Casado & Martínez-Rubio (2019) found in their experiments that the optimization benefits from using a different learning rate for the parameters in the skew symmetric matrix than for other parameters of the model. In our case we do not optimize the parameters of the skew symmetric matrix directly, but that observation suggests that using the Cayley or the exponential map may affect the optimization behavior of our model as well. Hence we decided to perform a learning rate sweep—we have run one training run with each of the following learning rates $10^{-3} \times [5, 2, 0.8, 0.5, 0.2, 0.08, 0.05]$ on each of the three datasets, and then 3 runs with the learning rate of 2×10^{-4}, which performed best on the validation set across all of the datasets.

B. Time benchmarking results

We also empirically compare the computational cost of each of the methods in the Table 2. They are obtained by running 10 epochs of training, mean and standard deviation of time per epoch is reported.

C-SNF is only 30% more expensive than H-SNF while obtaining significantly better results. E-SNF and C-SNF are more expensive than the other methods in the comparison.

Table 2: Average time per epoch in seconds over 10 epochs. Variance is negligible.

Model	**MNIST**	**Omniglot**	**Caltech 101**
VAE	39.8	18.7	3.4
Planar	46.4	21.9	3.9
IAF	59.6	27.6	4.9
O-SNF	62.7	29.2	5.3
H-SNF	101.6	46.6	8.4
T-SNF	74.7	34.8	6.2
C-SNF	131.5	61.3	10.9
E-SNF	210.1	99.7	17.8

C. Qualitative Results

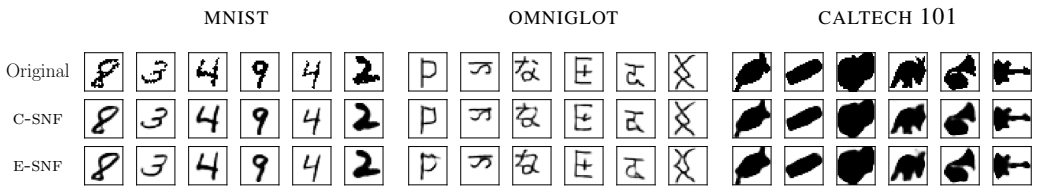


Figure 3: Reconstructions of the samples from the three datasets (top row) formed by a VAE using an approximate posterior formed by C-SNF (middle row) and E-SNF (bottom row).