

Expanding the Capabilities of Normalizing Flows in Deep Generative Models and Variational Inference

Anthony Lawrence Caterini

St. Hugh's College
University of Oxford

*A thesis submitted for the degree of
Doctor of Philosophy*

Trinity 2021

Abstract

Deep generative models and variational Bayesian inference are two frameworks for reasoning about observed high-dimensional data, which may even be combined. A fundamental requirement of either approach is the parametrization of an expressive family of density models. Normalizing flows, sometimes also referred to as invertible neural networks, are one class of models providing this: they are formulated to be bijective and differentiable, and thus produce a tractable density model via the change-of-variable formula. Beyond just deep generative modelling and variational inference, normalizing flows have shown promise as a plug-in density model in other settings such as approximate Bayesian computation and lossless compression. However, the bijectivity constraint can pose quite a restriction on the expressiveness of these approaches, and forces the learned distribution to have full support over the ambient space which is not well-aligned with the common assumption that low-dimensional manifold structure is embedded within high-dimensional data.

In this thesis, we challenge this requirement of strict bijectivity over the space of interest to modify normalizing flow models. The first work focuses on the setting of variational inference, defining a normalizing flow based on a discretized time-inhomogeneous Hamiltonian dynamics over an extended position-momentum space. This enables the flow to be guided by the true posterior unlike baseline flow-based models, thus requiring fewer parameters in the inference model to achieve comparable improvements in inference. The next chapter proposes a new deep generative model which relaxes the bijectivity requirement of normalizing flows by injecting learned noise at each layer, sacrificing easy evaluation of the density for expressiveness. We show, theoretically and empirically, the benefits of these models in density estimation over baseline flows. We then demonstrate in the next chapter that the benefits of this model class extend to the setting of variational inference, relying on auxiliary methods to train our models. Finally, the last paper in this thesis addresses the issue of full support in the ambient space and proposes injective flow models directly embedding low-dimensional structure into high dimensions. Our method is the first to optimize the injective change-of-variable term and produces promising results on out-of-distribution detection, which had previously eluded deep generative models. We conclude with some directions for future work and a broader perspective on the field.

Expanding the Capabilities of Normalizing Flows in Deep Generative Models and Variational Inference



Anthony Lawrence Caterini

St. Hugh's College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2021

To Allie

Without you, this would not have been possible

Acknowledgements

I'd first like to thank my wife Allie, whose unwavering support during my degree, and beyond, has been invaluable. You have always pushed me to be the best version of myself and for that I am grateful.

Next, I'll thank the rest of my family. To my sister, thank you for blazing the trail on doctoral studies and giving me great advice throughout all stages of my DPhil, and life in general. To my parents, I appreciate you fostering my interest in mathematics, science, and higher education, and providing me with the tools I needed to succeed all along the way. To my grandparents, thank you for supporting my move abroad and always caring for me. To my in-laws, I appreciate how you have always welcomed me like I was your own son/brother. And lastly, to my dogs: "Man's best friend(s)" is no overstatement!

Now I'd like to thank my supervisors, Arnaud Doucet and Dino Sejdinovic. You have both been excellent mentors throughout my studies, providing me with more support early on when I was new to the field and then admitting me increased autonomy as I progressed. Thank you for showing me how to reason about problems in machine learning and setting me up for a career as an independent researcher.

I'd also like to thank my collaborators and friends. To Rob Cornish, thank you for setting a spectacular example of productivity and hard work in the CIF project, and for being such a reliable friend. To Gabriel Loaiza Ganem, thank you for making my internship at Layer 6 so enjoyable that I decided to come back full-time, and for your important contributions to the rectangular flows project. To my other good friends Paul Vanetti, Dominic Richards, Emile Mathieu, Philippe Gagnon, Alex Shestopaloff, Jean-Francois Ton, Adam Golinski, Yuan Zhou, Sam Davenport, Schyan Zafar, Robert Hu, Emilien Dupont, Leon Law, Michael Li, Rachael Morey, and Jane Liew: you have all made my time in Oxford amazing, and I miss you already.

I'd finally like to thank my examiners David Barber and Yee Whye Teh for graciously giving their time to assess this thesis and provide helpful feedback, and the Commonwealth Scholarship Commission for funding my studies.

Abstract

Deep generative models and variational Bayesian inference are two frameworks for reasoning about observed high-dimensional data, which may even be combined. A fundamental requirement of either approach is the parametrization of an expressive family of density models. Normalizing flows, sometimes also referred to as invertible neural networks, are one class of models providing this: they are formulated to be bijective and differentiable, and thus produce a tractable density model via the change-of-variable formula. Beyond just deep generative modelling and variational inference, normalizing flows have shown promise as a plug-in density model in other settings such as approximate Bayesian computation and lossless compression. However, the bijectivity constraint can pose quite a restriction on the expressiveness of these approaches, and forces the learned distribution to have full support over the ambient space which is not well-aligned with the common assumption that low-dimensional manifold structure is embedded within high-dimensional data.

In this thesis, we challenge this requirement of strict bijectivity over the space of interest to modify normalizing flow models. The first work focuses on the setting of variational inference, defining a normalizing flow based on a discretized time-inhomogeneous Hamiltonian dynamics over an extended position-momentum space. This enables the flow to be guided by the true posterior unlike baseline flow-based models, thus requiring fewer parameters in the inference model to achieve comparable improvements in inference. The next chapter proposes a new deep generative model which relaxes the bijectivity requirement of normalizing flows by injecting learned noise at each layer, sacrificing easy evaluation of the density for expressiveness. We show, theoretically and empirically, the benefits of these models in density estimation over baseline flows. We then demonstrate in the next chapter that the benefits of this model class extend to the setting of variational inference, relying on auxiliary methods to train our models. Finally, the last paper in this thesis addresses the issue of full support in the ambient space and proposes injective flow models directly embedding low-dimensional structure into high dimensions. Our method is the first to optimize the injective change-of-variable term and produces promising results on out-of-distribution detection, which had previously eluded deep generative models. We conclude with some directions for future work and a broader perspective on the field.

Contents

1	Introduction	1
1.1	Variational Inference	2
1.1.1	Bayesian Inference	2
1.1.2	Standard Variational Inference	3
1.1.3	Auxiliary Variational Inference	5
1.2	Deep Generative Models	6
1.2.1	Variational Auto-Encoders	7
1.2.2	Normalizing Flows	8
1.3	Contributions	11
1.3.1	Hamiltonian Variational Auto-Encoder	12
1.3.2	Continuously-Indexed Normalizing Flows	12
1.3.3	Rectangular Normalizing Flows	13
1.4	Summary and Outline	14
2	Hamiltonian Variational Auto-Encoder	16
3	Relaxing Bijectivity Constraints with Continuously Indexed Normalising Flows	32
4	Variational Inference with Continuously-Indexed Normalizing Flows	76
5	Rectangular Flows for Manifold Learning	97
6	Conclusion	125
6.1	Contributions	125
6.2	Challenges and Future Work	127
6.2.1	Hamiltonian Variational Auto-Encoder	127
6.2.2	Continuously-Indexed Normalizing Flows	128
6.2.3	Rectangular Normalizing Flows	129
6.3	Final Thoughts and Broader Context	131
	Bibliography	132

1

Introduction

This thesis is concerned with developing algorithms that automatically extract structure from high-dimensional data. Bayesian inference is one framework for intuitively reasoning about underlying factors of variation in observed data, but traditional methods can struggle to scale since computational requirements increase sharply with dimension. Circumventing this limitation are *variational* approaches, which introduce some bias but empirically converge much faster, particularly in high-dimensional settings. Some approaches to variational inference are also able to incorporate deep neural networks, casting them more generally into the realm of *deep generative models*. These methods provide a complementary perspective on unsupervised analysis of high-dimensional data, affording additional flexibility to the variational family of posteriors.

Throughout this thesis, we will focus on one class of deep generative model in particular: *normalizing flows* [Tabak and Vanden-Eijnden, 2010, Rezende and Mohamed, 2015]. Normalizing flows admit exact likelihoods and have thus risen to popularity as a family of plug-in density models in a number of settings, including variational Bayesian inference and directly as generative models of observed data. However, despite their successes, they can still struggle to model high-dimensional distributions with complicated topological structure in a numerically stable way; since they also do not explicitly include information about the target distribution,

they may experience difficulties in variational inference as well. We will highlight in this section how we can ameliorate some of these issues by either *augmenting* or *reducing* the dimensionality of the flow problem where appropriate, deferring the details to the papers constituting the body of this thesis. We begin by briefly reviewing the key modelling and inferential frameworks we will consider in this thesis: variational Bayesian inference, deep generative models, and normalizing flows. Since this thesis is in the integrated format, the full literature review is deferred to the individual chapters.

1.1 Variational Inference

In this section we introduce variational inference, as two of the chapters in this thesis are focused on applying normalizing flows to this setting. We start with a brief overview of Bayesian inference, discuss how to build and optimize explicit variational models of the posterior, and lastly discuss how to train variational models on extended spaces.

1.1.1 Bayesian Inference

Bayesian inference provides a framework for reasoning about data probabilistically, for example as a means for inferring the latent processes underlying the creation of observed data, or generating predictions with quantified uncertainty [Gelman et al., 2013]. In this setting, we assume we have access to the following:

- A *prior* distribution with density¹ p_Z over some space $\mathcal{Z}$ describing a *latent* variable Z .
- A *likelihood* model $p_{X|Z}(\cdot | Z)$ describing how the *data* X defined on some space $\mathcal{X}$ is generated from the latent variable Z .

¹Although densities are not necessarily required here, we will assume throughout that latent variables admit a density as it simplifies the notation and presentation.

Then, given observed data² $x \in \mathcal{X}$, the object of interest becomes the *posterior* distribution, which by Bayes' rule has density

$$p_{Z|X}(z | x) = \frac{p_{X,Z}(x, z)}{\int_{\mathcal{Z}} p_{X,Z}(x, z') \, dz'}, \quad (1.1)$$

where we define the joint distribution $p_{X,Z}(x, z) := p_{X|Z}(x | z) \cdot p_Z(z)$ for all $x \in \mathcal{X}$ and $z \in \mathcal{Z}$.

When performing Bayesian analysis, we are typically interested in computing expectations of the form $\mathbb{E}_{p_{Z|X}(\cdot | x)} f(Z)$ for some function f defined on $\mathcal{Z}$, where $p_{Z|X}(\cdot | x)$ is the posterior. However, the integral appearing in the denominator of (1.1) is numerically intractable in many cases. Markov chain Monte Carlo (MCMC) [Brooks et al., 2011] is a popular approach to sidestep this issue, relying on the fact that we only require *samples* from the posterior and not the entire functional form to compute the required expectation. MCMC methods construct a Markov chain with invariant distribution given by the posterior [Geyer, 2011]; this chain will converge to the posterior given enough computation time. Yet despite this asymptotic exactness, the number of samples (and therefore computation time) required for good coverage of the posterior can increase rapidly with dimension [Neal, 2011, Section 4.4] and thus we focus on alternative approaches here.

Stochastic expectation maximization methods [Cappé and Moulines, 2009] are yet another class of algorithms for approximate inference, but are less generally applicable in that they require exponential family distributions with a particular structure to ensure tractability [Chen et al., 2018]. We will therefore not study these approaches further in this thesis.

1.1.2 Standard Variational Inference

Instead of attempting to just sample from the posterior, we may elect to directly parametrize an approximate posterior as is done in *variational inference* [Wainwright and Jordan, 2008, Blei et al., 2017]. The goal now is to optimize the parameters of

²Traditionally this refers to an observation of an entire dataset, but may also denote just a single datapoint in contexts like factor modelling or amortized inference. We discuss the latter in subsection 1.2.1.

this approximate posterior – here denoted q_Z – by minimizing the KL divergence from the true posterior $p_{Z|X}(\cdot | x)$ for observed data x . For a fixed model $p_{X,Z}$, this is equivalent to maximizing the *evidence lower bound* (ELBO), given by

$$\mathcal{L}(x) := \mathbb{E}_{z \sim q_Z} [\log p_{X,Z}(x, z) - \log q_Z(z)], \quad (1.2)$$

which can be shown to be a lower bound on the (typically) intractable log-evidence $\log p_X(x) := \log \int p_{X,Z}(x, z) dz$. This objective introduces bias not present in MCMC methods – as parametric assumptions must be made, which can limit the class of posteriors our variational model can represent – but has also demonstrated the ability to computationally scale to higher-dimensional settings [Hoffman et al., 2013, Blei et al., 2017].

Much of the research in variational inference (see Blei et al. [2017] for a review) focuses on specifying the family of variational models over which we optimize. The challenge here involves parametrizing a class of distributions for which you can calculate (or at least estimate) $\nabla \mathcal{L}(x)$, where ∇ denotes differentiation with respect to the parameters of q_Z . Mean-field approaches, wherein $q_Z := \prod_i q_{Z_i}$ is factored independently along the dimensions of latent variable $Z \equiv (Z_i)_i$, were the first successful type of variational methods. These models are attractive for their simplicity: $\mathcal{L}(x)$ may be available in closed-form, and even if not, its gradient is typically straightforward to estimate. However, simplicity comes at the cost of representational power, as we disregard any possible dependence between coordinates in latent space. This lack of representational power, combined with the mode-seeking behaviour of the KL objective, can in turn lead to underestimation of the variance of the posterior [Yin and Zhou, 2018]. Improvements have been made on the mean-field model to introduce dependence between variables – including using normalizing flows [Rezende and Mohamed, 2015] which we will discuss later – but the lack of expressiveness inherent in the standard approach of parametrizing explicit q_Z remains a difficult limitation to overcome.

1.1.3 Auxiliary Variational Inference

Explicit models are convenient for optimizing (1.2) since $\log q_Z$ can be evaluated directly. However, to gain expressiveness, we might consider defining q_Z *implicitly*, i.e. introduce into the variational model an *auxiliary* latent variable U defined on some space $\mathcal{U}$ so that, for all $z \in \mathcal{Z}$, $q_Z(z) := \int_{\mathcal{U}} q_{Z,U}(z, u) du$ is defined via the joint distribution $q_{Z,U}$ over latent and auxiliary variables. If we are able to specify tractable $q_{Z,U}$, we can then use the framework of *auxiliary variational inference* [Agakov and Barber, 2004, Ranganath et al., 2016] to optimize our approximate posterior. Since (1.2) can no longer easily be estimated, these methods propose optimizing an *auxiliary* ELBO instead, given below as

$$\mathcal{L}'(x) := \mathbb{E}_{(z,u) \sim q_{Z,U}} [\log p_{X,Z}(x, z) - \log q_{Z,U}(z, u) + \log r_{U|Z}(u | z)], \quad (1.3)$$

where we introduce an *auxiliary inference distribution* $r_{U|Z}$ to infer the auxiliary variables given the latent variables. It is not hard to show (e.g. [Lawson et al., 2019, Eq. (4)]) that (1.2) relates to (1.3) via

$$\mathcal{L}(x) = \mathcal{L}'(x) + \mathbb{E}_{z \sim q_Z} \left[D_{\text{KL}} \left(q_{U|Z}(\cdot | z) \parallel r_{U|Z}(\cdot | z) \right) \right]. \quad (1.4)$$

Thus, optimizing $\mathcal{L}'$ instead of $\mathcal{L}$ incurs additional bias, but the improved expressiveness from using implicit q_Z can result in a lower KL divergence from q_Z to the true posterior $p_{Z|X}$ – for example when using continuously-indexed flows (CIFs) instead of normalizing flows for variational inference as in chapter 4 – and thus a better inference model overall. Furthermore, in certain cases we can control the bias by structuring $r_{U|Z}$ to match the form of the typically intractable $q_{U|Z}$, which we also leverage in chapter 4 when stacking layers of CIFs.

It can be shown that many recent developments in variational inference are instances of the auxiliary variational inference framework [Lawson et al., 2019, Section 5]. Methods such as Hamiltonian variational inference [Salimans et al., 2015], the importance weighted auto-encoder [Burda et al., 2016], and semi-implicit variational inference [Yin and Zhou, 2018] can all be seen as optimizing a bound of the same form as $\mathcal{L}'$ in training. Auxiliary variational inference thus remains

a promising direction for improving practical performance of inference methods and is a major focus of this thesis.

1.2 Deep Generative Models

We now shift away from a purely Bayesian approach and enter the realm of deep generative models, which are a class of probabilistic methods containing neural-network-based components [Ruthotto and Haber, 2021]. Given an observed set of data $\mathcal{D} := \{x_i\}_{i=1}^n$, where now *each* datapoint x_i satisfies $x_i \in \mathcal{X}$, these methods typically attempt to parametrize the *density* underlying the unknown data-generating distribution. Thus, deep generative models can broadly be considered as methods for *density estimation*, and many of them are trained using either exact maximum likelihood estimation or an approximation thereof.

Deep generative models are becoming particularly popular for density estimation in high dimensions, as traditional methods such as the histogram [Birgé and Rozenholc, 2006] or the kernel density estimator [Zambom and Ronaldo, 2013] are plagued by the *curse of dimensionality*: they provably require the number of observed samples to increase exponentially with the problem dimension to produce acceptable density estimates [Nagler and Czado, 2016]. Deep belief networks [Hinton et al., 2006] were among the first deep approaches to probabilistic modelling and provided useful representations for classification, but required a more antiquated layer-wise training scheme. These models more generally belong to the class of *energy-based models* [Teh et al., 2003, LeCun et al., 2006], which has recently re-emerged as a popular and expressive framework for deep probabilistic modelling [Du and Mordatch, 2019, Nijkamp et al., 2019], including as an extension to other deep generative frameworks [Xiao et al., 2020, Arbel et al., 2020]. However, intractability of the parametrized density makes training these models a challenge [Song and Kingma, 2021], requiring careful use of MCMC techniques which are again tough to apply in high-dimensional settings. Generative adversarial networks [Goodfellow et al., 2014] are yet another prevalent deep generative model with striking results on image generation [Karras et al., 2019, Brock et al., 2019], but are themselves

difficult to train and do not admit tractable likelihoods. Therefore in this thesis we focus on other approaches, such as variational autoencoders or normalizing flows (further described in subsection 1.2.1 and subsection 1.2.2, respectively) which result in densities that are either tractable to compute or straightforward to estimate, and are thus easier to train.

1.2.1 Variational Auto-Encoders

As noted earlier, Bayesian inference provides a general framework for reasoning about data. The principles of variational inference can also be used as part of a larger generative modelling scheme which does not treat all parameters in a Bayesian fashion, as the ELBO (or auxiliary ELBO) can serve as a surrogate loss function for maximum likelihood estimation. This is the idea underlying the variational auto-encoder (VAE) [Kingma and Welling, 2014, Rezende et al., 2014], which we describe in this section. Here, we model each point in the dataset as the X -marginal of the generative process

$$Z \sim p_Z, \quad X \sim p_{X|Z}(\cdot | Z), \quad (1.5)$$

where p_Z is the *prior* distribution of some lower-dimensional latent variable, and $p_{X|Z}$ is a *likelihood* model comprised of neural networks with parameters θ . The goal here is to maximize the intractable marginal log-likelihood $\log p_X$ with respect to the parameters of θ over the observed dataset $\mathcal{D}$.

Now, given an inference model q_Z with parameters ϕ , we can learn both θ and ϕ concurrently by maximizing the ELBO (1.2), which provides a lower bound on $\log p_X$. This allows us to perform Bayesian inference for the latent variables while simultaneously learning θ in a frequentist, albeit biased, manner, with the amount of bias moderated by the quality of posterior approximation. Although we would optimally choose a completely different inference model for each datapoint [Cremer et al., 2018], such an approach is unable to share information across examples and may lead to overfitting, increased computational cost, and difficulties in training the model. Thus, early approaches instead *amortized* the cost of inference across the

entire dataset, building a shared approximate posterior $q_{Z|X}$ generally also endowed with neural network structure [Kingma and Welling, 2014, Rezende et al., 2014]. Yet early work still constrained $q_{Z|X}$ to have a multivariate Gaussian structure with diagonal covariance, even further limiting the expressiveness of the variational family and thus increasing the bias of the training procedure.

Improving on this explicit mean-field Gaussian parametrization has been one of the major drivers of recent VAE (and, more generally, variational inference) research. Normalizing flows [Rezende and Mohamed, 2015] provide a direct method to increase the flexibility of posterior approximation while keeping the ELBO tractable and will be described further in subsection 1.2.2. Auxiliary variational methods, such as those described in subsection 1.1.3, have also featured as an expressive *implicit* variational family in VAEs, although they often require auxiliary objectives to train and introduce additional bias. Two of the chapters of this thesis – chapter 2 and chapter 4 – focus on methods sitting at the intersection of these approaches, inheriting a degree of tractability from the inclusion of normalizing flows while parametrizing expressive auxiliary variational families.

1.2.2 Normalizing Flows

In this section, we now introduce *normalizing flows* [Tabak and Vanden-Eijnden, 2010, Challis and Barber, 2012, Tabak and Turner, 2013, Dinh et al., 2014, Rezende and Mohamed, 2015], which have emerged as a flexible, general-purpose framework for modelling or estimating densities; as stated earlier, models of this type feature heavily throughout the thesis as we work with them in each body chapter. Normalizing flows construct a density model p_X as the pushforward of a simple *base distribution*³ p_W over $\mathbb{R}^d$ through a parametrized diffeomorphism – i.e. a differentiable bijection – $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$, or equivalently as the X -marginal of the generative process

$$W \sim p_W, \quad X := f(W). \quad (1.6)$$

³Note that we use the non-standard variable W here to avoid notational clashes with the latent variable Z from Bayesian modelling and the auxiliary variable U from auxiliary variational inference and subsection 1.3.2.

The function f is typically referred to as the *flow*. The generative model here directly parametrizes a bijective mapping from latent to data space, thus allowing for exact inference and inversion of the model, in contrast with the VAE model shown in (1.5) which parametrizes a likelihood model and thus has more freedom to represent a wider range of densities. However, the most attractive property of normalizing flows is that they theoretically admit exact densities for p_X – unlike VAEs – via the change-of-variable formula

$$p_X(x) = p_W(f^{-1}(x)) \cdot |\det Df(f^{-1}(x))|, \quad (1.7)$$

where Df denotes the Jacobian of f ; flows are designed such that the determinant of this term is efficient to compute, which then allows for straightforward training via maximum likelihood for density estimation. Normalizing flows may also be used as components of larger modelling frameworks requiring density models, such as variational inference (the use of flows here is described further in section 1.2.2), approximate Bayesian computation [Papamakarios et al., 2019b], lossless compression [Ho et al., 2019], and reinforcement learning [Singh et al., 2020]. More details on specific flow implementations and applications are available in the survey papers Kobyzev et al. [2020] and Papamakarios et al. [2019a].

While bijectivity is convenient for computing densities, it also provably limits the class of distributions that the model can express, as we will elaborate on in subsection 1.3.2 and chapter 3. This can cause poor conditioning in the Jacobian matrix and non-invertibility in the flow, as has been empirically observed in [Behrmann et al., 2021] as well. In this case, (1.7) becomes unreliable, at which point we no longer achieve the exact likelihoods motivating normalizing flows in the first place. One way to avoid this issue is to enforce bijectivity on an even higher-dimensional space and then integrate out the extra variables; a technique following this principle is introduced in subsection 1.3.1 and chapter 2, and has been shown in the literature [Dupont et al., 2019, Zhang et al., 2019, Chen et al., 2020] to provide both theoretical and empirical benefits in certain settings. An alternative, but related, approach, which we will briefly describe in subsection 1.3.2

(and in more detail in chapter 3), is to relax the bijectivity constraint by directly augmenting any standard flow-based generative model with additional noise at each layer. This construction has shown empirical improvements in density estimation. We can see that in either case, we modify the flow with additional dimensions of randomness to achieve benefits in expressiveness at the cost of inexact likelihoods.

However, both of these techniques – along with the original normalizing flow model – still define a distribution having full support over $\mathbb{R}^d$. When modelling high-dimensional data, this construction violates the *manifold hypothesis*, which states that such data typically lies on a low-dimensional manifold embedded in the ambient space. We can thus instead use flow-based methods to define densities directly on manifolds. Previous work [Rezende et al., 2020, Mathieu and Nickel, 2020] has accomplished this by first assuming that the data assumes a particular manifold structure, and then fitting a density on it. Unfortunately this approach has not been shown to scale to high-dimensional settings of interest in this thesis where the manifold is unknown. On the other hand, in subsection 1.3.3 and chapter 5 we present a method based on *injective*, or *rectangular*, flows, which simultaneously learns a generic data manifold and an associated density with respect to the Riemannian measure on this manifold. This allows for direct and flexible embedding of low-dimensional structure within a high-dimensional ambient space. We find that our approach also correctly performs out-of-distribution detection on certain image datasets – which had previously been a well-known failure mode of full-support likelihood-based generative models [Nalisnick et al., 2018] – further motivating the need for manifold-supported models of high-dimensional data.

Normalizing Flows in Variational Inference

In this section we elaborate further on normalizing flows for variational inference, which historically had motivated early research on normalizing flows [Tabak and Vanden-Eijnden, 2010, Rezende and Mohamed, 2015]. Here, a flow model is used to parametrize the variational posterior q_Z (or $q_{Z|X}$ when including *amortization*). A typical approach is to first take a pre-existing explicit variational model, such

as a mean-field Gaussian, as the base distribution, and then apply a flow on top of samples from this distribution to introduce complex dependencies between coordinates of the variational distribution. Assuming the identity transformation is in the class of possible flows, this technique generalizes the pre-existing variational inference procedure at the cost of additional parameters. Normalizing flows can even be used to improve the inference model q_Z in VAEs (as proposed by Rezende and Mohamed [2015]), which as noted above had originally prescribed a neural-network based mean-field Gaussian inference distribution.

However, despite their improved expressiveness, vanilla normalizing flow models may not be well-suited to approximate complicated target posterior distributions because the flows themselves do not take into account this target in their construction. Only the ELBO training objective from (1.2) provides information about the target distribution to the flow, which is itself quite indirect. This can hamstring the performance of flows in variational inference as training these often heavily-parametrized models becomes very challenging, akin to shooting in the dark. There is also no natural way to *amortize* individual flow steps, with methods often relying on data-dependent hypernetworks – i.e. neural networks whose outputs are themselves the parameters of a different neural network – with additional hyperparameters to parametrize the flow at each layer [Rezende and Mohamed, 2015, Kingma et al., 2016, Van Den Berg et al., 2018]. In subsection 1.3.1 and chapter 2, we demonstrate how to overcome these issues and introduce a flow method specifically for variational inference which uses time-inhomogenous Hamiltonian dynamics guided by the true posterior. Our approach also provides a natural mechanism for amortization, avoiding the use of temperamental hypernetworks to output the flow parameters.

1.3 Contributions

Now we briefly discuss the contributions made in this thesis.

1.3.1 Hamiltonian Variational Auto-Encoder

Recall as discussed in section 1.2.2 that one of the major limitations of using normalizing flows in variational inference is that the flow itself is not guided by information about the target distribution. Thus, the first paper in this thesis – featured in chapter 2 – presents a method augmenting the inference distribution of a VAE with discretized Hamiltonian dynamics, which naturally incorporates information about the true posterior via the potential energy function contained in the Hamiltonian. The flow of the dynamics can be seen as a target-informed normalizing flow on the extended position-momentum space. It is also straightforward to include amortization in this framework. We build on previous work [Salimans et al., 2015] which used Hamiltonian Monte Carlo [Neal, 2011] to parametrize the approximate posterior, but avoid taking MCMC steps and thus lower the variance of our ELBO estimators by calculating optimal reverse kernels. Our method can be seen as an application of Hamiltonian importance sampling [Neal, 2005], but with the introduction of a learned variational tempering scheme which demonstrates benefits over fixed tempering in practice.

1.3.2 Continuously-Indexed Normalizing Flows

Next, in chapter 3, we theoretically demonstrate the failure of bijective flow-based models: to exactly model target distributions with support non-homeomorphic to the base distribution, the flow must necessarily approach non-invertibility. This is problematic because it appears quite unlikely that the support of any complex high-dimensional dataset is homeomorphic to that of the typical Gaussian base distribution. Further, this cannot be simply solved by using a base distribution with the correct topology, as determining this is highly non-trivial. We thus propose an alternative flow based model, a *continuously-indexed flow* (CIF), which augments (1.6) with additional randomness and defines p_X as the X -marginal of

$$W \sim p_W, \quad U \sim p_{U|W}(\cdot | W), \quad X = F(W; U). \quad (1.8)$$

Here, $p_{U|W}$ is a parametrized distribution over some continuous *indexing* space $\mathcal{U}$, and $F : \mathbb{R}^d \times \mathcal{U} \rightarrow \mathbb{R}^d$ is defined such that $F(\cdot; u)$ is a diffeomorphism for each $u \in \mathcal{U}$. The price we pay for additional expressiveness is that the density model p_X is no longer tractable, but we can still train these models using amortized variational inference as an approximation to maximum likelihood (similarly to the VAE), and at test time we can estimate likelihood to arbitrary accuracy using importance sampling. We can additionally stack the generative process (1.8) to achieve even more expressive power, while not suffering on inference since bijectivity allows us to at least structure the approximate posterior over indexing variables in the same form as the true posterior. We verify empirically that adding continuous indexing to normalizing flows provides the capacity to learn a far richer set of distributions than just the baseline flows themselves on a variety of density estimation benchmarks.

However, it is not immediately obvious how we might apply continuous indexing in other settings that normalizing flows have proven useful, such as variational inference, since we no longer have access to an explicit density model. On the other hand, we do have access to a tractable joint distribution over the variables of interest and the *indexing* variables, thus making CIFs a candidate for *auxiliary* variational inference, which we described in subsection 1.1.3. We develop such an approach in chapter 4. Beyond just defining a tractable joint distribution, we also outline how the model for inferring the CIF indexing variables can serve as the auxiliary inference distribution $r_{U|Z}$, further revealing the natural fit of CIFs here. We can transfer many of the design choices for CIFs in density estimation to this setting, although we provide additional support for amortization and reparametrization not required in the original model. We show improvements over baseline flows for inferring variables with topologically complex support, thus hinting that CIFs may be able to improve on flows in other applications as well.

1.3.3 Rectangular Normalizing Flows

The last paper in this thesis – alluded to in section 1.2.2 and provided in chapter 5 – focuses on the problem of making flow models in high dimension compatible with

the manifold hypothesis. We use a construction similar to [Brehmer and Cranmer, 2020] to model observed data in $\mathbb{R}^d$ as the pushforward of a simple density on $\mathbb{R}^{d'}$, with $d' < d$, through an injection g . This directly defines a d' -dimensional manifold $\mathcal{M} := g(\mathbb{R}^{d'})$ embedded in d -dimensional ambient space, and admits a density on $\mathcal{M}$ via the manifold change-of-variable formula [Krantz and Parks, 2008]. However, the volume-change term in this formulation is much tougher to evaluate than the corresponding term in normalizing flows (1.7), and thus Brehmer and Cranmer [2020] propose a heuristic training procedure to avoid calculating it. We argue that this method discards important information that makes learning the model challenging, and empirically verify our claims. We also show that exact evaluation of this volume-change term is more tractable than previously thought, and that approximation of the gradient of this term can still provide useful signal to the optimization procedure at a fraction of the cost of the exact method. Both of these approaches rely on careful use of techniques from automatic differentiation. We study the trade-offs between our two proposed methods, and show that they both outperform the baseline heuristic procedure for optimizing injective flow models. Finally, equipped with the ability to evaluate injective likelihoods exactly, we discover that injective flow models in general are more proficient at out-of-distribution detection than their bijective counterparts, thus suggesting that directly encoding low-dimensional structure in high-dimensional data may be the correct modelling assumption.

1.4 Summary and Outline

To recap, this thesis studies normalizing flow models in variational inference and density estimation, with a particular interest in modifying the requirement of bijectivity in the data space. This thesis is in the integrated format, meaning that each of the body chapters are self-contained works, with the final chapter providing a conclusion and noting some directions for future work. The references for the body chapters, in order, are as follows:

- Anthony L Caterini, Arnaud Doucet, and Dino Sejdinovic. Hamiltonian Variational Auto-Encoder. *Advances in Neural Information Processing Systems*, 32, 2018.
- Rob Cornish, Anthony L Caterini, George Deligiannidis, and Arnaud Doucet. Relaxing Bijectivity Constraints with Continuously Indexed Normalising Flows. In *International Conference on Machine Learning*, pages 2133-2143. PMLR, 2020.
- Anthony L Caterini, Rob Cornish, Dino Sejdinovic, and Arnaud Doucet. Variational Inference with Continuously-Indexed Normalizing Flows. In *37th Conference on Uncertainty in Artificial Intelligence (UAI)*. Association for Uncertainty in Artificial Intelligence (AUAI), 2021.
- Anthony L Caterini, Gabriel Loaiza-Ganem, Geoff Pleiss, and John P Cunningham. Rectangular Flows for Manifold Learning. *arXiv preprint arXiv:2106.01413*, 2021.

2

Hamiltonian Variational Auto-Encoder

Hamiltonian Variational Auto-Encoder

Anthony L. Caterini¹, Arnaud Doucet^{1,2}, Dino Sejdinovic^{1,2}

¹Department of Statistics, University of Oxford

²Alan Turing Institute for Data Science

{anthony.caterini, doucet, dino.sejdinovic}@stats.ox.ac.uk

Abstract

Variational Auto-Encoders (VAEs) have become very popular techniques to perform inference and learning in latent variable models: they allow us to leverage the rich representational power of neural networks to obtain flexible approximations of the posterior of latent variables as well as tight evidence lower bounds (ELBOs). Combined with stochastic variational inference, this provides a methodology scaling to large datasets. However, for this methodology to be practically efficient, it is necessary to obtain low-variance unbiased estimators of the ELBO and its gradients with respect to the parameters of interest. While the use of Markov chain Monte Carlo (MCMC) techniques such as Hamiltonian Monte Carlo (HMC) has been previously suggested to achieve this [28, 31], the proposed methods require specifying reverse kernels which have a large impact on performance. Additionally, the resulting unbiased estimator of the ELBO for most MCMC kernels is typically not amenable to the reparameterization trick. We show here how to optimally select reverse kernels in this setting and, by building upon Hamiltonian Importance Sampling (HIS) [21], we obtain a scheme that provides low-variance unbiased estimators of the ELBO and its gradients using the reparameterization trick. This allows us to develop a Hamiltonian Variational Auto-Encoder (HVAE). This method can be re-interpreted as a target-informed normalizing flow [25] which, within our context, only requires a few evaluations of the gradient of the sampled likelihood and trivial Jacobian calculations at each iteration.

1 Introduction

Variational Auto-Encoders (VAEs), introduced by Kingma and Welling [16] and Rezende et al. [26], are popular techniques to carry out inference and learning in complex latent variable models. However, the standard mean-field parametrization of the approximate posterior distribution can limit its flexibility. Recent work has sought to *augment* the VAE approach by sampling from the VAE posterior approximation and transforming these samples through mappings with additional trainable parameters to achieve richer posterior approximations. The most popular application of this idea is the Normalizing Flows (NFs) approach [25] in which the samples are deterministically evolved through a series of parameterized invertible transformations called a *flow*. NFs have demonstrated success in various domains [3, 17], but the flows do not explicitly use information about the target posterior. Therefore, it is unclear whether the improved performance is caused by an accurate posterior approximation or simply a result of overparametrization. The related Hamiltonian Variational Inference (HVI) [28] instead stochastically evolves the base samples according to Hamiltonian Monte Carlo (HMC) [22] and thus uses target information, but relies on defining reverse dynamics in the flow, which, as we will see, turns out to be unnecessary and suboptimal.

One of the key components in the formulation of VAEs is the maximization of the evidence lower bound (ELBO). The main idea put forward in Salimans et al. [28] is that it is possible to use K MCMC iterations to obtain an unbiased estimator of the ELBO and its gradients. This estimator

is obtained using an importance sampling argument on an augmented space, with the importance distribution being the joint distribution of the $K + 1$ states of the ‘forward’ Markov chain, while the augmented target distribution is constructed using a sequence of ‘reverse’ Markov kernels such that it admits the original posterior distribution as a marginal. The performance of this estimator is strongly dependent on the selection of these forward and reverse kernels, but no clear guideline for selection has been provided therein. By linking this approach to earlier work by Del Moral et al. [7], we show how to select these components. We focus, in particular, on the use of time-inhomogeneous Hamiltonian dynamics, proposed originally in Neal [21]. This method uses reverse kernels which are optimal for reducing variance of the likelihood estimators and allows for simple calculation of the approximate posteriors of the latent variables. Additionally, we can easily use the reparameterization trick to calculate unbiased gradients of the ELBO with respect to the parameters of interest. The resulting method, which we refer to as the Hamiltonian Variational Auto-Encoder (HVAE), can be thought of as a Normalizing Flow scheme in which the flow depends explicitly on the target distribution. This combines the best properties of HVI and NFs, resulting in target-informed and inhomogeneous deterministic Hamiltonian dynamics, while being scalable to large datasets and high dimensions.

2 Evidence Lower Bounds, MCMC and Hamiltonian Importance Sampling

In this section, we develop our approach to building a *target-informed* normalizing flow via time-inhomogeneous Hamiltonian dynamics to parametrize a family of approximate posteriors for variational inference. Our construction admits lower-variance estimators of the ELBO than the alternative approach of using full-fledged Hamiltonian Monte Carlo (HMC) [21] as the variational family. Also, in contrast to the approach of directly using HMC to sample from the true posterior, we inherit the speed benefits of amortized inference – particularly in the case where the forward model also has trainable parameters – although at the cost of introducing some bias via the variational objective.

2.1 Unbiased likelihood and evidence lower bound estimators

For data $x \in \mathcal{X} \subseteq \mathbb{R}^d$ and parameter $\theta \in \Theta$, consider the likelihood function

$$p_\theta(x) = \int p_\theta(x, z) dz = \int p_\theta(x|z) p_\theta(z) dz,$$

where $z \in \mathcal{Z}$ are some latent variables. If we assume we have access to a strictly positive unbiased estimate of $p_\theta(x)$, denoted $\hat{p}_\theta(x|u)$, then

$$\int \hat{p}_\theta(x|u) q_{\theta, \phi}(u|x) du = p_\theta(x), \quad (1)$$

with $u \sim q_{\theta, \phi}(\cdot)$, $u \in \mathcal{U}$ denoting all the random variables used to compute $\hat{p}_\theta(x|u)$. Here, ϕ denotes additional parameters of the sampling distribution. We emphasize that $\hat{p}_\theta(x|u)$ depends on ϕ , and we will also notationally suppress the u dependence throughout the rest of this section. By applying Jensen’s inequality to (1), we thus obtain, for all θ and ϕ ,

$$\mathcal{L}_{\text{ELBO}}(\theta, \phi; x) := \int \log \hat{p}_\theta(x) q_{\theta, \phi}(u|x) du \leq \log p_\theta(x) =: \mathcal{L}(\theta; x). \quad (2)$$

It can be shown that $|\mathcal{L}_{\text{ELBO}}(\theta, \phi; x) - \mathcal{L}(\theta; x)|$ decreases as the variance of $\hat{p}_\theta(x)$ decreases; see, e.g., [4, 19]. The standard variational framework corresponds to $\mathcal{U} = \mathcal{Z}$ and $\hat{p}_\theta(x) = p_\theta(x, z)/q_{\theta, \phi}(z|x)$, while the Importance Weighted Auto-Encoder (IWAE) [4] with L importance samples corresponds to $\mathcal{U} = \mathcal{Z}^L$, $q_{\theta, \phi}(u|x) = \prod_{i=1}^L q_{\theta, \phi}(z_i|x)$ and $\hat{p}_\theta(x) = \frac{1}{L} \sum_{i=1}^L p_\theta(x, z_i)/q_{\theta, \phi}(z_i|x)$.

In the general case, we do not have an analytical expression for $\mathcal{L}_{\text{ELBO}}(\theta, \phi; x)$. When performing stochastic gradient ascent for variational inference, however, we only require an unbiased estimator of $\nabla_\theta \mathcal{L}_{\text{ELBO}}(\theta, \phi; x)$. This is given by $\nabla_\theta \log \hat{p}_\theta(x)$ if the reparameterization trick [9, 16] is used, i.e. $q_{\theta, \phi}(u|x) = q(u)$, and $\hat{p}_\theta(x)$ is a ‘smooth enough’ function of u . As a guiding principle, one should attempt to obtain a low-variance estimator of $p_\theta(x)$, which typically translates into a low-variance estimator of $\nabla_\theta \mathcal{L}_{\text{ELBO}}(\theta, \phi; x)$. We can analogously optimize $\mathcal{L}_{\text{ELBO}}(\theta, \phi; x)$ with respect to ϕ through stochastic gradient ascent to obtain tighter bounds.

2.2 Unbiased likelihood estimator using time-inhomogeneous MCMC

Salimans et al. [28] propose to build an unbiased estimator of $p_\theta(x)$ by sampling a (potentially time-inhomogeneous) ‘forward’ Markov chain of length $K + 1$ using $z_0 \sim q_{\theta,\phi}^0(\cdot)$ and $z_k \sim q_{\theta,\phi}^k(\cdot|z_{k-1})$ for $k = 1, \dots, K$. Using artificial ‘reverse’ Markov transition kernels $r_{\theta,\phi}^k(z_k|z_{k+1})$ for $k = 0, \dots, K - 1$, it follows easily from an importance sampling argument that

$$\hat{p}_\theta(x) = \frac{p_\theta(x, z_K) \prod_{k=0}^{K-1} r_{\theta,\phi}^k(z_k|z_{k+1})}{q_{\theta,\phi}^0(z_0) \prod_{k=1}^K q_{\theta,\phi}^k(z_k|z_{k-1})} \quad (3)$$

is an unbiased estimator of $p_\theta(x)$ as long as the ratio in (3) is well-defined. In the framework of the previous section, we have $\mathcal{U} = \mathcal{Z}^{K+1}$ and $q_{\theta,\phi}(u|x)$ is given by the denominator of (3). Note that *any* distribution admitting a pointwise-evaluable Lebesgue density may serve as a reverse kernel in this setting. Although we did not use measure-theoretic notation, the *forward* kernels $q_{\theta,\phi}^k$ are typically MCMC kernels which do not admit a density with respect to the Lebesgue measure (e.g. the Metropolis–Hastings kernel). This makes it difficult to define reverse kernels for which (3) is well-defined as evidenced in Salimans et al. [28, Section 4] or Wolf et al. [31]. The estimator (3) was originally introduced in Del Moral et al. [7] where generic recommendations are provided for this estimator to admit a low relative variance: select $q_{\theta,\phi}^k$ as MCMC kernels which are invariant, or approximately invariant as in [10], with respect to $p_\theta^k(x, z_k)$, where $p_{\theta,\phi}^k(z|x) \propto [q_{\theta,\phi}^0(z)]^{1-\beta_k} [p_\theta(x, z)]^{\beta_k}$ is a sequence of artificial densities bridging $q_{\theta,\phi}^0(z)$ to $p_\theta(z|x)$ smoothly using $\beta_0 = 0 < \beta_1 < \dots < \beta_{K-1} < \beta_K = 1$. It is also established in Del Moral et al. [7] that, given any sequence of kernels $\{q_{\theta,\phi}^k\}_k$, the sequence of reverse kernels minimizing the variance of $\hat{p}_\theta(x)$ is given by $r_{\theta,\phi}^{k,\text{opt}}(z_k|z_{k+1}) = q_{\theta,\phi}^k(z_k)q_{\theta,\phi}^{k+1}(z_{k+1}|z_k)/q_{\theta,\phi}^{k+1}(z_{k+1})$, where $q_{\theta,\phi}^k(z_k)$ denotes the marginal density of z_k under the forward dynamics, yielding

$$\hat{p}_\theta(x) = \frac{p_\theta(x, z_K)}{q_{\theta,\phi}^K(z_K)}. \quad (4)$$

For stochastic forward transitions, it is typically not possible to compute $r_{\theta,\phi}^{k,\text{opt}}$ and the corresponding estimator (4) as the marginal densities $q_{\theta,\phi}^k(z_k)$ do not admit closed-form expressions. However this suggests that $r_{\theta,\phi}^k$ should be approximating $r_{\theta,\phi}^{k,\text{opt}}$ and various schemes are presented in [7]. As noticed by Del Moral et al. [7] and Salimans et al. [28], Annealed Importance Sampling (AIS) [20] – also known as the Jarzynski-Crooks identity ([5, 13]) in physics – is a special case of (3) using, for $q_{\theta,\phi}^k$, a $p_\theta^k(z|x)$ -invariant MCMC kernel and the reversal of this kernel as the reverse transition kernel $r_{\theta,\phi}^{k-1,1}$. This choice of reverse kernels is suboptimal but leads to a simple expression for estimator (3). AIS provides state-of-the-art estimators of the marginal likelihood and has been widely used in machine learning. Unfortunately, it typically cannot be used in conjunction with the reparameterization trick. Indeed, although it is very often possible to reparameterize the forward simulation of $(z_1, \dots, z_T)$ in terms of the deterministic transformation of some random variables $u \sim q$ independent of θ and ϕ , this mapping is not continuous because the MCMC kernels it uses typically include singular components. In this context, although (1) holds, $\nabla_\theta \log \hat{p}_\theta(x)$ is *not an unbiased estimator of* $\nabla_\theta \mathcal{L}_{\text{ELBO}}(\theta, \phi; x)$; see, e.g., Glasserman [9] for a careful discussion of these issues. Note that without the reparameterization trick, we must rely on REINFORCE-like gradient estimators to train the model, which are typically high-variance [18].

2.3 Using Hamiltonian dynamics

Given the empirical success of Hamiltonian Monte Carlo (HMC) [12, 22], various contributions have proposed to develop algorithms exploiting Hamiltonian dynamics to obtain unbiased estimates of the ELBO and its gradients when $\mathcal{Z} = \mathbb{R}^\ell$. This was proposed in Salimans et al. [28]. However, the algorithm suggested therein relies on a time-homogeneous leapfrog where momentum resampling is

¹The reversal of a μ -invariant kernel $K(z'|z)$ is given by $K_{\text{rev}}(z'|z) = \frac{\mu(z')K(z|z')}{\mu(z)}$. If K is μ -reversible then $K_{\text{rev}} = K$.

performed at each step and no Metropolis correction is used. It also relies on learned reverse kernels. To address the limitations of Salimans et al. [28], Wolf et al. [31] have proposed to include some Metropolis acceptance steps, but they still limit themselves to homogeneous dynamics and their estimator is not amenable to the reparameterization trick. Finally, in Hoffman [11], an alternative approach is used where the gradient of the true likelihood, $\nabla_{\theta} \mathcal{L}(\theta; x)$, is directly approximated by using Fisher’s identity and HMC to obtain approximate samples from $p_{\theta}(z|x)$. However, the MCMC bias can be very significant when one has multimodal latent posteriors and is strongly dependent on both the initial distribution and θ .

Here, we follow an alternative approach where we use Hamiltonian dynamics that are time-inhomogeneous as in [7] and [20], and use optimal reverse Markov kernels to compute $\hat{p}_{\theta}(x)$. This estimator can be used in conjunction with the reparameterization trick to obtain an unbiased estimator of $\nabla \mathcal{L}_{\text{ELBO}}(\theta, \phi; x)$. This method is based on the Hamiltonian Importance Sampling (HIS) scheme proposed in Neal [21]; one can also find several instances of related ideas in physics [14, 29]. We work in an extended space $(z, \rho) \in \mathcal{U} := \mathbb{R}^{\ell} \times \mathbb{R}^{\ell}$, introducing *momentum* variables ρ to pair with the *position* variables z , with new target

$$\bar{p}_{\theta}(x, z, \rho) := p_{\theta}(x, z) \mathcal{N}(\rho|0, I_{\ell}). \quad (5)$$

Essentially, the idea is to sample using *deterministic* transitions $q_{\theta, \phi}^k((z_k, \rho_k)|(z_{k-1}, \rho_{k-1})) = \delta_{\Phi_{\theta, \phi}^k(z_{k-1}, \rho_{k-1})(z_k, \rho_k)}$ so that $(z_K, \rho_K) = \mathcal{H}_{\theta, \phi}(z_0, \rho_0) := (\Phi_{\theta, \phi}^K \circ \dots \circ \Phi_{\theta, \phi}^1)(z_0, \rho_0)$, where $(z_0, \rho_0) \sim q_{\theta, \phi}^0(\cdot, \cdot)$ and $(\Phi_{\theta, \phi}^k)_{k \geq 1}$, define diffeomorphisms corresponding to a time-discretized and inhomogeneous Hamiltonian dynamics. In this case, it is easy to show that

$$q_{\theta, \phi}^K(z_K, \rho_K) = q_{\theta, \phi}^0(z_0, \rho_0) \prod_{k=1}^K |\det \nabla \Phi_{\theta, \phi}^k(z_k, \rho_k)|^{-1} \quad \text{and} \quad \hat{p}_{\theta}(x) = \frac{\bar{p}_{\theta}(x, z_K, \rho_K)}{q_{\theta, \phi}^K(z_K, \rho_K)}. \quad (6)$$

It can also be shown that this is nothing but a special case of (3) (on the extended position-momentum space) using the optimal reverse kernels² $r_{\theta, \phi}^{k, \text{opt}}$. This setup is similar to the one of Normalizing Flows [25], except here we use a flow informed by the target distribution. Salimans et al. [28] is in fact mentioned in Rezende and Mohamed [25], but the flow therein is homogeneous and yields a high-variance estimator of the normalizing constants even if $r_{\theta}^{k, \text{opt}}$ is used, as demonstrated in our simulations in section 4.

Under these dynamics, the estimator $\hat{p}_{\theta}(x)$ defined in (6) can be rewritten as

$$\hat{p}_{\theta}(x) = \frac{\bar{p}_{\theta}(x, \mathcal{H}_{\theta, \phi}(z_0, \rho_0))}{q_{\theta, \phi}^0(z_0, \rho_0)} \prod_{k=1}^K |\det \nabla \Phi_{\theta, \phi}^k(z_k, \rho_k)|. \quad (7)$$

Hence, if we can simulate $(z_0, \rho_0) \sim q_{\theta, \phi}^0(\cdot, \cdot)$ using $(z_0, \rho_0) = \Psi_{\theta, \phi}(u)$, where $u \sim q$ and $\Psi_{\theta, \phi}$ is a smooth mapping, then we can use the reparameterization trick since $\Phi_{\theta, \phi}^k$ are also smooth mappings.

In our case, the deterministic transformation $\Phi_{\theta, \phi}^k$ has two components: a leapfrog step, which discretizes the Hamiltonian dynamics, and a tempering step, which adds inhomogeneity to the dynamics and allows us to explore isolated modes of the target [21]. To describe the leapfrog step, we first define the potential energy of the system as $U_{\theta}(z|x) \equiv -\log p_{\theta}(x, z)$ for a single datapoint $x \in \mathcal{X}$. Leapfrog then takes the system from (z, ρ) into (z', ρ') via the following transformations:

$$\tilde{\rho} = \rho - \frac{\varepsilon}{2} \odot \nabla U_{\theta}(z|x), \quad (8)$$

$$z' = z + \varepsilon \odot \tilde{\rho}, \quad (9)$$

$$\rho' = \tilde{\rho} - \frac{\varepsilon}{2} \odot \nabla U_{\theta}(z'|x), \quad (10)$$

where $\varepsilon \in (\mathbb{R}^+)^{\ell}$ are the individual leapfrog step sizes per dimension, $\odot$ denotes elementwise multiplication, and the gradient of $U_{\theta}(z|x)$ is taken with respect to z . The composition of equations (8) - (10) has unit Jacobian since each equation describes a shear transformation. For the tempering

²Since this is a deterministic flow, the density can be evaluated directly. However, direct evaluation corresponds to optimal reverse kernels in the deterministic case.

portion, we multiply the momentum output of each leapfrog step by $\alpha_k \in (0, 1)$ for $k \in [K]$ where $[K] \equiv \{1, \dots, K\}$. We consider two methods for setting the values α_k . First, *fixed tempering* involves allowing an inverse temperature $\beta_0 \in (0, 1)$ to vary, and then setting $\alpha_k = \sqrt{\beta_{k-1}/\beta_k}$, where each β_k is a deterministic function of β_0 and $0 < \beta_0 < \beta_1 < \dots < \beta_K = 1$. In the second method, known as *free tempering*, we allow each of the α_k values to be learned, and then set the initial inverse temperature to $\beta_0 = \prod_{k=1}^K \alpha_k^2$. For both methods, the tempering operation has Jacobian α_k^ℓ . We obtain $\Phi_{\theta, \phi}^k$ by composing the leapfrog integrator with the cooling operation, which implies that the Jacobian is given by $|\det \nabla \Phi_{\theta, \phi}^k(z_k, \rho_k)| = \alpha_k^\ell = (\beta_{k-1}/\beta_k)^{\ell/2}$, which in turns implies

$$\prod_{k=1}^K |\det \nabla \Phi_{\theta, \phi}^k(z_k, \rho_k)| = \prod_{k=1}^K \left(\frac{\beta_{k-1}}{\beta_k} \right)^{\ell/2} = \beta_0^{\ell/2}.$$

The only remaining component to specify is the initial distribution. We will set $q_{\theta, \phi}^0(z_0, \rho_0) = q_{\theta, \phi}^0(z_0) \cdot \mathcal{N}(\rho_0|0, \beta_0^{-1}I_\ell)$, where $q_{\theta, \phi}^0(z_0)$ will be referred to as the *variational prior* over the latent variables and $\mathcal{N}(\rho_0|0, \beta_0^{-1}I_\ell)$ is the *canonical momentum distribution* at inverse temperature β_0 . The full procedure to generate an unbiased estimate of the ELBO from (2) on the extended space $\mathcal{U}$ for a single point $x \in \mathcal{X}$ and fixed tempering is given in Algorithm 1. The set of variational parameters to optimize contains the flow parameters β_0 and ε , along with additional parameters of the variational prior.³ We can see from (7) that we will obtain unbiased gradients with respect to θ and ϕ from our estimate of the ELBO if we write $(z_0, \rho_0) = (z_0, \gamma_0/\sqrt{\beta_0})$, for $z_0 \sim q_{\theta, \phi}^0(\cdot)$ and $\gamma_0 \sim \mathcal{N}(\cdot|0, I_\ell) \equiv \mathcal{N}_\ell(\cdot)$, provided we are not also optimizing with respect to parameters of the variational prior. We will require additional reparameterization when we elect to optimize with respect to the parameters of the variational prior, but this is generally quite easy to implement on a problem-specific basis and is well-known in the literature; see, e.g. [16, 25, 26] and section 4.

Algorithm 1 Hamiltonian ELBO, Fixed Tempering

Require: $p_\theta(x, \cdot)$ is the unnormalized posterior for $x \in \mathcal{X}$ and $\theta \in \Theta$

Require: $q_{\theta, \phi}^0(\cdot)$ is the variational prior on $\mathbb{R}^\ell$

```

function HIS( $x, \theta, K, \beta_0, \varepsilon$ )
  Sample  $z_0 \sim q_{\theta, \phi}^0(\cdot), \gamma_0 \sim \mathcal{N}_\ell(\cdot)$ 
   $\rho_0 \leftarrow \gamma_0/\sqrt{\beta_0}$   $\triangleright \rho_0 \sim \mathcal{N}(\cdot|0, \beta_0^{-1}I_\ell)$ 
  for  $k \leftarrow 1$  to  $K$  do  $\triangleright$  Run  $K$  steps of alternating leapfrog and tempering
     $\tilde{\rho} \leftarrow \rho - \varepsilon/2 \odot \nabla U_\theta(z_{k-1}|x)$   $\triangleright$  Start of leapfrog; Equation (8)
     $z_k \leftarrow z_{k-1} + \varepsilon \odot \tilde{\rho}$   $\triangleright$  Equation (9)
     $\rho' \leftarrow \tilde{\rho} - \varepsilon/2 \odot \nabla U_\theta(z_k|x)$   $\triangleright$  Equation (10)
     $\sqrt{\beta_k} \leftarrow \left( \left(1 - \frac{1}{\sqrt{\beta_0}}\right) \cdot k^2/K^2 + \frac{1}{\sqrt{\beta_0}} \right)^{-1}$   $\triangleright$  Quadratic tempering scheme
     $\rho_k \leftarrow \sqrt{\beta_{k-1}/\beta_k} \cdot \rho'$ 
   $\bar{p} \leftarrow p_\theta(x, z_K) \mathcal{N}(\rho_K|0, I_\ell)$ 
   $\bar{q} \leftarrow q_{\theta, \phi}^0(z_0) \mathcal{N}(\rho_0|0, \beta_0^{-1}I_\ell) \beta_0^{-\ell/2}$   $\triangleright$  Equation (6), left side
   $\hat{\mathcal{L}}_{\text{ELBO}}^H(\theta, \phi; x) \leftarrow \log \bar{p} - \log \bar{q}$   $\triangleright$  Take the log of equation (6), right side
  return  $\hat{\mathcal{L}}_{\text{ELBO}}^H(\theta, \phi; x)$   $\triangleright$  Can take unbiased gradients of this estimate wrt  $\theta, \phi$ 

```

Sampling Once the model is trained, we may return z_K Algorithm 1 as a sample from our variational model. Each sample requires K steps of HIS to produce, although K is often not very large (see experiments), and practically the computation can easily be parallelized over modern GPU hardware. We thus expect the cost of training the model plus the sample generation procedure to be quite a bit less than sampling directly via HMC, as we would require reasonably well-converged chains at each θ update during training in this case. Further, in the case of amortized inference, we can generate new samples given new input data considerably faster than the naïve HMC approach, which requires building a new Markov chain for each new datapoint.

³We avoid reference to a mass matrix M throughout this formulation because we can capture the same effect by optimizing individual leapfrog step sizes per dimension as pointed out in [22, Section 4.2].

Auxiliary inference viewpoint It is also important to note that our method can be viewed as a form of auxiliary variational inference [2, 24]. From this perspective, our method introduces exactly *one* backward model: the standard normal distribution for the final momentum variables. We control the extra variance introduced by this model by initializing the momentum variables at a higher-temperature normal distribution $\mathcal{N}(0, \beta_0^{-1}I)$ and then performing tempering at each step of HIS so that the momentum particles end at a temperature of 1. We also provide an approach which learns the tempering scheme automatically, which may be seen as implicitly training the final momentum distribution so that it matches our prescribed form via ELBO maximization. We leave it as future work to consider parametrizing an explicitly trainable distribution for the final momentum particles.

3 Stochastic Variational Inference

We will now describe how to use Algorithm 1 within a stochastic variational inference procedure, moving to the setting where we have a dataset $\mathcal{D} = \{x_1, \dots, x_N\}$ and $x_i \in \mathcal{X}$ for all $i \in [N]$. In this case, we are interested in finding

$$\theta^* \in \operatorname{argmax}_{\theta \in \Theta} \mathbf{E}_{x \sim \nu_{\mathcal{D}}(\cdot)} [\mathcal{L}(\theta; x)], \quad (11)$$

where $\nu_{\mathcal{D}}(\cdot) \equiv \frac{1}{N} \sum_{i=1}^N \delta_{x_i}(\cdot)$ is the empirical measure of the data. We must resort to variational methods since $\mathcal{L}(\theta; x)$ cannot generally be calculated exactly and instead maximize the surrogate ELBO objective function

$$\mathcal{L}_{\text{ELBO}}(\theta, \phi) \equiv \mathbf{E}_{x \sim \nu_{\mathcal{D}}(\cdot)} [\mathcal{L}_{\text{ELBO}}(\theta, \phi; x)] \quad (12)$$

for $\mathcal{L}_{\text{ELBO}}(\theta, \phi; x)$ defined as in (2). We can now turn to stochastic gradient ascent (or a variant thereof) to jointly maximize (12) with respect to θ and ϕ by approximating the expectation over $\nu_{\mathcal{D}}(\cdot)$ using *minibatches* of observed data.

For our specific problem, we can reduce the variance of the ELBO calculation by analytically evaluating some terms in the expectation (i.e. Rao-Blackwellization) as follows:

$$\begin{aligned} \mathcal{L}_{\text{ELBO}}^H(\theta, \phi; x) &= \mathbf{E}_{(z_0, \rho_0) \sim q_{\theta, \phi}^0(\cdot, \cdot)} \left[\log \left(\frac{\bar{p}_{\theta}(x, z_K, \rho_K) \beta_0^{\ell/2}}{q_{\theta, \phi}^0(z_0, \rho_0)} \right) \right] \\ &= \mathbf{E}_{z_0 \sim q_{\theta, \phi}^0(\cdot), \gamma_0 \sim \mathcal{N}_{\ell}(\cdot)} \left[\log p_{\theta}(x, z_K) - \frac{1}{2} \rho_K^T \rho_K - \log q_{\theta, \phi}^0(z_0) \right] + \frac{\ell}{2}, \end{aligned} \quad (13)$$

where we write $(z_K, \rho_K) = \mathcal{H}_{\theta, \phi}(z_0, \gamma_0 / \sqrt{\beta_0})$ under reparameterization. We can now consider the output of Algorithm 1 as taking a sample from the inner expectation for a given sample x from the outer expectation. Algorithm 2 provides a full procedure to stochastically optimize (13). In practice, we take the gradients of (13) using automatic differentiation packages. This is achieved by using TensorFlow [1] in our implementation.

4 Experiments

In this section, we discuss the experiments used to validate our method. We first test HVAE on an example with a tractable full log likelihood (where no neural networks are needed), and then perform larger-scale tests on the MNIST dataset. Code is available online.⁴ All models were trained using TensorFlow [1].

4.1 Gaussian Model

The generative model that we will consider first is a Gaussian likelihood with an offset and a Gaussian prior on the mean, given by

$$\begin{aligned} z &\sim \mathcal{N}(0, I_{\ell}), \\ x_i | z &\sim \mathcal{N}(z + \Delta, \Sigma) \quad \text{independently,} \quad i \in [N] \end{aligned}$$

⁴<https://github.com/anthonycaterini/hvae-nips>

Algorithm 2 Hamiltonian Variational Auto-Encoder

Require: $p_\theta(x, \cdot)$ is the unnormalized posterior for $x \in \mathcal{X}$ and $\theta \in \Theta$

```
function HVAE( $\mathcal{D}, K, n_B$ ) ▷  $n_B$  is minibatch size
  Initialize  $\theta, \phi$ 
  while  $\theta, \phi$  not converged do ▷ Stochastic optimization loop
    Sample  $\{x_1, \dots, x_{n_B}\} \sim \nu_{\mathcal{D}}(\cdot)$  independently
     $\hat{\mathcal{L}}_{\text{ELBO}}^H(\theta, \phi) \leftarrow 0$  ▷ Average ELBO estimators over mini-batch
    for  $i \leftarrow 1$  to  $n_B$  do
       $\hat{\mathcal{L}}_{\text{ELBO}}^H(\theta, \phi) \leftarrow \text{HIS}(x_i, \theta, K, \beta_0, \varepsilon) + \hat{\mathcal{L}}_{\text{ELBO}}^H(\theta, \phi)$ 
     $\hat{\mathcal{L}}_{\text{ELBO}}^H(\theta, \phi) \leftarrow \hat{\mathcal{L}}_{\text{ELBO}}^H(\theta, \phi)/n_B$ 
    ▷ Optimize the ELBO using gradient-based techniques such as RMSProp, ADAM, etc.
     $\theta \leftarrow \text{UPDATE\_THETA}(\nabla_\theta \hat{\mathcal{L}}_{\text{ELBO}}^H(\theta, \phi), \theta)$ 
     $\phi \leftarrow \text{UPDATE\_PHI}(\nabla_\phi \hat{\mathcal{L}}_{\text{ELBO}}^H(\theta, \phi), \phi)$ 
  return  $\theta, \phi$ 
```

where Σ is constrained to be diagonal. We will again write $\mathcal{D} \equiv \{x_1, \dots, x_N\}$ to denote an observed dataset under this model, where each $x_i \in \mathcal{X} \subseteq \mathbb{R}^d$. In this example, we have $\ell = d$. The goal of the problem is to learn the model parameters $\theta \equiv \{\Sigma, \Delta\}$, where $\Sigma = \text{diag}(\sigma_1^2, \dots, \sigma_d^2)$ and $\Delta \in \mathbb{R}^d$.

Here, we have only one latent variable generating the entire set of data. Thus, our variational lower bound is now given by

$$\mathcal{L}_{\text{ELBO}}(\theta, \phi; \mathcal{D}) := \mathbf{E}_{z \sim q_{\theta, \phi}(\cdot | \mathcal{D})} [\log p_\theta(\mathcal{D}, z) - \log q_{\theta, \phi}(z | \mathcal{D})] \leq \log p_\theta(\mathcal{D}),$$

for the variational posterior approximation $q_{\theta, \phi}(\cdot | \mathcal{D})$. We note that this is not exactly the same as the auto-encoder setting, in which an individual latent variable is associated with each observation, however it provides a tractable framework to analyze effectiveness of various variational inference methods. We also note that we can calculate the log-likelihood $\log p_\theta(\mathcal{D})$ exactly in this case, but we use variational methods for the sake of comparison.

From the model, we see that the logarithm of the unnormalized target is given by

$$\log p_\theta(\mathcal{D}, z) = \sum_{i=1}^N \log \mathcal{N}(x_i | z + \Delta, \Sigma) + \log \mathcal{N}(z | 0, I_d).$$

For this example, we will use a HVAE with variational prior equal to the true prior, i.e. $q^0 = \mathcal{N}(0, I_\ell)$, and fixed tempering. The potential, given by $U_\theta(z | \mathcal{D}) = \log p_\theta(\mathcal{D}, z)$, has gradient

$$\nabla U_\theta(z | \mathcal{D}) = z + N\Sigma^{-1}(z + \Delta - x).$$

The set of variational parameters here is $\phi \equiv \{\varepsilon, \beta_0\}$, where $\varepsilon \in \mathbb{R}^d$ contains the per-dimension leapfrog stepsizes and $\beta_0 \in (0, 1)$ is the initial inverse temperature. We constrain each of the leapfrog step sizes such that $\varepsilon_j \in (0, \xi)$ for some $\xi > 0$, for all $j \in [d]$ – this is to prevent the leapfrog discretization from entering unstable regimes. Note that $\phi \in \mathbb{R}^{d+1}$ in this example; in particular, we do not optimize any parameters of the variational prior and thus require no further reparameterization.

We will compare HVAE with a basic Variational Bayes (VB) scheme with mean-field approximate posterior $q_{\phi_V}(z | \mathcal{D}) = \mathcal{N}(z | \mu_Z, \Sigma_Z)$, where Σ_Z is diagonal and $\phi_V \equiv \{\mu_Z, \Sigma_Z\}$ denotes the set of learned variational parameters. We will also include a planar normalizing flow of the form of equation (10) in Rezende and Mohamed [25], but with the same flow parameters across iterations to keep the number of variational parameters of the same order as the other methods. The variational prior here is also set to the true prior as in HVAE above. The log variational posterior $\log q_{\phi_N}(z | \mathcal{D})$ is given by equation (13) of Rezende and Mohamed [25], where $\phi_N \equiv \{\mathbf{u}, \mathbf{v}, b\}^5 \in \mathbb{R}^{2d+1}$.

We set our true offset vector to be $\Delta = (-\frac{d-1}{2}, \dots, \frac{d-1}{2})/5$, and our scale parameters to range quadratically from $\sigma_1 = 1$, reaching a minimum at $\sigma_{(d+1)/2} = 0.1$, and increasing back to $\sigma_d = 1$.⁶

⁵Boldface vectors used to match notation of Rezende and Mohamed [25].

⁶When d is even, $\sigma_{(d+1)/2}$ does not exist, although we could still consider $(d+1)/2$ to be the location of the minimum of the parabola defining the true standard deviations.

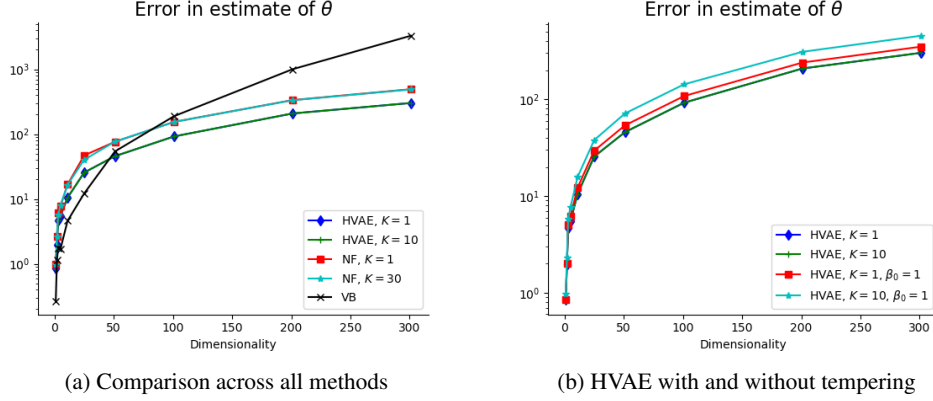


Figure 1: Averages of $\|\theta - \hat{\theta}\|_2^2$ for several variational methods and choices of dimensionality d , where $\hat{\theta}$ is the estimated maximizer of the ELBO for each method and θ is the true parameter.

All experiments have $N = 10,000$ and all training was done using RMSProp [30] with a learning rate of 10^{-3} .

To compare the results across methods, we train each method ten times on different datasets. For each training run, we calculate $\|\theta - \hat{\theta}\|_2^2$, where $\hat{\theta}$ is the estimated value of θ given by the variational method on a particular run, and plot the average of this across the 10 runs for various dimensions in Figure 1a. We note that, as the dimension increases, HVAE performs best in parameter estimation. The VB method suffers most on prediction of Δ as the dimension increases, whereas the NF method does poorly on predicting Σ .

We also compare HVAE with tempering to HVAE without tempering, i.e. where β_0 is fixed to 1 in training. This has the effect of making our Hamiltonian dynamics homogeneous in time. We perform the same comparison as above and present the results in Figure 1b. We can see that the tempered methods perform better than their non-tempered counterparts; this shows that time-inhomogeneous dynamics are a key ingredient in the effectiveness of the method.

4.2 Generative Model for MNIST

The next experiment that we consider is using HVAE to improve upon a convolutional variational auto-encoder (VAE) for the binarized MNIST handwritten digit dataset. Again, our training data is $\mathcal{D} = \{x_1, \dots, x_N\}$, where each $x_i \in \mathcal{X} \subseteq \{0, 1\}^d$ for $d = 28 \times 28 = 784$. The generative model is as follows:

$$z_i \sim \mathcal{N}(0, I_\ell),$$

$$x_i|z_i \sim \prod_{j=1}^d \text{Bernoulli}((x_i)_j | \pi_\theta(z_i)_j),$$

for $i \in [N]$, where $(x_i)_j$ is the j^{th} component of x_i , $z_i \in \mathcal{Z} \equiv \mathbb{R}^\ell$ is the latent variable associated with x_i , and $\pi_\theta : \mathcal{Z} \rightarrow \mathcal{X}$ is a convolutional neural network (i.e. the *generative network*, or *encoder*) parametrized by the model parameters θ . This is the standard generative model used in VAEs in which each pixel in the image x_i is conditionally independent given the latent variable. The VAE approximate posterior – and the HVAE *variational prior* across the latent variables in this case – is given by $q_{\theta, \phi}(z_i|x_i) = \mathcal{N}(z_i|\mu_\phi(x_i), \Sigma_\phi(x_i))$, where μ_ϕ and Σ_ϕ are separate outputs of the same neural network (the *inference network*, or *encoder*) parametrized by ϕ , and Σ_ϕ is constrained to be diagonal.

We attempted to match the network structure of Salimans et al. [28]. The inference network consists of three convolutional layers, each with filters of size 5×5 and a stride of 2. The convolutional layers output 16, 32, and 32 feature maps, respectively. The output of the third layer is fed into a

Table 1: Estimated NLL values for HVAE on MNIST. The base VAE achieves an NLL of 83.20. We see that *free* tempering plus ϵ varied across layers performs the best and allows for selecting larger values for K , and thus defining a more expressive variational family. A more detailed version of this table is included in the supplementary material.

	ϵ fixed across layers			ϵ varied across layers		
	$T = \text{Free}$	$T = \text{Fixed}$	$T = \text{None}$	$T = \text{Free}$	$T = \text{Fixed}$	$T = \text{None}$
$K = 1$	N/A	83.32	83.17	N/A	N/A	N/A
$K = 5$	83.09	83.26	83.68	83.01	82.94	83.35
$K = 10$	82.97	83.26	83.40	82.62	82.87	83.25
$K = 15$	82.78	83.56	83.82	82.62	83.09	82.94
$K = 20$	82.93	83.18	83.33	82.83	82.85	82.93

fully-connected layer with hidden dimension $n_h = 450$, whose output is then fully connected to the output means and standard deviations each of size ℓ . Softplus activation functions are used throughout the network except immediately before the outputted mean. The generative network mirrors this structure in reverse, replacing the stride with upsampling as in Dosovitskiy et al. [8] and replicated in Salimans et al. [28].

We apply HVAE on top of the base convolutional VAE. We evolve samples from the variational prior according to Algorithm 1 and optimize the new objective given in (13). We reparameterize $z_0|x \sim \mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$ as $z_0 = \mu_\phi(x) + \Sigma_\phi^{1/2}(x) \cdot \epsilon$, for $\epsilon \sim \mathcal{N}(0, I_\ell)$ and $x \in \mathcal{X}$, to generate unbiased gradients of the ELBO with respect to ϕ . We select various values for K and set $\ell = 64$. In contrast with normalizing flows, we do not need our flow parameters ϵ and β_0 to be outputs of the inference network because our flow is guided by the target. This allows our method to have fewer overall parameters than normalizing flow schemes. We use the standard stochastic binarization of MNIST [27] as training data, and train using Adamax [15] with learning rate 10^{-3} . We also employ early stopping by halting the training procedure if there is no improvement in the loss on validation data over 100 epochs.

To evaluate HVAE after training is complete, we estimate out-of-sample negative log likelihoods (NLLs) using 1000 importance samples from the HVAE approximate posterior. For each trained model, we estimate NLL three times, noting that the standard deviation over these three estimates is no larger than 0.12 nats. We report the average NLL values over either two or three different initializations (in addition to the three NLL estimates for each trained model) for several choices of tempering and leapfrog steps in Table 1. A full accounting of the tests is given in the supplementary material. We also consider an HVAE scheme in which we allow ϵ to vary across layers of the flow and report the results.

From Table 1, we notice that generally increasing the inhomogeneity in the dynamics improves the test NLL values. For example, free tempering is the most successful tempering scheme, and varying the leapfrog step size ϵ across layers also improves results. We also notice that increasing the number of leapfrog steps does not always improve the performance, as $K = 15$ provides the best results in free tempering schemes. We believe that the improvement in HVAE over the base VAE scheme can be attributed to a more expressive approximate posterior, as we can see that samples from the HVAE approximate posterior exhibit non-negligible covariance across dimensions. As in Salimans et al. [28], we are also able to improve upon the base model by adding our time-inhomogeneous Hamiltonian dynamics on top, but in a simplified regime without referring to learned reverse kernels. Rezende and Mohamed [25] report only lower bounds on the log-likelihood for NFs, which are indeed lower than our log-likelihood estimates, although they use a much larger number of variational parameters.

5 Conclusion and Discussion

We have proposed a principled way to exploit Hamiltonian dynamics within stochastic variational inference. Contrary to previous methods [28, 31], our algorithm does not rely on learned reverse Markov kernels and benefits from the use of tempering ideas. Additionally, we can use the reparam-

eterization trick to obtain unbiased estimators of gradients of the ELBO. The resulting HVAE can be interpreted as a target-driven normalizing flow which requires the evaluation of a few gradients of the log-likelihood associated to a single data point at each stochastic gradient step. However, the Jacobian computations required for the ELBO are trivial. In our experiments, the robustness brought about by the use of target-informed dynamics can reduce the number of parameters that must be trained and improve generalizability.

We note that, although we have fewer parameters to optimize, the memory cost of using HVAE and target-informed dynamics could become prohibitively large if the memory required to store evaluations of $\nabla_z \log p_\theta(x, z)$ is already extremely large. Evaluating these gradients is not a requirement of VAEs or standard normalizing flows. However, we have shown that in the case of a fairly large generative network we are still able to evaluate gradients and backpropagate through the layers of the flow. Further tests explicitly comparing HVAE with VAEs and normalizing flows in various memory regimes are required to determine in what cases one method should be used over the other.

There are numerous possible extensions of this work. Hamiltonian dynamics preserves the Hamiltonian and hence also the corresponding target distribution, but there exist other deterministic dynamics which leave the target distribution invariant but not the Hamiltonian. This includes the Nosé-Hoover thermostat. It is possible to directly use these dynamics instead of the Hamiltonian dynamics within the framework developed in subsection 2.3. In continuous-time, related ideas have appeared in physics [6, 23, 29]. This comes at the cost of more complicated Jacobian calculations. The ideas presented here could also be coupled with the methodology proposed in [10] – we conjecture that this could reduce the variance of the estimator (3) by an order of magnitude.

Acknowledgments

Anthony L. Caterini is a Commonwealth Scholar, funded by the UK government.

References

- [1] Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL <https://www.tensorflow.org/>. Software available from tensorflow.org.
- [2] Felix V Agakov and David Barber. An auxiliary variational method. In *International Conference on Neural Information Processing*, pages 561–566. Springer, 2004.
- [3] Rianne van den Berg, Leonard Hasenclever, Jakub M Tomczak, and Max Welling. Sylvester normalizing flows for variational inference. *arXiv preprint arXiv:1803.05649*, 2018.
- [4] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. In *The 4th International Conference on Learning Representations (ICLR)*, 2016.
- [5] Gavin E Crooks. Nonequilibrium measurements of free energy differences for microscopically reversible Markovian systems. *Journal of Statistical Physics*, 90(5-6):1481–1487, 1998.
- [6] Michel A Cuendet. Statistical mechanical derivation of Jarzynski’s identity for thermostated non-hamiltonian dynamics. *Physical Review Letters*, 96(12):120602, 2006.
- [7] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.
- [8] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 1538–1546. IEEE, 2015.
- [9] Paul Glasserman. *Gradient estimation via perturbation analysis*, volume 116. Springer Science & Business Media, 1991.
- [10] Jeremy Heng, Adrian N Bishop, George Deligiannidis, and Arnaud Doucet. Controlled sequential Monte Carlo. *arXiv preprint arXiv:1708.08396*, 2017.
- [11] Matthew D Hoffman. Learning deep latent Gaussian models with Markov chain Monte Carlo. In *International Conference on Machine Learning*, pages 1510–1519, 2017.

- [12] Matthew D Hoffman and Andrew Gelman. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.
- [13] Christopher Jarzynski. Nonequilibrium equality for free energy differences. *Physical Review Letters*, 78(14):2690, 1997.
- [14] Christopher Jarzynski. Hamiltonian derivation of a detailed fluctuation theorem. *Journal of Statistical Physics*, 98(1-2):77–102, 2000.
- [15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [16] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. In *The 2nd International Conference on Learning Representations (ICLR)*, 2014.
- [17] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751, 2016.
- [18] Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M Blei. Automatic differentiation variational inference. *The Journal of Machine Learning Research*, 18(1):430–474, 2017.
- [19] Chris J Maddison, John Lawson, George Tucker, Nicolas Heess, Mohammad Norouzi, Andriy Mnih, Arnaud Doucet, and Yee Teh. Filtering variational objectives. In *Advances in Neural Information Processing Systems*, pages 6576–6586, 2017.
- [20] Radford M Neal. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, 2001.
- [21] Radford M Neal. Hamiltonian importance sampling. www.cs.toronto.edu/pub/radford/his-talk.ps, 2005. Talk presented at the Banff International Research Station (BIRS) workshop on Mathematical Issues in Molecular Dynamics.
- [22] Radford M Neal et al. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11), 2011.
- [23] Piero Procacci, Simone Marsili, Alessandro Barducci, Giorgio F Signorini, and Riccardo Chelli. Crooks equation for steered molecular dynamics using a Nosé-Hoover thermostat. *The Journal of Chemical Physics*, 125(16):164101, 2006.
- [24] Rajesh Ranganath, Dustin Tran, and David Blei. Hierarchical variational models. In *International Conference on Machine Learning*, pages 324–333. PMLR, 2016.
- [25] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538, 2015.
- [26] Danilo Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286, 2014.
- [27] Ruslan Salakhutdinov and Iain Murray. On the quantitative analysis of deep belief networks. In *Proceedings of the 25th international conference on Machine learning*, pages 872–879. ACM, 2008.
- [28] Tim Salimans, Diederik P Kingma, and Max Welling. Markov chain Monte Carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, pages 1218–1226, 2015.
- [29] E Schöll-Paschinger and Christoph Dellago. A proof of Jarzynski’s nonequilibrium work theorem for dynamical systems that conserve the canonical distribution. *The Journal of Chemical Physics*, 125(5):054105, 2006.

- [30] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2): 26–31, 2012.
- [31] Christopher Wolf, Maximilian Karl, and Patrick van der Smagt. Variational inference with Hamiltonian Monte Carlo. *arXiv preprint arXiv:1609.08203*, 2016.

Supplementary Material

A Full List of Tests on MNIST

Table 2 and Table 3 display the list of test runs of HVAE on MNIST. The number of flow steps is denoted by K . The total number of epochs varies in training because of early stopping. The ELBO and NLL estimates are generated using 1000 importance samples from the HVAE approximate posterior; this procedure is run 3 times for each seed. The average and standard deviation of these estimates over the three runs is displayed. Table 2 refers to HVAE tests in which ε was fixed across flow layers, whereas Table 3 refers to HVAE tests in which ε was allowed to vary across flow layers.

Table 2: List of tests of HVAE for ε fixed across flow layers

K	Tempering	Seed	Total Epochs	ELBO Estimate	NLL Estimate
1	fixed	85547	1158	86.49 (0.05)	83.30 (0.06)
1	fixed	12345	1131	86.53 (0.02)	83.33 (0.02)
1	none	85547	1374	86.43 (0.08)	83.29 (0.07)
1	none	12345	1519	86.21 (0.04)	83.04 (0.04)
5	fixed	85547	998	86.69 (0.01)	83.36 (0.02)
5	fixed	12345	1070	86.40 (0.04)	83.16 (0.04)
5	free	85547	1046	86.45 (0.05)	83.16 (0.04)
5	free	12345	1141	86.31 (0.11)	83.02 (0.10)
5	none	85547	975	86.82 (0.04)	83.68 (0.05)
5	none	12345	959	86.82 (0.02)	83.68 (0.02)
10	fixed	85547	991	86.75 (0.03)	83.50 (0.03)
10	fixed	12345	1340	86.26 (0.05)	83.02 (0.04)
10	free	85547	1404	86.06 (0.05)	82.73 (0.05)
10	free	12345	819	86.65 (0.04)	83.18 (0.03)
10	free	98765	846	86.47 (0.08)	82.99 (0.08)
10	none	85547	1425	86.47 (0.07)	83.24 (0.06)
10	none	12345	1035	86.78 (0.02)	83.56 (0.03)
15	fixed	85547	666	87.11 (0.02)	83.93 (0.02)
15	fixed	12345	1204	86.51 (0.08)	83.19 (0.07)
15	free	85547	1303	86.19 (0.03)	82.75 (0.04)
15	free	12345	1028	86.16 (0.06)	82.81 (0.06)
15	none	85547	785	87.05 (0.06)	83.87 (0.06)
15	none	12345	989	86.95 (0.03)	83.76 (0.03)
20	fixed	85547	1103	86.30 (0.08)	83.18 (0.07)
20	fixed	12345	1126	86.32 (0.09)	83.18 (0.09)
20	free	85547	1028	86.28 (0.03)	82.96 (0.03)
20	free	12345	1131	86.27 (0.02)	82.89 (0.02)
20	none	85547	1318	86.36 (0.06)	83.30 (0.06)
20	none	12345	1299	86.44 (0.05)	83.35 (0.05)

Table 3: List of tests of HVAE for ε varied across flow layers

K	Tempering	Seed	Total Epochs	ELBO Estimate	NLL Estimate
5	fixed	85547	1229	86.28 (0.04)	83.10 (0.04)
5	fixed	12345	1579	85.89 (0.02)	82.77 (0.02)
5	free	85547	1116	86.24 (0.02)	83.11 (0.02)
5	free	12345	1302	86.07 (0.07)	82.91 (0.07)
5	none	85547	1014	86.50 (0.06)	83.38 (0.06)
5	none	12345	959	86.46 (0.01)	83.32 (0.01)
10	fixed	85547	1473	86.01 (0.05)	82.88 (0.05)
10	fixed	12345	1508	86.01 (0.01)	82.85 (0.01)
10	free	85547	1303	85.98 (0.03)	82.69 (0.03)
10	free	12345	1680	85.61 (0.02)	82.37 (0.02)
10	free	98765	846	86.17 (0.08)	82.80 (0.09)
10	none	85547	1035	86.44 (0.08)	83.35 (0.07)
10	none	12345	1276	86.29 (0.12)	83.14 (0.12)
15	fixed	85547	998	86.41 (0.01)	83.20 (0.02)
15	fixed	12345	1070	86.21 (0.02)	82.97 (0.02)
15	free	85547	1303	85.92 (0.03)	82.57 (0.02)
15	free	12345	1131	85.96 (0.03)	82.66 (0.03)
15	none	85547	1425	86.08 (0.06)	82.96 (0.05)
15	none	12345	1355	86.06 (0.01)	82.92 (0.01)
20	fixed	85547	1473	85.85 (0.05)	82.71 (0.06)
20	fixed	12345	1200	86.18 (0.06)	82.98 (0.05)
20	free	85547	1404	86.02 (0.05)	82.75 (0.05)
20	free	12345	1131	86.09 (0.03)	82.90 (0.02)
20	none	85547	1596	85.97 (0.03)	82.83 (0.02)
20	none	12345	1327	86.17 (0.06)	83.03 (0.06)

Table 4 displays the list of test runs of the base VAE on MNIST. ELBO and NLL estimates are again generated by importance sampling, but this time from the learned VAE approximate posterior.

Table 4: List of tests of VAE

Seed	Total Epochs	ELBO Estimate	NLL Estimate
85547	1524	86.41 (0.01)	83.25 (0.01)
98765	1301	86.42 (0.07)	83.19 (0.07)
12345	1381	86.43 (0.07)	83.15 (0.07)

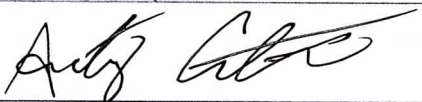
Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor (**only required where there isn't already a statement of contribution within the paper itself**).

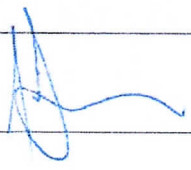
Title of Paper	Hamiltonian Variational Auto-Encoder
Publication Status	☒ Published ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and unsubmitted work written in a manuscript style
Publication Details	Caterini, A. L., Doucet, A., & Sejdinovic, D. (2018). Hamiltonian Variational Auto-Encoder. <i>Advances in Neural Information Processing Systems</i> .

Student Confirmation

Student Name:	Anthony Caterini		
Contribution to the Paper	Developed the details of the idea suggested by my supervisor Arnaud Doucet, including the relevant extended space ELBO. Formulated the variational tempering schedule. Wrote the majority of the manuscript, with Arnaud rewriting parts of Section 2 and my other supervisor Dino Sejdinovic providing general revision. Designed the experiments, wrote all the code, and managed their execution.		
Signature		Date	22 June 2021

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof. Arnaud Doucet			
Supervisor comments 			
Signature		Date	27/06/21

This completed form should be included in the thesis, at the end of the relevant chapter.

3

Relaxing Bijectivity Constraints with Continuously Indexed Normalising Flows

Relaxing Bijectivity Constraints with Continuously Indexed Normalising Flows

Rob Cornish¹ Anthony Caterini¹ George Deligiannidis^{1,2} Arnaud Doucet¹

Abstract

We show that normalising flows become pathological when used to model targets whose supports have complicated topologies. In this scenario, we prove that a flow must become arbitrarily numerically noninvertible in order to approximate the target closely. This result has implications for all flow-based models, and especially *residual flows* (ResFlows), which explicitly control the Lipschitz constant of the bijection used. To address this, we propose *continuously indexed flows* (CIFs), which replace the single bijection used by normalising flows with a continuously indexed family of bijections, and which can intuitively “clean up” mass that would otherwise be misplaced by a single bijection. We show theoretically that CIFs are not subject to the same topological limitations as normalising flows, and obtain better empirical performance on a variety of models and benchmarks.

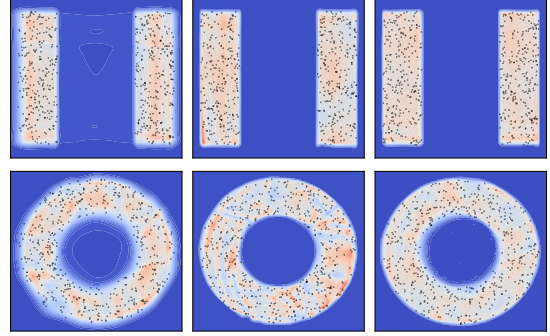


Figure 1: Densities learned by a 10-layer ResFlow (left), 100-layer ResFlow (middle), and 10-layer CIF-ResFlow (right) for two datasets (samples shown in black) that are not homeomorphic to the Gaussian prior. The 10-layer ResFlow visibly leaks mass outside of the support of the target due to its small bi-Lipschitz constant. The larger ResFlow improves on this, but still achieves smaller average log probability than the CIF-ResFlow, as is apparent from the greater homogeneity of the right-hand densities.

1 Introduction

Normalising flows (Rezende & Mohamed, 2015) have become popular methods for density estimation (Dinh et al., 2017; Papamakarios et al., 2017; Kingma & Dhariwal, 2018; Chen et al., 2019). These methods model an unknown target distribution P_X^* on a data space $\mathcal{X} \subseteq \mathbb{R}^d$ as the marginal of X obtained by the generative process

$$Z \sim P_Z, \quad X := f(Z), \quad (1)$$

where P_Z is a *prior* distribution on a space $\mathcal{Z} \subseteq \mathbb{R}^d$, and $f : \mathcal{X} \rightarrow \mathcal{Z}$ is a bijection. The use of a bijection means the density of X can be computed analytically by the change-of-variables formula, and the parameters of f can be learned by maximum likelihood using i.i.d. samples from P_X^* .

To be effective, a normalising flow model must specify an expressive family of bijections with tractable Jacobians.

¹University of Oxford, Oxford, United Kingdom ²The Alan Turing Institute, London, United Kingdom. Correspondence to: Rob Cornish <rcornish@robots.ox.ac.uk>.

Affine coupling layers (Dinh et al., 2015; 2017), autoregressive maps (Germain et al., 2015; Papamakarios et al., 2017), invertible linear transformations (Kingma & Dhariwal, 2018), ODE-based maps (Grathwohl et al., 2019), and invertible ResNet blocks (Behrmann et al., 2019; Chen et al., 2019) are all examples of such bijections that can be composed to produce expressive flows. These models have demonstrated significant promise in their ability to model complex datasets and to synthesise realistic data.

In all these cases, f and f^{-1} are both continuous. It follows that f is a *homeomorphism*, and therefore preserves the topology of its domain (Runde, 2007, Definition 3.3.10). As Dupont et al. (2019) and Dinh et al. (2019) mention, this seems intuitively problematic when P_Z and P_X^* are supported on domains with distinct topologies, which occurs for example when the supports differ in their number of connected components or “holes”, or when they are “knotted” differently. This seems inevitable in practice, as P_Z is usually quite simple (e.g. a Gaussian) while P_X^* is very complicated (e.g. a distribution over images).

As our first contribution, we make precise the consequences

of using a topologically misspecified prior. We confirm that in this case it is indeed impossible to recover the target perfectly if f is a homeomorphism. Moreover, in [Theorem 2.1](#) we prove that, in order to *approximate* such a target arbitrarily well, we must have $\text{BiLip } f \rightarrow \infty$, where $\text{BiLip } f$ denotes the *bi-Lipschitz constant* of f defined as the infimum over $M \in [1, \infty]$ such that

$$M^{-1}\|z - z'\| \leq \|f(z) - f(z')\| \leq M\|z - z'\| \quad (2)$$

for all $z, z' \in \mathcal{Z}$. [Theorem 2.1](#) applies essentially regardless of the training objective, and has implications for the case that P_Z and P_X^* both have full support but are heavily concentrated on regions that are not homeomorphic. Since $\text{BiLip } f$ is a natural measure of the “invertibility” of f ([Behrmann et al., 2020](#)), this result shows that the goal of designing neural networks with well-conditioned inverses is fundamentally at odds with the goal of designing neural networks that can approximate complicated densities.

[Theorem 2.1](#) also has immediate implications for *residual flows* (ResFlows) ([Behrmann et al., 2019](#); [Chen et al., 2019](#)), which have recently achieved state-of-the-art performance on several large-scale density estimation tasks. Unlike models based on triangular maps ([Jaini et al., 2019](#)), ResFlows have the attractive feature that the structure of their Jacobians is unconstrained, which may explain their greater expressiveness. However, as part of the construction, the bi-Lipschitz constant of f is bounded, and so these models must be composed many times in order to achieve overall the large bi-Lipschitz constant required for a complex P_X^* .¹

To address this problem we introduce *continuously indexed flows* (CIFs), which generalise (1) by replacing the single bijection f with an indexed family of bijections $\{F(\cdot; u)\}_{u \in \mathcal{U}}$, where the index set $\mathcal{U}$ is continuous. Intuitively, CIFs allow mass that would be erroneously placed by a single bijection to be rerouted into a more optimal location. We show that CIFs can learn the support of a given P_X^* exactly regardless of the topology of the prior, and without the bi-Lipschitz constant of any $F(\cdot; u)$ necessarily becoming infinite. CIFs do not specify the form of F , and can be used in conjunction with any standard normalising flow architecture directly.

Our use of a continuous index overcomes several limitations associated with alternative approaches based on a discrete index ([Dinh et al., 2019](#); [Duan, 2019](#)), which suffer either from a discontinuous loss landscape or an intractable computational complexity. However, as a consequence, we sacrifice the ability to compute the likelihood of our model analytically. To address this, we propose a variational approximation that exploits the bijective structure of the model and is suitable for training large-scale models in practice. We empirically evaluate CIFs applied to ResFlows, neural

spline flows (NSFs) ([Durkan et al., 2019](#)), masked autoregressive flows (MAFs) ([Papamakarios et al., 2017](#)), and RealNVPs ([Dinh et al., 2017](#)), obtaining improved performance in all cases. We observe a particular benefit for ResFlows: with a 10-layer CIF-ResFlow we surpass the performance of a 100-layer baseline ResFlow and achieve state-of-the-art results on several benchmark datasets.

2 Bi-Lipschitz Constraints on Pushforwards

Normalising flows fall into a larger class of density estimators based on *pushforwards*. Given a prior measure P_Z on $\mathcal{Z}$ and a mapping $f : \mathcal{Z} \rightarrow \mathcal{X}$, these models are defined as

$$P_X := f\#P_Z,$$

where the right-hand side denotes a distribution with $f\#P_Z(B) := P_Z(f^{-1}(B))$ for Borel $B \subseteq \mathcal{X}$. Normalising flows take f to be bijective, which under sufficient regularity yields a closed-form expression for the density² of P_X ([Billingsley, 2008](#), Theorem 17.2).

Intuitively, the pushforward map f *transports* the mass allocated by P_Z into $\mathcal{X}$ -space, thereby defining P_X based on where each unit of mass ends up. This imposes a global constraint on f if P_X is to match perfectly a given target P_X^* . In particular, denote by $\text{supp } P_Z$ the *support* of P_Z . While the precise definition of the support involves topological formalities (see [Section B.1](#) in the Supplement), intuitively this set defines the region of $\mathcal{Z}$ to which P_Z assigns mass. It is then straightforward to show that $P_X = P_X^*$ only if

$$\text{supp } P_X^* = \overline{f(\text{supp } P_Z)}, \quad (3)$$

where $\overline{A}$ denotes the closure of A in $\mathcal{X}$.³

The constraint (3) is especially onerous for normalising flows because of their bijectivity. In practice, f and f^{-1} are invariably both continuous, and so f is a *homeomorphism*. Consequently, for these models (3) entails⁴

$$\text{supp } P_X = \text{supp } P_X^* \text{ only if } \text{supp } P_Z \cong \text{supp } P_X^*, \quad (4)$$

where $\mathcal{A} \cong \mathcal{B}$ means that $\mathcal{A}$ and $\mathcal{B}$ are *homeomorphic*, i.e. isomorphic as topological spaces ([Runde, 2007](#), Definition 3.3.10). This means that $\text{supp } P_Z$ and $\text{supp } P_X^*$ must exactly share *all* topological properties, including number of connected components, number of “holes”, the way they are “knotted”, etc., in order to learn the target perfectly. Condition (4) therefore suggests that normalising flows are not optimally suited to the task of learning complex real-world densities, where such topological mismatch seems inevitable.

²Throughout, by “density” we mean Lebesgue density. We will write densities using lowercase, e.g. p_X for the measure P_X .

³See [Proposition B.3](#) in the Supplement for a proof.

⁴Note that $\overline{f(\text{supp } P_Z)} = f(\text{supp } P_Z)$ here since $\text{supp } P_Z$ is closed by [Proposition B.2](#) in the Supplement.

¹[Chen et al. \(2019\)](#) report using 100-200 layers to learn even simple 2D densities.

However, (4) only rules out the limiting case $P_X = P_X^*$. In practice it is likely enough to have $P_X \approx P_X^*$, and it is therefore relevant to consider the implications of a topologically misspecified prior in this case also. Intuitively, this seems to require f become *almost* nonbijective as P_X approaches P_X^* , but it is not immediately clear what this means, or whether this must occur for all models. Likewise, in practice it might be reasonable to assume the density of P_X^* is everywhere strictly positive. In this case, even if P_X^* is concentrated on some very complicated set, the constraint (4) would trivially be met if P_Z is Gaussian, for example. Nevertheless, it seems that infinitesimal regions of mass should not significantly change the behaviour required of f , and we would therefore like to extend (4) to apply here also.

The bi-Lipschitz constant (2) naturally quantifies the “invertibility” of f . Behrmann et al. (2020) recently showed a relationship between the bi-Lipschitz constant and the numerical invertibility of f . If f is injective and differentiable,

$$\text{BiLip } f = \max \left(\sup_{z \in \mathcal{Z}} \|Df(z)\|_{\text{op}}, \sup_{x \in f(\mathcal{Z})} \|Df^{-1}(x)\|_{\text{op}} \right),$$

where $Dg(y)$ is the Jacobian of g at y and $\|\cdot\|_{\text{op}}$ is the operator norm. A large bi-Lipschitz constant thus means f or f^{-1} “jumps” somewhere in its domain. More generally, if f is not injective, then $\text{BiLip } f = \infty$, while if $\text{BiLip } f < \infty$, then f is a homeomorphism from $\mathcal{Z}$ to $f(\mathcal{Z})$.⁵

The following theorem shows that if the supports of P_Z and P_X^* are not homeomorphic, then the bi-Lipschitz constant of f must grow arbitrarily large in order to approximate P_X^* . Here $\xrightarrow{\mathcal{D}}$ denotes weak convergence.

Theorem 2.1. *Suppose P_Z and P_X^* are probability measures on $\mathbb{R}^{d_Z}$ and $\mathbb{R}^{d_X}$ respectively, and that $\text{supp } P_Z \not\cong \text{supp } P_X^*$. Then for any sequence of measurable $f_n : \mathbb{R}^{d_Z} \rightarrow \mathbb{R}^{d_X}$, we can have $f_n \# P_Z \xrightarrow{\mathcal{D}} P_X^*$ only if*

$$\lim_{n \rightarrow \infty} \text{BiLip } f_n = \infty.$$

Weak convergence is implied by the minimisation of all standard statistical divergences used to train generative models, including the KL and Jensen-Shannon divergences and the Wasserstein metric (Arjovsky et al., 2017, Theorem 2). Thus, Theorem 2.1 states that these quantities can vanish only if the bi-Lipschitz constant of the learned mapping becomes arbitrarily large. Likewise, note that we do not assume $d_Z = d_X$ so that this result also applies to injective flow models (Kumar et al., 2019), as well as other pushforward-based models such as GANs (Goodfellow et al., 2014).⁶

⁵See Section B.2 in the Supplement for proofs.

⁶However, the implications for GANs seem less problematic since a GAN generator is not usually assumed to be bijective.

Theorem 2.1 also applies when $\text{supp } P_Z$ is *almost* not homeomorphic to $\text{supp } P_X^*$, as is made precise by the following corollary. Here ρ denotes any metric for the weak topology; see Chapter 6 of Villani (2008) for standard examples.

Corollary 2.2. *Suppose P_Z and P_X^0 are probability measures on $\mathbb{R}^{d_Z}$ and $\mathbb{R}^{d_X}$ respectively with $\text{supp } P_Z \not\cong \text{supp } P_X^0$. Then there exists nonincreasing $M : [0, \infty) \rightarrow [1, \infty]$ with $M(\epsilon) \rightarrow \infty$ as $\epsilon \rightarrow 0$ such that, for any probability measure P_X^* on $\mathbb{R}^{d_X}$, we have $\text{BiLip } f \geq M(\epsilon)$ whenever $\rho(P_X^*, P_X^0) \leq \epsilon$ and $\rho(f \# P_Z, P_X^*) \leq \epsilon$.*

In other words, if the target is close to a probability measure with non-homeomorphic support to that of the prior (i.e. $\rho(P_X^*, P_X^0)$ is small), and if the model is a good approximation of the target (i.e. $\rho(f \# P_Z, P_X^*)$ is small), then the Bi-Lipschitz constant of f must be large.

Proofs of these results are in Section B.3 of the Supplement.

2.1 Practical Implications

The results of this section indicate a limitation of existing flow-based density models. This is most direct for *residual flows* (ResFlows) (Behrmann et al., 2019; Chen et al., 2019), which take $f = f_L \circ \dots \circ f_1$ with each layer of the form

$$f_\ell^{-1}(x) = x + g_\ell(x), \quad \text{Lip } g_\ell \leq \kappa < 1. \quad (5)$$

Here Lip denotes the Lipschitz constant, which is bounded by a fixed constant κ throughout training. The Lipschitz constraint is enforced by spectral normalisation (Miyato et al., 2018; Gouk et al., 2018) and ensures each f_ℓ is bijective. However, it also follows (Behrmann et al., 2019, Lemma 2) that

$$\text{BiLip } f \leq \max(1 + \kappa, (1 - \kappa)^{-1})^L < \infty, \quad (6)$$

and Theorem 2.1 thus restricts how well a ResFlow can approximate P_X^* with non-homeomorphic support to P_Z . Figure 1 illustrates this in practice for simple 2-D examples.

It is possible to relax (6) by taking $\kappa \rightarrow 1$. However, this can have a detrimental effect on the variance of the Russian roulette estimator (Kahn, 1955) used by Chen et al. (2019) to compute the Jacobian, and in Section B.4 of the Supplement we give a simple example in which the variance is in fact infinite. Alternatively, we can also loosen the bound (6) by taking $L \rightarrow \infty$, and Figure 1 shows that this does indeed lead to better performance. However, greater depth means greater computational cost. In the next section we describe an alternative approach that allows relaxing the bi-Lipschitz constraint of Theorem 2.1 without modifying either κ or L , and thus avoids these potential issues.

Unlike ResFlows, most normalising flows used in practice have an unconstrained bi-Lipschitz constant (Behrmann et al., 2020). As a result, Theorem 2.1 does not prevent

these models from approximating non-homeomorphic targets arbitrarily well, and indeed several architectures have been proposed that can in principle do so (Huang et al., 2018; Jaini et al., 2019). Nevertheless, the constraint (4) shows that these models still face an underlying limitation in practice, and suggests we may improve performance more generally by relaxing the requirement of bijectivity. We verify empirically in Section 5 that, in addition to ResFlows, our proposed method also yields benefits for flows without an explicit bi-Lipschitz constraint.

Finally, Theorem 2.1 has implications for the numerical stability of normalising flows. It was recently pointed out by Behrmann et al. (2020) that, while having a well-defined mathematical inverse, many common flows can become *numerically* noninvertible over the course of training, leading to low-quality reconstructions and calling into question the accuracy of density values output by the change-of-variables formula. Behrmann et al. (2020) suggest explicitly constraining BiLip f in order to avoid this problem. Theorem 2.1 shows that this involves a fundamental tradeoff against expressivity: if greater numerical stability is required of our normalising flow, then we must necessarily reduce the set of targets we can represent arbitrarily well.

3 Continuously Indexed Flows

In this section we propose *continuously indexed flows* (CIFs) for relaxing the bijectivity of standard normalising flows. We begin by defining the model we consider, and then detail our suggested training and inference procedures. In the next section we discuss advantages over related approaches.

3.1 Model Specification

CIFs are obtained by replacing the single bijection f used by normalising flows with an indexed family $\{F(\cdot; u)\}_{u \in \mathcal{U}}$, where $\mathcal{U} \subseteq \mathbb{R}^{d_u}$ is our index set and each $F(\cdot; u) : \mathcal{Z} \rightarrow \mathcal{X}$ is a bijection. We then define the model P_X as the marginal of X obtained from the following generative process:

$$Z \sim P_Z, \quad U \sim P_{U|Z}(\cdot|Z), \quad X := F(Z; U). \quad (7)$$

Like (1), we assume a prior P_Z on $\mathcal{Z}$, but now also require conditional distributions $P_{U|Z}(\cdot|z)$ on $\mathcal{U}$ for each $z \in \mathcal{Z}$.

We can increase the complexity of (7) by taking P_Z itself to have the same form. This is directly analogous to the standard practice of composing simple bijections to obtain a richer class of normalising flows. In our context, stacking L layers of (7) corresponds to the generative process

$$Z_0 \sim P_{Z_0}, \quad U_\ell \sim P_{U_\ell|Z_{\ell-1}}(\cdot|Z_{\ell-1}), \quad Z_\ell := F_\ell(Z_{\ell-1}; U_\ell), \quad (8)$$

where $\ell \in \{1, \dots, L\}$. We then take P_X to be the marginal of $X := Z_L$. We have found this construction

to improve significantly the expressiveness of our models and make extensive use of it in our experiments below. Note that this corresponds to an instance of (7) where, defining $F^\ell(\cdot; u_1, \dots, u_\ell) := F_\ell(\cdot; u_\ell) \circ \dots \circ F_1(\cdot; u_1)$, we take $Z = Z_0$, $U = (U_1, \dots, U_L)$, $P_{U|Z}(\mathrm{d}u|z) = \prod_{\ell} P_{U_\ell|Z_{\ell-1}}(\mathrm{d}u_\ell|F^\ell(z; u_1, \dots, u_{\ell-1}))$, and $F = F^L$. We use this to streamline some of the discussion below.

Previous works, most notably RAD (Dinh et al., 2019), have considered related models with a discrete index set $\mathcal{U}$. We instead consider a *continuous* index. In particular, our $\mathcal{U}$ will be an open subset of $\mathbb{R}^{d_u}$, with each $P_{U|Z}(\cdot|z)$ having a density $p_{U|Z}(\cdot|z)$. A continuous index confers various advantages that we describe in Section 4. The choice also requires a distinct approach to training and inference that we describe in Section 3.2.

We require choices of $p_{U|Z}$ and F for each layer of our model. Straightforward possibilities are

$$F(z; u) = f \left(e^{-s(u)} \odot z - t(u) \right) \quad (9)$$

$$p_{U|Z}(\cdot|z) = \text{Normal}(\mu^p(z), \Sigma^p(z)) \quad (10)$$

for any bijection f (e.g. a ResFlow step) and appropriately defined neural networks s , t , μ^p , and Σ^p .⁷ Here the exponential of a vector is meant elementwise, and $\odot$ denotes elementwise multiplication. Note that (9) may be used with all existing normalising flow implementations out-of-the-box. These choices yielded strong empirical results despite their simplicity, but more sophisticated alternatives are certainly possible and may bring improvements in some applications.

3.2 Training and Inference

Heuristically,⁸ (7) yields the joint “density”

$$p_{X,U,Z}(x, u, z) := p_Z(z) p_{U|Z}(u|z) \delta(x - F(z; u)),$$

where p_Z is the density of P_Z and δ is the Dirac delta. If F is sufficiently regular, we can marginalise out the dependence on z by making the change of variable $x' := F(z; u)$, which means $\mathrm{d}z = |\det DF^{-1}(x'; u)| \mathrm{d}x'$.⁹ This yields a proper density for (X, U) by integrating over x' :

$$p_{X,U}(x, u) := p_Z(F^{-1}(x; u)) \times p_{U|Z}(u|F^{-1}(x; u)) |\det DF^{-1}(x; u)|. \quad (11)$$

For an L -layered model, an extension of this argument also gives the following joint density for each $(Z_\ell, U_{1:\ell})$:

$$p_{Z_\ell, U_{1:\ell}}(z_\ell, u_{1:\ell}) := p_{Z_{\ell-1}, U_{1:\ell-1}}(F_\ell^{-1}(z_\ell; u_\ell), u_{1:\ell-1}) \times p_{U_\ell|Z_{\ell-1}}(u_\ell|F_\ell^{-1}(z_\ell; u_\ell)) |\det DF_\ell^{-1}(z_\ell; u_\ell)|. \quad (12)$$

⁷Note this requires $\mathcal{Z} = \mathcal{X} = \mathbb{R}^d$ and $\mathcal{U} = \mathbb{R}^{d_u}$, i.e. these domains are not strict subsets. We assume this in all our experiments.

⁸We make this rigorous in Section B.5 of the Supplement.

⁹Here $DF(z; u)$ denotes the Jacobian with respect to z only.

Taking $X := Z_L$ as before we obtain $p_{X,U_{1:L}}$ and hence a density for P_X via

$$p_X(x) := \int p_{X,U_{1:L}}(x, u_{1:L}) du_{1:L}. \quad (13)$$

Since $\mathcal{U}$ is continuous, this is not analytically tractable. To facilitate likelihood-based training and inference, we make use of a variational scheme that we describe now.

Assuming an L -layered model (8), we introduce an approximate posterior density $q_{U_{1:L}|X} \approx p_{U_{1:L}|X}$ and consider the evidence lower bound (ELBO) of $\log p_X(x)$:

$$\mathcal{L}(x) := \mathbb{E}_{u_{1:L} \sim q_{U_{1:L}|X}(\cdot|x)} \left[\log \frac{p_{X,U_{1:L}}(x, u_{1:L})}{q_{U_{1:L}|X}(u_{1:L}|x)} \right]. \quad (14)$$

It is a standard result that $\mathcal{L}(x) \leq \log p_X(x)$ with equality if and only if $q_{U_{1:L}|X}$ is the exact posterior $p_{U_{1:L}|X}$. This allows learning an approximation to P_X^* by maximising $\sum_{i=1}^n \mathcal{L}(x_i)$ jointly in $p_{X,U_{1:L}}$ and $q_{U_{1:L}|X}$, where we assume a dataset of n i.i.d. samples $x_i \sim P_X^*$.

We now consider how to parametrise an effective $q_{U_{1:L}|X}$. Standard approaches to designing inference networks for variational autoencoders (VAEs) (Kingma & Welling, 2014; Rezende et al., 2014; Rezende & Mohamed, 2015; Kingma et al., 2016), while mathematically valid, would not exploit the conditional independencies induced by the bijective structure of (8). We therefore propose a novel inference network that is specifically targeted towards our model, which we compare with existing VAE approaches in Section 4.3. In particular, our $q_{U_{1:L}|X}$ has the following form:

$$q_{U_{1:L}|X}(u_{1:L}|x) := \prod_{\ell=1}^L q_{U_\ell|Z_\ell}(u_\ell|z_\ell), \quad (15)$$

with $z_L := x$ and $z_\ell := F_{\ell+1}^{-1}(z_{\ell+1}; u_{\ell+1})$ for $\ell \in \{1, \dots, L-1\}$, and $q_{U_\ell|Z_\ell}$ can be any parameterised conditional density. We show in Section B.6 of the Supplement that the posterior $p_{U_{1:L}|X}$ factors in the same way as (15), so that we do not lose any generality. Observe also that this scheme shares parameters between $q_{U_{1:L}|X}$ and $p_{X,U_{1:L}}$ in a natural way, since the same F_ℓ are used in both.

We assume each $q_{U_\ell|Z_\ell}$ can be suitably reparametrised (Kingma & Welling, 2014; Rezende et al., 2014) so that, for some function H_ℓ and some density η_ℓ that does not depend on the parameters of $q_{U_{1:L}|X}$ and $p_{X,U_{1:L}}$, we have $H_\ell(\epsilon_\ell, z_\ell) \sim q_{U_\ell|Z_\ell}(\cdot|z_\ell)$ when $\epsilon_\ell \sim \eta_\ell$. We can then obtain unbiased estimates of $\mathcal{L}(x)$ using Algorithm 1, which corresponds to a single-sample approximation to the expectation in (14). It is straightforward to see that Algorithm 1 has $\Theta(L)$ complexity. Differentiating through this procedure allows maximising $\sum_{i=1}^n \mathcal{L}(x_i)$ via stochastic gradient descent. At test time, we can also estimate $\log p_X(x)$ directly using importance sampling as described by Rezende

et al. (2014, (40)). In particular, letting $\hat{\mathcal{L}}^{(1)}, \dots, \hat{\mathcal{L}}^{(m)}$ denote the result of separate calls to ELBO(x), we have

$$m^{-1} \text{LogSumExp}(\hat{\mathcal{L}}^{(1)}, \dots, \hat{\mathcal{L}}^{(m)}) \rightarrow \log p_X(x) \quad (16)$$

almost surely as $m \rightarrow \infty$.

Algorithm 1 Unbiased estimation of $\mathcal{L}(x)$

function ELBO(x)

$z_L \leftarrow x$

$\Delta \leftarrow 0$

for $\ell = L, \dots, 1$ **do**

$\epsilon \sim \eta_\ell$

$u \leftarrow H_\ell(\epsilon, z_\ell)$

$z_{\ell-1} \leftarrow F_\ell^{-1}(z_\ell; u)$

$\Delta \leftarrow \Delta + \log p_{U_\ell|Z_{\ell-1}}(u|z_{\ell-1}) - \log q_{U_\ell|Z_\ell}(u|z_\ell)$
 $+ \log |\det DF_\ell^{-1}(z_\ell; u)|$

end for

return $\Delta + \log p_{Z_0}(z_0)$

end function

In all our experiments we used

$$q_{U_\ell|Z_\ell}(\cdot|z_\ell) = \text{Normal}(\mu_\ell^q(z_\ell), \Sigma_\ell^q(z_\ell)) \quad (17)$$

for appropriate neural networks μ_ℓ^q and Σ_ℓ^q , which is immediately reparameterisable as described e.g. by Kingma & Welling (2014). We found this gave good enough performance that we did not require alternatives such as IAF (Kingma et al., 2016), but such options may also be useful.

Finally, Algorithm 1 requires an expression for $\log |\det DF_\ell^{-1}(z_\ell; u_\ell)|$. For (9) this is

$$\log \left| \det Df_\ell^{-1} \left(e^{s_\ell(u_\ell)} \odot (z_\ell + t_\ell(u_\ell)) \right) \right| + \sum_{i=1}^d [s_\ell(u_\ell)]_i,$$

where $[x]_i$ denotes the i^{th} dimension of x .

4 Comparison with Related Models

4.1 Comparison with Normalising Flows

We now compare CIFs with normalising flows, and in particular describe how CIFs relax the constraints of bijectivity identified in Section 2.

4.1.1 ADVANTAGES

Observe that (7) generalises normalising flows: if $F(\cdot; u)$ does not depend on u , then we obtain (1). Moreover, training with the ELBO in this case does not reduce performance compared with training a flow directly, as the following result shows. Here the components of our model $F_\theta, p_{U|Z}^\theta$, and $q_{U|X}^\theta$ are parameterised by $\theta \in \Theta$, and for a given choice of parameters θ we will denote by P_X^θ and $\mathcal{L}^\theta$ the corresponding distribution and ELBO (14) respectively.

Proposition 4.1. Suppose there exists $\phi \in \Theta$ such that, for some bijection $f : \mathcal{Z} \rightarrow \mathcal{X}$, $F_\phi(\cdot; u) = f(\cdot)$ for all $u \in \mathcal{U}$. Likewise, suppose $p_{U|Z}^\phi$ and $q_{U|X}^\phi$ are such that, for some density r on $\mathcal{U}$, $p_{U|Z}^\phi(\cdot|z) = q_{U|X}^\phi(\cdot|x) = r(\cdot)$ for all $z \in \mathcal{Z}$ and $x \in \mathcal{X}$. If $\mathbb{E}_{x \sim P_X^*}[\mathcal{L}^\theta(x)] \geq \mathbb{E}_{x \sim P_X^*}[\mathcal{L}^\phi(x)]$, then

$$D_{\text{KL}}(P_X^* \parallel P_X^\theta) \leq D_{\text{KL}}(P_X^* \parallel f \# P_Z).$$

Simply stated, in the limit of infinite data, optimising the ELBO will yield at least as performant a model (as measured by the KL) as any normalising flow our model family can express. The proof is in [Section B.7](#) of the Supplement. In practice, our choices (9), (10), and (17) can easily realise the conditions of [Proposition 4.1](#) by zeroing out the output weights of the neural networks (other than f) involved. Thus, for a given f , we have reason to expect a comparative or better performing model (as measured by average log-likelihood) when trained as a CIF rather than as a normalising flow.

We expect this will in fact lead to *improved* performance because, intuitively, $P_{U|Z}$ can reroute z that would otherwise map outside of $\text{supp } P_X^*$. To illustrate, fix f in (9) and choose some $z \in \mathcal{Z}$. If $f(z) \in \text{supp } P_X^*$, then setting $F(z; u) = f(z)$ for all $u \in \mathcal{U}$ as described above ensures $F(z; U) \in \text{supp } P_X^*$ when $U \sim P_{U|Z}(\cdot|z)$. If conversely $f(z) \notin \text{supp } P_X^*$, then we *still* have $F(z; U) \in \text{supp } P_X^*$ almost surely if $P_{U|Z}(\cdot|z)$ is supported on $\{u \in \mathcal{U} : F(z; u) \in \text{supp } P_X^*\}$. Of course, if f is too simple, then $P_{U|Z}$ must heuristically become very complex in order to obtain this behaviour. This would seem to make inference harder, leading to a looser ELBO (14) and thus overall worse performance after training. We therefore expect CIFs to work well for f that, like the 10-layer ResFlow in [Figure 1](#), can learn a close approximation to the support of the target but “leak” some mass outside of it due to (4) or [Theorem 2.1](#). A CIF can then use $P_{U|Z}$ to “clean up” these small extraneous regions of mass.

We provide empirical support for this argument in [Section 5](#). We also summarise our discussion above with the following precise result. Here ∂A denotes the boundary of a set A .

Proposition 4.2. If $P_X^*(\partial \text{supp } P_X^*) = 0$ and $(z, u) \mapsto F(z; u)$ is jointly continuous with

$$\overline{F(\text{supp } P_Z \times \mathcal{U})} \supseteq \text{supp } P_X^*, \quad (18)$$

then there exists $P_{U|Z}$ such that $\text{supp } P_X = \text{supp } P_X^*$ if and only if, for all $z \in \text{supp } P_Z$, there exists $u \in \mathcal{U}$ with

$$F(z; u) \in \text{supp } P_X^*. \quad (19)$$

The assumptions here are fairly minimal: the boundary condition ensures P_X^* is not pathological, and if (18) does not hold, then $D_{\text{KL}}(P_X^* \parallel P_X) = \infty$ for every $P_{U|Z}$.¹⁰

¹⁰See [Proposition B.1](#) and [Proposition B.3](#) in the Supplement.

Additionally, the following result gives a sufficient condition under which it is possible to learn the target exactly.

Proposition 4.3. If $F(z; \cdot) : \mathcal{U} \rightarrow \mathcal{X}$ is surjective for each $z \in \mathcal{Z}$, then there exists $P_{U|Z}$ such that $P_X = P_X^*$.

See [Section B.8](#) of the Supplement for proofs. These results do not require $\text{supp } P_Z \cong \text{supp } P_X^*$, thereby showing CIFs relax the constraint (4) for standard normalising flows.

Of course, in practice, our parameterisation (9) does not necessarily ensure that F will satisfy these conditions, and our parameterisation (10) may not be expressive enough to instantiate the $P_{U|Z}$ that is required. However, these results show that CIFs provide at least a *mechanism* for correcting a topologically misspecified prior. When F and $P_{U|Z}$ are sufficiently expressive, we can expect that they will learn to approximate these conditions over the course of training if doing so produces a better density estimate. We therefore anticipate CIFs will improve performance for ResFlows, where [Theorem 2.1](#) applies, and may have benefits more generally, since all flows are ultimately constrained by (4).

4.1.2 DISADVANTAGES

On the other hand, CIFs introduce additional overhead compared with regular normalising flows. It therefore remains to show we obtain better performance on a fixed computational budget, which requires using a smaller model. Empirically this holds for the models and datasets we consider in [Section 5](#), but there are likely cases where it does not, particularly if the topologies of the target and prior are similar.

Likewise, CIFs sacrifice the exactness of normalising flows. We do not see this as a significant problem for the task of density estimation, since the importance sampling estimator (16) means that at test time we can obtain arbitrary accuracy by taking m to be large. However, the lack of a closed-form density does limit the use of CIFs in some downstream tasks. In particular, CIFs cannot immediately be plugged in to a variational approximation in the manner of [Rezende & Mohamed \(2015\)](#), since this requires exact likelihoods. However, it may be possible to use CIFs in the context of an extended-space variational framework along the lines of [Agakov & Barber \(2004\)](#), and we leave this for future work.

4.2 Comparison with Discretely Indexed Models

Similar models to CIFs have been proposed that use a discrete index space. In the context of Bayesian inference, [Duan \(2019\)](#) proposes a single-layer ($L = 1$) model consisting of (7) with $\mathcal{U} = \{1, \dots, I\}$ and $F(\cdot; i) = f_i$ for separate normalising flows $f_1, \dots, f_I$. A special case of this framework is given by *deep Gaussian mixture models* ([Van den Oord & Schrauwen, 2014](#); [van den Oord & Dambre, 2015](#)), which corresponds to using invertible linear transformations for each f_i . In this case, (13) becomes a summation that can

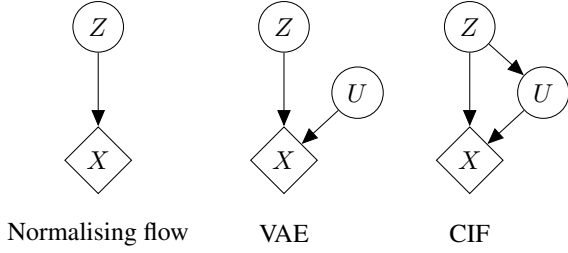


Figure 2: Comparison of related generative models. Circular nodes are random and diamond nodes are deterministic. CIFs generalise both normalising flows and VAEs as shown.

be computed analytically. However, this quickly becomes intractable as L grows larger, since the cost to compute this is seen to be $\Theta(I^L)$. Unlike for a continuous u , this cannot easily be reduced to $\Theta(L)$ using a variational approximation as in Section 3.2, since a discrete $q_{U|X}$ is not amenable to the reparameterisation trick. In addition, the use of separate bijections also means that the number of parameters of the model grows as I increases. In contrast, a continuous index allows a natural mechanism for sharing parameters across different $F(\cdot; u)$ as in (9).

Prior to Duan (2019), Dinh et al. (2019) proposed RAD as a means to mitigate the $\Theta(I^L)$ cost of naïvely stacking discrete layers. RAD partitions $\mathcal{X}$ into I disjoint subsets $B_1, \dots, B_I$ and defines bijections $f_i : \mathcal{Z} \rightarrow B_i$ for each i . The model is then taken to be the marginal of X in

$$Z \sim P_Z, \quad U \sim P_{U|Z}(\cdot|Z), \quad X := f_U(Z),$$

where each $P_{U|Z}(\cdot|z)$ is a discrete distribution on $\{1, \dots, I\}$. Note that this is not an instance of our model (7), since we require each $F(\cdot; u)$ to be surjective onto $\mathcal{X}$. The use of partitioning means that (13) is a summation with only a single term, which reduces the cost for L layers to $\Theta(L)$. However, partitioning also makes p_X discontinuous. This leads to a very difficult optimisation problem and Dinh et al. (2019) only report results for simple 2-D densities. Additionally, partitioning requires ad-hoc architectural changes to existing normalising flows, and does not directly address the increasing parameter cost as I grows large.

4.3 Comparison with Variational Autoencoders

CIFs also generalise a broad family of variational autoencoders (VAEs) (Kingma & Welling, 2014; Rezende et al., 2014). Recall that VAEs take

$$p_X(x) := \int p_U(u) p_{X|U}(x|u) du \quad (20)$$

for some choices of densities p_U and $p_{X|U}$.¹¹ For instance, a mean-field Gaussian observation density has

$$p_{X|U}(\cdot|u) := \text{Normal}\left(t(u), \text{diag}\left(e^{s(u)}\right)\right),$$

where $t, s : \mathcal{U} \rightarrow \mathcal{X}$, and $\text{diag}(v)$ denotes the matrix with diagonal $v \in \mathbb{R}^d$ and zeros elsewhere. If P_Z is a standard Gaussian, if each $P_{U|Z}(\cdot|z)$ has independent density p_U , and if F is (9) with f the identity, then it follows that (7) has marginal density (20) (modulo the signs of s and t).¹²

More generally, every VAE model (20) with each $p_{X|U}(\cdot|u)$ strictly positive corresponds to an instance of (7) where U is sampled independently of Z . To see this, let p_Z be any strictly positive density on $\mathcal{Z}$, and let each $F(\cdot; u)$ be the Knothe-Rosenblatt coupling (Villani, 2008) of p_Z and $p_{X|U}(\cdot|u)$. By construction each $F(\cdot; u)$ is invertible and gives $F(Z; u) \sim p_{X|U}(\cdot|u)$ when $Z \sim p_Z$. As a result, (7) again yields X with a marginal density defined by (20). Consequently, CIFs generalise the VAE framework by adding an additional edge in the graphical model as shown in Figure 2.

On the other hand, CIFs differ from VAEs in the way they are composed. Whereas CIFs stack by taking p_Z to be a CIF, VAEs are typically stacked by taking p_U to be a VAE (Rezende et al., 2014; Kingma et al., 2014; Burda et al., 2016; Sønderby et al., 2016). This has implications for the design of the inference network $q_{U_{1:L}|X}$. In particular, a hierarchical VAE obtained in this way is *Markovian*, so that

$$p_{U_{1:L}|X}(u_{1:L}|X) = p_{U_L|X}(u_L|x) \prod_{\ell=1}^L p_{U_\ell|U_{\ell-1}}(u_\ell|u_{\ell-1})$$

where L is the number of layers. This directly allows specifying $q_{U_{1:L}|X}$ to be of the same form without any loss of generality (Kingma et al., 2014; Burda et al., 2016) – i.e. using a “bottom-up approach” – although practically (Sønderby et al., 2016) it has been shown that providing some reverse “top-down” structure to the inference model as well provides improved performance. Conversely, CIFs do not factor in this way, which motivates our alternative approach in Section 3.2 and allows for efficient inference with a natural weight-sharing scheme between both the generation and inference direction.

Note finally that CIFs should not be conflated with the large class of methods that use normalising flows to improve the inference procedure in VAEs (Rezende & Mohamed, 2015; Kingma et al., 2016; van den Berg et al., 2018). These approaches are orthogonal to ours and indeed may be useful for improving our own inference procedure by replacing (17) with a more expressive model.

¹¹Note that this notation is nonstandard for VAEs in order to align with the rest of the paper. Here our U corresponds to z as used by Kingma & Welling (2014).

¹²Here Z corresponds to ϵ as used by Kingma & Welling (2014).

4.4 Other Related Work

Additional related methods have been proposed. Within a classification context, Dupont et al. (2019) identify topological problems related to ODE-based mappings (Chen et al., 2018), which like normalising flows are homeomorphisms and hence preserve the topology of their input. To avoid this, Dupont et al. (2019) propose augmenting the data by appending auxiliary dimensions and learning a new mapping on this space. In contrast, CIFs may be understood as augmenting not the data but instead the *model* by considering a family of individual bijections on the *original* space.

In addition, Ho et al. (2019) use a variational scheme to improve on the standard dequantisation method proposed by Theis et al. (2016) for modelling image datasets with normalising flows. This approach is potentially complementary to CIFs, but we do not make use of it in our experiments.

5 Experiments

We evaluated the performance of CIFs on several problems of varying difficulty, including synthetic 2-D data, several tabular datasets, and three image datasets. In all cases we took $\mathcal{Z} = \mathcal{X} = \mathbb{R}^d$ with d the dimension of the dataset. We used the stacked architecture (8) with the prior P_{Z_0} a Gaussian. At each layer, F had form (9) with f a primitive flow step from a baseline architecture (e.g. a single residual block for ResFlow). Each $p_{U|Z}$ and $q_{U|X}$ had form (10) and (17) respectively. We provide an overview of our results for the tabular and image datasets here. Full experimental details, including additional 2-D figures along the lines of Figure 1, are in Section C of the Supplement. See github.com/jrmcornish/cif for our code.

5.1 Tabular Datasets

We tested the performance of CIFs on the tabular datasets used by Papamakarios et al. (2017). For each dataset, we trained 10 and 100-layer baseline fully connected ResFlows, and corresponding 10-layer CIF-ResFlows. The CIF-ResFlows had roughly 1.5-4.5% more parameters (depending on the dimension of the dataset) than the otherwise identical 10-layer ResFlows, and roughly 10% of the parameters of the 100-layer ResFlows. Table 1 reports the average log-probability of the test set that we obtained for each model. Observe that in all cases CIF-ResFlows significantly outperform both baseline models. Moreover, for all but GAS, the CIF-ResFlows achieve state-of-the-art performance based on the results reported by Durkan et al. (2019, Table 1). This is particularly noticeable for POWER and BSDS300, where CIF-ResFlow improves on the best results of Durkan et al. (2019) by 0.94 and 2.77 nats respectively.

We additionally tried using *masked autoregressive flows* (MAFs) (Papamakarios et al., 2017) and *neural spline flows*

(NSFs) (Durkan et al., 2019) for f . In each case, we closely match the experimental settings of the baselines and augment using CIFs, controlling for the number of parameters used by the CIF extensions. Table 1 reports the average log-probability across the test set for each experiment. Here, CIF-NSF-1 is a CIF with the same number of parameters as the baseline, and CIF-NSF-2 is a model using a baseline configuration for f (but having more parameters overall). We see that CIF-MAFs consistently outperform MAFs across datasets; CIF-NSFs do not improve upon NSFs as dramatically, although we still notice improvements and would expect to improve further with more hyperparameter tuning. Lastly it is important to notice that MAFs and NSFs do not restrict the Lipschitz constant of f . These results show that CIFs can yield benefits for normalising flows even if Theorem 2.1 is not directly a limitation.

Finally, for ablation purposes we tried taking f to be the identity. We obtained consistently worse performance than for CIF-ResFlows and CIF-MAF in this case, which aligns with our conjecture in Section 4.1.1 that a performant CIF requires an expressive base flow f . Details and results are given in Section C.1.4 of the Supplement.

5.2 Image Datasets

We also considered CIFs applied to the MNIST (LeCun, 1998), Fashion-MNIST (Xiao et al., 2017), and CIFAR-10 (Krizhevsky & Hinton, 2009) datasets. Following our tabular experiments, we trained a multi-scale convolutional ResFlow and a corresponding CIF-ResFlow, as well as a larger baseline ResFlow to account for the additional parameters and depth introduced by our method. Note that these models were significantly smaller than those used by Chen et al. (2019): e.g. for CIFAR10, the ResFlow used by Chen et al. (2019) had 25M parameters, while our two baseline ResFlows and our CIF-ResFlow had 2.4M, 6.2M, and 5.6M parameters respectively. We likewise considered RealNVPs with the same multi-scale convolutional architecture used by Dinh et al. (2017) for their CIFAR-10 experiments. For these runs we trained baseline RealNVPs, corresponding CIF-RealNVPs, and larger baseline RealNVPs with more depth and parameters.

The results are given in Table 2 and Table 3. Observe CIFs outperformed the baseline models for all datasets, which shows that our approach can scale to high dimensions. For the CIF-ResFlows, we also obtained better performance than Chen et al. (2019) on MNIST and better performance than Glow (Kingma & Dhariwal, 2018) on CIFAR10, despite using a much smaller model. Samples from all models are shown in Section C.2 of the Supplement. We note that our samples are neither better nor worse than the baselines, but also point out that our model is optimized for performance on log-likelihood, and thus would not necessarily

Table 1: Mean $\pm$ standard error (over 3 seeds) of average test set log-likelihood (in nats). Higher is better. Best performing runs for each group are shown in bold. A $\star$ indicates state-of-the-art performance according to Durkan et al. (2019, Table 1).

	POWER	GAS	HEPMASS	MINIBOONE	BSDS300
RESFLOW ($L = 10$)	-2.73 ± 0.03	4.16 ± 0.08	-20.68 ± 0.02	-14.2 ± 0.10	123.51 ± 0.09
RESFLOW ($L = 100$)	0.48 ± 0.00	10.57 ± 0.17	-16.67 ± 0.05	-11.16 ± 0.04	148.05 ± 0.61
CIF-RESFLOW ($L = 10$)	$1.60 \pm 0.21^\star$	12.12 ± 0.10	$-13.74 \pm 0.03^\star$	$-8.10 \pm 0.04^\star$	$160.50 \pm 0.08^\star$
MAF	0.19 ± 0.02	9.23 ± 0.07	-18.33 ± 0.10	-10.98 ± 0.03	156.13 ± 0.00
CIF-MAF	0.48 ± 0.01	12.02 ± 0.10	-16.63 ± 0.09	-9.93 ± 0.04	156.67 ± 0.02
NSF	0.69 ± 0.00	13.01 ± 0.02	-14.30 ± 0.05	-10.68 ± 0.06	157.59 ± 0.02
CIF-NSF-1	0.68 ± 0.01	12.94 ± 0.01	-13.83 ± 0.10	-9.93 ± 0.06	157.60 ± 0.02
CIF-NSF-2	0.69 ± 0.00	13.08 ± 0.00	-14.18 ± 0.09	-10.80 ± 0.01	157.56 ± 0.02

Table 2: Average test bits per dimension.¹³ Lower is better.

	MNIST	CIFAR-10
RESFLOW (SMALL)	1.074	3.474
RESFLOW (BIG)	1.018	3.422
CIF-RESFLOW	0.922	3.334

Table 3: Mean $\pm$ standard error of average test set bits per dimension over 3 random seeds. Lower is better.

	FASHION-MNIST	CIFAR-10
REALNVP (SMALL)	2.944 ± 0.003	3.565 ± 0.001
REALNVP (BIG)	2.946 ± 0.002	3.554 ± 0.001
CIF-REALNVP	2.823 ± 0.003	3.477 ± 0.019

be expected to produce better samples even with better log-likelihood numbers (Theis et al., 2016). We leave producing better samples with CIFs to future work.

6 Conclusion and Future Work

The constraint (4) shows that normalising flows are unable to exactly model targets whose topology differs from that of the prior. Moreover, in order to approximate such targets closely, Theorem 2.1 shows that the bi-Lipschitz constant of a flow must become arbitrarily large. To address these problems, we have proposed CIFs, which can “clean up” regions of mass that are placed outside the support of the target by a standard flow. CIFs perform well in practice and outperform baseline flows on several benchmark datasets.

While we have focussed on the use of CIFs for density estimation in this paper, it would also be interesting to apply CIFs in other contexts where normalising flows have been used successfully. As CIFs do not have an analytically available density, this would likely require the modification

¹³Only one seed was used per run due to computational limitations. However, the results were not cherry-picked.

of existing numerical frameworks, but the expressiveness benefits provided by CIFs might make this additional effort worthwhile. We leave this direction for future work.

Acknowledgements

Rob Cornish is supported by the EPSRC Centre for Doctoral Training in Autonomous Intelligent Machines & Systems (EP/L015897/1) and NVIDIA. Anthony Caterini is a Commonwealth Scholar supported by the U.K. Government. Arnaud Doucet is partially supported by the U.S. Army Research Laboratory, the U.S. Army Research Office, and by the U.K. Ministry of Defence (MoD) grant EP/R013616/1 and the U.K. EPSRC under grant numbers EP/R034710/1 and EP/R018561/1.

References

- Agakov, F. V. and Barber, D. An auxiliary variational method. In *International Conference on Neural Information Processing*, pp. 561–566. Springer, 2004.
- Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pp. 214–223, 2017.
- Bache, K. and Lichman, M. UCI machine learning repository, 2013. URL <http://archive.ics.uci.edu/ml>.
- Beatson, A. and Adams, R. P. Efficient optimization of loops and limits with randomized telescoping sums. In *International Conference on Machine Learning*, pp. 534–543, 2019.
- Behrmann, J., Grathwohl, W., Chen, R. T., Duvenaud, D., and Jacobsen, J.-H. Invertible residual networks. In *International Conference on Machine Learning*, pp. 573–582, 2019.
- Behrmann, J., Vicol, P., Wang, K.-C., Grosse, R. B., and Jacobsen, J.-H. On the invertibility of invertible neural

- networks, 2020. URL <https://openreview.net/forum?id=BJlVeyHFwH>.
- Billingsley, P. *Probability and Measure*. John Wiley & Sons, 2008.
- Burda, Y., Grosse, R., and Salakhutdinov, R. Importance weighted autoencoders. In *ICLR*, 2016.
- Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pp. 6571–6583, 2018.
- Chen, T. Q., Behrmann, J., Duvenaud, D. K., and Jacobsen, J.-H. Residual flows for invertible generative modeling. In *Advances in Neural Information Processing Systems*, pp. 9913–9923, 2019.
- Dinh, L., Krueger, D., and Bengio, Y. Nice: Non-linear independent components estimation. In *ICLR Workshop*, 2015.
- Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real NVP. In *ICLR*, 2017.
- Dinh, L., Sohl-Dickstein, J., Pascanu, R., and Larochelle, H. A RAD approach to deep mixture models. In *ICLR Workshop*, 2019.
- Duan, L. L. Transport Monte Carlo. *arXiv preprint arXiv:1907.10448*, 2019.
- Dudley, R. M. *Real Analysis and Probability*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2 edition, 2002.
- Dupont, E., Doucet, A., and Teh, Y. W. Augmented neural ODEs. In *Advances in Neural Information Processing Systems*, pp. 3134–3144, 2019.
- Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. Neural spline flows. In *Advances in Neural Information Processing Systems*, pp. 7509–7520, 2019.
- Germain, M., Gregor, K., Murray, I., and Larochelle, H. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pp. 881–889, 2015.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.
- Gouk, H., Frank, E., Pfahringer, B., and Cree, M. Regularisation of neural networks by enforcing lipschitz continuity. *arXiv preprint arXiv:1804.04368*, 2018.
- Grathwohl, W., Chen, R. T., Betterncourt, J., Sutskever, I., and Duvenaud, D. FFJORD: Free-form continuous dynamics for scalable reversible generative models. In *ICLR*, 2019.
- He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016a.
- He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pp. 630–645. Springer, 2016b.
- Ho, J., Chen, X., Srinivas, A., Duan, Y., and Abbeel, P. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning*, pp. 2722–2730, 2019.
- Huang, C.-W., Krueger, D., Lacoste, A., and Courville, A. Neural autoregressive flows. In *International Conference on Machine Learning*, pp. 2083–2092, 2018.
- Hutchinson, M. F. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 19(2):433–450, 1990.
- Jaini, P., Selby, K. A., and Yu, Y. Sum-of-squares polynomial flow. In *International Conference on Machine Learning*, pp. 3009–3018, 2019.
- Kahn, H. Use of different Monte Carlo sampling techniques. Technical report, Rand Corporation, 1955.
- Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2015.
- Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pp. 10215–10224, 2018.
- Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. In *ICLR*, 2014.
- Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pp. 3581–3589, 2014.
- Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pp. 4743–4751, 2016.

- Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.
- Kumar, A., Poole, B., and Murphy, K. Learning generative samplers using relaxed injective flow. In *ICML Workshop on Invertible Neural Nets and Normalizing Flows*, 2019.
- LeCun, Y. The MNIST database of handwritten digits, 1998. URL <http://yann.lecun.com/exdb/mnist/>.
- Lyne, A.-M., Girolami, M., Atchadé, Y., Strathmann, H., Simpson, D., et al. On Russian roulette estimates for Bayesian inference with doubly-intractable likelihoods. *Statistical science*, 30(4):443–467, 2015.
- Martin, D., Fowlkes, C., Tal, D., and Malik, J. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pp. 416–423. IEEE, 2001.
- Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. In *ICLR*, 2018.
- Papamakarios, G., Pavlakou, T., and Murray, I. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pp. 2338–2347, 2017.
- Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pp. 1530–1538, 2015.
- Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pp. 1278–1286, 2014.
- Rhee, C.-h. and Glynn, P. W. Unbiased estimation with square root convergence for SDE models. *Operations Research*, 63(5):1026–1043, 2015.
- Rudin, W. *Principles of Mathematical Analysis*, volume 3. McGraw-hill New York, 1964.
- Rudin, W. *Real and Complex Analysis*. Tata McGraw-hill education, 2006.
- Runde, V. *A Taste of Topology*. Springer, 2007.
- Skilling, J. The eigenvalues of mega-dimensional matrices. In *Maximum Entropy and Bayesian Methods*, pp. 455–466. Springer, 1989.
- Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., and Winther, O. Ladder variational autoencoders. In *Advances in Neural Information Processing Systems*, pp. 3738–3746, 2016.
- Theis, L., Oord, A. v. d., and Bethge, M. A note on the evaluation of generative models. In *ICLR*, 2016.
- van den Berg, R., Hasenclever, L., Tomczak, J. M., and Welling, M. Sylvester normalizing flows for variational inference. In *UAI 2018: The Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 393–402, 2018.
- van den Oord, A. and Dambre, J. Locally-connected transformations for deep GMMs. In *International Conference on Machine Learning (ICML): Deep learning Workshop*, pp. 1–8, 2015.
- Van den Oord, A. and Schrauwen, B. Factoring variations in natural images with deep Gaussian mixture models. In *Advances in Neural Information Processing Systems*, pp. 3518–3526, 2014.
- Villani, C. *Optimal Transport: Old and New*, volume 338. Springer Science & Business Media, 2008.
- Xiao, H., Rasul, K., and Vollgraf, R. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Relaxing Bijectivity Constraints with Continuously Indexed Normalising Flows: Supplementary Material

A Guide to Notation

(a_n)	A sequence of elements $a_1, a_2, \dots$
$a(n) = \Theta(b(n))$	$a(n)$ differs from $b(n)$ by at most a constant factor as $n \rightarrow \infty$
$u \odot v$	The elementwise product of tensors u and v
$\text{LogSumExp}(a_1, \dots, a_m)$	$\log(\sum_{i=1}^m \exp(a_i))$
e^v , where $v \in \mathbb{R}^d$	$(e^{v_1}, \dots, e^{v_d})$
$\ v\ $	The norm of a vector $v \in \mathbb{R}^d$ (our results are agnostic to the specific choice of $\ \cdot\ $)
$\ A\ _{\text{op}}$	The operator norm of a matrix $A \in \mathbb{R}^{d_1 \times d_2}$ induced by $\ \cdot\ $
I_d	The $d \times d$ identity matrix
$\det A$	The determinant of a square matrix A
$Df(z)$	The Jacobian matrix of a function f evaluated at z
$DF(z; u)$	The Jacobian matrix of a function $DF(\cdot; u)$ (i.e. with u fixed) evaluated at z
$\text{Lip } f$	The Lipschitz constant of a function f
$\text{BiLip } f$	The bi-Lipschitz constant of a function f
$\mathcal{A} \cong \mathcal{B}$	The topological spaces $\mathcal{A}$ and $\mathcal{B}$ are homeomorphic
$\overline{B}$	The topological closure of a set B
$\text{int}(B)$	The interior of a set B
∂B	The boundary of a set B
$\text{supp } \mu$	The support of a measure μ
$f\#\mu$	The pushforward of a measure μ by a function f
$\mu_n \xrightarrow{\mathcal{D}} \mu$	Weak convergence of the measures μ_n to μ

B Proofs

B.1 Preliminaries

We require some basic results that we include here for completeness. We will make use of standard definitions and results from topology and real analysis. A complete background to these topics can be found in [Dudley \(2002\)](#).

B.1.1 SUPPORTS OF MEASURES

Recall that for a Borel measure μ on a topological space $\mathcal{Z}$, the *support* of μ , denoted $\text{supp } \mu$, is the set of all $z \in \mathcal{Z}$ such that $\mu(N_z) > 0$ for every open set N_z containing z .

The following is an immediate consequence:

Proposition B.1. *Suppose μ and ν are Borel measures with μ absolutely continuous with respect to ν . Then*

$$\text{supp } \mu \subseteq \text{supp } \nu.$$

Proof. Suppose $z \notin \text{supp } \nu$. Then there exists an open set N_z containing z such that $\nu(N_z) = 0$. By absolute continuity, we have also that $\mu(N_z) = 0$ and hence $z \notin \text{supp } \mu$. $\square$

In general the converse need not hold. For example, the Dirac measure on 0 has support contained within the Lebesgue measure on $\mathbb{R}$ (which has full support), but is not absolutely continuous with respect to it.

The following characterisation is useful:

Proposition B.2. *For any Borel measure μ ,*

$$(\text{supp } \mu)^c = \bigcup_{\substack{A \text{ open:} \\ \mu(A)=0}} A, \quad (\text{B.1})$$

and hence $\text{supp } \mu$ is closed.

Proof. This follows directly from the definitions, since $z \notin \text{supp } \mu$ if and only if there exists open N_z with $z \in N_z$ and $\mu(N_z) = 0$, which is just another way of saying that z is contained in the right-hand side of (B.1). It follows that $(\text{supp } \mu)^c$ is open, and hence $\text{supp } \mu$ is closed. $\square$

We mainly care about how the support of a measure is transformed by a pushforward function. The following proposition characterises what occurs in this case.

Proposition B.3. *Suppose $\mathcal{Z}$ and $\mathcal{X}$ are topological spaces. If μ is a Borel measure on $\mathcal{Z}$ such that $\mu((\text{supp } \mu)^c) = 0$, and if $f : \mathcal{Z} \rightarrow \mathcal{X}$ is continuous, then*

$$\text{supp } f\#\mu = \overline{f(\text{supp } \mu)}.$$

Proof. Suppose $x \notin \overline{f(\text{supp } \mu)}$. Then x must have an open neighbourhood N_x such that

$$N_x \cap f(\text{supp } \mu) = \emptyset.$$

This implies

$$\begin{aligned} f^{-1}(N_x) \cap \text{supp } \mu &\subseteq f^{-1}(N_x) \cap f^{-1}(f(\text{supp } \mu)) \\ &= f^{-1}(N_x \cap f(\text{supp } \mu)) \\ &= f^{-1}(\emptyset) \\ &= \emptyset. \end{aligned}$$

We then have

$$f\#\mu(N_x) = \mu(f^{-1}(N_x)) = \mu(f^{-1}(N_x) \cap \text{supp } \mu) = 0,$$

where the second equality follows since we assumed $\mu((\text{supp } \mu)^c) = 0$, and hence $x \notin \text{supp } f\#\mu$. Consequently

$$\text{supp } f\#\mu \subseteq \overline{f(\text{supp } \mu)}.$$

In the other direction, suppose $x \in \overline{f(\text{supp } \mu)}$, so that $x = f(z)$ for some $z \in \text{supp } \mu$. Given an open neighbourhood N_x it then follows from continuity that $f^{-1}(N_x)$ is an open neighbourhood of z , and so

$$f\#\mu(N_x) = \mu(f^{-1}(N_x)) > 0$$

since $z \in \text{supp } \mu$. This entails $\text{supp } f\#\mu \supseteq f(\text{supp } \mu)$, which means

$$\text{supp } f\#\mu = \overline{\text{supp } f\#\mu} \supseteq \overline{f(\text{supp } \mu)}$$

by Proposition B.2. $\square$

Note that in general we need not have $\text{supp } f\#\mu = f(\text{supp } \mu)$. For example, if μ is Gaussian and $f = \arctan$, then

$$f(\text{supp } \mu) = (-1, 1) \neq [-1, 1] = \text{supp } f\#\mu.$$

Likewise, in general we do require the assumption $\mu((\text{supp } \mu)^c) = 0$. This is because there exist examples of nontrivial Borel measures μ such that $\text{supp } \mu = \emptyset$. Taking $f \equiv x_0$ to be any constant $x_0 \in \mathcal{X}$ (in which case f is certainly continuous) then gives

$$\overline{f(\text{supp } \mu)} = \emptyset \neq \{x_0\} = \text{supp } f\#\mu.$$

However, for our purposes, the following proposition shows that this is not a restriction.

Proposition B.4. Suppose μ is a Borel measure on a separable metric space $\mathcal{Z}$. Then

$$\mu((\text{supp } \mu)^c) = 0.$$

Proof. Throughout the proof, for each z and $r > 0$, we will denote by $B(z, r)$ an open ball of radius r centered at z . Likewise, for each $z \notin \text{supp } \mu$, let

$$r^*(z) := \sup\{r > 0 \mid \mu(B(z, r)) = 0\}.$$

Observe that r^* is well-defined (but possibly infinite) since $z \notin \text{supp } \mu$ means there must exist some $r > 0$ such that $\mu(B(z, r)) = 0$.

We first show that $\mu(B(z, r^*(z))) = 0$ for all $z \notin \text{supp } \mu$. To this end, fix z and choose a sequence $r_m \uparrow r^*(z)$ with $r_m < r^*(z)$. We then have

$$B(z, r^*(z)) = \bigcup_{m=1}^{\infty} B(z, r_m),$$

and so

$$\mu(B(z, r^*(z))) = \lim_{m \rightarrow \infty} \mu(B(z, r_m)) = 0$$

by continuity of measure.

Now, by separability, we can choose a countable sequence $(z_k) \subseteq (\text{supp } \mu)^c$ such that $\overline{\{z_k\}} = \overline{(\text{supp } \mu)^c}$. We show that

$$(\text{supp } \mu)^c = \bigcup_{k=1}^{\infty} B(z_k, r^*(z_k)),$$

from which the result follows by countable subadditivity. It is clear from (B.1) that the left-hand side is a superset of the right. In the other direction, let $z \in (\text{supp } \mu)^c$. By construction of (z_k) , there exists a subsequence $(z_{k'})$ such that $z_{k'} \rightarrow z$. For all k' large enough we then have $z_{k'} \in B(z, r^*(z)/2)$ and hence

$$B(z_{k'}, r^*(z)/2) \subseteq B(z, r^*(z))$$

by triangle inequality. It follows that for such k' we have

$$\mu(B(z_{k'}, r^*(z)/2)) \leq \mu(B(z, r^*(z))) = 0,$$

and so $r^*(z_{k'}) \geq r^*(z)/2$ since $r^*(z_{k'})$ is the supremum. But then we have

$$z \in B(z_{k'}, r^*(z)/2) \subseteq B(z_{k'}, r^*(z_{k'})),$$

so that

$$z \in \bigcup_{k=1}^{\infty} B(z_k, r^*(z_k))$$

and we are done. $\square$

B.2 Lipschitz and Bi-Lipschitz Functions

We assume that $\mathcal{Z} \subseteq \mathbb{R}^{d_{\mathcal{Z}}}$, $\mathcal{X} \subseteq \mathbb{R}^{d_{\mathcal{X}}}$, and $f : \mathcal{Z} \rightarrow \mathcal{X}$. Recall that the *Lipschitz* constant of f , denoted $\text{Lip } f$, is defined as the infimum over $M \in [0, \infty]$ such that

$$\|f(z) - f(z')\| \leq M\|z - z'\|$$

for all $z, z' \in \mathcal{Z}$. Likewise the *bi-Lipschitz* constant $\text{BiLip } f$ is defined as the infimum over $M \in [1, \infty]$ such that

$$M^{-1}\|z - z'\| \leq \|f(z) - f(z')\| \leq M\|z - z'\|$$

for all $z, z' \in \mathcal{Z}$. We prove some basic properties that follow from this definition.

Proposition B.5. $\text{BiLip } f < \infty$ if and only if f is injective and $\max(\text{Lip } f, \text{Lip } f^{-1}) < \infty$, where $f^{-1} : f(\mathcal{Z}) \rightarrow \mathcal{Z}$. For all injective f , we then have $\text{BiLip } f = \max(\text{Lip } f, \text{Lip } f^{-1})$.

Proof. For the first statement, suppose $\text{BiLip } f < \infty$. It is immediate that $\text{BiLip } f \geq \text{Lip } f$. To see that f is injective, note that for $z \neq z'$ we have

$$\|f(z) - f(z')\| \geq (\text{BiLip } f)^{-1} \|z - z'\| > 0$$

and so $f(z) \neq f(z')$. On the other hand, for $x, x' \in f(\mathcal{Z})$, we have

$$(\text{BiLip } f)^{-1} \|f^{-1}(x) - f^{-1}(x')\| \leq \|f(f^{-1}(x)) - f(f^{-1}(x'))\| = \|x - x'\|,$$

which gives that $\text{BiLip } f \geq \text{Lip } f^{-1}$. Altogether we have

$$\max(\text{Lip } f, \text{Lip } f^{-1}) \leq \text{BiLip } f < \infty, \quad (\text{B.2})$$

which gives the forward direction.

Next suppose f is injective and that

$$M := \max(\text{Lip } f, \text{Lip } f^{-1}) < \infty.$$

For $z, z' \in \mathcal{Z}$, we certainly have

$$\|f(z) - f(z')\| \leq M \|z - z'\|.$$

Likewise, since $f(z), f(z') \in f(\mathcal{Z})$,

$$\|z - z'\| = \|f^{-1}(f(z)) - f^{-1}(f(z'))\| \leq M \|f(z) - f(z')\|,$$

so that

$$M^{-1} \|z - z'\| \leq \|f(z) - f(z')\|$$

because injectivity of f means that $M > 0$. From this it follows that

$$\text{BiLip } f \leq M < \infty, \quad (\text{B.3})$$

which gives the reverse direction, proving the first statement.

For the second statement, suppose f is injective. Then if $\text{BiLip } f < \infty$, (B.2) and (B.3) together give

$$\text{BiLip } f = \max(\text{Lip } f, \text{Lip } f^{-1}).$$

On the other hand, if $\text{BiLip } f = \infty$ then $\max(\text{Lip } f, \text{Lip } f^{-1}) = \infty$ since we would otherwise obtain a contradiction by the first statement of the proposition. This completes the proof. $\square$

It follows directly that if $\text{BiLip } f < \infty$, then f is a homeomorphism from $\mathcal{Z}$ to $f(\mathcal{Z})$.¹⁴ Moreover, in this case f maps closed sets to closed sets, as the following result shows:

Proposition B.6. *If $\text{BiLip } f < \infty$ and $\mathcal{Z}$ is closed in $\mathbb{R}^{dz}$, then $f(\mathcal{Z})$ is closed in $\mathbb{R}^{dx}$.*

Proof. It is a straightforward consequence of Proposition B.5 that if $(x_n) \subseteq f(\mathcal{Z})$ is Cauchy, then $(f^{-1}(x_n))$ is Cauchy. Consequently $(f^{-1}(x_n))$ converges to some $z_\infty \in \mathcal{Z}$, since $\mathcal{Z}$ is a closed subset of a complete space and therefore complete. But then

$$\begin{aligned} \|x_n - f(z_\infty)\| &= \|f(f^{-1}(x_n)) - f(z_\infty)\| \\ &\leq M \|f^{-1}(x_n) - z_\infty\| \\ &\rightarrow 0 \end{aligned}$$

as $n \rightarrow \infty$. Consequently $f(\mathcal{Z})$ is complete, and so $f(\mathcal{Z})$ is closed as desired since the ambient space $\mathbb{R}^{dx}$ is complete. $\square$

The Lipschitz constant can be computed from the *operator norm* $\|\cdot\|_{\text{op}}$ of the Jacobian of f . Recall that $\|\cdot\|_{\text{op}}$ is defined as for a matrix $A \in \mathbb{R}^{dx \times dz}$ as

$$\|A\|_{\text{op}} := \sup_{\substack{v \in \mathbb{R}^{dz}: \\ \|v\|=1}} \|Av\|$$

where we think of elements of $\mathbb{R}^{dz}$ as column vectors.

¹⁴Note however that the converse is not true in general: for example, $\exp$ is a homeomorphism from $\mathbb{R}$ to $(0, \infty)$, but $\text{BiLip } \exp = \infty$.

Proposition B.7. *If $\mathcal{Z} = \mathbb{R}^{d_Z}$, $\mathcal{X} = \mathbb{R}^{d_X}$, and f is everywhere differentiable, then*

$$\text{Lip } f = \sup_{z \in \mathcal{Z}} \|\text{D}f(z)\|_{\text{op}}.$$

Proof. If $v \in \mathcal{Z}$ with $\|v\| = 1$, then

$$\begin{aligned} \|\text{D}f(z)v\| &= \lim_{t \rightarrow 0} \frac{\|f(z + tv) - f(z)\|}{|t|} \\ &\leq \lim_{t \rightarrow 0} \frac{(\text{Lip } f)\|(z + tv) - z\|}{|t|} \\ &= \text{Lip } f. \end{aligned}$$

It follows directly that

$$\|\text{D}f(z)\|_{\text{op}} \leq \text{Lip } f.$$

On the other hand, suppose $\text{Lip } f > M$. Then there exists $z, z' \in \mathcal{Z}$ such that

$$\|f(z) - f(z')\| > M\|z - z'\|.$$

Since f is differentiable, so too is the map $\varphi : [0, 1] \rightarrow \mathcal{X}$ defined by

$$\varphi(t) := f(tz + (1 - t)z').$$

By Theorem 5.19 of [Rudin \(1964\)](#), there exists $t_0 \in (0, 1)$ such that the derivative φ' satisfies

$$\|\varphi'(t_0)\| \geq \|f(z') - f(z)\| > M\|z - z'\|.$$

But, letting $z_0 := t_0 z + (1 - t_0)z'$, observe that

$$\begin{aligned} \varphi'(t_0) &= \lim_{t \rightarrow 0} \frac{f(z_0 + t(z - z')) - f(z_0)}{t} \\ &= [\text{D}f(z_0)](z - z'), \end{aligned}$$

where we think of z, z' as column vectors. As such,

$$\begin{aligned} \|\text{D}f(z_0)\|_{\text{op}}\|z - z'\| &\geq \|[\text{D}f(z_0)](z - z')\| \\ &= \|\varphi'(t_0)\| \\ &> M\|z - z'\| \end{aligned}$$

and so

$$\sup_{z \in \mathcal{Z}} \|\text{D}f(z)\|_{\text{op}} > M.$$

Since M was arbitrary this means that

$$\text{Lip } f \leq \sup_{z \in \mathcal{Z}} \|\text{D}f(z)\|_{\text{op}}$$

which gives the result. □

[Proposition B.5](#) and [Proposition B.7](#) then immediately entail the following:

Corollary B.8. *Suppose $\mathcal{Z} = \mathbb{R}^{d_Z}$ and $\mathcal{X} = \mathbb{R}^{d_X}$. If f is injective, and if f and $f^{-1} : f(\mathcal{Z}) \rightarrow \mathcal{Z}$ are everywhere differentiable, then*

$$\text{BiLip } f = \max \left(\sup_{z \in \mathcal{Z}} \|\text{D}f(z)\|_{\text{op}}, \sup_{x \in f(\mathcal{Z})} \|\text{D}f^{-1}(x)\|_{\text{op}} \right).$$

B.2.1 ARZELÀ-ASCOLI

Our proof of [Theorem 2.1](#) makes use of the Arzelà-Ascoli theorem. This is a standard and foundational result in analysis, but we include a statement here for completeness. To this end, suppose we have a sequence of functions $f_n : \mathcal{Z} \subseteq \mathbb{R}^{d_Z} \rightarrow \mathcal{X} \subseteq \mathbb{R}^{d_X}$. We say that (f_n) is *pointwise bounded* if, for all $z \in \mathcal{Z}$,

$$\sup_n \|f_n(z)\| < \infty.$$

Likewise, (f_n) is *uniformly equicontinuous* if for every $\epsilon > 0$ there exists $\delta > 0$ such that, for all n ,

$$\|f_n(z) - f_n(z')\| < \epsilon$$

whenever $\|z - z'\| < \delta$.

Theorem B.9 (Arzelà-Ascoli). *If a sequence of functions $f_n : \mathcal{Z} \subseteq \mathbb{R}^{d_Z} \rightarrow \mathcal{X} \subseteq \mathbb{R}^{d_X}$ is pointwise bounded and uniformly equicontinuous, then there exists a subsequence of (f_n) that converges uniformly on every compact subset of $\mathcal{Z}$.*

Proof. The case $d = 1$ is proven for example by [Rudin \(2006, Theorem 11.28\)](#). This can be extended to the case $d > 1$ by a standard argument. In particular, write

$$f_n =: (f_{n,1}, \dots, f_{n,d}),$$

where $f_{n,i} : \mathcal{Z} \rightarrow \mathbb{R}$. Then extract a subsequence (f_{n_1}) of (f_n) such that $f_{n_1,1}$ converges uniformly on every compact subset of $\mathcal{Z}$. Then extract a subsequence of (f_{n_1}) such that the same holds for $f_{n_1,2}$, and so on. The result is a subsequence $(f_{n'})$ such that each $f_{n',i}$ converges uniformly on compact subsets of $\mathcal{Z}$, from which the same holds for $f_{n'}$ also by the triangle inequality. $\square$

B.3 Pushforward Maps Require Unbounded Bi-Lipschitz Constants

Theorem 2.1. *Suppose P_Z and P_X^* are probability measures on $\mathbb{R}^{d_Z}$ and $\mathbb{R}^{d_X}$ respectively, and that $\text{supp } P_Z \not\cong \text{supp } P_X^*$. Then for any sequence of measurable $f_n : \mathbb{R}^{d_Z} \rightarrow \mathbb{R}^{d_X}$, we can have $f_n \# P_Z \xrightarrow{\mathcal{D}} P_X^*$ only if*

$$\lim_{n \rightarrow \infty} \text{BiLip } f_n = \infty.$$

Proof. We suppose that $f_n \# P_Z \xrightarrow{\mathcal{D}} P_X^*$ and prove the contrapositive. That is, without loss of generality (pass to a subsequence if necessary) we assume

$$M := \sup_n \text{BiLip } f_n < \infty, \tag{B.4}$$

and prove that $\text{supp } P_Z \cong \text{supp } P_X^*$.

We first show that (f_n) is pointwise bounded. To this end, observe that Prokhorov's theorem ([Dudley, 2002, Proposition 9.3.4](#)) means that P_Z is tight and that the sequence $(f_n \# P_Z)$ is uniformly tight. As such, there exists compact $K \subseteq \mathbb{R}^{d_Z}$ such that $P_Z(K) > 0$, and compact $K' \subseteq \mathbb{R}^{d_X}$ such that

$$\inf_n f_n \# P_Z(K') > 1 - P_Z(K).$$

For each n , we must then have some $z_n \in K$ such that $f_n(z_n) \in K'$; otherwise $K' \subseteq f_n(K)^c$ and so

$$\begin{aligned} f_n \# P_Z(K') &\leq f_n \# P_Z(f_n(K)^c) \\ &= 1 - f_n \# P_Z(f_n(K)) \\ &= 1 - P_Z(f_n^{-1}(f_n(K))) \\ &= 1 - P_Z(K) \end{aligned}$$

since f_n is injective by [Proposition B.5](#). But for any fixed $z \in \mathbb{R}^{d_Z}$, this entails

$$\begin{aligned} \sup_n \|f_n(z)\| &\leq \sup_n \|f_n(z_n)\| + \|f_n(z) - f_n(z_n)\| \\ &\leq \sup_{x \in K'} \|x\| + \sup_{z \in K} M \|z - z_n\| \\ &\leq \sup_{x \in K'} \|x\| + 2M \sup_{z \in K} \|z\| \\ &< \infty \end{aligned}$$

since K and K' are compact.

Next, observe that (B.4) easily means (f_n) is uniformly equicontinuous. In particular, for $\epsilon > 0$, choosing $\delta := \epsilon/M$ gives

$$\|f_n(z) - f_n(z')\| \leq M \|z - z'\| < \epsilon$$

for all n whenever $\|z - z'\| < \delta$

[Theorem B.9](#) now entails the existence of a subsequence $(f_{n'})$ that converges uniformly on every compact subset of $\mathbb{R}^{d_Z}$. In particular, $(f_{n'})$ converges pointwise to a limit that we denote by f_∞ . Moreover, f_∞ is bi-Lipschitz. To see this, recall that for all n' and $z, z' \in \mathbb{R}^{d_Z}$ we have

$$\frac{1}{M} \|z - z'\| \leq \|f_{n'}(z) - f_{n'}(z')\| \leq M \|z - z'\|,$$

by our assumption (B.4). Taking $n' \rightarrow \infty$ shows that $\text{BiLip } f_\infty \leq M < \infty$.

We also have that

$$f_{n'} \# P_Z \xrightarrow{\mathcal{D}} f_\infty \# P_Z. \quad (\text{B.5})$$

This follows from the Portmanteau theorem ([Dudley, 2002](#), Theorem 11.3.3). In particular, suppose h is a bounded Lipschitz function, and let $B_r \subseteq \mathbb{R}^{d_Z}$ denote a ball of radius $r > 0$ at the origin. Then

$$\begin{aligned} \left| \int h(x) f_{n'} \# P_Z(dx) - \int h(x) f_\infty \# P_Z(dx) \right| &= \left| \int h(f_{n'}(z)) - h(f_\infty(z)) P_Z(dz) \right| \\ &\leq \int_{B_r} |h(f_{n'}(z)) - h(f_\infty(z))| P_Z(dz) \\ &\quad + \int_{B_r^c} |h(f_{n'}(z))| + |h(f_\infty(z))| P_Z(dz) \\ &\leq P_Z(B_r) (\text{Lip } h) \sup_{z \in B_r} \|f_n(z) - f_\infty(z)\| + 2P_Z(B_r^c) \sup_{z \in \mathbb{R}^{d_Z}} |h(z)|. \end{aligned}$$

Hence

$$\limsup_{n' \rightarrow \infty} \left| \int h(x) f_{n'} \# P_Z(dx) - \int h(x) f_\infty \# P_Z(dx) \right| \leq 2P_Z(B_r^c) \sup_{z \in \mathbb{R}^{d_Z}} |h(z)|$$

by the uniform convergence of $f_{n'}$ to f_∞ on compact subsets, and since $\text{Lip } h < \infty$. Taking $r \rightarrow \infty$, the right-hand side vanishes since h is bounded, and we obtain (B.5).

We are now ready to complete the proof. Since f_∞ is bi-Lipschitz, [Proposition B.5](#) means that f_∞ is a homeomorphism from $\mathbb{R}^{d_Z}$ to $f_\infty(\mathbb{R}^{d_Z})$. This certainly gives

$$\text{supp } P_Z \cong f_\infty(\text{supp } P_Z).$$

But now [Proposition B.6](#) means

$$f_\infty(\text{supp } P_Z) = \overline{f_\infty(\text{supp } P_Z)}$$

where the closure is taken in $\mathbb{R}^{d_X}$. However, from (B.5) we have

$$P_X^* = f_\infty \# P_Z,$$

which by [Proposition B.3](#) means that

$$\text{supp } P_X^\star = \text{supp } f_\infty \# P_Z = \overline{f_\infty(\text{supp } P_Z)}.$$

Consequently

$$\text{supp } P_X^\star = f_\infty(\text{supp } P_Z) \cong \text{supp } P_Z$$

as desired. $\square$

The following corollary extends the above result to the case where $\text{supp } P_X^\star$ may be homeomorphic to $\text{supp } P_Z$, but $P_X^\star$ is very *close* to a probability measure with non-homeomorphic support to P_Z . Here ρ denotes any metric for the weak topology. In other words, ρ must be a metric on the space of distributions that satisfies $\rho(P_n, P) \rightarrow 0$ as $n \rightarrow \infty$ if and only if $P_n \xrightarrow{\mathcal{D}} P$. The Lévy-Prokhorov and bounded Lipschitz metrics provide standard examples of such ρ ([Villani, 2008](#), Definition 3.3.10).

Corollary 2.2. *Suppose P_Z and P_X^0 are probability measures on $\mathbb{R}^{d_Z}$ and $\mathbb{R}^{d_X}$ respectively with $\text{supp } P_Z \not\cong \text{supp } P_X^0$. Then there exists nonincreasing $M : [0, \infty) \rightarrow [1, \infty]$ with $M(\epsilon) \rightarrow \infty$ as $\epsilon \rightarrow 0$ such that, for any probability measure $P_X^\star$ on $\mathbb{R}^{d_X}$, we have $\text{BiLip } f \geq M(\epsilon)$ whenever $\rho(P_X^\star, P_X^0) \leq \epsilon$ and $\rho(f \# P_Z, P_X^\star) \leq \epsilon$.*

Proof. Define $M : [0, \infty) \rightarrow [1, \infty]$ by

$$M(\epsilon) := \inf \{ \text{BiLip } f \mid f : \mathbb{R}^{d_Z} \rightarrow \mathbb{R}^{d_X}, \rho(f \# P_Z, P_X^0) \leq 2\epsilon \},$$

with $M(\epsilon) := \infty$ if the infimum is taken over the empty set. Certainly M is nonincreasing. If we have both $\rho(P_X^\star, P_X^0) \leq \epsilon$ and $\rho(f \# P_Z, P_X^\star) \leq \epsilon$, then the triangle inequality gives

$$\rho(f \# P_Z, P_X^0) \leq \rho(f \# P_Z, P_X^\star) + \rho(P_X^\star, P_X^0) \leq 2\epsilon$$

and so $\text{BiLip } f \geq M(\epsilon)$ since the right-hand side is an infimum. It remains only to show that $M(\epsilon) \rightarrow \infty$ as $\epsilon \rightarrow 0$. For contradiction, suppose there exists $\epsilon_n \rightarrow 0$ such that $\sup_n M(\epsilon_n) < \infty$. From the definition of M , this means that for each n there exists $f_n : \mathbb{R}^{d_Z} \rightarrow \mathbb{R}^{d_X}$ such that $\rho(f_n \# P_Z, P_X^0) \leq 2\epsilon_n$ and $\text{BiLip } f_n \leq M(\epsilon_n) + 1$. It follows directly that $\rho(f_n \# P_Z, P_X^0) \rightarrow 0$ as $n \rightarrow \infty$, which in turn means $f_n \# P_Z \xrightarrow{\mathcal{D}} P_X^0$ since ρ is a metric for the weak topology. At the same time we have

$$\sup_n \text{BiLip } f_n \leq \sup_n M(\epsilon_n) + 1 < \infty,$$

which contradicts [Theorem 2.1](#), since we assumed $\text{supp } P_Z \not\cong \text{supp } P_X^0$. $\square$

B.4 Variance of the Russian Roulette Estimator

In this section we briefly review the Russian roulette estimator used in [Chen et al. \(2019\)](#), and then discuss some scenarios in which we expect the variance of this estimator to increase unboundedly.

B.4.1 RUSSIAN ROULETTE ESTIMATOR

Residual Flows ([ResFlows](#), ([Chen et al., 2019](#))), building off of Invertible Residual Networks (iResNets, ([Behrmann et al., 2019](#))), model the data by repeatedly stacking bijections of the form $f_\ell^{-1}(x) = x + g_\ell(x)$, where $\text{Lip } g_\ell =: \kappa < 1$, as mentioned in (5). The change-of-variable formula for one layer of flow reads as, for $x \in \mathbb{R}^d$,

$$\log p_X(x) = \log p_Z(f_\ell^{-1}(x)) + \text{tr} \left(\sum_{j=1}^{\infty} \frac{(-1)^{j+1}}{j} \text{D}g_\ell(x)^j \right). \quad (\text{B.6})$$

To deal with this infinite series, iResNets truncate after a fixed number of terms – this provides a biased estimate of the log-likelihood of a point x under the model. ResFlows rely on an alternative method of estimating (B.6), first using a Russian roulette procedure to rewrite the series as follows:

$$\sum_{j=1}^{\infty} \frac{(-1)^{j+1}}{j} \text{tr} (\text{D}g_\ell(x)^j) = \mathbb{E}_N \left[\sum_{j=1}^N \frac{(-1)^{j+1}}{j} \frac{\text{tr} (\text{D}g_\ell(x)^j)}{p_j} \right] =: S(x),$$

where $N \sim \text{Geom}(p)$ is a geometric random variable, and $p_k := \mathbb{P}(N \geq k)$. Then, taking a single sample $N \sim \text{Geom}(p)$, an unbiased estimator of S is given as S_N , where S_n is defined for any $n \in \mathbb{N}$ and $x \in \mathbb{R}$ as

$$S_n(x) := \sum_{j=1}^n \frac{(-1)^{j+1}}{j} \frac{\text{tr}(\text{D}g_\ell(x)^j)}{p_j} \quad (\text{B.7})$$

for any $x \in \mathbb{R}^d$. We will study the variance of S_N in this section.¹⁵

First, however, define the quantity $\alpha_j(x)$ for $j \in \mathbb{N}$, $x \in \mathbb{R}^d$ as

$$\alpha_j(x) := \frac{(-1)^{j+1}}{j} \text{tr}(\text{D}g(x)^j), \quad (\text{B.8})$$

where we now drop the dependence of g on ℓ . Then, $S(x) = \sum_{j=1}^{\infty} \alpha_j(x)$, and $S_N(x) = \sum_{j=1}^N \alpha_j(x)/p_j$.

B.4.2 WHAT MIGHT HAPPEN WHEN $\kappa \rightarrow 1$?

We begin with an informal discussion on the variance of S_N as $\kappa \rightarrow 1$. First of all we know that, as $\kappa \rightarrow 1$, the mapping f^{-1} gets arbitrarily close to a non-invertible mapping: consider e.g. $g(x) = -\kappa x$, then $f^{-1} = (1 - \kappa)\text{Id} \rightarrow 0$ as $\kappa \rightarrow 1$. This near non-invertibility has implications for the speed of convergence of both $S(x)$ and its gradient,¹⁶ as noted in these two results from [Behrmann et al. \(2019\)](#):

1. **Theorem 3:** $\left| \sum_{j=1}^n \alpha_j(x) - \log \det(I + \text{D}g(x)) \right| \leq -d \left(\log(1 - \kappa) + \sum_{j=1}^n \frac{\kappa^j}{j} \right),$
2. **Theorem 4:** $\|\nabla_\theta (\alpha_j(x) - \log \det(I + \text{D}g(x)))\|_\infty = \mathcal{O}(\kappa^n).$

We can see that both bounds become very loose as $\kappa \rightarrow 1$, implying we cannot guarantee the fast convergence of either series. It then follows that we cannot invoke the results from [Rhee & Glynn \(2015\)](#) and [Beatson & Adams \(2019\)](#) to argue that the variance of the Russian roulette estimator S_N will be small. Indeed, in the next section, we will look at a specific example where this variance becomes *infinite*.

B.4.3 A SPECIFIC EXAMPLE OF INFINITE VARIANCE

Now consider the case where $d = 1$. We will show that when $\kappa^2 > 1 - p$, there is a set of x having positive Lebesgue measure such that $S_N(x)$ from (B.7) has infinite variance.

We note that here we have $\text{tr}(\text{D}g(x)^j) = (g'(x))^j$ for any $j \in \mathbb{N}$. We can thus rewrite α_j from (B.8) as

$$\alpha_j(x) := \frac{(-1)^{j+1}}{j} (g'(x))^j. \quad (\text{B.9})$$

Also recall that $N \sim \text{Geom}(p)$ and $p_j := \mathbb{P}(N \geq j)$ for all $j \in \mathbb{N}$.

Proposition B.10. *For any $x \in \mathbb{R}$ and random variable N satisfying $\text{supp } N = \mathbb{N}$, $S_N(x)$ has finite expectation if $\kappa < 1$.*

Proof. Refer to [Lyne et al. \(2015, Proposition A.1\)](#). □

Proposition B.11. *Under the same conditions as [Proposition B.10](#),*

$$\text{Var} S_N(x) \geq \lim_{n \rightarrow \infty} 2 \sum_{j=1}^n \alpha_j(x) S_{j-1}(x) - \mathbb{E}[S_N(x)]^2.$$

¹⁵[Chen et al. \(2019\)](#) additionally approximate $\text{tr}(\text{D}g_\ell(x)^j)$ by the Hutchinson's trace estimator $v^T \text{D}g_\ell(x)^j v$ for $v \sim \mathcal{N}(0, I)$. Since v is independent of N , their estimator has strictly higher variance than (B.7).

¹⁶With respect to the flow parameters θ

Proof. This proof is taken from [Lyne et al. \(2015, Proposition A.2\)](#); we mostly rewrite the proof but adapt it to our specific setting and notation. Note that we will drop the dependence of S_j and α_j on x throughout the proof.

We know from [Proposition B.10](#) that $\mathbb{E}[S_N(x)]$ is finite. Thus we will simply lower-bound $\mathbb{E}[S_N(x)^2]$.

We will first use induction to show the following holds for any $n \in \mathbb{N}$:

$$\sum_{j=1}^n S_j^2(p_j - p_{j+1}) = \alpha_1^2 + \sum_{j=2}^n \frac{\alpha_j^2}{p_j} + 2 \sum_{j=2}^n \alpha_j S_{j-1} - S_n^2 p_{n+1}. \quad (\text{B.10})$$

The base case is

$$S_1^2(p_1 - p_2) = \frac{\alpha_1^2}{p_1^2} p_1 - S_1^2 p_2 = \alpha_1^2 - S_1^2 p_2$$

since $p_1 = 1$. Now, assume (B.10) holds for some $m \in \mathbb{N}$. Then, for $n = m + 1$,

$$\begin{aligned} \sum_{j=1}^{m+1} S_j^2(p_j - p_{j+1}) &= \sum_{j=1}^m S_j^2(p_j - p_{j+1}) + S_{m+1}^2(p_{m+1} - p_{m+2}) \\ &= \alpha_1^2 + \sum_{j=2}^m \frac{\alpha_j^2}{p_j} + 2 \sum_{j=2}^m \alpha_j S_{j-1} - S_m^2 p_{m+1} \\ &\quad + S_{m+1}^2(p_{m+1} - p_{m+2}) \end{aligned} \quad (\text{B.11})$$

by the inductive hypothesis. We also have

$$\begin{aligned} p_{m+1}(S_m^2 - S_{m+1}^2) &= p_{m+1}(S_m - S_{m+1})(S_m + S_{m+1}) \\ &= p_{m+1} \frac{\alpha_{m+1}}{p_{m+1}} \left(2S_m + \frac{\alpha_{m+1}}{p_{m+1}} \right) \\ &= \frac{\alpha_{m+1}^2}{p_{m+1}} + 2\alpha_{m+1} S_m. \end{aligned}$$

Substituting this result into (B.11) completes the induction and proves (B.10) for all $n \in \mathbb{N}$.

Now, by Jensen's inequality,

$$S_n^2 = \left(\sum_{j=1}^n \frac{p_j \frac{\alpha_j}{p_j}}{p_j} \right)^2 \leq \frac{\sum_{j=1}^n \frac{\alpha_j^2}{p_j}}{\sum_{j=1}^n p_j}.$$

This implies

$$p_{n+1} S_n^2 \leq p_n S_n^2 \leq \frac{p_n}{\sum_{j=1}^n p_j} \sum_{j=1}^n \frac{\alpha_j^2}{p_j} \leq \sum_{j=1}^n \frac{\alpha_j^2}{p_j}$$

since (p_n) is a positive sequence.

This finally implies the following lower bound for any $n \in \mathbb{N}$:

$$\begin{aligned}
 \sum_{j=1}^n S_j^2 \mathbb{P}(N = j) &= \sum_{j=1}^n S_j^2 (p_j - p_{j+1}) \\
 &= \alpha_1^2 + \sum_{j=2}^n \frac{\alpha_j^2}{p_j} + 2 \sum_{j=2}^n \alpha_j S_{j-1} - S_n^2 p_{n+1} \\
 &\geq \alpha_1^2 + \sum_{j=2}^n \frac{\alpha_j^2}{p_j} + 2 \sum_{j=2}^n \alpha_j S_{j-1} - \sum_{j=1}^n \frac{\alpha_j^2}{p_j} \\
 &= \alpha_1^2 (1 - p_1^{-1}) + 2 \sum_{j=2}^n \alpha_j S_{j-1} \\
 &= 2 \sum_{j=2}^n \alpha_j S_{j-1},
 \end{aligned}$$

where the final line follows because $p_1 = 1$.

Since $\mathbb{E}[S_N^2] = \lim_{n \rightarrow \infty} \sum_{j=1}^n S_j^2 \mathbb{P}(N = j)$, the proof is complete. $\square$

We are about ready to prove the main result but require one more auxiliary result first.

Proposition B.12. *Suppose $|b| > 1$. Then,*

$$\lim_{n \rightarrow \infty} \frac{n}{b^n} \sum_{j=1}^{n-1} \frac{b^j}{j} = \frac{1}{b-1}.$$

Proof. We will first show that the limit exists, and then show that it equals $(b-1)^{-1}$. Let

$$c_n = \frac{n}{b^n} \sum_{j=1}^{n-1} \frac{b^j}{j}.$$

We can rewrite this as follows:

$$c_n = \sum_{j=1}^{n-1} \frac{n}{b^{n-j} j} = \sum_{j=1}^{n-1} \frac{n}{b^j (n-j)} = \sum_{j=1}^{n-1} \frac{1}{b^j} + \sum_{j=1}^{n-1} \frac{j}{b^j (n-j)}.$$

Since $b > 1$, the first sum is a convergent geometric series as $n \rightarrow \infty$. We can decompose the second sum into its positive and negative terms:

$$\sum_{j=1}^{n-1} \frac{j}{b^j (n-j)} = \sum_{j \geq 1: b^j > 0} \frac{j}{b^j (n-j)} + \sum_{j \geq 1: b^j < 0} \frac{j}{b^j (n-j)} \equiv \textcircled{1}_n + \textcircled{2}_n.$$

We can see, for all $n \in \mathbb{N}$,

$$\textcircled{1}_n \geq - \sum_{j=1}^{n-1} \frac{j}{|b|^j (n-j)} \quad \text{and} \quad \textcircled{2}_n \leq \sum_{j=1}^{n-1} \frac{j}{|b|^j (n-j)}.$$

Furthermore, for all $j \in \{1, \dots, n-1\}$, we have

$$\frac{j}{n-j} \leq j.$$

Now notice that the series $\sum_{j=1}^{\infty} \frac{j}{|b|^j}$ converges by the ratio test:

$$\lim_{j \rightarrow \infty} \left| \frac{\frac{j+1}{|b|^{j+1}}}{\frac{j}{|b|^j}} \right| = \lim_{j \rightarrow \infty} \frac{j+1}{j|b|} = \frac{1}{|b|} < 1.$$

This implies the existence of $\lim_{n \rightarrow \infty} \sum_{j=1}^{n-1} \frac{j}{|b|^j(n-j)}$. Since the sequence $(\textcircled{1}_n)$ (resp. $(\textcircled{2}_n)$) is negative, non-increasing, and bounded below (resp. positive, non-decreasing, and bounded above), this implies the existence of $\lim_{n \rightarrow \infty} \textcircled{1}_n$ (resp. $\lim_{n \rightarrow \infty} \textcircled{2}_n$). Altogether, this implies the existence of

$$\lim_{n \rightarrow \infty} \left(\sum_{j=1}^{n-1} \frac{1}{b^j} + \sum_{j=1}^{n-1} \frac{j}{b^j(n-j)} \right) = \lim_{n \rightarrow \infty} c_n =: c_\infty.$$

Now we will determine its precise value. Note the following recurrence for all $n \in \mathbb{N}$:

$$c_{n+1} = \frac{n+1}{bn} (1 + c_n).$$

Taking the limit of both sides as $n \rightarrow \infty$ gives

$$c_\infty = \frac{1}{b} (1 + c_\infty).$$

Solving this gives us $c_\infty = \frac{1}{b-1}$, which completes the proof. $\square$

Proposition B.13. *Suppose $N \sim \text{Geom}(p)$, g is continuously differentiable, and $1 - p < \kappa^2 < 1$. Then*

$$\{x \in \mathbb{R} \mid \text{Var} S_N(x) = \infty\}$$

has positive Lebesgue measure.

Proof. From [Proposition B.11](#), for a given $x \in \mathbb{R}$, we can see that showing $\sum_{n=2}^{\infty} \alpha_n(x) S_{n-1}(x)$ diverges is sufficient to prove $\text{Var} S_N(x)$ is infinite.

Consider using the ratio test to assess the convergence of the above series, with terms defined as $a_n(x) := \alpha_n(x) S_{n-1}(x)$. We have the following for any $n \geq 2$:

$$\begin{aligned} \left| \frac{a_{n+1}(x)}{a_n(x)} \right| &= \left| \frac{\alpha_{n+1}(x) S_n(x)}{\alpha_n(x) S_{n-1}(x)} \right| \\ &= \frac{\frac{|(g'(x))^{n+1}|}{n+1}}{\frac{|(g'(x))^n|}{n}} \cdot \left| \frac{\sum_{j=1}^n \frac{\alpha_j(x)}{p_j}}{\sum_{j=1}^{n-1} \frac{\alpha_j(x)}{p_j}} \right| \\ &= \frac{n|g'(x)|}{n+1} \cdot \left| \frac{(-1)^{n+1} \cdot (g'(x))^n}{np_n} \left(\sum_{j=1}^{n-1} \frac{(-1)^{j+1} \cdot (g'(x))^j}{jp_j} \right)^{-1} + 1 \right|. \end{aligned}$$

Recall $p_j = (1-p)^{j-1} \equiv q^{j-1}$. Then, writing $b = -\frac{g'(x)}{q}$, we have

$$\frac{(-1)^{n+1} \cdot (g'(x))^n}{np_n} \left(\sum_{j=1}^{n-1} \frac{(-1)^{j+1} \cdot (g'(x))^j}{jp_j} \right)^{-1} = \frac{1}{n} b^n \left(\sum_{j=1}^{n-1} \frac{1}{j} b^j \right)^{-1}.$$

Now let us assume that $|g'(x)|^2 > q$. We can see that $|g'(x)|^2 > q \implies |g'(x)| > q$ since $q \in (0, 1)$, which then entails $|b| > 1$. Therefore, by [Proposition B.12](#),

$$\lim_{n \rightarrow \infty} \frac{n}{b^n} \sum_{j=1}^{n-1} \frac{b^j}{j} = \frac{1}{b-1}.$$

This then implies

$$\begin{aligned} \lim_{n \rightarrow \infty} \left| \frac{a_{n+1}(x)}{a_n(x)} \right| &= \lim_{n \rightarrow \infty} \frac{n|g'(x)|}{n+1} \left| \frac{1}{\frac{n}{b^n} \sum_{j=1}^{n-1} \frac{b^j}{j}} + 1 \right| \\ &= |g'(x)| \left| \frac{1}{\frac{1}{b-1}} + 1 \right| = \frac{|g'(x)|^2}{q} > 1 \end{aligned}$$

since we have assumed that $|g'(x)|^2 > q$. Thus, for all x in the set

$$V_{g,q} := \{x \in \mathbb{R} \mid |g'(x)|^2 > q\},$$

the series $\sum_{n=2}^{\infty} \alpha_n(x) S_{n-1}(x)$ diverges by the ratio test. This means that $\text{Var} S_N(x) = \infty$ for all $x \in V_{g,q}$.

Finally, we will prove the set $\{x \in \mathbb{R} \mid \text{Var} S_N(x) = \infty\}$ has positive Lebesgue measure. Recall that $\text{Lip } g = \kappa$, which directly implies $\sup_{x \in \mathbb{R}} |g'(x)| = \kappa$ from [Proposition B.7](#) and thus $\sup_{x \in \mathbb{R}} |g'(x)|^2 = \kappa^2$. Then, since $\kappa^2 > q$, there exists $x_0 \in \mathbb{R}$ such that $|g'(x_0)|^2 \in (q, \kappa^2)$. By the continuity of $|g'|$, there is open ball of nonzero radius around x_0 , denoted $\mathcal{B}(x_0)$, such that $|g'(x)|^2 > q$ for all $x \in \mathcal{B}(x_0)$. Since $\mathcal{B}(x_0)$ is open and non-empty, it has positive Lebesgue measure. The inclusions

$$\mathcal{B}(x_0) \subseteq V_{g,q} \subseteq \{x \in \mathbb{R} \mid \text{Var} S_N(x) = \infty\}$$

thus conclude the proof. □

B.4.4 DISCUSSION

Changing p as κ increases An obvious strategy to avoid satisfying the conditions of [Proposition B.13](#) is to set p such that $1 - \kappa^2 > p$. However, lowering p in this way incurs additional computational cost: the average number of iterations per training step is equal to p^{-1} , or is lower-bounded by $(1 - \kappa^2)^{-1}$ if $p < 1 - \kappa^2$. Thus, if we send $\kappa \rightarrow 1$ to mitigate the bi-Lipschitz constraint (6), we will either incur an infinite computational cost or run the risk of encountering infinite variance.

Higher dimensions Although [Proposition B.13](#) only applies for $d = 1$, it is conceivable that similar results can be derived for $d > 1$, especially when considering the discussion in [Section B.4.2](#). We leave a deeper investigation for future work.

B.5 Density of a CIF

We make precise our heuristic derivation of the density (11) via the following result.

Proposition B.14. *Suppose $\mathcal{Z}, \mathcal{X} \subseteq \mathbb{R}^d$ are open, and that $F(\cdot; u) : \mathcal{Z} \rightarrow \mathcal{X}$ is a continuously differentiable bijection with everywhere invertible Jacobian for each $u \in \mathcal{U}$. Under the generative model (8), (X, U) has joint density*

$$p_Z(F^{-1}(x; u)) p_{U|Z}(u|F^{-1}(x; u)) |\det DF^{-1}(x; u)|.$$

Proof. Suppose $h : \mathcal{X} \times \mathcal{U} \rightarrow \mathbb{R}$ is a bounded measurable test function. Then

$$\begin{aligned} \mathbb{E}[h(X, U)] &= \mathbb{E}[h(F(Z; U), U)] \\ &= \int \left[\int h(F(z; u), u) p_Z(z) p_{U|Z}(u|z) dz \right] du \\ &= \int h(x, u) p_Z(F^{-1}(x; u)) p_{U|Z}(u|F^{-1}(x; u)) |\det DF^{-1}(x; u)| dz du, \end{aligned}$$

where in the third line we substitute $x := F(z; u)$ on the inner integral, which is valid by Theorem 17.2 of [Billingsley \(2008\)](#). Now for $A \subseteq \mathcal{X} \times \mathcal{U}$, let $h := \mathbb{I}_A$. It follows that

$$\mathbb{P}((X, U) \in A) = \mathbb{E}[\mathbb{I}_A(X, U)] = \int_A p_Z(F^{-1}(x; u)) p_{U|Z}(u|F^{-1}(x; u)) |\det DF^{-1}(x; u)| dz du,$$

which gives the result since A was arbitrary. □

B.6 Our Approximate Posterior Does Not Sacrifice Generality

The following result shows that our parameterisation of the approximate posterior $q_{U_{1:L}|X}$ in (15) does not lose generality. In particular, provided each $q_{U_\ell|Z_\ell}$ is sufficiently expressive, we can always recover the exact posterior.

Proposition B.15. *Under the generative model (8), the posterior factors like*

$$p_{U_{1:L}|X}(u_{1:L}|x) = \prod_{\ell=1}^L p_{U_\ell|Z_\ell}(u_\ell|z_\ell),$$

where $z_L := x$ and $z_\ell := F_{\ell+1}^{-1}(z_{\ell+1}; u_{\ell+1})$ for $\ell \in \{1, \dots, L-1\}$.

Proof. Writing $p_{U_{1:L}|X}$ autoregressively gives

$$p_{U_{1:L}|X}(u_{1:L}|x) = \prod_{\ell=1}^L p_{U_\ell|U_{\ell+1:L}, X}(u_\ell|u_{\ell+1:L}, x).$$

But now it is clear from the generative model (8) that U_ℓ is conditionally independent of $(U_{\ell+1:L}, X)$ given Z_ℓ , and as such

$$p_{U_\ell|U_{\ell+1:L}, X}(u_\ell|u_{\ell+1:L}, x) = p_{U_\ell|Z_\ell}(u_\ell|z_\ell).$$

Substituting this into the above expression then gives the result. $\square$

B.7 Conditions for a CIF to Outperform an Underlying Normalising Flow

For this result, the components of our model are assumed to be parameterised by $\theta \in \Theta$, which we will indicate by F_θ , $p_{U|Z}^\theta$, and $q_{U|X}^\theta$. We will also use θ to indicate quantities that result from the choice of parameters θ (e.g. P_X^θ for the distribution obtained), and will denote by $\mathcal{L}^\theta$ the corresponding ELBO (14).

Proposition 4.1. *Suppose there exists $\phi \in \Theta$ such that, for some bijection $f : \mathcal{Z} \rightarrow \mathcal{X}$, $F_\phi(\cdot; u) = f(\cdot)$ for all $u \in \mathcal{U}$. Likewise, suppose $p_{U|Z}^\phi$ and $q_{U|X}^\phi$ are such that, for some density r on $\mathcal{U}$, $p_{U|Z}^\phi(\cdot|z) = q_{U|X}^\phi(\cdot|x) = r(\cdot)$ for all $z \in \mathcal{Z}$ and $x \in \mathcal{X}$. If $\mathbb{E}_{x \sim P_X^*}[\mathcal{L}^\theta(x)] \geq \mathbb{E}_{x \sim P_X^*}[\mathcal{L}^\phi(x)]$, then*

$$D_{\text{KL}}(P_X^* \parallel P_X^\theta) \leq D_{\text{KL}}(P_X^* \parallel f \# P_Z).$$

Proof. Observe from (11) that

$$p_{X,U}^\phi(x, u) = p_Z(f^{-1}(x)) |\det Df^{-1}(x)| p_{U|Z}(u|f^{-1}(x)).$$

It then follows from (13) that, under ϕ , the model has density

$$p_X^\phi(x) = p_Z(f^{-1}(x)) |\det Df^{-1}(x)| \underbrace{\int p_{U|Z}(u|f^{-1}(x)) du}_{=1}$$

which is exactly the density of the normalising flow $f \# P_Z$. We also obtain the posterior

$$\begin{aligned} p_{U|X}^\phi(u|x) &= \frac{p_{X,U}^\phi(x, u)}{p_X^\phi(x)} \\ &= p_{U|Z}(u|f^{-1}(x)) \\ &= r(u). \end{aligned}$$

Since each $q_{U|X}^\phi(\cdot|x) = r(\cdot)$ also, it follows that $\mathcal{L}^\phi$ is tight, so that $\mathcal{L}^\phi(x) = \log p_X^\phi(x)$ for all $x \in \mathcal{X}$.

Now suppose some $\theta \in \Theta$ has

$$\mathbb{E}_{x \sim P_X^*}[\mathcal{L}^\theta(x)] \geq \mathbb{E}_{x \sim P_X^*}[\mathcal{L}^\phi(x)].$$

It follows that

$$\mathbb{E}_{x \sim P_X^*}[\log p_X^\theta(x)] \geq \mathbb{E}_{x \sim P_X^*}[\mathcal{L}^\phi(x)] = \mathbb{E}_{x \sim P_X^*}[\log p_X^\phi(x)].$$

Subtracting $\mathbb{E}_{x \sim P_X^*}[\log p_X^*(x)]$ from both sides and negating gives

$$D_{\text{KL}}(P_X^* \parallel P_X^\theta) \leq D_{\text{KL}}(P_X^* \parallel P_X^\phi) = D_{\text{KL}}(P_X^* \parallel f \# P_Z).$$

$\square$

B.8 CIFs Can Learn Target Supports Exactly

In this section we give necessary and sufficient conditions for a CIF to learn the support of a target distribution exactly, without needing changes to F . However, our argument applies more generally and does not make specific use of the bijective structure of F . To make this clear, we formulate our result here in terms of a generalisation of the model (7). In particular, we will take P_X as the marginal in X of

$$Z \sim P_Z, \quad U \sim P_{U|Z}(\cdot|Z), \quad X := G(Z, U), \quad (\text{B.12})$$

where $G : \mathcal{Z} \times \mathcal{U} \rightarrow \mathcal{X}$. We will assume that

- $\mathcal{Z} \subseteq \mathbb{R}^{d_Z}$, $\mathcal{U} \subseteq \mathbb{R}^{d_U}$, and $\mathcal{X} \subseteq \mathbb{R}^{d_X}$ are equipped with the subspace topology;
- P_Z and each $P_{U|Z}(\cdot|z)$ are Borel probability measures on $\mathcal{Z}$ and $\mathcal{U}$ respectively;
- G is continuous with respect to the product topology $\mathcal{Z} \times \mathcal{U}$.

We then have the following formula for $\text{supp } P_X$:

Lemma B.16. *Under the model (B.12),*

$$\text{supp } P_X = \overline{\bigcup_{z \in \text{supp } P_Z} G(\{z\} \times \text{supp } P_{U|Z}(\cdot|z))}.$$

Proof. Denote the joint distribution of (Z, U) by $P_{Z,U}$. Observe from Proposition B.3 that

$$\text{supp } P_X = \overline{G(\text{supp } P_{Z,U})}.$$

Let

$$B := \bigcup_{z \in \text{supp } Z} \{z\} \times \text{supp } P_{U|Z}(\cdot|z).$$

The result follows if we can show that

$$\text{supp } P_{Z,U} = \overline{B},$$

since $\overline{G(\overline{B})} = \overline{G(B)}$ because G is continuous.

We first show that $\text{supp } P_{Z,U} \supseteq \overline{B}$. Suppose $(z, u) \in B$, and let $N_{(z,u)} \subseteq \mathcal{Z} \times \mathcal{U}$ be an open set containing (z, u) . Then there exists open N_z and N_u containing z and u respectively such that $N_z \times N_u \subseteq N_{(z,u)}$, since the open rectangles form a base for the product topology. It follows that

$$\begin{aligned} P_{Z,U}(N_{(z,u)}) &\geq P_{Z,U}(N_z \times N_u) \\ &= \int_{N_z} P_{U|Z}(N_u|z') P_Z(dz') \\ &> 0, \end{aligned}$$

since by the definition of B we have $P_Z(N_z) > 0$ and $P_{U|Z}(N_u|z) > 0$ for each $u \in N_z$. From this we have $\text{supp } P_{Z,U} \supseteq B$, and taking the closure of each side gives $\text{supp } P_{Z,U} \supseteq \overline{B}$.

In the other direction, suppose that $(z, u) \notin \overline{B}$. Then there exist open sets N_z and N_u containing z and u respectively such that

$$(N_z \times N_u) \cap \overline{B} = \emptyset.$$

By the definition of B , it follows that if $(z', u') \in N_z \times N_u$ and $z' \in \text{supp } P_Z$, then $u' \notin \text{supp } P_{U|Z}(\cdot|z')$. Otherwise stated, if $z' \in N_z \cap \text{supp } P_Z$, then

$$N_u \cap \text{supp } P_{U|Z}(\cdot|z') = \emptyset.$$

Thus

$$\begin{aligned} P_{Z,U}(N_z \times N_u) &= \int_{N_z} \left[\int_{N_u} P_{U|Z}(du'|z') \right] P_Z(dz') \\ &= \int_{N_z \cap \text{supp } P_Z} \left[\int_{N_u \cap \text{supp } P_{U|Z}(\cdot|z')} P_{U|Z}(du'|z') \right] P_Z(dz') \\ &= 0, \end{aligned}$$

where the second line follows from [Proposition B.4](#). Consequently $(z, u) \notin \text{supp } P_{Z,U}$, which gives $\text{supp } P_{Z,U} \subseteq \overline{B}$. $\square$

We now give necessary and sufficient conditions for the model [\(B.12\)](#) to learn a given target support exactly.

Proposition B.17. *Suppose $P_X^*(\partial \text{supp } P_X^*) = 0$ and that*

$$\overline{G(\text{supp } P_Z \times \mathcal{U})} \supseteq \text{supp } P_X^*. \quad (\text{B.13})$$

Then there exists $P_{U|Z}$ such that $\text{supp } P_X = \text{supp } P_X^$ if and only if, for all $z \in \text{supp } P_Z$, there exists $u \in \mathcal{U}$ with*

$$G(z, u) \in \text{supp } P_X^*.$$

Proof. $(\Rightarrow)$ Choose $P_{U|Z}$ such that $\text{supp } P_X = \text{supp } P_X^*$. [Lemma B.16](#) gives

$$\bigcup_{z \in \text{supp } P_Z} G(\{z\} \times \text{supp } P_{U|Z}(\cdot|z)) \subseteq \text{supp } P_X^*.$$

Suppose $z \in \text{supp } P_Z$. Then for indeed all $u \in \text{supp } P_{U|Z}(\cdot|z)$ we must have $G(z, u) \in \text{supp } P_X^*$, which proves this direction since $\text{supp } P_{U|Z}(\cdot|z) \neq \emptyset$ by [Proposition B.4](#).

$(\Leftarrow)$ For $z \in \text{supp } P_Z$, let

$$A_z := \{u \in \mathcal{U} : G(z, u) \in \text{int}(\text{supp } P_X^*)\}, \quad (\text{B.14})$$

where int denotes the *interior* operator. If $A_z = \emptyset$, define $P_{U|Z}(\cdot|z)$ to be Dirac on some u such that $G(z, u) \in \text{supp } P_X^*$, which exists by assumption. Otherwise, we let $P_{U|Z}(\cdot|z)$ be a probability measure with support $\overline{A_z}$. To show that such a measure exists, observe that A_z is open since G is continuous. Since $\mathcal{U}$ is separable, we can therefore write

$$A_z = \bigcup_{n=1}^{\infty} B_n$$

for a countable collection of open sets $B_n \subseteq A_z$. We can then define a probability measure μ by

$$\mu(C) := \sum_{n=1}^{\infty} 2^{-n} \mathbb{I}(C \cap B_n \neq \emptyset)$$

for measurable $C \subseteq \mathcal{U}$. Since $A_z \neq \emptyset$, it is straightforward to see that this is a probability measure with $\mu(A_z) = 1$. Consequently $\text{supp } \mu = \overline{A_z}$ by [Proposition B.2](#), since A_z is open and $\text{supp } \mu$ is the smallest closed set with μ -probability 1.

We show this construction gives $\text{supp } P_X \subseteq \text{supp } P_X^*$. To this end, we first prove that if $z \in \text{supp } P_Z$ and $u \in \text{supp } P_{U|Z}(\cdot|z)$ then

$$G(z, u) \in \text{supp } P_X^*.$$

If $A_z = \emptyset$ this is immediate. Otherwise, since $\text{supp } P_{U|Z}(\cdot|z) = \overline{A_z}$, there exists $(u_n) \subseteq A_z$ such that $u_n \rightarrow u$. By [\(B.14\)](#), each $G(z, u_n) \in \text{supp } P_X^*$. By continuity we then have

$$G(z, u_n) \rightarrow G(z, u) \in \text{supp } P_X^*$$

since $\text{supp } P_X^*$ is closed. It follows that

$$\bigcup_{z \in \text{supp } P_Z} G(\{z\} \times \text{supp } P_{U|Z}(\cdot|z)) \subseteq \text{supp } P_X^*,$$

which gives $\text{supp } P_X \subseteq \text{supp } P_X^*$ from [Lemma B.16](#) since $\text{supp } P_X^*$ is closed.

We now show $\text{supp } P_X \supseteq \text{supp } P_X^*$. Since $P_X^*(\partial \text{supp } P_X^*) = 0$ we have

$$\text{supp } P_X^* = \overline{\text{int}(\text{supp } P_X^*)}$$

by [Proposition B.2](#), so that $\text{supp } P_X \supseteq \text{supp } P_X^*$ if $\text{supp } P_X \supseteq \text{int}(\text{supp } P_X^*)$. Now suppose $x \in \text{int}(\text{supp } P_X^*)$. Then there exists $(z_n) \subseteq \text{supp } P_Z$ and $(u_n) \subseteq \mathcal{U}$ such that $G(z_n, u_n) \rightarrow x$ by [\(B.13\)](#). But then we must have $G(z_n, u_n) \in \text{int}(\text{supp } P_X^*)$ for n large enough because x lies in the interior. Consequently, for n large enough,

$$u_n \in A_{z_n} \subseteq \text{supp } P_{U|Z}(\cdot|z_n)$$

and hence $G(z_n, u_n) \in \text{supp } P_X$ by [Lemma B.16](#). This means $x \in \text{supp } P_X$ since $\text{supp } P_X$ is closed. $\square$

The following proposition then gives a straightforward condition under which it is additionally possible to recover the *target* exactly (i.e. not just its support). In our experiments we do not enforce this condition explicitly. However, since we learn the parameters of G here, we can expect our model will approximate this behaviour if doing so produces a better density estimator.

Proposition B.18. *If $G(z, \cdot)$ is surjective for each $z \in \mathcal{Z}$, then there exists $P_{U|Z}$ such that $P_X = P_X^*$.*

Proof. Fix $z \in \mathcal{Z}$. Surjectivity of $G(z, \cdot)$ means that, for $x \in \mathcal{X}$, there exists $u \in \mathcal{U}$ such that $G(z, u) = x$. Thus we can define $H_z : \mathcal{X} \rightarrow \mathcal{U}$ such that

$$G(z, H_z(x)) = x$$

for all $x \in \mathcal{X}$. We then define each

$$P_{U|Z}(\cdot|z) := H_z \# P_X^*.$$

From this it follows that $P_X = P_X^*$. For, letting $B \subseteq \mathcal{X}$ be measurable,

$$\begin{aligned} P_X(B) &= \int_{G^{-1}(B)} P_{U|Z}(du|z) P_Z(dz) \\ &= \int \left[\int \mathbb{I}_B(G(z, u)) H_z \# P_X^*(du) \right] P_Z(dz) \\ &= \int \left[\int \mathbb{I}_B(G(z, H_z(x))) P_X^*(dx) \right] P_Z(dz) \\ &= \int \left[\int \mathbb{I}_B(x) P_X^*(dx) \right] P_Z(dz) \\ &= P_X^*(B), \end{aligned}$$

which gives the result. $\square$

C Experimental Details

Our choices [\(10\)](#) and [\(17\)](#) required parameterising s , t , μ^p , Σ^p , μ^q , and Σ^q . Since these terms are naturally paired, at each layer of our model we set

$$\begin{aligned} [s(u), t(u)] &:= \text{NN}_F(u), \\ [\mu^p(z), \varsigma^p(z)] &:= \text{NN}_p(z), \\ \Sigma^p(z) &:= \text{diag}(e^{\varsigma^p(z)}), \\ [\mu^q(x), \varsigma^q(x)] &:= \text{NN}_q(x), \\ \Sigma^q(x) &:= \text{diag}(e^{\varsigma^q(x)}), \end{aligned}$$

where NN denotes a separate neural network and $\varsigma^p(z), \varsigma^q(x) \in \mathbb{R}^d$.

In all experiments we trained our models to maximise either the log-likelihood (for the baseline flows) or the ELBO (for the CIFs) using the ADAM optimiser (Kingma & Ba, 2015) with default hyperparameters and no weight decay. The ELBO was estimated using a single sample per datapoint (i.e. a single call to Algorithm 1). We used a held-out validation set and trained each model until its validation score stopped improving, except for the NSF tabular data experiments where we train for a fixed number of epochs as specified in Durkan et al. (2019). After training, we used validation performance to select the best parameters found during training for use at test time (again except for the NSF experiments, where we just test with the final model). Both validation and test scores were computed using the exact log-likelihood for the baseline and the importance sampling estimate (16) for the CIFs, with $m = 5$ samples for validation and $m = 100$ for testing.

C.1 Tabular Data Experiments

Following Papamakarios et al. (2017), we experimented with the POWER, GAS, HEPMASS, and MINIBOONE datasets from the UCI repository (Bache & Lichman, 2013), as well as a dataset of 8×8 image patches extracted from the BSDS300 dataset (Martin et al., 2001). We preprocessed these datasets identically to Papamakarios et al. (2017), and used the same train/validation/test splits. For all CIF-ResFlow models, we used a batch size of 1000 and a learning rate of 10^{-3} . For the MAF experiments, we used a batch size of 1000 and a learning rate of 10^{-3} , except for BSDS300 where we used a learning rate of 10^{-4} to control the instability of the baseline. For the NSF experiments, we used batch sizes and learning rates as dictated by Durkan et al. (2019, Table 5), along with their cosine learning rate annealing scheme.

Also, for all CIF models, each U_ℓ had the same dimension d_U , which we took to be roughly a quarter of the dimensionality of the data (except in Section C.1.4 for which $d_U = d_X$). In particular, we set $d_U := 2$ for POWER and GAS, $d_U := 5$ for HEPMASS, $d_U := 10$ for MINIBOONE, and $d_U := 15$ for BSDS300.

C.1.1 RESIDUAL FLOWS

The residual blocks in all ResFlow models used multilayer perceptrons (MLPs) with 4 hidden layers of 128 hidden units (denoted 4×128), LipSwish nonlinearities (Chen et al., 2019, (10)) before each linear layer, and a residual connection from the input to the output. We did not use any kind of normalisation (e.g. ActNorm or BatchNorm) for these experiments. For all models we set $\kappa = 0.9$ in (5) to match the value for the 2-D experiments in the codebase of Chen et al. (2019). Other design choices followed Chen et al. (2019). In particular:

- We always exactly computed several terms at the beginning of the series expansion of the log Jacobian, and then used Russian Roulette sampling (Kahn, 1955) to estimate the sum of the remaining terms. In particular, at training time we computed 2 exact terms, while at test time we computed 20 exact terms;
- We used a geometric distribution with parameter 0.5 for the number of terms to compute in our Russian Roulette estimators;
- We used the Skilling-Hutchinson trace estimator (Skilling, 1989; Hutchinson, 1990) to estimate the trace in the log Jacobian term;
- At both training and test time, we used a single Monte Carlo sample of (n, v) to estimate (6) of Chen et al. (2019);

However, note that for these experiments, for the sake of simplicity, we did not use the memory-saving techniques in (8) and (9) of Chen et al. (2019), nor the adaptive power iteration scheme described in their Appendix E.

For NN_F , NN_p , and NN_q we used 2×10 MLPs with tanh nonlinearities. These networks were much smaller than 4×128 , and hence the CIF-ResFlows had only roughly 1.5-4.5% more parameters (depending on the dimension of the dataset) than the otherwise identical 10-layer ResFlows, and roughly 10% of the parameters of the 100-layer ResFlows.

The 100-layer ResFlows were significantly slower to train than the 10-layer models, and for POWER, GAS, and BSDS300 we were forced to stop these before their validation loss had converged. However, to ensure a fair comparison, we allocated more total computing power to these models than to the 10-layer models, which were terminated properly. In particular, we trained each 100-layer ResFlow on POWER and GAS for a total of 10 days on a single NVIDIA GeForce GTX 1080 Ti, and on BSDS300 for a total of 7 days. In contrast, the 10-layer ResFlows converged after around 1 day on POWER, 4.5 days on GAS, and around 3 days on BSDS300. Likewise, the 10-layer CIF-ResFlows converged after around 1 day on POWER, 6 days on GAS, and 2 days on BSDS300.

Table C.4: MAF and CIF-MAF parameter configurations for POWER and GAS.

	LAYERS (L)	AUTOREGRESSIVE NETWORK SIZE	NN_p SIZE	NN_q SIZE	NN_F SIZE
MAF	5, 10, 20	$2 \times 100, 2 \times 200, 2 \times 400$	-	-	-
CIF-MAF	5, 10	2×128	$2 \times 100, 2 \times 200$	$2 \times 100, 2 \times 200$	2×128

Table C.5: MAF and CIF-MAF parameter configurations for HEPMASS and MINIBOONE

	LAYERS (L)	AUTOREGRESSIVE NETWORK SIZE	NN_p SIZE	NN_q SIZE	NN_F SIZE
MAF	5, 10, 20	$2 \times 128, 2 \times 512, 2 \times 1024$	-	-	-
CIF-MAF	5, 10	2×128	$2 \times 128, 2 \times 512$	$2 \times 128, 2 \times 512$	2×128

C.1.2 MASKED AUTOREGRESSIVE FLOWS

The experiment comparing MAF baselines to CIF-MAFs was inspired by the experimental setup in [Papamakarios et al. \(2017\)](#). For each dataset, we specified a set of hyperparameters over which to search for both the baselines and the CIFs; these hyperparameters are provided in [Table C.4](#), [Table C.5](#), and [Table C.6](#). Then, we trained each model until no validation improvement had been observed for 50 epochs. We then evaluated the model with the best validation score among all candidate models on the test dataset to obtain a log-likelihood score. We performed this procedure with three separate random seeds, and report the average and standard error across the runs in [Table 1](#).

We searched over all combinations of parameters listed in [Table C.4](#), [Table C.5](#), and [Table C.6](#). For example, on HEPMASS or MINIBOONE, our set of candidate MAF models included: for $L = 5$, an autoregressive network of size of either 2×128 , 2×512 , or 2×1024 ; for $L = 10$, an autoregressive network size of either 2×128 , 2×512 , or 2×1024 ; and for $L = 20$, again an autoregressive network size of either 2×128 , 2×512 , or 2×1024 ; this gave us a total of 9 candidate MAF models for each seed. The set of candidate CIF-MAF models can similarly be determined via the table and gave us a total of 8 candidate models for each seed. We maintained this split of 9 candidates for MAF and 8 candidates for CIF-MAF across datasets to fairly compare against the baseline by allowing them more configurations. We also considered deeper and wider MAF models to compensate for the additional parameters introduced by NN_F , NN_p , and NN_q in the CIF-MAFs. Finally, we allowed the baseline MAF models to use batch normalization between MADE layers as recommended by [Papamakarios et al. \(2017\)](#), but we do not use them within CIF-MAFs as the structure of our F generalises this transformation.

We should note that our evaluation of models is slightly different from [Papamakarios et al. \(2017\)](#). For the model which scores best on the validation set, [Papamakarios et al. \(2017\)](#) report the average and standard deviation of log-likelihood across the points in the test dataset. However, our error bars emerge as the error in average test-set log-likelihood across *multiple* runs of the same experiment; this style of evaluation is often employed in other works as well (e.g. FFJORD ([Grathwohl et al., 2019](#)), NAF ([Huang et al., 2018](#)), and SOS ([Jaini et al., 2019](#)) as noted in [Durkan et al. \(2019, Table 1\)](#)).

C.1.3 NEURAL SPLINE FLOWS

The experiment comparing NSF baselines to CIF-NSFs mirrors the experimental setup in [Durkan et al. \(2019\)](#). Specifically, we constructed baseline NSFs that exactly copied the settings in [Durkan et al. \(2019, Table 5\)](#). We also built CIF-NSFs using these baseline settings, although for the CIF-NSF-1 model we lowered the number of hidden channels in the autoregressive networks so that the total number of trainable parameters matched that of the baseline. Our parameter settings are provided in [Table C.7](#); note that parameter settings are homogeneous across datasets, besides MINIBOONE for which we reduced

Table C.6: MAF and CIF-MAF parameter configurations for BSDS300

	LAYERS (L)	AUTOREGRESSIVE NETWORK SIZE	NN_p SIZE	NN_q SIZE	NN_F SIZE
MAF	5, 10, 20	$2 \times 512, 2 \times 1024, 2 \times 2048$	-	-	-
CIF-MAF	5, 10	2×512	$2 \times 128, 2 \times 512$	$2 \times 128, 2 \times 512$	2×128

Continuously Indexed Flows

Table C.7: CIF-NSF configurations for all tabular datasets. The number of hidden features in the autoregressive network is referred to as n_h .

	NN_p SIZE	NN_q SIZE	NN_F SIZE	n_h VS. BASELINE
CIF-NSF-1 (MINIBOONE)	3×50	2×10	3×25	FEWER
CIF-NSF-1 (NON-MINIBOONE)	3×200	2×10	3×100	FEWER
CIF-NSF-2	3×200	2×10	3×100	SAME

Table C.8: Mean $\pm$ standard error of average test set log-likelihood (higher is better). Best performing runs are shown in bold. CIF-Id-1 had $s \equiv 0$ and $t = \text{Id}$. CIF-Id-2 had $s \equiv 0$ and $t = \text{NN}_F$. CIF-Id-3 had $(s, t) = \text{NN}_F$.

	POWER	GAS	HEPMASS	MINIBOONE
CIF-Id-1 ($\text{NN}_q = 10 \times 2$)	0.43 ± 0.01	10.92 ± 0.10	-17.06 ± 0.05	-11.26 ± 0.03
CIF-Id-1 ($\text{NN}_q = 100 \times 4$)	0.42 ± 0.01	10.86 ± 0.16	-17.44 ± 0.09	-10.91 ± 0.04
CIF-Id-2 ($\text{NN}_q = 10 \times 2$)	0.45 ± 0.01	10.43 ± 0.08	-17.63 ± 0.10	-11.13 ± 0.08
CIF-Id-2 ($\text{NN}_q = 100 \times 4$)	0.47 ± 0.01	10.89 ± 0.18	-17.51 ± 0.09	-10.75 ± 0.07
CIF-Id-3 ($\text{NN}_q = 10 \times 2$)	0.50 ± 0.01	11.32 ± 0.14	-17.08 ± 0.02	-10.45 ± 0.04
CIF-Id-3 ($\text{NN}_q = 100 \times 4$)	0.50 ± 0.01	11.58 ± 0.12	-16.68 ± 0.07	-10.01 ± 0.04

the size NN_p and NN_F by a factor of 4 as per Durkan et al. (2019)¹⁷. We trained both NSF and CIF-NSFs for a number of training epochs corresponding to the number of training steps divided by the number of batches in the training set, i.e.

$$n_e = \lceil n_s / (n_t / n_b) \rceil,$$

where n_e is the number of epochs, n_s is the number of training steps, n_b is the batch size, and n_t is the number of training data points. Note that n_s and n_b are from Durkan et al. (2019, Table 5), and n_t is fixed by the pre-processing steps from Papamakarios et al. (2017). We then evaluated the test-set performance of each model after the pre-specified number of epochs, averaging across three seeds, and put the results in Table 1. We again average randomness across seeds, rather than across points in the test set, as discussed in the previous section.

We quickly note here that we selected our parameters after trying a few settings on various UCI datasets. There were other settings which performed better for individual datasets that are not included here, as we would like the proposed configurations to be as homogeneous as possible. It appeared as though the NSF models were already fairly good at modelling the data, which allowed us to make NN_q much smaller while still achieving good inference.

We also should note that we wrapped our code around the NSF bijection code from <https://github.com/bayesiains/nsf>. We also disable weight decay in all of these experiments without observing any problems with convergence.

C.1.4 ABLATING f

We ran ablation experiments to gain some insight into the relative importance of f in (9). In particular, we considered a 10 layer model ($L = 10$) where at each layer U_ℓ had the same dimension as the data and $f = \text{Id}$ was the identity. We refer to this model as CIF-Id.

We considered three parameterisations of CIF-Id. The first had $s \equiv 0$ and $t = \text{Id}$, which from our choice (10) of $p_{U|Z}$ corresponds to stacking the following generative process:

$$\begin{aligned} Z &\sim P_Z \\ \epsilon &\sim \text{Normal}(0, I_d) \\ X &:= Z - \mu^p(Z) - e^{\zeta^p(Z)} \odot \epsilon. \end{aligned} \tag{C.1}$$

¹⁷Indeed, there was no choice of n_h which would allow us to achieve the same number of parameters as the baseline for the models noted in row 2 of Table C.7.

Observe this generalise ResFlows, since (5) can be realised by sending $\varsigma^p \rightarrow -\infty$ and having $\mu^p < 1$. Accordingly, we took NN_p to be a 4×128 MLP to match the size of the residual blocks used in our tabular ResFlow experiments.

The second CIF-Id parameterisation had $s \equiv 0$ and $t = \text{NN}_F$, which amounts to replacing (C.1) with

$$X := Z - t \left(\mu^p(Z) + e^{\varsigma^p(Z)} \odot \epsilon \right).$$

To align with the first CIF-Id, we took NN_F and NN_p to be 2×128 MLPs, and zeroed out the s output of NN_F to obtain $s \equiv 0$. The third parameterisation had $(s, t) = \text{NN}_F$, which replaces (C.1) with

$$X := \exp \left(-s \left(\mu^p(Z) + e^{\varsigma^p(Z)} \odot \epsilon \right) \right) \odot Z - t \left(\mu^p(Z) + e^{\varsigma^p(Z)} \odot \epsilon \right).$$

Again, we took NN_F and NN_p to be 2×128 MLPs in this case.

We ran all configurations with two different choices of NN_q : a 2×10 MLP as in our tabular ResFlow experiments, as well as a 4×100 MLP. The results are given in Table C.8.¹⁸ Observe that these models performed comparably or better than the 100-layer ResFlows, but worse than the CIF-ResFlows and CIF-MAFs in Table 1. As discussed in Section 4.1.1, we conjecture this occurs because a CIF-Id requires greater complexity from $p_{U|Z}$ to make up for its simple choice of f , which in turn makes inference harder and hence the ELBO (14) looser, resulting in a poorer model that is learned overall. Likewise, note that the best performance in all cases was obtained when $(s, t) = \text{NN}_F$. This provides some justification for the generality of our choice of (9), as opposed to simpler alternatives that omit s or t .

C.2 Image Experiments

In all our image experiments we applied the same uniform dequantisation scheme as Theis et al. (2016), after which we applied the logit transform of Dinh et al. (2017) with $\alpha = 10^{-5}$ for Fashion-MNIST and $\alpha = 0.05$ for CIFAR10.

C.2.1 RESFLOW

For our baseline ResFlow experiments we used the same architecture as Chen et al. (2019). In particular, our convolutional residual blocks (denoted Conv-ResBlock) had the form

$$\text{LipSwish} \rightarrow 3 \times 3 \text{ Conv} \rightarrow \text{LipSwish} \rightarrow 1 \times 1 \text{ Conv} \rightarrow \text{LipSwish} \rightarrow 3 \times 3 \text{ Conv},$$

while our fully connected residual blocks (denoted FC-ResBlock) had the form

$$\text{LipSwish} \rightarrow \text{Linear} \rightarrow \text{LipSwish} \rightarrow \text{Linear},$$

with a residual connection from the input to the output in both cases. The overall architecture of the flow in all cases was:

$$\text{Image} \rightarrow \text{LogitTransform}(\alpha) \rightarrow k \times \text{Conv-ResBlock} \rightarrow [\text{Squeeze} \rightarrow k \times \text{Conv-ResBlock}] \times 2 \rightarrow 4 \times \text{FC-ResBlock},$$

where the Squeeze operation was as defined by Dinh et al. (2017). Like Chen et al. (2019), we used ActNorm layers (Kingma & Dhariwal, 2018) before and after each residual block.

Due to computational constraints, the models we considered were smaller than those used by Chen et al. (2019). In particular, our smaller ResFlow models used 128 hidden channels in their Conv-ResBlocks, 64 hidden channels in the linear layers of their FC-ResBlocks, and had $k = 4$. Our larger ResFlow models used 256 hidden channels in their Conv-ResBlocks, 128 hidden channels in the linear layers of their FC-ResBlocks, and had $k = 6$. In contrast, Chen et al. (2019) used 512 hidden channels in their Conv-ResBlocks, 128 hidden channels in their FC-ResBlocks, and had $k = 16$.

As described for our tabular experiments, we used the same estimation scheme as Chen et al. (2019). Additionally:

- We took $\kappa = 0.98$;
- We used the Neumann gradient series expression for the log Jacobian (Chen et al., 2019, (8)) and computed gradients in the forward pass (Chen et al., 2019, (9)) to reduce memory overhead;

¹⁸Due to computational constraints we did not run these experiments on BSDS300.

- We used an adaptive rather than a fixed number of power iterations for spectral normalisation (Gouk et al., 2018), with a tolerance of 0.001;

For the CIF-ResFlows, we augmented the smaller baseline ResFlow by treating each composition of ActNorm $\rightarrow$ ResBlock, as well as the final ActNorm, as an instance of f in (9). Each NN_F , NN_p , and NN_q was a ResNet (He et al., 2016a;b) consisting of 2 residual blocks with 32 hidden channels (denoted 2×32). We gave each U_ℓ the same shape as a single channel of Z_ℓ , and upsampled to the dimension of Z_ℓ by adding channels at the output of each NN_F . Note that we did not experiment with using the larger baseline ResFlow model as the basis for a CIF.

For all models we used a learning rate of 10^{-3} and a batch size of 64.

Figure C.3 through to Figure C.8 show samples synthesised from the ResFlow and CIF-ResFlow density models trained on MNIST and CIFAR-10.

C.2.2 REALNVP

For our RealNVP-based image experiments, we took the baseline to be a RealNVP with the same architecture used by Dinh et al. (2017) for their CIFAR-10 experiments. In particular, we used 10 affine coupling layers with the corresponding alternating channelwise and checkerboard masks. Each coupling layer used a ResNet (He et al., 2016a;b) consisting of 8 residual blocks of 64 channels (denoted 8×64). We replicated the multi-scale architecture of Dinh et al. (2017), squeezing the channel dimension after the first 3 coupling layers, and splitting off half the dimensions after the first 6. This model had 5.94M parameters for Fashion-MNIST and 6.01M parameters for CIFAR-10.

For the CIF-RealNVP, we considered each affine coupling layer to be an instance of f in (9). When choosing the size of our networks, we sought to maintain roughly the same depth over which gradients were propagated as in the baseline. To this end, our coupling networks were 4×64 ResNets, each NN_p and NN_q were 2×64 ResNets, and each NN_F was a 2×8 ResNet. We gave each U_ℓ the same shape as a single channel of Z_ℓ , and upsampled to the dimension of Z_ℓ by adding channels at the output of NN_F . Our model had 5.99M parameters for Fashion-MNIST and 6.07M parameters for CIFAR-10.

For completeness, we also trained a RealNVP model with coupler networks of size 4×64 to match our CIF-RealNVP configuration. This model had 2.99M parameters for Fashion-MNIST and 3.05M for CIFAR-10.

In all cases for these experiments we used a learning rate of 10^{-4} and a batch size of 100.

Figure C.9 through to Figure C.14 show samples synthesised from the RealNVP and CIF-RealNVP density models trained on Fashion-MNIST and CIFAR-10.

C.3 2-D Experiments

To gain intuition about our model, we ran experiments on some simple 2-D datasets. For the datasets in Figure 1, we used a 10-layer ResFlow, a 100-layer ResFlow, and 10-layer CIF-ResFlow. For the CIF-ResFlows we took $d_{\mathcal{U}} = 1$. Other architectural and training details were the same as for the tabular experiments described in Section 5.1 and Section C.1.1. The resulting average test set log-likelihoods for the top dataset were:

- -1.501 for the 10-layer ResFlow
- -1.419 for the 100-layer ResFlow
- -1.409 for the 10-layer CIF-ResFlow

The final average test set log likelihoods for the bottom dataset were:

- -2.357 for the 10-layer ResFlow
- -2.287 for the 100-layer ResFlow
- -2.275 for the 10-layer CIF-ResFlow

Note that in both cases the CIF-ResFlow slightly outperformed the 100-layer ResFlow.

We additionally ran several experiments comparing a baseline MAF against a CIF-MAF on the 2-D datasets shown in Figure C.15. The baseline MAFs had 20 autoregressive layers, while the CIF-MAFs had 5. The network used at each layer had 4 hidden layers of 50 hidden units (denoted 4×50). For the CIF-MAF, we took $d_{\mathcal{U}} = 1$, and used 2×10 MLPs for NN_F and 4×50 MLPs for NN_p and NN_q . In total the baseline MAF had 160160 parameters, while our model had 119910 parameters.

The results of these experiments are shown in Figure C.15. Observe that CIF-MAF consistently produces a more faithful representation of the target distribution than the baseline, and in all cases achieved higher average test set log probability. A failure mode of our approach is exhibited in the spiral dataset, where our model still lacks the power to fully capture the topology of the target. However, we did not find it difficult to improve on this: by increasing the size of NN_p to 8×50 (and keeping all other parameters fixed), we were able to obtain the result shown in Figure C.16. This model had a total of 221910 parameters. We also tried a larger MAF model with autoregressive networks of size 8×50 , (obtaining 364160 parameters total). This model diverged after approximately 160 epochs. The result after 150 epochs is shown in Figure C.16.



Figure C.3: Synthetic MNIST samples generated by the small baseline ResFlow model



Figure C.4: Synthetic MNIST samples generated by the large baseline ResFlow model

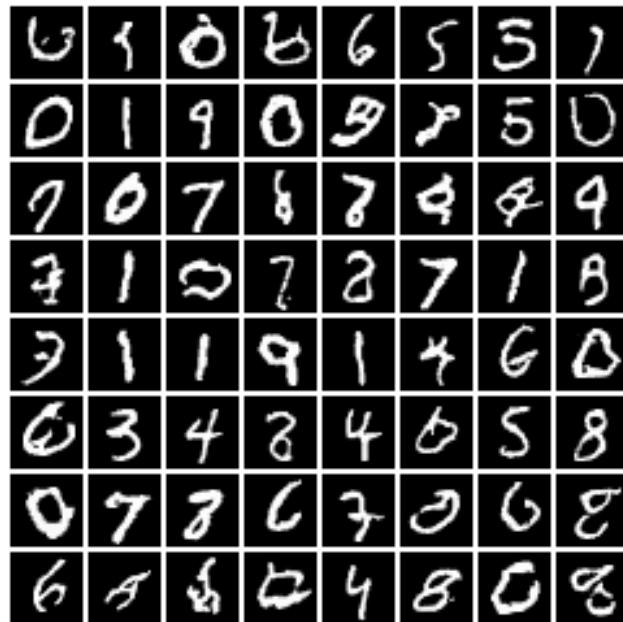


Figure C.5: Synthetic MNIST samples generated by the CIF-ResFlow model

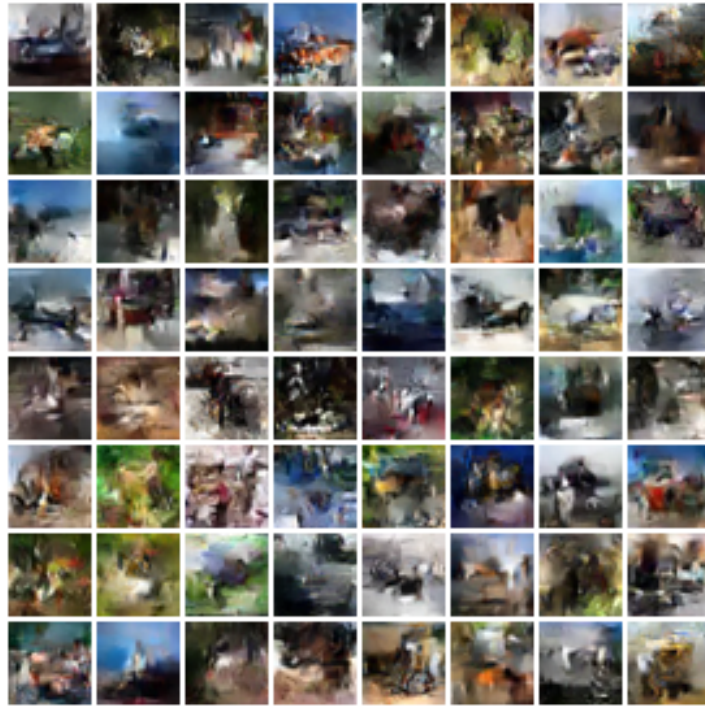


Figure C.6: Synthetic CIFAR-10 samples generated by the small baseline ResFlow model

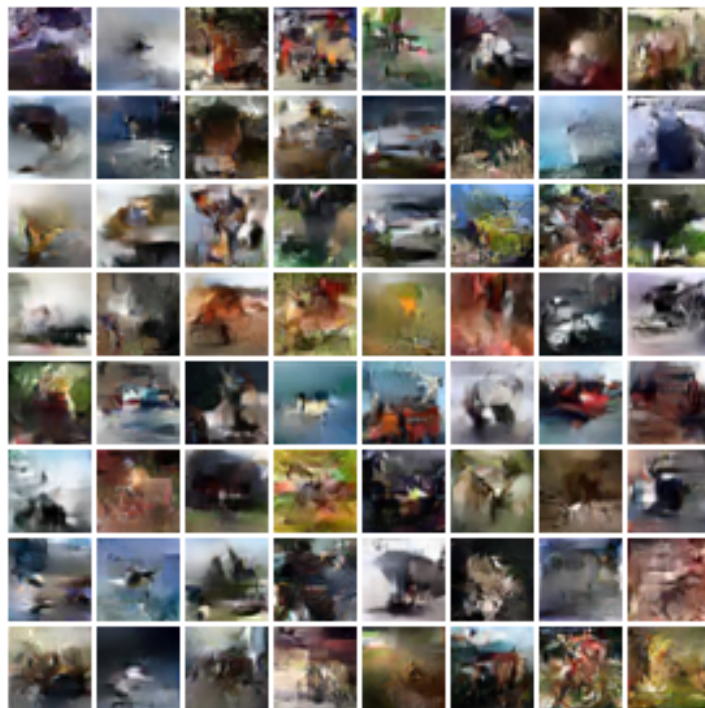


Figure C.7: Synthetic CIFAR-10 samples generated by the large baseline ResFlow model

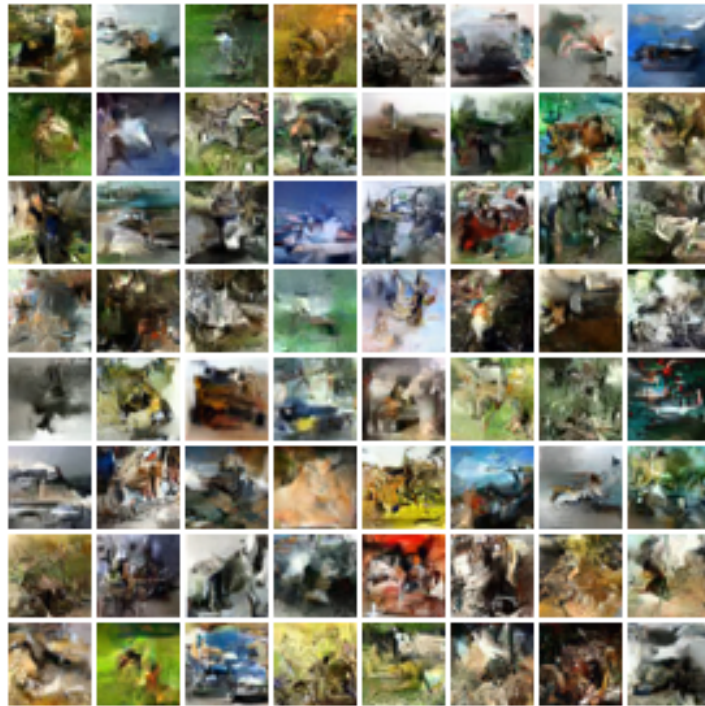


Figure C.8: Synthetic CIFAR-10 samples generated by the CIF-ResFlow model



Figure C.9: Synthetic Fashion-MNIST samples generated by RealNVP with coupling networks of size 4×64



Figure C.10: Synthetic Fashion-MNIST samples generated by RealNVP with coupling networks of size 8×64



Figure C.11: Synthetic Fashion-MNIST samples generated by CIF-RealNVP



Figure C.12: Synthetic CIFAR-10 samples generated by RealNVP with coupling networks of size 4×64



Figure C.13: Synthetic CIFAR-10 samples generated by RealNVP with coupling networks of size 8×64



Figure C.14: Synthetic CIFAR-10 samples generated by CIF-RealNVP

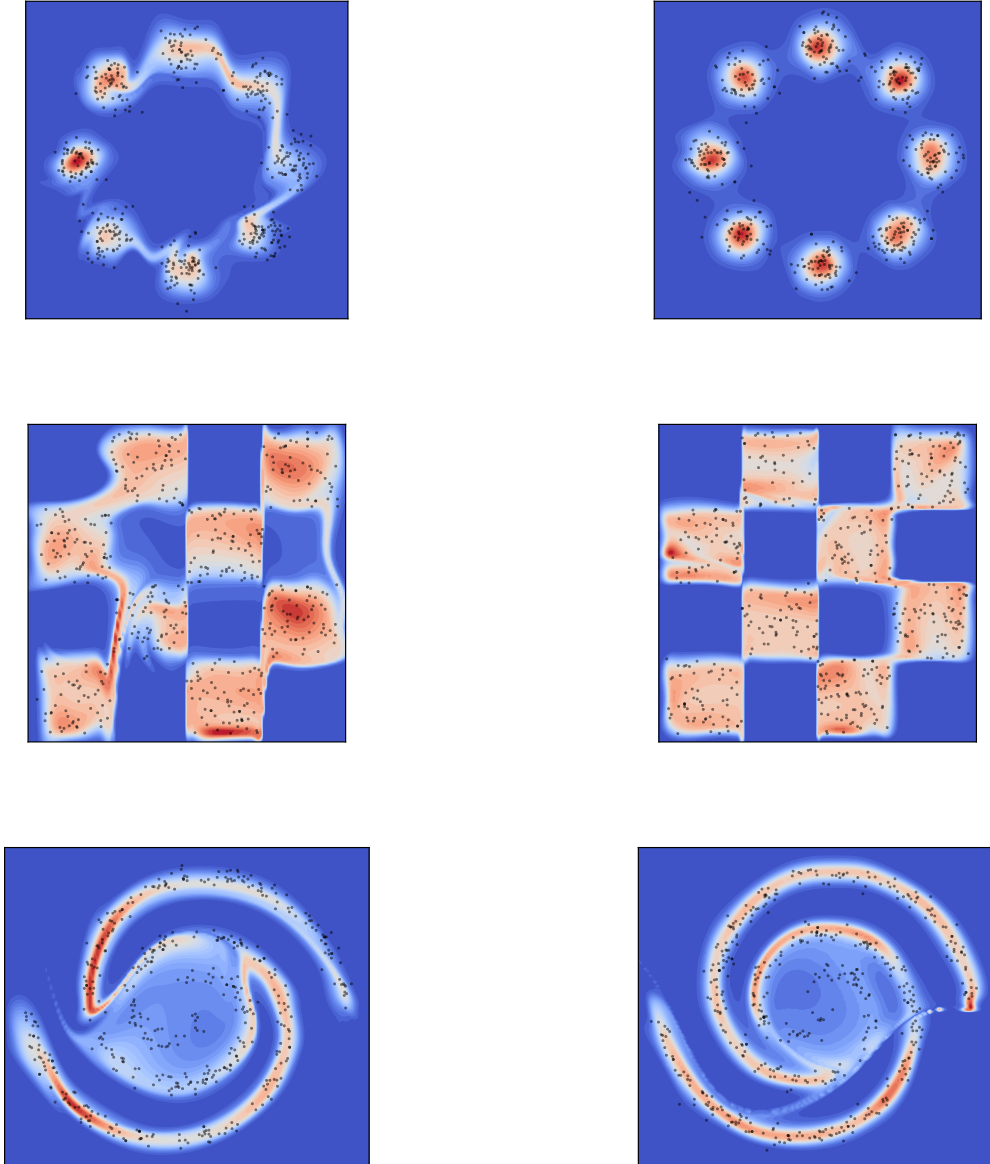


Figure C.15: Density models learned by a standard 20 layer MAF (left) and by a 5 layer CIF-MAF (right) for a variety of 2-D target distributions. Samples from the target are shown in black.

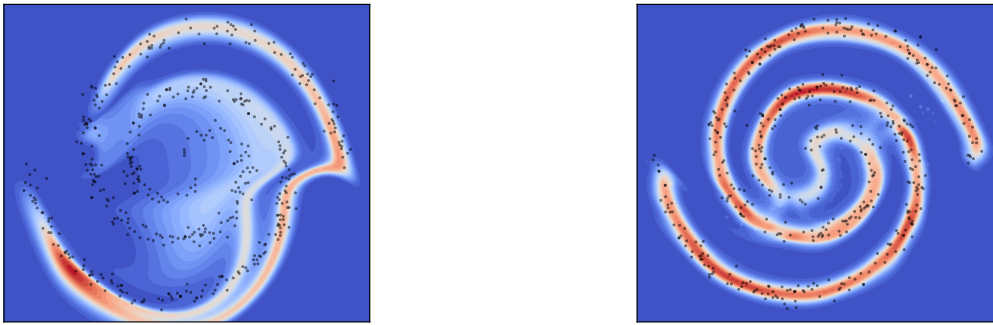


Figure C.16: Density models learned by a larger 20 layer MAF (left) and a larger 5 layer CIF-MAF (right) for the spirals dataset.

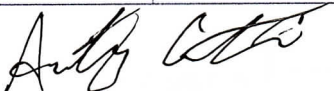
Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor (**only required where there isn't already a statement of contribution within the paper itself**).

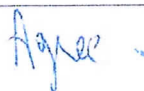
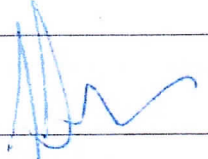
Title of Paper	Relaxing Bijectivity Constraints with Continuously Indexed Normalising Flows
Publication Status	☒ Published ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and unsubmitted work written in a manuscript style
Publication Details	Cornish, R., Caterini, A. , Deligiannidis, G., & Doucet, A. (2020). Relaxing Bijectivity Constraints with Continuously Indexed Normalising Flows. <i>International Conference on Machine Learning</i> .

Student Confirmation

Student Name:	Anthony Caterini
Contribution to the Paper	Developed the model jointly from the start with the lead author Rob Cornish, including: formulating the generative model; deriving the joint density, inference model, and ELBO objective for optimization; writing and revising the paper; writing code; and designing and running experiments. Took particular ownership of related work section, running the neural spline flow experiments, and studying the variance of the Russian roulette estimators in the residual flows method.
Signature 	Date 22 June 2021

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof. Arnaud Doucet	
Supervisor comments 	
Signature 	Date 22/06/2021

This completed form should be included in the thesis, at the end of the relevant chapter.

4

Variational Inference with Continuously-Indexed Normalizing Flows

Variational Inference with Continuously-Indexed Normalizing Flows

Anthony Caterini¹

Rob Cornish¹

Dino Sejdinovic¹

Arnaud Doucet¹

¹Department of Statistics, University of Oxford

Abstract

Continuously-indexed flows (CIFs) have recently achieved improvements over baseline normalizing flows on a variety of density estimation tasks. CIFs do not possess a closed-form marginal density, and so, unlike standard flows, cannot be plugged in directly to a variational inference (VI) scheme in order to produce a more expressive family of approximate posteriors. However, we show here how CIFs can be used as part of an *auxiliary* VI scheme to formulate and train expressive posterior approximations in a natural way. We exploit the conditional independence structure of multi-layer CIFs to build the required auxiliary inference models, which we show empirically yield low-variance estimators of the model evidence. We then demonstrate the advantages of CIFs over baseline flows in VI problems when the posterior distribution of interest possesses a complicated topology, obtaining improved results in both the Bayesian inference and surrogate maximum likelihood settings.

1 INTRODUCTION

Variational inference (VI) has emerged as a fast, albeit biased, alternative to Markov chain Monte Carlo for Bayesian inference. VI methods attempt to minimize the KL divergence from a parametrized family of distributions to a true posterior over latent variables. The expressiveness of this family is essential for good performance, with under-expressive models leading to both increased bias and under-estimation of posterior variance (Yin and Zhou, 2018).

If the density of the approximate posterior is available in closed-form, then the variational family is said to be *explicit*. Explicit models allow for straightforward estimation of the VI objective, but can often lead to reduced expressiveness, which limits their performance overall. Mean-field VI (Blei

et al., 2017), for example, imposes restrictive independence assumptions between the latent variables of interest.

Normalizing flows (Tabak et al., 2010; Rezende and Mohamed, 2015) provide an alternative family of explicit density models that yield improved expressiveness compared with mean field alternatives. These methods push samples from a simple base distribution (typically Gaussian) through parametrized bijections to produce complex, yet still exact, density models. Normalizing flows have performed well in tasks requiring explicit density models (e.g. (Louizos and Welling, 2017; Papamakarios et al., 2017; Ho et al., 2019)), including VI, where flows have demonstrated the ability to improve the quality of approximate posteriors (Rezende and Mohamed, 2015; Durkan et al., 2019).

Although normalizing flows can directly improve the expressiveness of mean-field VI schemes, their inherent bijectivity remains quite restrictive. We can overcome this limitation by instead using continuously-indexed flows (CIFs) (Cornish et al., 2020). CIFs relax the bijectivity constraint of standard normalizing flows by augmenting them with continuous index variables, thus parametrizing an *implicit* density model defined as the marginalization over these additional indexing variables. Beyond being well-grounded theoretically, CIFs also have empirically demonstrated the ability to outperform relevant normalizing flow baselines in the context of density estimation, and thus it is sensible to investigate the performance of CIFs in VI.

A difficulty in applying CIFs to VI – and implicit models more generally – is that their marginal distribution is intractable, precluding evaluation of the standard VI objective. However, conveniently, CIFs still admit a tractable *joint* distribution over the variables of interest (latent variables in VI) and the auxiliary indexing variables. We can therefore appeal to the framework of *auxiliary variational inference* (AVI) (Agakov and Barber, 2004), which facilitates the training of implicit models with tractable joint densities as variational inference models. CIFs also *already* prescribe a model for inferring auxiliary variables – typically required

in AVI schemes – suggesting that CIFs are a natural fit here. AVI methods more generally have shown improved expressiveness over the explicit counterparts in several settings (Burda et al., 2016; Yin and Zhou, 2018), and are becoming more popular with the rise of implicit models overall (Tran et al., 2017; Lawson et al., 2019; Kleinegesse et al., 2020), suggesting that this framework is able to overcome any supposed drawbacks associated with not having access to explicit densities.

In this work, we show that these benefits are also realized when CIFs are applied within the AVI framework. We first describe how CIFs can be used as the variational family in AVI, naturally incorporating the components of CIF models designed for density estimation, and we explain how we can also *amortize* these inference models. We then empirically demonstrate the advantages of using CIFs over standard normalizing flows for modelling posteriors with complicated topologies, and additionally how CIFs can facilitate maximum likelihood estimation of the parameters of complex latent-variable generative models.

2 CONTINUOUSLY-INDEXED FLOWS FOR VARIATIONAL INFERENCE

In this section we first review necessary background on variational inference (VI) – including auxiliary variational inference (AVI) – and continuously-indexed flows (CIFs). We then describe how CIFs naturally fit in as a class of auxiliary variational posteriors, and extend to include amortization. We summarize the results of this section in Algorithm 1.

2.1 VARIATIONAL INFERENCE

Given a joint probability density $p_{X,Z}$, with observed data $X \in \mathcal{X}$ and latent variable $Z \in \mathcal{Z}$, variational inference (VI) provides us with a means to approximate the intractable posterior $p_{Z|X}(\cdot | x)$. This is accomplished by introducing a parametrized approximate posterior¹ q_Z , and maximizing the evidence lower bound (ELBO)

$$\mathcal{L}_1(x) := \mathbb{E}_{z \sim q_Z} [\log p_{X,Z}(x, z) - \log q_Z(z)] \quad (1)$$

with respect to the parameters of q_Z . This is equivalent to minimizing the KL divergence between q_Z and the true posterior $p_{Z|X}(\cdot | x)$.

Explicit VI methods, such as mean-field approaches or normalizing flow models, define q_Z in such a way that it can be evaluated pointwise. Although this approach is computationally convenient, the expressiveness of the resulting methods can often be limited. To improve on this, implicit

¹We may also *amortize* q_Z and replace it with the conditional $q_{Z|X}$, especially when using VI to facilitate generative modelling. Further discussion on amortization is deferred to Subsection 2.4.

methods define q_Z typically through some type of sampling process with intractable marginal distribution, such as the pushforward of a simple distribution through an unrestricted deep neural network. These methods can be quite powerful but also challenging to optimize, especially in the context of VI (Tran et al., 2017), as we lose the tractability of (1).

Auxiliary Variational Inference In contexts where q_Z is obtained as $q_Z(z) := \int q_{Z,U}(z, u) du$ for some joint density $q_{Z,U}$ that can be sampled from and evaluated pointwise, its parameters can be learned via *auxiliary variational inference* (AVI) (Agakov and Barber, 2004). We refer to U here as an *auxiliary* variable. These approaches introduce an auxiliary inference distribution $r_{U|Z}$ and optimize

$$\mathcal{L}_2(x) := \mathbb{E}_{(z,u) \sim q_{Z,U}} \left[\log \frac{p_{X,Z}(x, z) \cdot r_{U|Z}(u | z)}{q_{Z,U}(z, u)} \right]. \quad (2)$$

Key to this approach is the fact that $\mathcal{L}_1(x) \geq \mathcal{L}_2(x)$, and that this bound is tight when $r_{U|Z} = q_{U|Z}$, which holds because

$$\mathcal{L}_1(x) = \mathcal{L}_2(x) + \mathbb{E}_{z \sim q_Z} [D_{\text{KL}}(q_{U|Z}(\cdot | z) || r_{U|Z}(\cdot | z))]. \quad (3)$$

As such, optimizing the parameters of $r_{U|Z}$ jointly with those of $q_{Z,U}$ will encourage learning better approximations to the true posterior $p_{Z|X}$. Note that, although we now are optimizing a lower bound on $\mathcal{L}_1$, we are also optimizing over a larger family of approximate posteriors which may end up yielding a better optimum (cf. Proposition 3.1 below).

2.2 CONTINUOUSLY-INDEXED FLOWS

We now describe in detail the *continuously-indexed flow* (CIF) model (Cornish et al., 2020), which we intend to incorporate into an AVI scheme. CIFs define a density q_Z over $\mathcal{Z}$ as the Z -marginal of

$$W \sim q_W, \quad U \sim q_{U|W}(\cdot | W), \quad Z = G(W; U), \quad (4)$$

where q_W is a noise distribution over $\mathcal{Z}$, $q_{U|W}$ is a conditional distribution over $\mathcal{U}$ describing an auxiliary indexing variable, and $G : \mathcal{Z} \times \mathcal{U} \rightarrow \mathcal{Z}$ is a function such that $G(\cdot; u)$ is a bijection for each $u \in \mathcal{U}$. For all $z \in \mathcal{Z}$, the density model q_Z is then given by the intractable integral $q_Z(z) := \int q_{Z,U}(z, u) du$ over the tractable joint density $q_{Z,U}$ given by

$$q_{Z,U}(z, u) = q_W(G^{-1}(z; u)) \times q_{U|W}(u | G^{-1}(z; u)) |\det D_z G^{-1}(z; u)| \quad (5)$$

for all $z \in \mathcal{Z}$ and $u \in \mathcal{U}$, where G^{-1} denotes the inverse of G (and $D_z G^{-1}$ the Jacobian of G^{-1}) with respect to its first argument z (see Appendix A for a derivation). Typically, $q_{U|W}$ is chosen to be conditionally Gaussian with mean and covariance as the outputs of neural networks taking the conditioning variables W and Z as input, and

$$G(w; u) := e^{s(u)} \odot (g(w) + t(u)), \quad (6)$$

where $g : \mathcal{Z} \rightarrow \mathcal{Z}$ is some base bijection, $s, t : \mathcal{U} \rightarrow \mathcal{Z}$ are arbitrary neural networks, and $\odot$ denotes elementwise multiplication. Cornish et al. (2020) used the model (4) in the context of density estimation to model the generative process of a set of i.i.d. data.

Multi-layer CIFs Cornish et al. (2020) also propose to improve the expressiveness of (4) by taking the noise distribution q_W to be a CIF model itself. Applying this recursively L times, we can take q_Z to be the W_L -marginal in the following model:

$$\begin{aligned} W_0 &\sim q_{W_0}, \quad U_\ell \sim q_{U_\ell|W_{\ell-1}}(\cdot | W_{\ell-1}) \\ W_\ell &= G_\ell(W_{\ell-1}; U_\ell), \end{aligned} \quad (7)$$

where $\ell \in \{1, \dots, L\}$. Here q_{W_0} is typically a mean-field Gaussian and each $G_\ell : \mathcal{Z} \times \mathcal{U} \rightarrow \mathcal{Z}$ is bijective in its first argument. Practically, multi-layer CIF models have demonstrated far more representational power than single-layer versions, although we note that we can still view this multi-layer model as an instance of (4) for certain choices of $q_{U|W}$ and G (as in Appendix B).

Auxiliary Inference Distribution The intractability of q_Z arising from both (4) and (7) precludes direct maximum likelihood estimation. Cornish et al. (2020) therefore introduce an auxiliary *backward* distribution, either $r_{U|Z}$ or $r_{U_{1:L}|Z}$ respectively, to enable training of CIFs through an amortized ELBO. Particularly noteworthy is the structure of this distribution in the multi-layer case. The optimal choice for $r_{U_{1:L}|Z}$ would be $q_{U_{1:L}|Z}$, which can be shown to factorize as $q_{U_{1:L}|Z}(u_{1:L} | z) = \prod_{\ell=1}^L q_{U_\ell|W_\ell}(u_\ell | w_\ell)$, where $w_L := z$ and $w_\ell := G_{\ell+1}^{-1}(w_{\ell+1}; u_{\ell+1})$ recursively for $\ell \in \{1, \dots, L-1\}$. Although this gives us the form of $q_{U_{1:L}|Z}$, the backward distributions $q_{U_\ell|W_\ell}$ are not generally available in closed form. However this does at least motivate defining $r_{U_{1:L}|Z}$ to have the same form, which can be done by introducing (reparametrizable) densities $r_{U_\ell|W_\ell}$ and setting

$$r_{U_{1:L}|Z}(u_{1:L} | z) := \prod_{\ell=1}^L r_{U_\ell|W_\ell}(u_\ell | w_\ell) \quad (8)$$

with w_ℓ defined as above. The densities for $r_{U_\ell|W_\ell}$ are also taken to be parametrized conditional Gaussians. This structured inference procedure induces a natural weight-sharing scheme between the forward and backward directions of the model, as both are defined using G_ℓ .

2.3 CIF MODELS IN AVI

We can use CIFs as the family of approximate posteriors q_Z in VI by appealing to the framework of AVI. Starting with the single-layer version, we see from (5) that CIFs admit a tractable joint distribution $q_{Z,U}$ over latent and auxiliary

variables. We can then plug this distribution into (2), noting also that CIFs already prescribe a form for $r_{U|Z}$ and thus are a natural fit within an AVI scheme. However, we must take one additional step to formulate an objective amenable to optimization, as naively substituting $q_{Z,U}$ into (2) produces an expectation over a distribution containing the parameters of G itself. To address this, we show in Appendix A how to rewrite this as an expectation over $q_{W,U}$ rather than $q_{Z,U}$, obtaining the objective

$$\mathbb{E}_{(w,u) \sim q_{W,U}} \left[\log \frac{p_{X,Z}(x, z) \cdot r_{U|Z}(u | z)}{q_{W,U}(w, u) \cdot |\det D_w G(w; u)|^{-1}} \right], \quad (9)$$

where we write $z := G(w; u)$ for readability. We always select $q_{W,U}$ to be reparametrizable (Kingma and Welling, 2014), which makes the objective straightforward to optimize via stochastic gradient descent with respect to the parameters of q, r , and G . Note however that $r_{U|Z}$ need not necessarily be reparametrizable. This “direction” of reparametrization contrasts with CIF models for density estimation which require $r_{U|Z}$ – not $q_{U|W}$ – to be reparametrizable. Further discussion demonstrating that CIF models in density estimation and VI can be viewed as “opposites” of each other is provided in Subsection 3.3 and Appendix D.

Multi-layer CIFs in AVI We can also use multi-layer CIFs as part of an AVI scheme. Now, each of the $q_{U_\ell|W_{\ell-1}}$ distributions in (7) is chosen to be reparametrizable, again contrasting with (Cornish et al., 2020) which does not require reparametrizable distributions here. Figure 1a graphically displays the joint model $q_{Z,U_{1:L}}$. We also adopt the form of $r_{U_{1:L}|Z}$ from (8) and demonstrate this auxiliary inference procedure in Figure 1b, although we do not require the individual $r_{U_\ell|W_\ell}$ distributions to be reparametrizable. Being able to have $r_{U_{1:L}|Z}$ match the structure of the true auxiliary posterior $q_{U_{1:L}|Z}$ is likely useful in lowering the variance of estimators of the ELBO and gradients thereof.

We can now substitute our definitions for $q_{Z,U_{1:L}}$ (implied by (7)) and $r_{U_{1:L}|Z}$ into (2) to derive an optimization objective for training multi-layer CIFs as the approximate posterior in VI. We again must be careful about reparametrization, as we need to write the objective as an expectation over $q_{W_0,U_{1:L}}$ instead of $q_{Z,U_{1:L}}$ to be able to optimize all parameters of q , analogously to (9). Further details on how to do this are provided in Appendix B, with the full objective given in (1) there. Algorithm 1 describes how to compute an unbiased estimator of this objective, from which we can then obtain unbiased gradients via automatic differentiation.

2.4 AMORTIZATION

VI methods can also be used to provide a surrogate objective for maximum likelihood estimation of the parameters of latent-variable models, particularly for deep generative models such as the variational auto-encoder

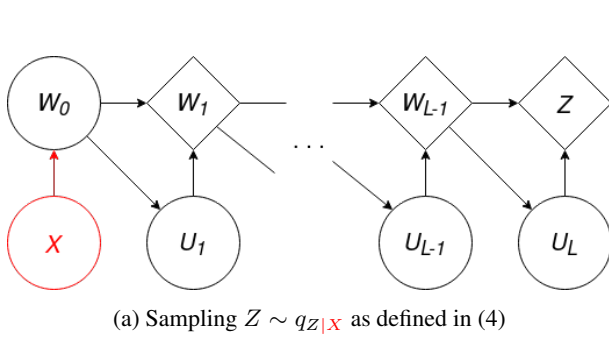
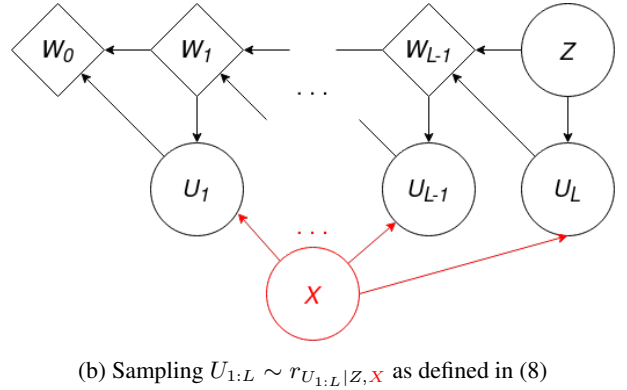


Figure 1: Diagrams demonstrating how to sample from the CIF approximate posterior (left) and the auxiliary inference model (right). The red highlighting corresponds to amortization – these can be ignored for models not requiring amortization.

(VAE) (Kingma and Welling, 2014; Rezende et al., 2014). In these settings, the goal is to maximize the marginal log-likelihood $\sum_i \log p_X(x_i)$ over the observed data $\{x_i\}_i$, with respect to the parameters of p , where $\log p_X(x) := \log \int p_{X,Z}(x, z) dz$ is a density model containing parametrized $p_{X,Z}(x, z)$. This integral is often intractable, and thus we resort to maximizing the ELBO (1) (or (2)) with respect to both the parameters of p and q as it bounds the marginal log-likelihood from below. In this case, we would like to *amortize* the cost of variational inference across an entire dataset, rather than compute a brand new approximate posterior for each datapoint, and so we parametrize our variational distribution as an explicit function of the data.

We can readily incorporate amortization into the single-layer CIF by replacing q_W with $q_{W|X}$ in (4), and $r_{U|Z}$ with $r_{U|Z,X}$ in (9) since the true auxiliary posterior $q_{U|Z,X}$ will now carry an explicit dependence on the data X . For multi-layer CIFs, it is again straightforward to incorporate amortization into the model for q by replacing q_{W_0} with $q_{W_0|X}$ in (7). Additional care must be taken when constructing the *auxiliary* inference model r , however, as the explicit dependence on data will appear in each term of the factorization of the true auxiliary posterior $q_{U_{1:L}|Z,X}$. We thus structure $r_{U_{1:L}|Z,X}$ similarly:

$$r_{U_{1:L}|Z,X}(u_{1:L} | z, x) := \prod_{\ell=1}^L r_{U_\ell|W_\ell,X}(u_\ell | w_\ell, x),$$

where w_ℓ is as defined in (8). The full amortized objective is given in (2) in the Appendix. Figure 1 graphically demonstrates how to incorporate amortization into both q and r , while Algorithm 1 includes a provision for this case as well.

3 COMPARISON TO RELATED WORK

In this section we first compare against methods using explicit normalizing flow models for variational inference,

Algorithm 1 Unbiased L -layer CIF ELBO estimator

```

function ELBO( $x$ , amortized)
  if amortized then
     $q_0 \leftarrow q_{W_0|X}(\cdot | x)$ 
  else
     $q_0 \leftarrow q_{W_0}$ 
   $w_0 \sim q_0$ 
   $\Delta \leftarrow -\log q_0(w)$ 
  for  $\ell = 1, \dots, L$  do
     $u \sim q_{U_\ell|W_{\ell-1}}(\cdot | w_{\ell-1})$ 
     $w_\ell \leftarrow G_\ell(u; w_{\ell-1})$ 
    if amortized then
       $r_\ell \leftarrow r_{U_\ell|W_\ell,X}(\cdot | w_\ell, x)$ 
    else
       $r_\ell \leftarrow r_{U_\ell|W_\ell}(\cdot | w_\ell)$ 
     $\Delta \leftarrow \Delta + \log r_\ell(u) - \log q_{U_\ell|W_{\ell-1}}(u | w_{\ell-1})$ 
       $+ \log |\det DG_\ell(w_{\ell-1}; u)|$ 
  return  $\Delta + \log p_{X,Z}(x, w_L)$ 

```

then move on to a discussion of implicit VI methods, and lastly compare the structure of CIFs in basic density estimation to CIFs in VI.

3.1 NORMALIZING FLOWS FOR VI

Normalizing flows (NFs) originally became popular as a method for increasing the expressiveness of explicit variational inference models (Rezende and Mohamed, 2015). NF methods define q_Z as the Z -marginal of

$$W \sim q_W, \quad Z = g(W), \quad (10)$$

where $g : \mathcal{Z} \rightarrow \mathcal{Z}$ is a bijection. We can equivalently write q_Z as $q_Z := g_{\#} q_W$, where $g_{\#} q_W$ denotes the *pushforward* of the distribution q_W under the map g . Using the change of variable formula, we can rewrite (1) here as

$$\mathcal{L}_1(x) = \mathbb{E}_{w \sim q_W} \left[\log \frac{p_{X,Z}(x, g(w))}{q_W(w) \cdot |\det Dg(w)|^{-1}} \right]. \quad (11)$$

This objective is a simplified version of the CIF VI objective (9). The following proposition, which we adapt here to the VI setting from Cornish et al. (2020, Proposition 4.1), shows that generalizing from (11) to (9) is beneficial, as a CIF model trained by this auxiliary bound will perform at least as well in inference as its corresponding baseline flow trained via maximization of (11).

Proposition 3.1. *Assume a CIF inference model with components $q_{U|W}^\phi$, $r_{U|Z}^\phi$, and G_ϕ is parametrized by $\phi \in \Phi$, with associated objective (9) denoted as $\mathcal{L}_2^\phi$. Suppose there exists $\psi \in \Phi$ such that for some bijection g , $G_\psi(\cdot; u) = g(\cdot)$ for all $u \in \mathcal{U}$. Similarly, suppose $q_{U|W}^\psi$ and $r_{U|Z}^\psi$ are such that, for some density ρ on $\mathcal{U}$, $q_{U|W}^\psi(\cdot | w) = r_{U|Z}^\psi(\cdot | z) = \rho(\cdot)$ for all $w, z \in \mathcal{Z}$. For a given $x \in \mathcal{X}$, if $\mathcal{L}_2^\phi(x) \geq \mathcal{L}_2^\psi(x)$,*

$$D_{\text{KL}}(q_Z^\phi \| p_{Z|X}(\cdot | x)) \leq D_{\text{KL}}(g_{\#}q_W \| p_{Z|X}(\cdot | x)).$$

The proof of this result, from which we also see that $\mathcal{L}_2^\psi(x) = \mathcal{L}_1(x)$ (where $\mathcal{L}_1(x)$ is as written in (11)) for all $x \in \mathcal{X}$, is provided in Appendix C. This shows that optimizing a CIF using the auxiliary ELBO $\mathcal{L}_2$ will produce at least as good of an inference model (as measured by the KL divergence) as a baseline normalizing flow optimized using the marginal ELBO (11), in the limit of infinite samples from the inference model. Note that our choices of G from (6) and $q_{U|W}$ and $r_{U|Z}$ as conditionally Gaussian will usually entail the conditions of Proposition 3.1, since for example we have $G(w; u) = g(w)$ in (6) if the final layer weights in the s and t networks are zero. We also empirically confirm that Proposition 3.1 holds in the experiments.

Beyond the discussion above, we also note that the bijectivity constraint of baseline normalizing flows can lead to problems when modelling a density that is concentrated on a region with complicated topological structure (Cornish et al., 2020, Corollary 2.2), and may cause flows to become numerically non-invertible in this case (Behrmann et al., 2020). Many models such as neural spline flows (NSFs) (Durkan et al., 2019) and *universal* flows (Huang et al., 2018; Jaini et al., 2019) have been proposed to improve expressiveness within the standard framework based on a single bijection. CIFs, on the other hand, use auxiliary variables to provide a mechanism for circumventing the limitations of using a single bijection, but lose analytical tractability as a result.

3.2 IMPLICIT VI METHODS

Several other AVI methods exist that, like our approach, also require the specification of parametrized auxiliary inference distribution $r_{U|Z}$. Hierarchical variational models (HVMs) (Ranganath et al., 2016) are one such example, which take $q_{Z,U}(z, u) := q_{Z|U}(z | u) \cdot q_U(u)$ for parametrized distributions $q_{Z|U}$ and q_U both analytically tractable. Although both CIFs and HVMs specify tractable $q_{Z,U}$, the CIF joint

distribution (5) does not admit such a simple factorization, which may therefore increase expressiveness. Furthermore, unlike CIFs, HVMs do not admit a natural mechanism for matching the auxiliary inference model $r_{U|Z}$ to the structure of the true auxiliary posterior $q_{U|Z}$ when considering multiple levels of hierarchy.

Related to these are approaches are Hamiltonian-based VI methods (Salimans et al., 2015; Caterini et al., 2018), which build $q_{Z,U}$ by numerically integrating Hamiltonian dynamics, inducing a flow that is bijective now on the extended space $\mathcal{Z} \times \mathcal{U}$ instead of just $\mathcal{Z}$. In contrast, CIFs can be used to augment any type of normalizing flow (not just Hamiltonian dynamics), and are not restricted to a specific family of bijections G . Hamiltonian methods also suffer from greatly increasing computational requirements as the number of parameters in $p_{X,Z}$ grows, since they require $D_z \log p_{X,Z}(x, z)$ at every flow step.

There also exist methods which that do not parametrize $r_{U|Z}$, but instead build an auxiliary inference distribution in VI by drawing extra samples from the approximate posterior q_Z and re-weighting (as noted in Lawson et al. (2019)). These methods, including the importance-weighted autoencoder (IWAE) (Burda et al., 2016) and semi-implicit variational inference (Yin and Zhou, 2018), effectively perform inference over an extended space consisting of K copies of the original latent space (Domke and Sheldon, 2018). These approaches may thus require far more memory to train than parametrized AVI methods, and often require care to ensure the variance of estimators of the objective (and gradients thereof) is controlled (Rainforth et al., 2018b; Tucker et al., 2019). That being said, it may be possible to combine multi-sample bounds with CIF models using a framework such as the one in Sobolev and Vetrov (2019), which demonstrates how to use IWAE-like approaches within HVMs.

A separate class of implicit VI models proposes expressive but intractable joint densities requiring density ratio estimation to train (Huszár, 2017; Tran et al., 2017). CIFs, along with other AVI methods, avoid density ratio estimation by instead constructing a tractable joint density $q_{Z,U}$.

3.3 CIFS FOR DENSITY ESTIMATION

As mentioned earlier, CIFs were originally proposed as a model for density estimation (DE), a setting in which we have access to a set of observed data $\{x_i\}_i$ over which we would like to build a density model p_X maximizing the marginal likelihood. This constitutes the key distinction between this work and Cornish et al. (2020): here, we only use CIFs for parametrizing an *inference* model q_Z , assuming we *already* have access to a forward density model $p_{X,Z}$.

However, the inference procedure required to *train* CIFs for DE is actually very closely related to the model (4). In particular, if we relabel the forward CIF model for DE as r

(instead of p used by Cornish et al. (2020)), the single-layer CIF density estimation objective is equivalent to

$$\mathbb{E}_{(x,u) \sim q_{X,U}} \left[\log \frac{r_Z(G(x;u)) \cdot r_{U|Z}(u | G(x;u))}{q_X^*(x) \cdot q_{U|X}(u | x) \cdot |\det D_x G(x;u)|^{-1}} \right], \quad (12)$$

where q_X^* is the unknown data-generating distribution from which we have i.i.d. samples, and $q_{X,U}(x,u) := q_X^*(x) \cdot q_{U|X}(u | x)$. See Appendix D for a derivation. Comparing this with (9), we see that CIFs for density estimation may be interpreted as performing AVI targeting r_Z with an amortized inference model defined as the Z -marginal of

$$X \sim q_X^*, \quad U \sim q_{U|X}(\cdot | X), \quad Z = G(X;U). \quad (13)$$

Furthermore, despite the aesthetic similarities between (7) of Cornish et al. (2020) defining p for DE, and (4) here defining q for VI, it is actually the q models that share a natural correspondence with each other. In both cases, q refers to an inference model that must be reparametrized, whereas neither p in DE nor r here require this. We might even consider using a CIF as the inference distribution for a CIF density model, which may yield additional benefits from added compositionality, although we leave these considerations as future work.

4 EXPERIMENTS

In this section, we investigate using CIFs to build more expressive variational models in posterior sampling and maximum likelihood estimation of generative models. We compare inference models based on the Masked Autoregressive Flow (MAF) (Papamakarios et al., 2017) and the autoregressive variant of the Neural Spline Flow (NSF) (Durkan et al., 2019) to CIF-based extensions. Both of these baseline models empirically provide good performance in general-purpose density estimation. We use the ADAM optimizer (Kingma and Ba, 2015) throughout. Hyperparameters for all experiments are available in Appendix E. Code will be made available at <https://github.com/anthonycaterini/cif-vi>.

4.1 TOY MIXTURE OF GAUSSIANS

Our first example looks at using VI to sample from a toy mixture of Gaussians. Given component means $\{\mu_k\}_k$ and covariances $\{\Sigma_k\}_k$, we directly define the “posterior”² $p_{Z|X}(z | x) := \sum_{k=1}^K \mathcal{N}(z; \mu_k, \Sigma_k) / K$, where K is the total number of components, so that the joint target is $p_{X,Z}(x,z) \propto p_{Z|X}(z | x)$. We work in two dimensions with component means adequately spaced out in a square lattice. Although the support of $p_{Z|X}$ is all of $\mathbb{R}^2$, it is concentrated on a subset of K disconnected components, which

²Note that there is no data x in this example – we define the “posterior” directly. Details are in Appendix E.

is not homeomorphic to $\mathbb{R}^2$, and thus we anticipate difficulties in using just a normalizing flow as the approximate posterior. We compare baseline NSF models to CIF-based extensions.

The initial distribution for both the NSF and CIF models is given by $q_W := \mathcal{N}(0, \sigma_0^2 \mathbf{I})$, with σ_0 taken as either a fixed hyperparameter or a trainable variational parameter. The CIF extension includes an auxiliary variable $u \in \mathbb{R}$ at each layer, conditional Gaussian distributions for $q_{U_\ell|W_{\ell-1}}$ and $r_{U_\ell|W_\ell}$ parametrized by small neural networks, and a single small two-headed neural network to output s and t in (6) at each layer, adding only 8.5% more parameters on top of the baseline NSF model.

Marginal ELBO Estimator For all experiments in this section, we will measure the trained models on estimates of the marginal ELBO (1). When using an explicit variational method, such as an NSF, this is readily estimated by basic Monte Carlo (MC) with N i.i.d. samples $z^{(i)} \sim q_Z$ for $i \in \{1, \dots, N\}$:

$$\hat{\mathcal{L}}(x) := \frac{1}{N} \sum_{i=1}^N \log \frac{p_{X,Z}(x, z^{(i)})}{q_Z(z^{(i)})}. \quad (14)$$

However, recall that in implicit methods q_Z is not available in closed form, which precludes direct evaluation of (14). Thus, we first must build an estimator of $q_Z(z)$ for all $z \in \mathcal{Z}$ to use within (14). We can do this via importance sampling, taking M i.i.d. samples $u^{(j)} \sim r_{U|Z}(\cdot | z)$ for $j \in \{1, \dots, M\}$ from our trained auxiliary inference model:

$$q_Z(z) \approx \frac{1}{M} \sum_{j=1}^M \frac{q_{Z,U}(z, u^{(j)})}{r_{U|Z}(u^{(j)} | z)} =: \hat{q}_Z(z). \quad (15)$$

The full estimator of the marginal ELBO for auxiliary models is then obtained by substituting (15) into (14); this is written out in full in Appendix F. Although this estimator is positively biased (because it includes the negative logarithm of an unbiased MC estimator), it is still *consistent*, and its bias is naturally controlled by the training procedure which encourages $r_{U|Z}$ to match the intractable $q_{U|Z}$. We can mitigate any further bias by increasing M (Rainforth et al., 2018a). A table displaying estimates of the marginal ELBO on a single trained model for various choices of N and M is also available in Appendix F; we choose $N = 10,000$ and $M = 100$ based on these results.

Results For our first experiment, we select $K = 9$ and fix σ_0 throughout training to either 0.1, 1, or 10. We train both NSF baselines and CIF-NSF extensions with three different random seeds for each setting of σ_0 . We show a kernel density estimation of the approximate posterior of the average case model on each configuration in Figure 2 and report the average of the marginal ELBO estimates across all three runs in the titles of the plots. We can clearly see

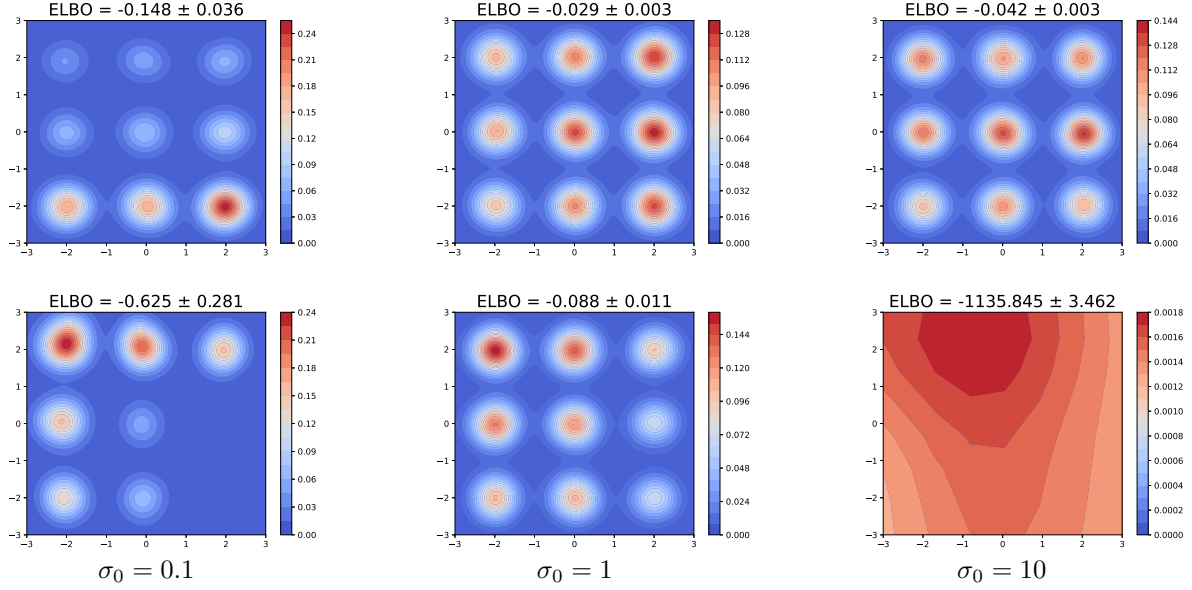


Figure 2: Samples from the trained inference models visualized using a KDE plot for a range of σ_0 values. We ran each configuration 3 times, displaying the average case of the three runs in the image, with the average plus/minus standard error of the marginal ELBO across the three runs shown in the title of the plot (higher is better). Models in the top row are CIF-NSFs, and those in the bottom row are baseline NSF. We can see that when $\sigma_0 = 0.1$, the NSF does not have enough initial noise to consistently cover the target, and when $\sigma_0 = 10$, the NSF has too much noise and cannot locate the target. The CIF-NSF at least locates each mode in all cases and provides higher-quality approximations across the board.

from both the ELBO values and the plots themselves that the CIF extensions are more consistently producing higher-quality variational approximations across the range of σ_0 , as form of (6) allows the model to directly control the noise of the outputted samples. The NSF baselines only produce reliable models for $\sigma_0 = 1$.

In this example it is quite clear how the parametrization of the CIF model “cleans up” a major deficiency of the baseline method by rescaling the initial noise. However, we might also allow σ_0 to be learned as part of the overall variational inference procedure to further probe the effectiveness of CIFs, and we experiment with this on a more challenging problem ($K = 16$). We find that the trained CIF models again outperform the baseline NSF (estimated marginal ELBO over 3 runs of -0.116 ± 0.021 for CIFs vs. -0.562 ± 0.008 for baseline NSF), thus demonstrating the increased expressiveness of CIFs beyond just rescaling.

4.2 GENERATIVE MODELLING OF IMAGES

For our second example, we use amortized variational inference to facilitate the training of a generative model of image data in the style of the variational auto-encoder (VAE) method (Kingma and Welling, 2014). We attempt to build models of the MNIST (LeCun et al., 1998) and Fashion-MNIST (Xiao et al., 2017) datasets, which both contain 256-bit greyscale images of size 28×28 . We employ dynamic binarization of these greyscale images at each training

step. The likelihood function to describe an image relies on a neural network “decoder” $\pi : \mathcal{Z} \rightarrow [0, 1]^d$, such that

$$Z \sim \mathcal{N}(0, \mathbf{I}), \quad X \sim \bigotimes_{j=1}^d \text{Ber}(\cdot | \pi_j(Z))$$

is the generative process for an image X . In our experiments, we consider two different types of decoders: a small convolutional network with only one hidden layer, and a larger convolutional network with several residual blocks as in e.g. Durkan et al. (2019). For the experiments with the smaller decoder, we use a 20-dimensional latent space $\mathcal{Z}$, and for the larger decoder, we increase to 32 dimensions.

Inference Methods We consider several models of inference to aid in surrogate maximum likelihood estimation of the parameters of π . First we consider a VAE inference model, where

$$q_{Z|X}(\cdot | x) := \mathcal{N}(\mu_Z(x), \text{diag } \sigma_Z^2(x))$$

with an “encoder” neural network taking in image data x and outputting both μ_Z and $\log \sigma_Z$. The encoder that we use in all experiments is a single-hidden-layer convolutional network which “matches” the structure of the small decoder; we keep the encoder small since the VAE here is just a base upon which we build more complicated inference models. We also consider an importance-weighted version of this VAE model (IWAE) with $K = 5$ importance samples

Table 1: Test-set average marginal log-likelihood (plus/minus one standard error) over three runs. Runs that are within one standard error of the best-performing model are shown in bold.

Model	Small Target		Large Target	
	MNIST	Fashion-MNIST	MNIST	Fashion-MNIST
VAE	-94.83 ± 0.05	-238.54 ± 0.11	-86.27 ± 0.04	-229.72 ± 0.03
IWAE ($K = 5$)	-93.14 ± 0.10	-237.03 ± 0.05	-84.23 ± 0.09	-227.80 ± 0.02
Small MAF	-91.98 ± 0.19	-237.09 ± 0.15	-83.41 ± 0.09	-228.74 ± 0.24
Large MAF	-92.68 ± 0.26	-237.57 ± 0.03	-83.38 ± 0.12	-228.72 ± 0.27
CIF-MAF	-90.87 ± 0.05	-236.31 ± 0.14	-82.70 ± 0.12	-227.64 ± 0.05
Small NSF	-91.12 ± 0.15	-236.65 ± 0.17	-83.06 ± 0.05	-228.58 ± 0.18
Large NSF	-90.79 ± 0.02	-236.48 ± 0.13	-83.12 ± 0.10	-228.46 ± 0.07
CIF-NSF	-90.82 ± 0.09	-236.48 ± 0.20	-83.31 ± 0.17	-228.54 ± 0.12

(Burda et al., 2016), which we find roughly matches the computation time per epoch of the flow-based inference methods below.

The first flow-based model that we consider is a 5-layer masked autoregressive flow (MAF) (Papamakarios et al., 2017), which is equivalent to an inverse autoregressive flow (IAF) (Kingma et al., 2016) when removing the hypernetworks producing the flow parameters. We also run experiments with a 10-layer neural spline flow (NSF) (Durkan et al., 2019), for which we clip the norm of the gradients to a maximum of 5 – as suggested for tabular density estimation – for increased stability of training. Additional hyperparameter settings for each flow are available in Appendix E. As alluded to previously, for each of the flow-based methods we will use the small VAE encoder as a base distribution $q_{W_0|X}$ to project the image data into the dimension of the latent space; we do this rather than using a large VAE encoder as the base distribution in the large target experiments (as is typically done) to force the flow models to handle more of the inference. We also consider two baseline variants for each model, a larger and smaller version, which we control by changing the number of hidden channels in the autoregressive maps.

Finally, we consider amortized CIF-based extensions of the *smaller* variants of the flow models mentioned above, so that in the end our CIF models have approximately the same total number of parameters as the larger baseline flows. We use a 2-dimensional u at each flow step. We include parametrized conditional Gaussian distributions for $q_{U_\ell|W_{\ell-1}}$ and $r_{U_\ell|W_\ell, X}$ at each layer $\ell \in \{1, \dots, L\}$, with additional care taken in the structure of the r network to combine vector inputs W_ℓ with image inputs X – details are provided in Appendix E.3.3. We use a single neural network at each layer to parametrize s_ℓ and t_ℓ appearing in G_ℓ .

Results The results of the experiment are available in Table 1. We use the standard importance-sampling based estimator of the marginal likelihood from Rezende et al. (2014, Appendix E) with 1,000 samples, which we find empirically produces low-variance estimates for the small target model³ as noted in Appendix E.5. We see that, in each experiment, CIF models are either producing the best average performance as measured by test-set estimated average marginal likelihood, or are within error bars of the best. Importantly, we note that CIFs are outperforming the baseline models which they are built directly on top of across the board: CIF-MAF and CIF-NSF significantly improve upon Small MAF and Small NSF, respectively. This justifies the claims of Proposition 3.1, demonstrating that we are not penalized for using the auxiliary objective instead of the standard ELBO.

We also can see that the CIF models produce better results than the IWAE models, which can themselves be seen as a method for auxiliary VI as previously mentioned. Despite IWAE methods being more parameter-efficient, we found that increasing K for IWAE significantly increased training time per epoch over the CIF models.

5 CONCLUSION AND DISCUSSION

In this work, we have presented continuously-indexed flows (CIFs) as a novel parametrization of an approximate posterior for use within variational inference (VI). We did this by naturally incorporating the CIF model into the framework of AVI. We have shown that the theoretical and empirical benefits of CIFs over baseline flow models extend to the VI setting, as CIFs outperform baseline flows in both sampling from complicated target distributions and facilitating maximum likelihood estimation of parametrized latent-variable models. We now add a brief further discussion on CIFs in

³We expect the same low-variance behaviour to translate to the larger target model, but did not run this for computational reasons.

VI and consider some directions for future work.

Modelling Discrete Distributions One issue with CIFs for VI (indeed, CIFs more generally) is that they are currently only designed to model continuous distributions, unlike e.g. HVMs. It may be possible however to alleviate this constraint by using discrete flows (Hooeboom et al., 2019) as a component of the overall CIF model, although it remains to be seen if the theoretical and empirical benefits of CIFs over baseline flows would extend to this case.

CIFs in Other Applications This work can serve as a template for applying CIFs more generally in applications where NFs have proven effective, such as compression (Ho et al., 2019) and approximate Bayesian computation (Papamakarios et al., 2019). These approaches may require the formulation of appropriate, application-specific surrogate objectives, but the expressiveness gains could overcome the additional costs (as in VI and density estimation) and could therefore be investigated.

Acknowledgements

Anthony Caterini is a Commonwealth Scholar supported by the U.K. Government. Rob Cornish is supported by the Engineering and Physical Sciences Research Council (EPSRC) through the Bayes4Health programme Grant EP/R018561/1. Arnaud Doucet is supported by the EPSRC CoSInES (Computational Statistical Inference for Engineering and Security) grant EP/R034710/1

References

- Felix V Agakov and David Barber. An auxiliary variational method. In *International Conference on Neural Information Processing*, pages 561–566. Springer, 2004.
- Jens Behrmann, Paul Vicol, Kuan-Chieh Wang, Roger B. Grosse, and Jörn-Henrik Jacobsen. On the invertibility of invertible neural networks, 2020. URL <https://openreview.net/forum?id=BJlVeyHFwH>.
- David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.
- Yuri Burda, Roger B. Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. In *4th International Conference on Learning Representations*, 2016.
- Anthony L Caterini, Arnaud Doucet, and Dino Sejdinovic. Hamiltonian variational auto-encoder. In *Advances in Neural Information Processing Systems*, pages 8167–8177, 2018.
- Rob Cornish, Anthony L Caterini, George Deligiannidis, and Arnaud Doucet. Relaxing bijectivity constraints with continuously-indexed normalising flows. In *International Conference on Machine Learning*, 2020.
- Justin Domke and Daniel R Sheldon. Importance weighting and variational inference. In *Advances in Neural Information Processing Systems*, pages 4470–4479, 2018.
- Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In *Advances in Neural Information Processing Systems*, pages 7509–7520, 2019.
- Jonathan Ho, Evan Lohn, and Pieter Abbeel. Compression with flows via local bits-back coding. In *Advances in Neural Information Processing Systems*, pages 3874–3883, 2019.
- Emiel Hooeboom, Jorn Peters, Rianne van den Berg, and Max Welling. Integer discrete flows and lossless compression. In *Advances in Neural Information Processing Systems*, volume 32, 2019.
- Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural autoregressive flows. In *International Conference on Machine Learning*, pages 2078–2087, 2018.
- Ferenc Huszár. Variational inference using implicit distributions. *arXiv preprint arXiv:1702.08235*, 2017.
- Priyank Jaini, Kira A Selby, and Yaoliang Yu. Sum-of-squares polynomial flow. In *International Conference on Machine Learning*, pages 3009–3018, 2019.
- Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR*, 2015.
- Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *2nd International Conference on Learning Representations*, 2014.
- Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751, 2016.
- Steven Kleinegesse, Christopher Drovandi, and Michael U Gutmann. Sequential Bayesian experimental design for implicit models via mutual information. *arXiv preprint arXiv:2003.09379*, 2020.
- John Lawson, George Tucker, Bo Dai, and Rajesh Ranganath. Energy-inspired models: Learning with sampler-induced distributions. In *Advances in Neural Information Processing Systems*, pages 8499–8511, 2019.

- Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- Christos Louizos and Max Welling. Multiplicative normalizing flows for variational bayesian neural networks. In *International Conference on Machine Learning*, pages 2218–2227. PMLR, 2017.
- George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.
- George Papamakarios, David Sterratt, and Iain Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 837–848, 2019.
- Tom Rainforth, Rob Cornish, Hongseok Yang, Andrew Warrington, and Frank Wood. On nesting Monte Carlo estimators. In *International Conference on Machine Learning*, pages 4267–4276. PMLR, 2018a.
- Tom Rainforth, Adam Kosiorek, Tuan Anh Le, Chris Maddison, Maximilian Igl, Frank Wood, and Yee Whye Teh. Tighter variational bounds are not necessarily better. In *International Conference on Machine Learning*, pages 4277–4285, 2018b.
- Rajesh Ranganath, Dustin Tran, and David Blei. Hierarchical variational models. In *International Conference on Machine Learning*, pages 324–333, 2016.
- Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538, 2015.
- Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286, 2014.
- Tim Salimans, Diederik P. Kingma, and Max Welling. Markov chain Monte Carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, pages 1218–1226, 2015.
- Artem Sobolev and Dmitry P Vetrov. Importance weighted hierarchical variational inference. In *Advances in Neural Information Processing Systems*, pages 603–615, 2019.
- Esteban G Tabak, Eric Vanden-Eijnden, et al. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217–233, 2010.
- Dustin Tran, Rajesh Ranganath, and David Blei. Hierarchical implicit models and likelihood-free variational inference. In *Advances in Neural Information Processing Systems*, pages 5523–5533, 2017.
- George Tucker, Dieterich Lawson, Shixiang Gu, and Chris J. Maddison. Doubly reparameterized gradient estimators for Monte Carlo objectives. In *7th International Conference on Learning Representations, ICLR*, 2019.
- Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.
- Mingzhang Yin and Mingyuan Zhou. Semi-implicit variational inference. In *International Conference on Machine Learning*, pages 5660–5669, 2018.

Variational Inference with Continuously-Indexed Normalizing Flows

Supplementary Material

Anthony Caterini¹

Rob Cornish¹

Dino Sejdinovic¹

Arnaud Doucet¹

¹Department of Statistics, University of Oxford

A SINGLE-LAYER DENSITY AND OBJECTIVE FUNCTION

Here we demonstrate how to obtain the single-layer CIF density and associated objective function.

Single-layer density We can derive the joint density $q_{Z,U}$ by first considering the density over (Z, U, W) and integrating out W :

$$q_{Z,U}(z, u) = \int q_{Z,U,W}(z, u, w) \, dw = \int q_W(w) \cdot q_{U|W}(u | w) \cdot \delta(z - G(w; u)) \, dw.$$

Now if we perform the change of variable $w = G^{-1}(z'; u)$, we get $dw = |\det D_z G^{-1}(z'; u)| \, dz'$, which then gives

$$\begin{aligned} q_{Z,U}(z, u) &= \int q_W(G^{-1}(z'; u)) \cdot q_{U|W}(u | G^{-1}(z'; u)) \cdot \delta(z - z') \cdot |\det D_z G^{-1}(z'; u)| \, dz' \\ &= q_W(G^{-1}(z; u)) \cdot q_{U|W}(u | G^{-1}(z; u)) \cdot |\det D_z G^{-1}(z; u)|. \end{aligned}$$

Single-layer objective First, we substitute our model $q_{Z,U}$ into (3) to obtain

$$\mathcal{L}(x) = \mathbb{E}_{(z,u) \sim q_{Z,U}} \left[\frac{p_{X,Z}(x, z) \cdot r_{U|Z}(u | z)}{q_W(G^{-1}(z; u)) \cdot q_{U|W}(u | G^{-1}(z; u)) \cdot |\det D_z G^{-1}(z; u)|} \right].$$

Now, noting that $z = G(w; u)$ for some $w \sim q_W$ as per the sampling procedure (4), we can rewrite the above objective instead as an expectation over $q_{W,U}$ (using LOTUS) to obtain

$$\mathcal{L}(x) = \mathbb{E}_{(w,u) \sim q_{W,U}} \left[\log \frac{p_{X,Z}(x, G(w; u)) \cdot r_{U|Z}(u | G(w; u))}{q_W(w) \cdot q_{U|W}(u | w) \cdot |\det D_w G(w; u)|^{-1}} \right],$$

since $D_z G^{-1}(G(w; u); u) = D_w G(w; u)$, which recovers (9).

B MULTI-LAYER DENSITY AND OBJECTIVE FUNCTION

This section is much like the previous section, except this time for the multi-layer model. We demonstrate how to recursively calculate the density and provide the objective function for a multi-layer model in both the un-amortized and amortized settings.

Recursive multi-layer density We can derive the full joint density $q_{Z,U_{1:L}}$ by first considering an intermediate density $q_{W_\ell, U_{1:\ell}}$ for $\ell \in \{1, \dots, L\}$, then integrating over the variable $W_{\ell-1}$:

$$\begin{aligned} q_{W_\ell, U_{1:\ell}}(w_\ell, u_{1:\ell}) &= \int q_{W_\ell, U_\ell, W_{\ell-1}, U_{1:\ell-1}}(w_\ell, u_\ell, w_{\ell-1}, u_{1:\ell-1}) \, dw_{\ell-1} \\ &= \int q_{W_{\ell-1}, U_{1:\ell-1}}(w_{\ell-1}, u_{1:\ell-1}) \cdot q_{U_\ell|W_{\ell-1}}(u_\ell | w_{\ell-1}) \cdot \delta(w_\ell - G_\ell(w_{\ell-1}; u_\ell)) \, dw_{\ell-1}, \end{aligned}$$

where in the second line we use the fact that U_ℓ is conditionally independent of $U_{1:\ell-1}$ given $W_{\ell-1}$ (note that this fact is also used to derive the structure of the auxiliary posterior $q_{U_{1:L}|Z}$), and the base case of the recursion is given by $q_{W_0, U_{1:0}}(w_0, -) := q_{W_0}(w_0)$. Now, as in the previous section, we use the change of variable $w_{\ell-1} = G_\ell^{-1}(w'_\ell; u_\ell)$ to obtain $dw_{\ell-1} = |\det D_{w_\ell} G_\ell^{-1}(w'_\ell; u_\ell)| dw'_\ell$, and thus

$$q_{W_\ell, U_{1:\ell}}(w_\ell, u_{1:\ell}) = q_{W_{\ell-1}, U_{1:\ell-1}}(G_\ell^{-1}(w_\ell; u_\ell), u_{1:\ell-1}) \cdot q_{U_\ell|W_{\ell-1}}(u_\ell | G_\ell^{-1}(w_\ell; u_\ell)) \cdot |\det D_{w_\ell} G_\ell^{-1}(w_\ell; u_\ell)|.$$

We obtain our full inference model as the L^{th} step of the recursion, i.e. $q_{Z, U_{1:L}} \equiv q_{W_L, U_{1:L}}$.

Multi-layer objective function Given our joint model $q_{Z, U_{1:L}}$ and the factorized auxiliary inference model $r_{U_{1:L}|Z}$ from (8), we can write the objective function from (2) as

$$\mathcal{L}(x) = \mathbb{E}_{(z, u_{1:L}) \sim q_{Z, U_{1:L}}} \left[\log \frac{p_{X, Z}(x, z) \cdot r_{U_{1:L}|Z}(u_{1:L} | z)}{q_{Z, U_{1:L}}(z, u_{1:L})} \right].$$

However, as in the single-layer case, it is difficult to calculate unbiased gradients of the objective – as written in this form – with respect to the parameters of the bijections G_ℓ , as these bijections are also appearing in the distribution over which we take the expectation. Thus we write $w_\ell = G_\ell(w_{\ell-1}; u_\ell)$ recursively for $\ell \in \{1, \dots, L\}$, with $z := w_L$, to rewrite the objective function instead as an expectation over $q_{W_0, U_{1:L}}(u_{1:L}, w_0) := q_{W_0}(w_0) \cdot \prod_{\ell=1}^L q_{U_\ell|W_{\ell-1}}(u_\ell | w_{\ell-1})$ as below:

$$\begin{aligned} \mathcal{L}(x) &= \mathbb{E}_{(w_0, u_{1:L}) \sim q_{W_0, U_{1:L}}} \left[\log \frac{p_{X, Z}(x, z) \cdot \prod_{\ell=1}^L r_{U_\ell|W_\ell}(u_\ell | w_\ell)}{q_{W_0}(w_0) \cdot \prod_{\ell=1}^L \{q_{U_\ell|W_{\ell-1}}(u_\ell | w_{\ell-1}) \cdot |\det D_{w_{\ell-1}} G_\ell(w_{\ell-1}; u_\ell)|^{-1}\}} \right] \\ &= \mathbb{E}_{(w_0, u_{1:L}) \sim q_{W_0, U_{1:L}}} \left[-\log q_{W_0}(w_0) + \sum_{\ell=1}^L \log \frac{r_{U_\ell|W_\ell}(u_\ell | w_\ell) \cdot |\det D_{w_{\ell-1}} G_\ell(w_{\ell-1}; u_\ell)|}{q_{U_\ell|W_{\ell-1}}(u_\ell | w_{\ell-1})} + \log p_{X, Z}(x, z) \right]. \end{aligned} \quad (1)$$

The form of objective given in (1) demonstrates how Algorithm 1 works: initialize with $-\log q_{W_0}(w_0)$, collect r_ℓ, DG_ℓ , and q_ℓ terms at each step ℓ , and then finish by evaluating the joint target at the realized z value.

Amortization When using amortization, we can redefine the generative process (7) given X as follows:

$$W_0 \sim q_{W_0|X}(\cdot | X), \quad U_\ell \sim q_{U_\ell|W_{\ell-1}}(\cdot | W_{\ell-1}), \quad W_\ell = G_\ell(W_{\ell-1}; U_\ell),$$

where $Z := W_L$. Now, additionally conditioning our auxiliary inference model $r_{U_{1:L}|Z, X}$ on X , we can write the objective (1) for the amortized case as

$$\mathcal{L}(x) = \mathbb{E}_{(w_0, u_{1:L}) \sim q_{W_0, U_{1:L}} | X} \left[-\log q_{W_0|X}(w_0 | x) + \sum_{\ell=1}^L \log \frac{r_{U_\ell|W_\ell, X}(u_\ell | w_\ell, x) \cdot |\det D_{w_{\ell-1}} G_\ell(w_{\ell-1}; u_\ell)|}{q_{U_\ell|W_{\ell-1}}(u_\ell | w_{\ell-1})} + \log p_{X, Z}(x, z) \right]. \quad (2)$$

Multi-layer CIF as a single-layer CIF Lastly, we show here how the multi-layer model (7) corresponds to an instance of (4) for an L -layered extended space and bijection (as per Cornish et al. (2020, Section 3.1)): first define $G^\ell(\cdot; u_1, \dots, u_\ell) := G_\ell(\cdot; u_\ell) \circ \dots \circ G_1(\cdot; u_1)$, and then take $W := W_0, U := U_1, \dots, U_L, q_{U|W}(u | w) := \prod_{\ell} q_{U_\ell|W_{\ell-1}}(u_\ell | G^\ell(w; u_{1:\ell}))$, and $G := G^L$ in (4) to arrive at (7).

C CONTINUOUSLY-INDEXED FLOWS VERSUS BASELINE NORMALIZING FLOWS

Here, we provide a proof of Proposition 3.1. Throughout the proof, we consider x such that $\mathcal{L}_2^\phi(x) \geq \mathcal{L}_2^\psi(x)$ as per our assumption.

Let us first denote the normalizing flow objective (11) as $\mathcal{L}_1$. It is not hard to show that $\mathcal{L}_2^\psi$ reduces to $\mathcal{L}_1$:

$$\begin{aligned} \mathcal{L}_2^\psi(x) &= \mathbb{E}_{(w, u) \sim q_{W, U}^\phi} \left[\log \frac{p_{X, Z}(x, G_\psi(w; u)) \cdot r_{U|Z}(u | G_\psi(w; u))}{q_W(w) \cdot q_{U|W}^\psi(u | w) \cdot |\det D_w G_\psi(w; u)|^{-1}} \right], \\ &= \mathbb{E}_{w \sim q_W} \mathbb{E}_{u \sim \rho} \left[\log \frac{p_{X, Z}(x, g(w)) \cdot \rho(u)}{q_W(w) \cdot \rho(u) \cdot |\det Dg(w)|^{-1}} \right] \\ &= \mathcal{L}_1(x). \end{aligned}$$

From (3), we also know that

$$\mathbb{E}_{z \sim q_Z^\phi} [\log p_{X,Z}(x, z) - \log q_Z^\phi(z)] \geq \mathcal{L}_2^\phi(x),$$

as the left-hand-side is the intractable marginal ELBO obtained when using $q_Z^\phi(z) := \int q_{Z,U}^\phi(z, u) du$ as the variational distribution.

We get the final result by exploiting the standard relationship between the ELBO and the KL divergence:

$$\begin{aligned} D_{\text{KL}}(q_Z^\phi \parallel p_{Z|X}(\cdot|x)) &= \log p_X(x) - \mathbb{E}_{z \sim q_Z^\phi} [\log p_{X,Z}(x, z) - \log q_Z^\phi(z)] \\ &\leq \log p_X(x) - \mathcal{L}_2^\phi(x) \\ &\leq \log p_X(x) - \mathcal{L}_1(x) \\ &= D_{\text{KL}}(g_{\#} q_W \parallel p_{Z|X}(\cdot|x)). \end{aligned}$$

D RELATIONSHIP BETWEEN CONTINUOUSLY-INDEXED FLOWS FOR DENSITY ESTIMATION AND VARIATIONAL INFERENCE

When we are using CIFs for density estimation, we can write the single-layer generative process as

$$Z \sim r_Z, \quad U \mid Z \sim r_{U|Z}(\cdot \mid Z), \quad X = G^{-1}(Z; U),$$

so that $r_X(x) = \int r_{X,U}(x, u) du$ is the proposed density model of a dataset $\mathcal{D} = \{x_i\}_{i=1}^N$, with

$$r_{X,U}(x, u) = r_Z(G(x; u)) \cdot r_{U|Z}(u \mid G(x; u)) \cdot |\det DG(x; u)|.$$

Our goal is to maximize the average likelihood of $r_X(x)$ over the dataset, i.e. $\max \frac{1}{N} \sum_{i=1}^N \log r_X(x_i)$. However, since r_X is intractable, we must introduce a reparametrizable inference distribution $q_{U|X}$ and instead maximize the ELBO, given here for a datapoint $x \in \mathcal{D}$:

$$\mathcal{L}(x) = \mathbb{E}_{u \sim q_{U|X}(\cdot|x)} [\log r_{X,U}(x, u) - \log q_{U|X}(u \mid x)]. \quad (3)$$

Note that instead of maximizing the average of (3) over the dataset, we could instead theoretically maximize the average of (3) over the unknown “true” data-generating distribution – here denoted q_X^* – which admits the objective $\max \mathbb{E}_{q_X^*} \mathcal{L}(x)$. Maximizing this objective is equivalent to maximizing

$$\mathbb{E}_{x \sim q_X^*} [\mathcal{L}(x)] - \mathbb{E}_{x \sim q_X^*} [\log q_X^*(x)] \quad (4)$$

since q_X^* is independent of the parameters of the model. If we substitute (3) into this expression and expand the definition for $r_{X,U}$, we have

$$\begin{aligned} &\mathbb{E}_{x \sim q_X^*} [\mathcal{L}(x)] - \mathbb{E}_{x \sim q_X^*} [\log q_X^*(x)] \\ &= \mathbb{E}_{x \sim q_X^*} \left[\mathbb{E}_{u \sim q_{U|X}(\cdot|x)} [\log r_{X,U}(x, u) - \log q_{U|X}(u \mid x)] - \log q_X^*(x) \right] \\ &= \mathbb{E}_{x \sim q_X^*} \left[\mathbb{E}_{u \sim q_{U|X}(\cdot|x)} [\log r_Z(G(x; u)) + \log r_{U|Z}(u \mid G(x; u)) + \log |\det D_x G(x; u)| - \log q_{U|X}(u \mid x)] - \log q_X^*(x) \right] \\ &= \mathbb{E}_{(x,u) \sim q_{X,U}} \left[\log \frac{r_Z(G(x; u)) \cdot r_{U|Z}(u \mid G(x; u))}{q_X^*(x) \cdot q_{U|X}(u \mid x) \cdot |\det D_x G(x; u)|^{-1}} \right], \end{aligned}$$

which derives (12), where we define $q_{X,U}(x, u) := q_X^*(x) \cdot q_{U|X}(u \mid x)$.

Note also that maximizing (4) is equivalent to minimizing an upper bound on $D_{\text{KL}}(q_X^* \parallel r_X)$:

$$\begin{aligned} \mathbb{E}_{x \sim q_X^*} [\log q_X^*(x)] - \mathbb{E}_{x \sim q_X^*} [\mathcal{L}(x)] &= \mathbb{E}_{x \sim q_X^*} \left[\log q_X^*(x) - \mathbb{E}_{u \sim q_{U|X}(\cdot|x)} [\log r_{X,U}(x, u) - \log q_{U|X}(u \mid x)] \right] \\ &\geq \mathbb{E}_{x \sim q_X^*} [\log q_X^*(x) - \log r_X(x)] \quad (\text{Jensen}) \\ &= D_{\text{KL}}(q_X^* \parallel r_X). \end{aligned}$$

This is not surprising but at least motivates the use of (4) as a theoretical objective.

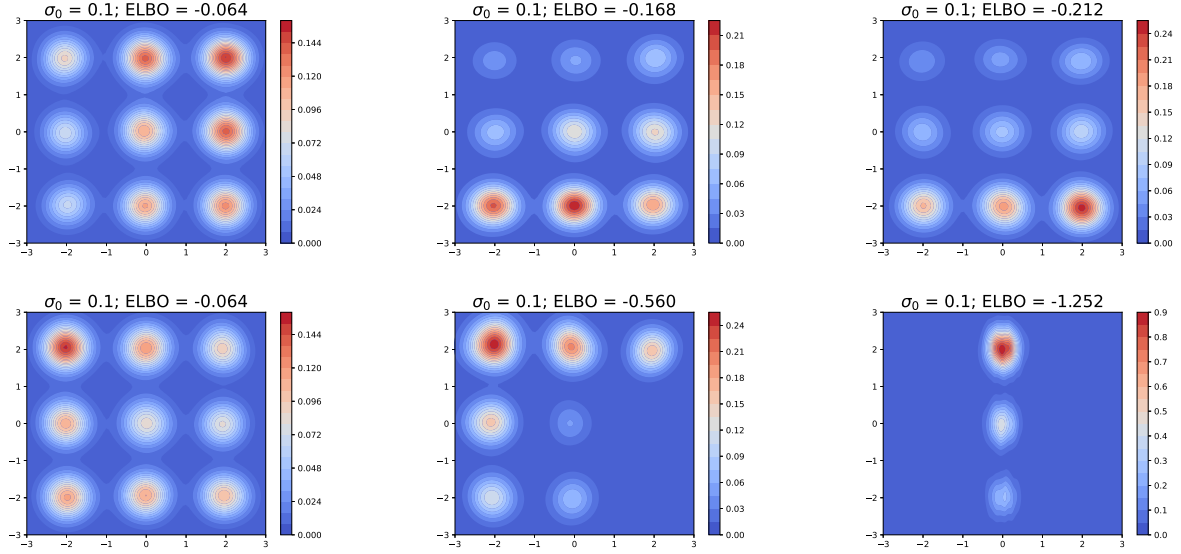


Figure 1: All runs of the mixture of Gaussians experiment for $\sigma_0 = 0.1$. CIF models are in the top row, NSF models in the bottom row.

E FURTHER EXPERIMENT DETAILS

We have included all details about the experiments from the main text in this section. We first include the approximate posterior plots from *all* runs of the $K = 9$ mixture of Gaussians experiment and not just the best of three random seeds. We next discuss the setup of both the image and mixture of Gaussians problems, then the specific structures used to build the baseline flow models, CIF extensions, and VAE models, then discuss the details of the optimization, and finally describe the log-likelihood estimator used to generate the values in Table 1.

E.1 MORE 2D MIXTURE PLOTS

We have included visualizations of the trained approximate posteriors for all three runs of each setting of σ_0 in Figure 1, Figure 2, and Figure 3. We notice consistently better performance from the trained CIF models, with the CIFs learning to cover the modes in all cases.

E.2 SETUP OF SPECIFIC PROBLEMS

E.2.1 Mixture of Gaussians Experiment

First of all, we note that the Mixture of Gaussians experiment may seem a bit unusual because we directly define the posterior and have no actual “data” x in the problem. However, we can easily imagine a Bayesian generative process which would essentially create such a posterior:

$$z \sim \sum_k \alpha_k \cdot \mathcal{N}(\mu_k, \Sigma_k), \quad x_i \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(z, \Sigma).$$

Then, $p_{Z|X_{1:n}}(\cdot | x_{1:n}) = \sum_k \omega_k(x_{1:n}) \cdot \mathcal{N}(\tilde{\mu}_k(\bar{x}), \tilde{\Sigma}_k)$ is another mixture of Gaussians for some weights ω_k and modified parameters $\tilde{\mu}_k, \tilde{\Sigma}_k$. Instead of defining the model in this way, we just directly specify the posterior as a mixture of Gaussians and perform inference.

For the $K = 9$ experiment, we evenly space the means out in a lattice within the $[-2, 2]^2$ square, i.e. $\{\mu_k\}_{k=1}^9 := \{-2, 0, 2\} \times \{-2, 0, 2\}$, and we select $\Sigma_k := \frac{1}{4^2} \mathbf{I}$ for all $k \in \{1, \dots, K\}$ so that the components had enough separation.

For the $K = 16$ experiment, we again evenly space the means out in a lattice, but this time within the $[-3, 3]^2$ square, i.e. $\{\mu_k\}_{k=1}^{16} := \{-3, -1, 1, 3\} \times \{-3, -1, 1, 3\}$, and again set $\Sigma_k := \frac{1}{4^2} \mathbf{I}$ for all $k \in \{1, \dots, K\}$.

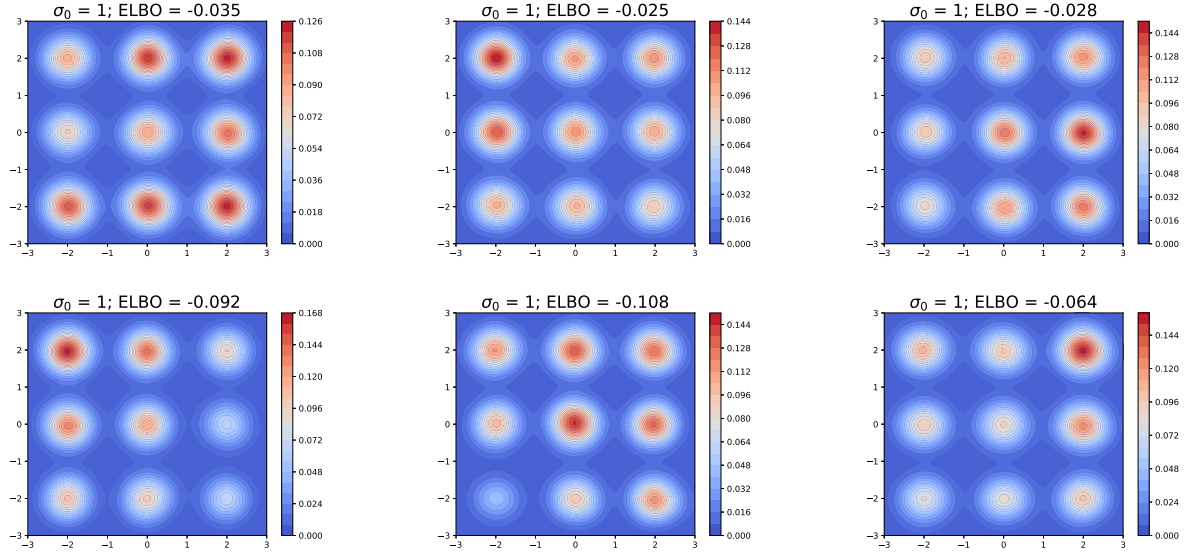


Figure 2: All runs of the mixture of Gaussians experiment for $\sigma_0 = 1$. CIF models are in the top row, NSF models in the bottom row.

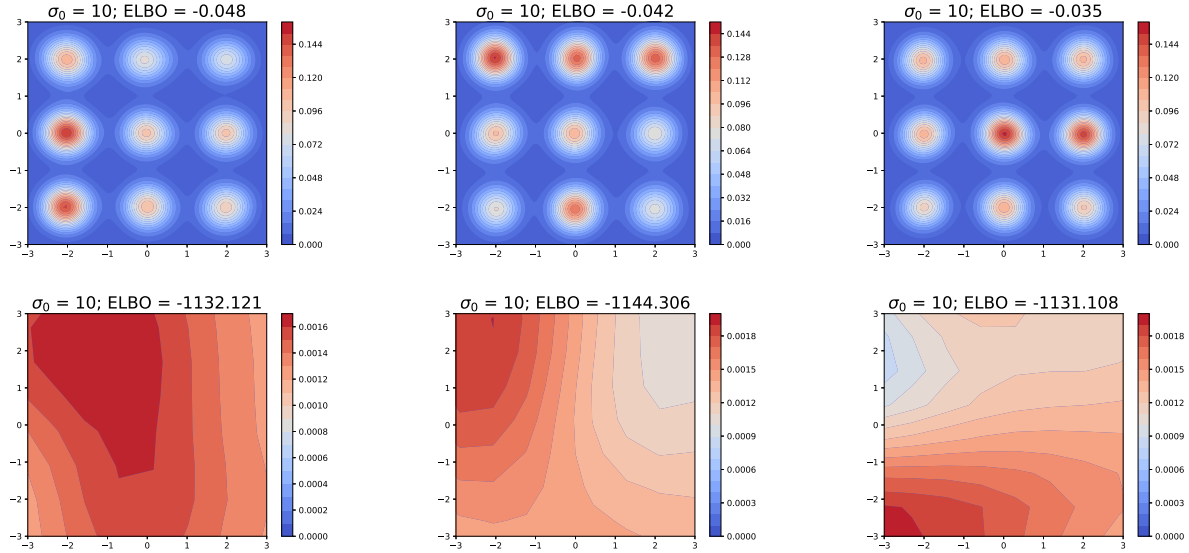


Figure 3: All runs of the mixture of Gaussians experiment for $\sigma_0 = 10$. CIF models are in the top row, NSF models in the bottom row.

Table 1: Hyperparameters used in the MAF bijections throughout the paper.

Hyperparameter	Value
Flow steps	5
Autoregressive networks	512×2 for Large MAF, 420×2 for all others (including CIF-MAF)
Batch normalization	True for MAFs, False for CIF-MAFs

Table 2: Hyperparameters used in the NSF bijections throughout the paper. Parameters have the same meaning as those from Durkan et al. (2019, Table 5), although we have additionally noted the tail bound used for the splines.

Hyperparameter	Value
Flow steps	5 for Gaussian mixture, 10 for images
Residual blocks	2
Hidden features	44 for Large NSF, 32 for all others (including CIF-NSF)
Bins	8
Dropout	0.0
Tail bound (B)	3

E.2.2 Image Experiment

We first re-iterate that we set $n_Z := 20$ for the small decoder, and $n_Z := 32$ for the large one, so that $\mathcal{Z} := \mathbb{R}^{n_Z}$.

The small decoder is a single-hidden-layer transposed convolutional network. It applies a fully-connected layer with tanh nonlinearity to transform the n_Z -dimensional latent variables into 8 feature maps of size 14×14 , and then applies a zero-padded transposed convolution with a 4×4 kernel and stride of 2 to project into size $1 \times 28 \times 28$ (the same size as the MNIST or Fashion-MNIST data). We use this output to directly parametrize the logits of a Bernoulli distribution.

The large decoder exactly matches the form from Durkan et al. (2019); indeed, we simply used the `ConvDecoder` directly from their codebase (<https://github.com/bayesiains/nsf>).

We use the standard train/test split for both MNIST and Fashion-MNIST, with 60,000 training points and 10,000 test points in each dataset. Of the 60,000 training points in each, we set aside 10% as validation points for early stopping.

E.3 MODEL DETAILS

Here we discuss the details of the models used. We note that all of the MAF, NSF, CIF-MAF, and CIF-NSF models use a distribution specified by the VAE encoder (Subsubsection E.3.4) as the initial distribution $q_{W|X}$ for the image experiments.

E.3.1 Masked Autoregressive Flow Bijection Settings

We note the hyperparameter settings that we used for the masked autoregressive flow bijections throughout the paper in Table 1. We insert batch normalization layers between flow steps in the MAF models as per the recommendation of Papamakarios et al. (2017), but do not use them in CIFs as the form of G_ℓ makes them unnecessary.

E.3.2 Neural Spline Flow Bijection Settings

We note the hyperparameter settings that we used for the neural spline flow bijections throughout the paper in Table 2. We clip the gradients at a norm of 5 in all models using NSF bijections as recommended by Durkan et al. (2019).

E.3.3 Continuously-Indexed Flow Settings

In this section, we describe the network configurations and hyperparameter settings that we use for the CIF extensions to the NSF bijections. Beyond what is required for the baseline flow, a multi-layer CIF additionally requires definitions of $q_{U_\ell|W_{\ell-1}}, r_{U_\ell|W_\ell}$ ($r_{U_\ell|W_\ell, X}$ when amortized), and s_ℓ, t_ℓ (from (6)) for $\ell \in \{1, \dots, L\}$, which we describe below.

For all experiments, we define the densities $q_{U_\ell|W_{\ell-1}}(\cdot | w) := \mathcal{N}(\mu_\ell^u(w), \text{diag}(\sigma_\ell^u(w)^2))$ for all $\ell \in \{1, \dots, L\}$ and $w \in \mathcal{Z}$, where $\mu_\ell^u(w)$ and $\sigma_\ell^u(w)$ are outputs of the same neural network: a 2-hidden-layer MLP with 10 hidden units in each layer. Similarly, s_ℓ and t_ℓ are two outputs of a 2-hidden-layer MLP with 10 hidden units in each layer.

The auxiliary inference model for the Gaussian mixture experiment is essentially the same as q above: $r_{U_\ell|W_\ell}(\cdot | w) := \mathcal{N}(\mu_\ell^r(w), \text{diag}(\sigma_\ell^r(w)^2))$ for all $\ell \in \{1, \dots, L\}$ and $w \in \mathcal{Z}$, where $\mu_\ell^r(w)$ and $\sigma_\ell^r(w)$ are outputs of a 2-hidden-layer MLP with 10 hidden units in each layer.

For the image experiment, the auxiliary inference model is now amortized, with $r_{U_\ell|W_\ell, X}(\cdot | w, x) := \mathcal{N}(\mu_\ell^r(w, x), \text{diag}(\sigma_\ell^r(w, x)^2))$ for all $\ell \in \{1, \dots, L\}, w \in \mathcal{Z}$, and $x \in \mathcal{X}$, where $\mu_\ell^r(w, x)$ and $\sigma_\ell^r(w, x)$ are again two outputs of the same neural network. However, this network has a more complicated structure as it is taking in both vector-valued and image-valued inputs; we describe the steps of the network in the list below:

1. Use a linear layer to project w into a shape amenable to upsampling into an image channel (here we selected $1 \times 7 \times 7$ as this shape).
2. Bilinearly upsample by a factor of 4 to size $1 \times 28 \times 28$ and append as an additional channel to the input x to get a new input $\tilde{x} \in \mathbb{R}^{2 \times 28 \times 28}$.
3. Feed $\tilde{x}$ into a network of the same form as the VAE encoder in Subsubsection E.3.4.

The encoder will output the parameters of the normal distribution as required. We note that the linear layer step could likely be made more parameter-efficient (e.g. map to $1 \times 4 \times 4$ and upsample by a factor of 7), and there are likely other ways to combine vector-valued w and image-valued x more sensibly. Nevertheless, the design choices made here performed well in practice.

We also need to specify the u dimension for a CIF: we add $u \in \mathbb{R}$ at each layer for the Gaussian mixture example, and $u \in \mathbb{R}^2$ for the image datasets. This provides a total u dimension of 5 for the Gaussian mixture example, 10 for the CIF-MAF, and 20 for the CIF-NSF.

E.3.4 VAE Encoder Settings

The structure of the encoder used in the VAE model essentially mirrors the structure of the small decoder network from Subsubsection E.2.2. In particular, given a $1 \times 28 \times 28$ image, a zero-padded convolution is performed using a 4×4 filter and stride length 2 with the tanh nonlinearity applied afterwards, outputting 8 feature maps each of size 14×14 . Then, a fully-connected linear layer is applied to map the feature maps to an output which is two times the size of the latent dimension, giving us the mean and (log) standard deviation of the approximate posterior.

E.4 OPTIMIZATION HYPERPARAMETERS

Table 3 notes the parameters used for optimizing the models across experiments. There are a few things to note:

1. An “epoch” for the mixture of Gaussians example is simply a single stochastic optimization step for a specified number of samples from the approximate posterior since there is no “data” in this example.
2. None of the image experiments actually reached the maximum number of epochs.
3. The hyperparameter choices below were essentially default choices.

E.5 ESTIMATION OF MARGINAL LOG-LIKELIHOOD

To generate the log-likelihood outputs in Table 1, we use an importance-sampling-based estimate as in e.g. Rezende et al. (2014, Appendix E) for each run, and then average the results of this estimator across three runs. Specifically, given the test

Table 3: Optimization hyperparameters used for each experiment. Note that an “epoch” for the mixture of Gaussians example is simply a single optimization step for a specified number of samples, as there is no “data”.

Hyperparameter	Mixture of Gaussians	Images
Learning rate	10^{-3}	10^{-3}
Weight decay	0	0
Training batch size	N/A	100
q samples per step	1,000	1
Early stopping	No	Yes
Early stopping epochs	N/A	50
Maximum epochs	20,000	1,000

Table 4: Average variance in log-likelihood estimators across models and datasets for the small decoder experiment. For each run of a particular model on a particular dataset, we calculate the estimator (either (5) or (6)) 3 separate times, and calculate the empirical variance across the outputted estimates. Then we average this variance across the original 3 runs for each model-dataset combination, arriving at the numbers in the table. For example, we have 3 Small VAE models trained with different random seeds on the MNIST dataset. For each of these models, we first calculate (5) three separate times obtaining the empirical variance of these estimates, and then we average the empirical variances across the 3 Small VAE models trained with different random seeds.

Model	MNIST	Fashion-MNIST
Small VAE	2.60×10^{-3}	2.51×10^{-3}
Small MAF	2.85×10^{-3}	6.77×10^{-3}
Large MAF	2.59×10^{-3}	4.93×10^{-3}
CIF-MAF	2.38×10^{-3}	6.45×10^{-4}
Small NSF	2.78×10^{-4}	3.84×10^{-3}
Large NSF	1.84×10^{-3}	$2, 27 \times 10^{-3}$
CIF-NSF	9.78×10^{-4}	4.47×10^{-3}

dataset $\mathcal{D}_{\text{test}} = \{x_i\}_{i=1}^{N_{\text{test}}}$ and a number of samples S , the average log-likelihood for a single run is given by

$$\frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \log \left(\frac{1}{S} \sum_{s=1}^S \frac{p_{X,Z}(x_i, z_i^{(s)})}{q_{Z|X}(z_i^{(s)} | x)} \right), \quad \text{where } z_i^{(s)} \sim q_{Z|X}(\cdot | x_i), \quad (5)$$

for explicit models $q_{Z|X}$ (e.g. VAEs and normalizing flows), and

$$\frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \log \left(\frac{1}{S} \sum_{s=1}^S \frac{p_{X,Z}(x_i, z_i^{(s)}) \cdot r_{U|Z,X}(u_i^{(s)} | z_i^{(s)}, x_i)}{q_{Z,U|X}(z_i^{(s)}, u_i^{(s)} | x)} \right), \quad \text{where } z_i^{(s)}, u_i^{(s)} \sim q_{Z,U|X}(\cdot, \cdot | x_i), \quad (6)$$

for implicit models $q_{Z,U|X}$ (e.g. CIFs). We take $S = 1000$ in practice, finding that this provides adequately low-variance estimators as noted in Table 4.

F FURTHER DETAILS ON THE MARGINAL ELBO ESTIMATOR

Here is the full version of the (positively) biased, but still consistent, estimator of (2) described in Subsection 4.1:

$$\hat{\mathcal{L}}(x) := \frac{1}{N} \sum_{i=1}^N \left(\log p_{X,Z}(x, z_i) - \log \left\{ \frac{1}{M} \sum_{j=1}^M \frac{q_{Z,U}(z_i, u_{i,j})}{r_{U|Z}(u_{i,j} | z_i)} \right\} \right), \quad (7)$$

Table 5: Average plus/minus standard deviation of 20 estimates of the **marginal** ELBO (7) from a single trained model of the mixture of Gaussians experiment. We vary the values of N and select M as either a linear ($M = N$) or square-root ($M = \sqrt{N}$) function of N . We include Monte Carlo estimates of the auxiliary ELBO from the sample of z_i for reference.

N	$M = N$		$M = \sqrt{N}$	
	Marginal	Auxiliary	Marginal	Auxiliary
1	-0.213 ± 0.665	-0.237 ± 0.668	-0.302 ± 0.361	-0.317 ± 0.363
5	-0.165 ± 0.180	-0.166 ± 0.183	-0.178 ± 0.221	-0.186 ± 0.223
10	-0.189 ± 0.180	-0.190 ± 0.187	-0.156 ± 0.136	-0.160 ± 0.145
50	-0.190 ± 0.078	-0.198 ± 0.076	-0.162 ± 0.094	-0.167 ± 0.096
100	-0.177 ± 0.057	-0.182 ± 0.059	-0.181 ± 0.056	-0.185 ± 0.053
500	-0.176 ± 0.022	-0.180 ± 0.022	-0.172 ± 0.018	-0.177 ± 0.019
1,000	-0.160 ± 0.018	-0.165 ± 0.019	-0.169 ± 0.020	-0.174 ± 0.020
5,000	-0.169 ± 0.006	-0.174 ± 0.006	-0.174 ± 0.006	-0.179 ± 0.006
10,000	-0.171 ± 0.006	-0.175 ± 0.006	-0.170 ± 0.005	-0.175 ± 0.005
50,000	-0.169 ± 0.002	-0.174 ± 0.002	-0.170 ± 0.002	-0.175 ± 0.002

where $(z_i, u'_i) \stackrel{\text{i.i.d.}}{\sim} q_{Z,U}$ for $i \in \{1, \dots, N\}$, and for each i , $u_{i,j} \stackrel{\text{i.i.d.}}{\sim} r_{U|Z}(\cdot | z_i)$ for $j \in \{1, \dots, M\}$. We need to be careful to make M large enough so that our estimates are not too biased, and at the same time make N large enough so that our estimates are not dominated by variance. We explore the relationship between values of the estimator (7) for a single trained model, and the settings of M and N in Table 5. For $N = 10,000$ and $M = 100$, we have both acceptably low variance and low bias as the estimator appears to not be significantly lowered by increasing M .

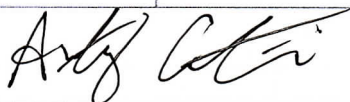
Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor (**only required where there isn't already a statement of contribution within the paper itself**).

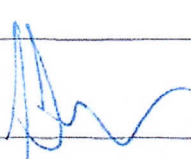
Title of Paper	Variational Inference with Continuously-Indexed Normalizing Flows
Publication Status	☐ Published ☒ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and unsubmitted work written in a manuscript style
Publication Details	Caterini, A., Cornish, R., Sejdinovic, D., & Doucet, A. (2021). Variational Inference with Continuously-Indexed Normalizing Flows. <i>Uncertainty in Artificial Intelligence</i> .

Student Confirmation

Student Name:	Anthony Caterini		
Contribution to the Paper	Responsible for the entire project, including the idea. My one supervisor Arnaud Doucet suggested the auxiliary variation bound early on, and the second author Rob Cornish provided helpful discussion throughout along with some revisions to the manuscript. My other supervisor Dino Sejdinovic also provided some revisions and helpful comments.		
Signature		Date	22 June 2021

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof. Arnaud Doucet			
Supervisor comments 			
Signature		Date	27/06/2021

This completed form should be included in the thesis, at the end of the relevant chapter.

5

Rectangular Flows for Manifold Learning

Rectangular Flows for Manifold Learning

Anthony L. Caterini*
University of Oxford & Layer 6 AI
anthony@layer6.ai

Gabriel Loaiza-Ganem*
Layer 6 AI
gabriel@layer6.ai

Geoff Pleiss
Columbia University
gmp2162@columbia.edu

John P. Cunningham
Columbia University
jpc2181@columbia.edu

Abstract

Normalizing flows are invertible neural networks with tractable change-of-volume terms, which allow optimization of their parameters to be efficiently performed via maximum likelihood. However, data of interest are typically assumed to live in some (often unknown) low-dimensional manifold embedded in a high-dimensional ambient space. The result is a modelling mismatch since – by construction – the invertibility requirement implies high-dimensional support of the learned distribution. Injective flows, mappings from low- to high-dimensional spaces, aim to fix this discrepancy by learning distributions on manifolds, but the resulting volume-change term becomes more challenging to evaluate. Current approaches either avoid computing this term entirely using various heuristics, or assume the manifold is known beforehand and therefore are not widely applicable. Instead, we propose two methods to tractably calculate the gradient of this term with respect to the parameters of the model, relying on careful use of automatic differentiation and techniques from numerical linear algebra. Both approaches perform end-to-end nonlinear manifold learning and density estimation for data projected onto this manifold. We study the trade-offs between our proposed methods, empirically verify that we outperform approaches ignoring the volume-change term by more accurately learning manifolds and the corresponding distributions on them, and show promising results on out-of-distribution detection. Our code is available at <https://github.com/layer6ai-labs/rectangular-flows>.

1 Introduction

In recent years, Normalizing Flows (NFs) have become a staple of generative modelling, being widely used for density estimation [14, 15, 45, 28, 16], variational inference [52, 30], maximum entropy modelling [37], and more [46, 31]. In density estimation, we typically have access to a set of points living in some high-dimensional space $\mathbb{R}^D$. NFs model the corresponding data-generating distribution as the pushforward of a simple distribution on $\mathbb{R}^D$ – often a Gaussian – through a smooth bijective mapping. Clever construction of these bijections allows for tractable density evaluation and thus maximum likelihood estimation of the parameters. However, as an immediate consequence of this choice, the learned distribution has support homeomorphic to $\mathbb{R}^D$; in particular, the resulting distribution is supported on a set of dimension D . This is not a realistic assumption in practice – especially for density estimation – as it directly contradicts the manifold hypothesis [6] which states that high-dimensional data lives on a lower-dimensional manifold embedded in ambient space.

* Authors contributed equally.

A natural idea to circumvent this misspecification is to consider *injective* instead of bijective flows, which now push forward a random variable on $\mathbb{R}^d$ with $d < D$ to obtain a distribution on some d -dimensional manifold embedded in $\mathbb{R}^D$. These mappings admit a change-of-variable formula bearing resemblance to that of bijective flows, but unfortunately the volume-change term becomes computationally prohibitive, which then impacts the tractability of maximum likelihood. While there have been recent efforts towards training flows where the resulting distribution is supported on a low-dimensional manifold [18, 53, 8, 35, 40, 12], these approaches either assume that the manifold is known beforehand or propose various heuristics to avoid the change-of-variable computation. Both of these are undesirable, because, while we should expect most high-dimensional data of interest to exhibit low-dimensional structure, this structure is almost always unknown. On the other hand, we argue that avoiding the volume-change term may result in learning a manifold to which it is difficult to properly assign density, and this approach further results in methods which do not take advantage of density evaluation, undermining the main motivation for using NFs in the first place.

We show that density estimation for injective flows based on maximum likelihood can be made tractable. By carefully leveraging forward- and backward-mode automatic differentiation [3], we propose two methods that allow backpropagating through the volume term arising from the injective change-of-variable formula. The first method involves exact evaluation of this term and its gradient which incurs a higher memory cost; the second uses conjugate gradients [43] and Hutchinson’s trace estimator [23] to obtain unbiased stochastic gradient estimates. Unlike previous work, our methods do not need the data manifold to be specified beforehand, but instead simultaneously estimate this manifold along with the distribution on it end-to-end, thus enabling maximum likelihood training to occur. To the best of our knowledge, ours are the first methods to scale backpropagation through the injective volume-change term to ambient dimensions D close to 3,000. We study the trade-off between memory and variance introduced by our methods and show empirical improvements over injective flow baselines for density estimation. We also show that injective flows obtain state-of-the-art performance for likelihood-based Out-of-Distribution (OoD) detection, assigning higher likelihoods to Fashion-MNIST (FMNIST) [57] than to MNIST [36] with a model trained on the former.

2 Background

2.1 Square Normalizing Flows

A normalizing flow [52, 15] is a diffeomorphism $\tilde{f}_\theta : \mathbb{R}^D \rightarrow \mathbb{R}^D$ parametrized by θ , that is, a differentiable bijection with differentiable inverse. Starting with a random variable $Z \sim p_Z$ for a simple density p_Z supported on $\mathbb{R}^D$, e.g. a standard Gaussian, the change-of-variable formula states that the random variable $X := \tilde{f}_\theta(Z)$ has density p_X on $\mathbb{R}^D$ given by:

$$p_X(x) = p_Z\left(\tilde{f}_\theta^{-1}(x)\right) \left| \det \mathbf{J}\left[\tilde{f}_\theta\right]\left(\tilde{f}_\theta^{-1}(x)\right) \right|^{-1}, \quad (1)$$

where $\mathbf{J}[\cdot]$ is the differentiation operator, so that $\mathbf{J}[\tilde{f}_\theta](\tilde{f}_\theta^{-1}(x)) \in \mathbb{R}^{D \times D}$ is the Jacobian of $\tilde{f}_\theta$ (with respect to the inputs and not θ) evaluated at $\tilde{f}_\theta^{-1}(x)$. We refer to this now standard setup as *square flows* since the Jacobian is a square matrix. The change-of-variable formula is often written in terms of the Jacobian of $\tilde{f}_\theta^{-1}$, but we use the form of (1) as it is more applicable for the next section. NFs are typically constructed in such a way that not only ensures bijectivity, but also so that the Jacobian determinant in (1) can be efficiently evaluated. When provided with a dataset $\{x_i\}_{i=1}^n \subset \mathbb{R}^D$, an NF models its generating distribution as the pushforward of p_Z through $\tilde{f}_\theta$, and thus the parameters can be estimated via maximum likelihood as $\theta^* := \arg \max_\theta \sum_{i=1}^n \log p_X(x_i)$.

2.2 Rectangular Normalizing Flows

As previously mentioned, square NFs unrealistically result in the learned density p_X having D -dimensional support. We follow the injective flow construction of Brehmer and Cranmer [8], where a smooth and injective mapping $g_\phi : \mathbb{R}^d \rightarrow \mathbb{R}^D$ with $d < D$ is constructed. In this setting, $Z \in \mathbb{R}^d$ is the low-dimensional variable used to model the data as $X := g_\phi(Z)$. A well-known result from differential geometry [32] provides an applicable change-of-variable formula:

$$p_X(x) = p_Z\left(g_\phi^{-1}(x)\right) \left| \det \mathbf{J}[g_\phi]^\top\left(g_\phi^{-1}(x)\right) \mathbf{J}[g_\phi]\left(g_\phi^{-1}(x)\right) \right|^{-1/2} \mathbf{1}(x \in \mathcal{M}_\phi), \quad (2)$$

where $\mathcal{M}_\phi := \{g_\phi(z) : z \in \mathbb{R}^d\}$. The Jacobian-transpose-Jacobian determinant now characterizes the change in volume from Z to X . We make several relevant observations: (i) The Jacobian matrix $\mathbf{J}[g_\phi](g_\phi^{-1}(x)) \in \mathbb{R}^{D \times d}$ is no longer a square matrix, and we thus refer to these flows as *rectangular*. (ii) Note that $g_\phi^{-1} : \mathcal{M}_\phi \rightarrow \mathbb{R}^d$ is only properly defined on $\mathcal{M}_\phi$ and not $\mathbb{R}^D$, and p_X is now supported on the d -dimensional manifold $\mathcal{M}_\phi$. (iii) We write the indicator $\mathbb{1}(x \in \mathcal{M}_\phi)$ explicitly to highlight the fact that this density is *not* a density with respect to the Lebesgue measure; rather, the dominating measure is a Riemannian measure on the manifold $\mathcal{M}_\phi$ [48]. (iv) One can clearly verify as a sanity check that when $d = D$, equation (2) reduces to (1).

Since data points x will almost surely not lie exactly on $\mathcal{M}_\phi$, we use a left inverse $g_\phi^\dagger : \mathbb{R}^D \rightarrow \mathbb{R}^d$ in place of g_ϕ^{-1} such that $g_\phi^\dagger(g_\phi(z)) = z$ for all $z \in \mathbb{R}^d$, which exists because g_ϕ is injective. This is properly defined on $\mathbb{R}^D$, unlike g_ϕ^{-1} which only exists over $\mathcal{M}_\phi$. Equation (2) then becomes:

$$p_X(x) = p_Z \left(g_\phi^\dagger(x) \right) \left| \det \mathbf{J}[g_\phi]^\top \left(g_\phi^\dagger(x) \right) \mathbf{J}[g_\phi] \left(g_\phi^\dagger(x) \right) \right|^{-1/2}. \quad (3)$$

Note that (3) is equivalent to projecting x onto $\mathcal{M}_\phi$ as $x \leftarrow g_\phi(g_\phi^\dagger(x))$, and then evaluating the density from (2) at the projected point.

Now, g_ϕ is injectively constructed as follows:

$$g_\phi = \tilde{f}_\theta \circ \text{pad} \circ h_\eta \quad \text{and} \quad g_\phi^\dagger = h_\eta^{-1} \circ \text{pad}^\dagger \circ \tilde{f}_\theta^{-1}, \quad (4)$$

where $\tilde{f}_\theta : \mathbb{R}^D \rightarrow \mathbb{R}^D$ and $h_\eta : \mathbb{R}^d \rightarrow \mathbb{R}^d$ are both square flows, $\phi := (\theta, \eta)$, and $\text{pad} : \mathbb{R}^d \rightarrow \mathbb{R}^D$ and $\text{pad}^\dagger : \mathbb{R}^D \rightarrow \mathbb{R}^d$ are defined as $\text{pad}(z) = (z, \mathbf{0})$ and $\text{pad}^\dagger(z, z') = z$, where $\mathbf{0}, z' \in \mathbb{R}^{D-d}$. Now, $\mathcal{M}_\phi$ depends only on θ and not η , so we write it as $\mathcal{M}_\theta$ from now on. Applying (3) yields:

$$p_X(x) = p_Z \left(g_\phi^\dagger(x) \right) \left| \det \mathbf{J}[h_\eta] \left(g_\phi^\dagger(x) \right) \right|^{-1} \left| \det \mathbf{J}[f_\theta]^\top \left(f_\theta^\dagger(x) \right) \mathbf{J}[f_\theta] \left(f_\theta^\dagger(x) \right) \right|^{-1/2}, \quad (5)$$

where $f_\theta = \tilde{f}_\theta \circ \text{pad}$ and $f_\theta^\dagger = \text{pad}^\dagger \circ \tilde{f}_\theta^{-1}$. We include a derivation of (5) in Appendix A, along with a note on why injective transformations cannot be stacked as naturally as bijective ones.

Evaluating likelihoods is seemingly intractable since constructing flows with a closed-form volume-change term is significantly more challenging than in the square case, even if the relevant matrix is now $d \times d$ instead of $D \times D$. Brehmer and Cranmer [8] thus propose a two-step training procedure to promote tractability wherein f_θ and h_η are trained separately. After observing that there is no term encouraging $x \in \mathcal{M}_\theta$, and that $x \in \mathcal{M}_\theta \iff x = g_\phi(g_\phi^\dagger(x)) \iff x = f_\theta(f_\theta^\dagger(x))$, they decide to simply train f_θ by minimizing the reconstruction error to encourage the observed data to lie on $\mathcal{M}_\theta$:

$$\theta^* = \arg \min_{\theta} \sum_{i=1}^n \left\| x_i - f_\theta \left(f_\theta^\dagger(x_i) \right) \right\|_2^2. \quad (6)$$

Note that the above requires computing both f_θ and $f_\theta^\dagger$, so that $\tilde{f}_\theta$ should be chosen as a flow allowing fast evaluation of both $\tilde{f}_\theta$ and $\tilde{f}_\theta^{-1}$. Architectures such as the Real NVP [15] or follow-up work [28, 16] are thus natural choices for $\tilde{f}_\theta$, while architectures with an autoregressive component [45, 30] should be avoided. Then, since h_η does not appear in the challenging determinant term in (5), h_η can be chosen as any normalizing flow, and optimization – for a fixed θ – can be tractably achieved by maximum likelihood over the lower-dimensional space:

$$\eta^* = \arg \max_{\eta} \sum_{i=1}^n \left\{ \log p_Z \left(g_\phi^\dagger(x_i) \right) - \log \left| \det \mathbf{J}[h_\eta] \left(g_\phi^\dagger(x_i) \right) \right| \right\}. \quad (7)$$

In practice, gradient steps in θ and η are alternated. This entire procedure circumvents evaluation of the Jacobian-transpose-Jacobian determinant term in (5), but as we show in section 3, avoiding this term by separately learning the manifold and the density on it comes with its downsides. We then show how to tractably estimate this term in section 4.

3 Related Work and Motivation

Low-dimensional and topological pathologies The mismatch between the dimension of the modelled support and that of the data-generating distribution has been observed throughout the literature in various ways. Dai and Wipf [13] show, in the context of variational autoencoders [29], that using flexible distributional approximators supported on $\mathbb{R}^D$ to model data living in a low-dimensional manifold results in pathological behavior where the manifold itself is learned, but not the distribution on it. Zhang et al. [58] demonstrate the issues with simply adding noise to the model when attempting to account for support mismatch, and propose a new family of divergences to handle this. Cornish et al. [11] demonstrate the drawbacks of normalizing flows for estimating the density of topologically-complex data, and provide a new numerically stable method for learning NFs when the support is not homeomorphic to $\mathbb{R}^D$. However, this approach still models the support as being D -dimensional. Behrmann et al. [5] show instabilities associated with NFs – particularly a lack of numerical invertibility, as also explained theoretically by Cornish et al. [11]. This is not too surprising, as attempting to learn a smooth invertible function mapping $\mathbb{R}^D$ to some low-dimensional manifold is an intrinsically ill-posed problem. This body of work strongly motivates the development of models whose support has matching topology – including dimension – to that of the true data distribution.

Manifold flows A challenge to overcome for obtaining NFs on manifolds is the Jacobian-transpose-Jacobian determinant computation. Current approaches for NFs on manifolds approach this challenge in one of two ways. The first assumes the manifold is known beforehand [18, 53, 40], severely limiting applicability to low-dimensional data where the true manifold can realistically be known. The second group circumvents the computation of the Jacobian-transpose-Jacobian entirely through various heuristics. Kumar et al. [35] use a potentially loose lower bound of the log-likelihood, and do not explicitly enforce injectivity, resulting in a method for which the change-of-variables almost surely does not hold. Cunningham et al. [12] propose to convolve the manifold distribution with Gaussian noise, which results in the model having high-dimensional support. Finally, Brehmer and Cranmer [8] propose the method we described in subsection 2.2, where manifold learning and density estimation are done separately in order to avoid the log determinant computation. Concurrently to our work, Ross and Cresswell [54] proposed a rectangular flow construction which sacrifices some expressiveness but allows for exact likelihood evaluation.

Why optimize the volume-change term? Learning f_θ and h_η separately without the Jacobian of f_θ is concerning: even if f_θ maps to the correct manifold, it might unnecessarily expand and contract volume in such a way that makes correctly learning h_η much more difficult than it needs to be. Looking ahead to our experiments, Figure 1 exemplifies this issue: the **top-middle** panel shows the ground truth density on a 1-dimensional circle in $\mathbb{R}^2$, and the **top-right** panel shows the distribution recovered by the two-step method of Brehmer and Cranmer [8]. We can see that, while the manifold is correctly recovered, the distribution on it is not. The **bottom-right** panel shows the speed at which f_{θ^*} maps $\mathbb{R}$ to $\mathcal{M}_{\theta^*}$: the top of the circle, which should have large densities, also has high speeds. Indeed, there is nothing in the objective discouraging f_θ to learn this behaviour, which implies that the corresponding low-dimensional distribution must be concentrated in a small region and thus making it harder to learn. The **bottom-middle** panel confirms this explanation: the learned low-dimensional distribution (dark red) does not match what it should (i.e. the distribution of $\{f_{\theta^*}^\dagger(x_i)\}_{i=1}^n$, in light red). This failure could have been avoided by learning the manifold in a density-aware fashion by including the Jacobian-transpose-Jacobian determinant in the objective.

4 Maximum Likelihood for Rectangular Flows: Taming the Gradient

4.1 Our Optimization Objective

We have argued that including the Jacobian-transpose-Jacobian in the optimization objective is sensible. However, as we previously mentioned, (5) corresponds to the density of the projection of x onto $\mathcal{M}_\theta$. Thus, simply optimizing the likelihood would not result in learning $\mathcal{M}_\theta$ in such a way that observed data lies on it, only encouraging *projected* data points to have high likelihood. We thus maximize the log-likelihood subject to the constraint that the reconstruction error should be smaller than some threshold, i.e. $\phi^* = \arg \max_\phi \sum_{i=1}^n \log p_X(x_i)$ subject to $\sum_{i=1}^n \|x_i - f_\theta(f_\theta^\dagger(x_i))\|_2^2 \leq \kappa$.

In practice, we use the KKT conditions [26, 34] and maximize the Lagrangian [7] instead:

$$\phi^* = \arg \max_{\phi} \sum_{i=1}^n \left\{ \log p_Z(g_{\phi}^{\dagger}(x_i)) - \log \left| \det \mathbf{J}[h_{\eta}](g_{\phi}^{\dagger}(x_i)) \right| - \frac{1}{2} \log \det J_{\theta}^{\top}(x_i) J_{\theta}(x_i) \right. \quad (8)$$

$$\left. - \beta \left\| x_i - f_{\theta}(f_{\theta}^{\dagger}(x_i)) \right\|_2^2 \right\},$$

where we treat $\beta > 0$ as a hyperparameter rather than κ , and denote $\mathbf{J}[f_{\theta}](f_{\theta}^{\dagger}(x_i))$ as $J_{\theta}(x_i)$ for simplicity. We have dropped the absolute value since $J_{\theta}^{\top}(x_i) J_{\theta}(x_i)$ is always symmetric positive definite, since $J_{\theta}(x_i)$ has full rank by injectivity of f_{θ} . We now make a technical but relevant observation about our objective: since our likelihoods are Radon-Nikodym derivatives with respect to a Riemannian measure on $\mathcal{M}_{\theta}$, different values of θ will result in different supports and dominating measures. One should thus be careful to compare likelihoods for models with different values of θ . However, thanks to the smoothness of the objective over θ , we should expect likelihoods for values of θ which are “close enough” to be comparable for practical purposes. In other words, comparisons remain reasonable locally, and the gradient of the volume-change term should contain relevant information to learn $\mathcal{M}_{\theta}$ in such a way that also facilitates learning h_{η} on the pulled-back dataset $\{f_{\theta}^{\dagger}(x_i)\}_{i=1}^n$.

4.2 Optimizing our Objective: Stochastic Gradients

Note that all the terms in (8) are straightforward to evaluate and backpropagate through except for the third one; in this section we show how to obtain unbiased stochastic estimates of its gradient. In what follows we drop the dependence of the Jacobian on x_i from our notation and write J_{θ} , with the understanding that the end computation will be parallelized over a batch of x_i s. We assume access to an efficient matrix-vector product routine, i.e. computing $J_{\theta}^{\top} J_{\theta} \epsilon$ can be quickly achieved for any $\epsilon \in \mathbb{R}^d$. We elaborate on how we obtain these matrix-vector products in the next section. It is a well known fact from matrix calculus [49] that:

$$\frac{\partial}{\partial \theta_j} \log \det J_{\theta}^{\top} J_{\theta} = \text{tr} \left((J_{\theta}^{\top} J_{\theta})^{-1} \frac{\partial}{\partial \theta_j} J_{\theta}^{\top} J_{\theta} \right), \quad (9)$$

where tr denotes the trace operator and θ_j is the j -th element of θ . Next, we can use Hutchinson’s trace estimator [23], which states that for any matrix $M \in \mathbb{R}^{d \times d}$, $\text{tr}(M) = \mathbb{E}_{\epsilon}[\epsilon^{\top} M \epsilon]$ for any $\mathbb{R}^d$ -valued random variable ϵ with zero mean and identity covariance matrix. We can thus obtain an unbiased stochastic estimate of our gradient as:

$$\frac{\partial}{\partial \theta_j} \log \det J_{\theta}^{\top} J_{\theta} \approx \frac{1}{K} \sum_{k=1}^K \epsilon_k^{\top} (J_{\theta}^{\top} J_{\theta})^{-1} \frac{\partial}{\partial \theta_j} J_{\theta}^{\top} J_{\theta} \epsilon_k, \quad (10)$$

where $\epsilon_1, \dots, \epsilon_K$ are typically sampled either from standard Gaussian or Rademacher distributions. Naïve computation of the above estimate remains intractable without explicitly constructing $J_{\theta}^{\top} J_{\theta}$. Fortunately, the $J_{\theta}^{\top} J_{\theta} \epsilon$ terms can be trivially obtained using the given matrix-vector product routine, avoiding the construction of $J_{\theta}^{\top} J_{\theta}$, and then $\partial/\partial \theta_j J_{\theta}^{\top} J_{\theta} \epsilon$ follows by taking the gradient w.r.t. θ .

There is however still the issue of computing $\epsilon^{\top} (J_{\theta}^{\top} J_{\theta})^{-1} = [(J_{\theta}^{\top} J_{\theta})^{-1} \epsilon]^{\top}$. We use conjugate gradients (CG) [43] in order to achieve this. CG is an iterative method to solve problems of the form $Au = \epsilon$ for given $A \in \mathbb{R}^{d \times d}$ (in our case $A = J_{\theta}^{\top} J_{\theta}$) and $\epsilon \in \mathbb{R}^d$; we include the CG algorithm in Appendix B for completeness. CG has several important properties. First, it is known to recover the solution (assuming exact arithmetic) after at most d steps, which means we can evaluate $A^{-1} \epsilon$. The solution converges exponentially (in the number of iterations τ) to the true value [55], so often $\tau \ll d$ iterations are sufficient for accuracy to many decimal places. In practice, if we can tolerate a certain amount of bias, we can further increase computational speed by stopping iterations early. Second, CG only requires a method to compute matrix-vector products against A , and does not require access to A itself. One such product is performed at each iteration, and CG thus requires at most d matrix-vector products, though again in practice $\tau \ll d$ products usually suffice. This results in $\mathcal{O}(\tau d^2)$ solve complexity—less than the $\mathcal{O}(d^3)$ required by direct inversion methods. We denote $A^{-1} \epsilon$ computed with conjugate gradients as $\text{CG}(A; \epsilon)$. We can then compute the estimator from (10) as:

$$\frac{\partial}{\partial \theta_j} \log \det J_{\theta}^{\top} J_{\theta} \approx \frac{1}{K} \sum_{k=1}^K \text{CG}(J_{\theta}^{\top} J_{\theta}; \epsilon_k)^{\top} \frac{\partial}{\partial \theta_j} J_{\theta}^{\top} J_{\theta} \epsilon_k. \quad (11)$$

In practice, we implement this term by noting that $\text{CG}(J_\theta^\top J_\theta; \epsilon)^\top \partial / \partial \theta_j J_\theta^\top J_\theta \epsilon = \partial / \partial \theta_j \text{stop_gradient}(\text{CG}(J_\theta^\top J_\theta; \epsilon)^\top) J_\theta^\top J_\theta \epsilon$, thereby taking advantage of the `stop_gradient` operation from Automatic Differentiation (AD) libraries and allowing us to avoid implementing a custom backward pass. We thus compute the contribution of a point x to the training objective as:

$$\begin{aligned} \log p_Z(g_\phi^\dagger(x)) - \log \left| \det \mathbf{J}[h_\eta](g_\phi^\dagger(x)) \right| - \beta \left\| x - f_\theta(g_\phi^\dagger(x)) \right\|_2^2 \\ - \frac{1}{2K} \sum_{k=1}^K \text{stop_gradient} \left(\text{CG}(J_\theta^\top J_\theta; \epsilon_k)^\top \right) J_\theta^\top J_\theta \epsilon_k \end{aligned} \quad (12)$$

which gives the correct gradient estimate when taking the derivative with respect to ϕ .

Linear solvers for Jacobian terms We note that linear solvers like CG have been used before to backpropagate through log determinant computations in the context of Gaussian processes [17], and more recently for square NFs with flexible architectures which do not allow for straightforward Jacobian determinant computations [22, 39]. However, none of these methods require the Jacobian-transpose-Jacobian-vector product routine that we derive in the next section, and to the best of our knowledge, these techniques have not been previously applied for training rectangular NFs. We also point out that recently Oktay et al. [44] proposed a method to efficiently obtain stochastic estimates of $J_\theta \epsilon$. While their method cannot be used as a drop-in replacement within our framework as it would result in a biased CG output, we believe this could be an interesting direction for future work. Finally, we note that CG has recently been combined with the Russian roulette estimator [25] to avoid having to always iterate d times while maintaining unbiasedness, again in the context of Gaussian processes [50]. We also leave the exploration of this estimator within our method for future work.

4.3 AD Considerations: The Exact Method and the Forward-Backward AD Trick

In this section we derive the aforementioned routine for vector products against $J_\theta^\top J_\theta$, as well as an exact method that avoids the need for stochastic gradients (for a given x) at the price of increased memory requirements. But first, let us ask: why are these methods needed in the first place? There is work using power series to obtain stochastic estimates of log determinants [20, 9], and one might consider using them in our setting. However, these series require knowledge of the singular values of $J_\theta^\top J_\theta$, to which we do not have access (constructing $J_\theta^\top J_\theta$ to obtain its singular values would defeat the purpose of using the power series in the first place), and we would thus not have a guarantee that the series are valid. Additionally, they have to be truncated and thus result in biased estimators, and using Russian roulette estimators to avoid bias [9] can result in infinite variance [11]. Finally, these series compute and backpropagate (w.r.t. θ) through products of the form $\epsilon^\top (J_\theta^\top J_\theta)^m \epsilon$ for different values of m , which can easily require more matrix-vector products than our methods. Behrmann et al. [4] address some of the issues with power series approximations as the result of controlling Lipschitz constants, although their estimates remain biased and potentially expensive.

Having motivated our approach, we now use commonly-known properties of AD to derive it; we briefly review these properties in Appendix C, referring the reader to Baydin et al. [3] for more detail. First, we consider the problem of explicitly constructing J_θ . This construction can then be used to evaluate $J_\theta^\top J_\theta$ and exactly compute its log determinant either for log density evaluation of a trained model, or to backpropagate (with respect to θ) through both the log determinant computation and the matrix construction, thus avoiding having to use stochastic gradients as in the previous section. We refer to this procedure as the *exact* method. Naïvely, one might try to explicitly construct J_θ using only backward-mode AD, which would require D vector-Jacobian products (vjps) of the form $v^\top J_\theta$ – one per basis vector $v \in \mathbb{R}^D$ (and then stacking the resulting row vectors vertically). A better way to explicitly construct J_θ is with forward-mode AD, which only requires d Jacobian-vector products (jvps) $J_\theta \epsilon$, again one per basis vector $\epsilon \in \mathbb{R}^d$ (and then stacking the resulting column vectors horizontally). We use a custom implementation of forward-mode AD in the popular PyTorch [47] library² for the exact method, as well as for the forward-backward AD trick described below.

We now explain how to combine forward- and backward-mode AD to obtain efficient matrix-vector products against $J_\theta^\top J_\theta$ in order to obtain the tractable gradient estimates from the previous section.

²PyTorch has a forward-mode AD implementation which relies on the “double backward” trick, which is known to be memory-inefficient. See <https://j-towns.github.io/2017/06/12/A-new-trick.html> for a description.

Table 1: Number of jvps and vjps (with respect to inputs) needed for forward and backward passes (with respect to θ), along with the corresponding variance of gradient entries.

Method	FORWARD	BACKWARD	VARIANCE
Exact (naïve)	D vjps	D vjps	0
Exact	d jvps	d jvps	0
Stochastic	$K(\tau + 1)$ jvps + $K(\tau + 1)$ vjps	K jvps + K vjps	$\propto 1/K$

Note that $v := J_\theta \epsilon$ can be computed with a single jvp call, and then $J_\theta^\top J_\theta \epsilon = [v^\top J_\theta]^\top$ can be efficiently computed using only a vjp call. We refer to this way of computing matrix-vector products against $J_\theta^\top J_\theta$ as the *forward-backward AD trick*. We summarize both of our gradient estimators in Appendix D. Note that (12) requires $K(\tau + 1)$ such matrix-vector products, which is seemingly less efficient as it is potentially greater than the d jvps required by the exact method. However, the stochastic method is much more memory-efficient than its exact counterpart when optimizing over θ : of the $K(\tau + 1)$ matrix-vector products needed to evaluate (12), only K require gradients with respect to θ . Thus only K jvps and K vjps, along with their intermediate steps, must be stored in memory over a training step. In contrast, the exact method requires gradients (w.r.t. θ) for every one of its d jvp computations, which requires storing these computations along with their intermediate steps in memory.

Our proposed methods thus offer a memory vs. variance trade-off. Increasing K in the stochastic method results in larger memory requirements which imply longer training times, as the batch size must be set to a smaller value. On the other hand, the larger the memory cost, the smaller the variance of the gradient. This still holds true for the exact method, which results in exact gradients, at the cost of increased memory requirements (as long as $K \ll d$; if K is large enough the stochastic method should never be used over the exact one). Table 1 summarizes this trade-off.

5 Experiments

We now compare our methods against the two-step baseline of Brehmer and Cranmer [8], and also study the memory vs. variance trade-off. We use the real NVP [15] architecture for all flows, except we do not use batch normalization [24] as it causes issues with vjp computations. We point out that all comparisons remain fair, and we include a detailed explanation of this phenomenon in Appendix E, along with all experimental details in Appendix G. Throughout, we use the abbreviations RNFs-ML for our maximum likelihood training method, RNFs-TS for the two-step method, and RNFs for rectangular NFs in general. For most runs, we found it useful to anneal the likelihood term(s). That is, at the beginning of training we optimize only the reconstruction term, and then slowly incorporate the other terms. This likelihood annealing procedure helped avoid local optima where the manifold is not recovered (large reconstruction error) but the likelihood of projected data is high.

5.1 Simulated Data

We consider a simulated dataset where we have access to ground truth, which allows us to empirically verify the deficiencies of RNFs-TS. We use a von Mises distribution, which is supported on the one-dimensional unit circle in $\mathbb{R}^2$. Figure 1 shows this distribution, along with its estimates from RNFs-ML (exact) and RNFs-TS. As previously observed, RNFs-TS correctly approximate the manifold, but fail to learn the right distribution on it. In contrast we can see that RNFs-ML, by virtue of including the Jacobian-transpose-Jacobian term in the optimization, manage to recover both the manifold and the distribution on it (**top left panel**), while also resulting in an easier-to-learn low-dimensional distribution (**bottom middle panel**) thanks to f_{θ^*} mapping to $\mathcal{M}_{\theta^*}$ at a more consistent speed (**bottom left panel**). We do point out that, while the results presented here are representative of usual runs for both methods, we did have runs with different results which we include in Appendix G for completeness. We finish with the observation that even though the line and the circle are not homeomorphic and thus RNFs are not perfectly able to recover the support, they manage to adequately approximate it.

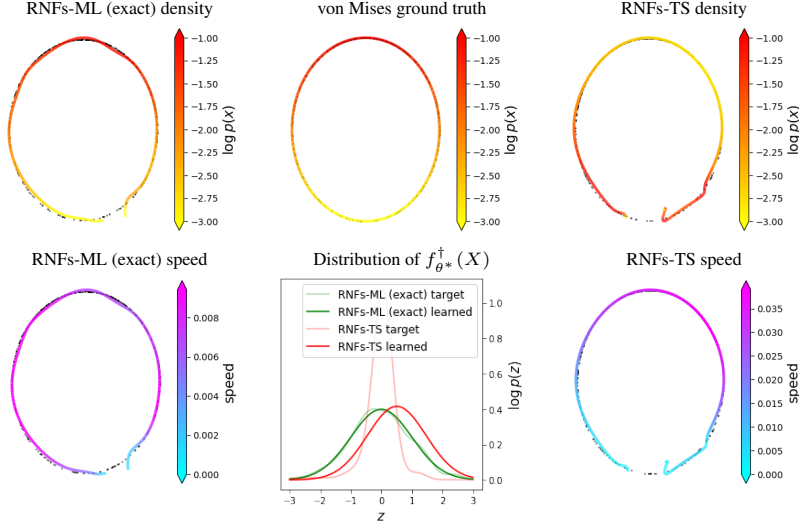


Figure 1: Top row: RNFs-ML (exact) (left), von Mises ground truth (middle), and RNF-TS (right). Bottom row: Speed at which f_{θ^*} maps to $\mathcal{M}_{\theta^*}$ (measured as l_2 distance between uniformly spaced consecutive points in $\mathbb{R}$ mapped through f_{θ^*}) for RNFs-ML (exact) (left), RNFs-TS (right), and distribution h_{η} has to learn in order to recover the ground truth, fixing θ^* (middle). See text for discussion.

Table 2: FID-like metric for tabular data (lower is better). Bolded runs are the best or overlap with it.

Method	POWER	GAS	HEPMASS	MINIBOONE
RNFs-ML (exact)	0.067 $\pm$ 0.016	0.138 $\pm$ 0.023	0.486 $\pm$ 0.032	0.978 $\pm$ 0.082
RNFs-ML ($K = 1$)	0.083 $\pm$ 0.015	0.110 $\pm$ 0.021	0.779 $\pm$ 0.191	1.001 $\pm$ 0.051
RNFs-ML ($K = 10$)	0.113 $\pm$ 0.037	0.140 $\pm$ 0.013	0.495 $\pm$ 0.055	0.878 $\pm$ 0.083
RNFs-TS	0.178 $\pm$ 0.024	0.161 $\pm$ 0.016	0.649 $\pm$ 0.081	1.085 $\pm$ 0.062

5.2 Tabular Data

We now turn our attention to the tabular datasets used by Papamakarios et al. [45], now a common benchmark for NFs as well. As previously mentioned, one should be careful when comparing models with different supports, as we cannot rely on test likelihoods as a metric. We take inspiration from the FID score [21], which is commonly used to evaluate quality of generated images when likelihoods are not available. The FID score compares the first and second moments of a well-chosen statistic – taken in practice to be the values of the last hidden layer of a pre-trained inception network [56] – from the model and data distributions using the squared Wasserstein-2 metric (between Gaussians). Here, we take the statistic to be the data itself instead of the final hidden units of a pre-trained classifier: in other words, our metric compares the mean and covariance of generated data against those of observed data with the same squared Wasserstein-2 metric. We include the mathematical formulas for computing both FID and our modified version for tabular data in Appendix F. We use early stopping with our FID-like score across all models. Our results are summarized in Table 2, where we can see that RNFs-ML consistently do a better job at recovering the underlying distribution. Once again, these results emphasize the benefits of including the Jacobian-transpose-Jacobian in the objective. Interestingly, except for HEPMASS, the results from our stochastic version with $K = 1$ are not significantly exceeded by the exact version or using a larger value of K , suggesting that the added variance does not result in decreased empirical performance. We highlight that no tuning was done (except on GAS for which we changed d from 4 to 2), RNFs-ML outperformed RNFs-TS out-of-the-box here (details are in Appendix G). We report training times in Appendix G, and observe that RNFs-ML take a similar amount of time as RNFs-TS to train for datasets with lower values of D , and while we do take longer to train for the other datasets, our training times remain reasonable and we often require fewer epochs to converge.

Table 3: FID scores (lower is better) and decision stump OoD accuracy (higher is better).

Method	FID			OoD ACCURACY	
	CIFAR-10	MNIST	FMNIST	MNIST $\rightarrow$ FMNIST	FMNIST $\rightarrow$ MNIST
RNFs-ML (exact)	643.31	36.09	296.01	92%	91%
RNFs-ML ($K = 1$)	830.94	33.98	288.39	97%	78%
RNFs-ML ($K = 4$)	-	42.90	342.91	77%	89%
RNFs-TS	731.46	35.52	318.59	98%	96%

5.3 Image Data and Out-of-Distribution Detection

We also compare RNFs-ML to RNFs-TS for image modelling on MNIST and FMNIST. We point out that these datasets have ambient dimension $D = 784$, and being able to fit RNFs-ML is in itself noteworthy: to the best of our knowledge no previous method has scaled optimizing the Jacobian-transpose-Jacobian term to these dimensions. We use FID scores both for comparing models and for early stopping during training. We also used likelihood annealing, with all experimental details again given in Appendix G. We report FID scores in Table 3, where we can see that we outperform RNFs-TS. Our RNFs-ML ($K = 1$) variant also outperforms its decreased-variance counterparts. This initially puzzling behaviour is partially explained by the fact that we used the $K = 1$ variant to tune the model (being the cheapest one to train), and then used the tuned hyperparameters for a single run of the other two variants. Nonetheless, once again these results suggest that the variance induced by our stochastic method is not empirically harmful, and that while using the exact method should be the default whenever feasible, using $K = 1$ otherwise is sensible. We also report training times where we can see the computational benefits of our stochastic method, as well as visualizations of samples, in Appendix G.

We also compare performance on the CIFAR-10 dataset [33], for which $D = 3,072$. Once again, being able to fit RNFs-ML in this setting is in itself remarkable. We do not include RNFs-ML ($K = 4$) results because of limited experimentation on CIFAR-10 due to computational cost (experimental details, including hyperparameters which we tried, are given in Appendix G). We can see that, while RNFs-TS outperformed RNFs-ML ($K = 1$) – which we hypothesize might be reversed given more tuning – our RNFs-ML (exact) version is the best performing model, yet again highlighting the importance of including the change-of-volume term in the objective.

We further evaluate the performance of RNFs for OoD detection. Nalisnick et al. [41] pointed out that square NFs trained on FMNIST assign higher likelihoods to MNIST than they do to FMNIST. While there has been research attempting to fix this puzzling behaviour [1, 2, 10, 51], to the best of our knowledge no method has managed to correct it using only likelihoods of trained models. Figure 2 shows that RNFs remedy this phenomenon, and that models trained on FMNIST assign higher test likelihoods to FMNIST than to MNIST. This correction does not come at the cost of strange behaviour now emerging in the opposite direction (i.e. when training on MNIST, see Appendix G for a histogram). Table 3 quantifies these results (arrows point from in-distribution datasets to OoD ones) with the accuracy of a decision stump using only log-likelihood, and we can see that the best-performing RNFs models essentially solve this OoD task. While we leave a formal explanation of this result for future work, we believe this discovery highlights the importance of properly specifying models and of ensuring the use of appropriate inductive biases, in this case low intrinsic dimensionality of the observed data. The strong performance of RNFs-TS here seems to indicate that this is a property of RNFs rather than of our ML training method specifically, although our exact approach is still used to compute these log-likelihoods at test time. We include additional results on OoD detection using reconstruction errors – along with a discussion – in Appendix G, where we found the opposite unexpected behaviour: FMNIST always has smaller reconstruction errors, regardless of which dataset was used for training.

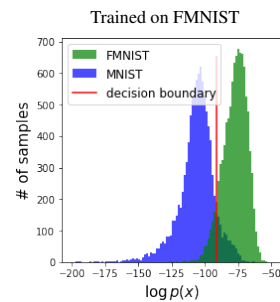


Figure 2: OoD detection with RNFs-ML (exact).

6 Scope and Limitations

In this paper we address the dimensionality-based misspecification of square NFs while properly using maximum likelihood as the training objective, thus providing an advancement in the training of RNFs. Our methods however remain *topologically* misspecified: even though we can better address dimensionality, we can currently only learn manifolds homeomorphic to $\mathbb{R}^d$. For example, one could conceive of the MNIST manifold as consisting of 10 connected components (one per digit), which cannot be learned by f_θ . It is nonetheless worth noting that this limitation is shared by other deep generative modelling approaches, for example GANs [19] result in connected supports (since the image of a connected set under a continuous function is connected). We observed during training in image data that the residuals of CG were not close to $\mathbf{0}$ numerically, even after d steps, indicating poor conditioning and thus possible numerical non-invertibility of the matrix $J_\theta^\top J_\theta$. We hypothesize that this phenomenon is caused by topological mismatch, which we also conjecture affects us more than the baseline as our CG-obtained (or from the exact method) gradients might point in an inaccurate direction. We thus expect our methods in particular to benefit from improved research on making flows match the target topology, for example via continuous indexing [11].

Additionally, while we have successfully scaled likelihood-based training of RNFs far beyond current capabilities, our methods – even the stochastic one – remain computationally expensive for higher dimensions, and further computational gains remain an open problem. We also attempted OoD detection on CIFAR-10 against the SVHN dataset [42], and found that neither RNFs-ML nor RNFs-TS has good performance, although anecdotally we may have at least improved on the situation outlined by Nalisnick et al. [41]. We hypothesize these results might be either caused by topological mismatch, or corrected given more tuning.

7 Conclusions and Broader Impact

In this paper we argue for the importance of likelihood-based training of rectangular flows, and introduce two methods allowing to do so. We study the benefits of our methods, and empirically show that they are preferable to current alternatives. Given the methodological nature of our contributions, we do not foresee our work having any negative ethical implications or societal consequences.

Acknowledgements

We thank Brendan Ross, Jesse Cresswell, and Maksims Volkovs for useful comments and feedback. We would also like to thank Rob Cornish for the excellent CIFs codebase upon which our code is built, and Emile Mathieu for plotting suggestions. GP and JPC are supported by the Simons Foundation, McKnight Foundation, the Grossman Center, and the Gatsby Charitable Trust.

References

- [1] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy. Deep variational information bottleneck. *ICLR*, 2017.
- [2] A. A. Alemi, I. Fischer, and J. V. Dillon. Uncertainty in the variational information bottleneck. *arXiv preprint arXiv:1807.00906*, 2018.
- [3] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. Automatic differentiation in machine learning: a survey. *Journal of machine learning research*, 18, 2018.
- [4] J. Behrmann, W. Grathwohl, R. T. Chen, D. Duvenaud, and J.-H. Jacobsen. Invertible residual networks. In *International Conference on Machine Learning*, pages 573–582. PMLR, 2019.
- [5] J. Behrmann, P. Vicol, K.-C. Wang, R. Grosse, and J.-H. Jacobsen. Understanding and mitigating exploding inverses in invertible neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 1792–1800. PMLR, 2021.
- [6] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

- [7] D. P. Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.
- [8] J. Brehmer and K. Cranmer. Flows for simultaneous manifold learning and density estimation. In *Advances in Neural Information Processing Systems*, volume 33, 2020.
- [9] R. T. Q. Chen, J. Behrmann, D. K. Duvenaud, and J.-H. Jacobsen. Residual flows for invertible generative modeling. In *Advances in Neural Information Processing Systems*, volume 32, 2019.
- [10] H. Choi, E. Jang, and A. A. Alemi. Waic, but why? generative ensembles for robust anomaly detection. *arXiv preprint arXiv:1810.01392*, 2018.
- [11] R. Cornish, A. Caterini, G. Deligiannidis, and A. Doucet. Relaxing bijectivity constraints with continuously indexed normalising flows. In *International Conference on Machine Learning*, pages 2133–2143. PMLR, 2020.
- [12] E. Cunningham, R. Zabounidis, A. Agrawal, I. Fiterau, and D. Sheldon. Normalizing flows across dimensions. *arXiv preprint arXiv:2006.13070*, 2020.
- [13] B. Dai and D. Wipf. Diagnosing and enhancing vae models. *ICLR*, 2019.
- [14] L. Dinh, D. Krueger, and Y. Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.
- [15] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real nvp. *ICLR*, 2017.
- [16] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios. Neural spline flows. In *Advances in Neural Information Processing Systems*, volume 32, 2019.
- [17] J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *Advances in Neural Information Processing Systems*, 31, 2018.
- [18] M. C. Gemici, D. Rezende, and S. Mohamed. Normalizing flows on riemannian manifolds. *arXiv preprint arXiv:1611.02304*, 2016.
- [19] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*, pages 2672–2680, 2014.
- [20] I. Han, D. Malioutov, and J. Shin. Large-scale log-determinant computation through stochastic chebyshev expansions. In *International Conference on Machine Learning*, pages 908–917. PMLR, 2015.
- [21] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, volume 30, 2017.
- [22] C.-W. Huang, R. T. Chen, C. Tsirigotis, and A. Courville. Convex potential flows: Universal probability distributions with optimal transport and convex optimization. *ICLR*, 2021.
- [23] M. F. Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.
- [24] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.
- [25] H. Kahn. *Use of different Monte Carlo sampling techniques*. Rand Corporation, 1955.
- [26] W. Karush. Minima of functions of several variables with inequalities as side constraints. *M. Sc. Dissertation. Dept. of Mathematics, Univ. of Chicago*, 1939.
- [27] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

- [28] D. P. Kingma and P. Dhariwal. Glow: generative flow with invertible 1×1 convolutions. In *Advances in Neural Information Processing Systems*, volume 31, 2018.
- [29] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *ICLR*, 2014.
- [30] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, volume 30, 2016.
- [31] I. Kobyzev, S. Prince, and M. Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.
- [32] S. G. Krantz and H. R. Parks. *Geometric integration theory*. Springer Science & Business Media, 2008.
- [33] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.
- [34] H. W. Kuhn and A. Tucker. W., 1951, "nonlinear programming,". In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*(University of California Press, Berkeley, CA), volume 481492, 1951.
- [35] A. Kumar, B. Poole, and K. Murphy. Regularized autoencoders via relaxed injective probability flow. In *International Conference on Artificial Intelligence and Statistics*, pages 4292–4301. PMLR, 2020.
- [36] Y. LeChun. The mnist database of handwritten digits, 1998. URL <http://yann.lecun.com/exdb/mnist>, 1998.
- [37] G. Loaiza-Ganem, Y. Gao, and J. P. Cunningham. Maximum entropy flow networks. *ICLR*, 2017.
- [38] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *ICLR*, 2019.
- [39] C. Lu, J. Chen, C. Li, Q. Wang, and J. Zhu. Implicit normalizing flows. *ICLR*, 2021.
- [40] E. Mathieu and M. Nickel. Riemannian continuous normalizing flows. In *Advances in Neural Information Processing Systems*, volume 33, 2020.
- [41] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan. Do deep generative models know what they don’t know? *ICLR*, 2019.
- [42] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011.
- [43] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.
- [44] D. Oktay, N. McGreivy, J. Aduol, A. Beatson, and R. P. Adams. Randomized automatic differentiation. *ICLR*, 2021.
- [45] G. Papamakarios, T. Pavlakou, and I. Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, volume 30, 2017.
- [46] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*, 2019.
- [47] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimeshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems* 32, volume 32. 2019.
- [48] X. Pennec. Intrinsic statistics on riemannian manifolds: Basic tools for geometric measurements. *Journal of Mathematical Imaging and Vision*, 25(1):127–154, 2006.

- [49] K. B. Petersen and M. S. Pedersen. The matrix cookbook, Oct. 2008. URL <http://www2.imm.dtu.dk/pubdb/p.php?3274>. Version 20081110.
- [50] A. Potapczynski, L. Wu, D. Biderman, G. Pleiss, and J. P. Cunningham. Bias-free scalable gaussian processes via randomized truncations. *International Conference on Machine Learning*, to appear, 2021.
- [51] J. Ren, P. J. Liu, E. Fertig, J. Snoek, R. Poplin, M. Depristo, J. Dillon, and B. Lakshminarayanan. Likelihood ratios for out-of-distribution detection. In *Advances in Neural Information Processing Systems*, volume 32, 2019.
- [52] D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR, 2015.
- [53] D. J. Rezende, G. Papamakarios, S. Racaniere, M. Albergo, G. Kanwar, P. Shanahan, and K. Cranmer. Normalizing flows on tori and spheres. In *International Conference on Machine Learning*, pages 8083–8092. PMLR, 2020.
- [54] B. L. Ross and J. C. Cresswell. Tractable density estimation on learned manifolds with conformal embedding flows. *arXiv preprint arXiv:2106.05275*, 2021.
- [55] J. R. Shewchuk et al. An introduction to the conjugate gradient method without the agonizing pain, 1994.
- [56] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.
- [57] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.
- [58] M. Zhang, P. Hayes, T. Bird, R. Habib, and D. Barber. Spread divergence. In *International Conference on Machine Learning*, pages 11106–11116. PMLR, 2020.

Checklist

1. For all authors...
 - (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [\[Yes\]](#)
 - (b) Did you describe the limitations of your work? [\[Yes\]](#) See Section 6.
 - (c) Did you discuss any potential negative societal impacts of your work? [\[Yes\]](#) Although we do not expect any. See Section 7.
 - (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [\[Yes\]](#)
2. If you are including theoretical results...
 - (a) Did you state the full set of assumptions of all theoretical results? [\[N/A\]](#)
 - (b) Did you include complete proofs of all theoretical results? [\[N/A\]](#)
3. If you ran experiments...
 - (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [\[Yes\]](#) As supplemental material, and we will release the code upon publication.
 - (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [\[Yes\]](#) See Appendix G.
 - (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [\[Yes\]](#) When computationally tractable to repeat experiments. See Section 5.
 - (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [\[Yes\]](#) See Appendix G.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
 - (a) If your work uses existing assets, did you cite the creators? [\[Yes\]](#) See the README in our code and the references in our paper.
 - (b) Did you mention the license of the assets? [\[No\]](#) Because the code we based ours on does not seem to have had a license. See the README in our code.
 - (c) Did you include any new assets either in the supplemental material or as a URL? [\[Yes\]](#) Just code.
 - (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [\[No\]](#) But we are using publicly available and widely used datasets only.
 - (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [\[No\]](#) But only because we do not believe it does.
5. If you used crowdsourcing or conducted research with human subjects...
 - (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [\[N/A\]](#)
 - (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [\[N/A\]](#)
 - (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [\[N/A\]](#)

A Injective Change-of-Variable Formula and Stacking Injective Flows

We first derive (5) from (3). By the chain rule, we have:

$$\mathbf{J}[g_\phi] \left(g_\phi^\dagger(x) \right) = \mathbf{J}[f_\theta] \left(f_\theta^\dagger(x) \right) \mathbf{J}[h_\eta] \left(g_\phi^\dagger(x) \right). \quad (13)$$

The Jacobian-transpose Jacobian term in (3) thus becomes:

$$\begin{aligned} & \left| \det \mathbf{J}[g_\phi]^\top \left(g_\phi^\dagger(x) \right) \mathbf{J}[g_\phi] \left(g_\phi^\dagger(x) \right) \right|^{-1/2} \\ &= \left| \det \mathbf{J}[h_\eta]^\top \left(g_\phi^\dagger(x) \right) \mathbf{J}[f_\theta]^\top \left(f_\theta^\dagger(x) \right) \mathbf{J}[f_\theta] \left(f_\theta^\dagger(x) \right) \mathbf{J}[h_\eta] \left(g_\phi^\dagger(x) \right) \right|^{-1/2} \\ &= \left| \det \mathbf{J}[h_\eta]^\top \left(g_\phi^\dagger(x) \right) \right|^{-1/2} \left| \det \mathbf{J}[f_\theta]^\top \left(f_\theta^\dagger(x) \right) \mathbf{J}[f_\theta] \left(f_\theta^\dagger(x) \right) \right|^{-1/2} \left| \det \mathbf{J}[h_\eta] \left(g_\phi^\dagger(x) \right) \right|^{-1/2} \\ &= \left| \det \mathbf{J}[h_\eta] \left(g_\phi^\dagger(x) \right) \right|^{-1} \left| \det \mathbf{J}[f_\theta]^\top \left(f_\theta^\dagger(x) \right) \mathbf{J}[f_\theta] \left(f_\theta^\dagger(x) \right) \right|^{-1/2}, \end{aligned} \quad (14)$$

where the second equality follows from the fact that $\mathbf{J}[h_\eta]^\top (g_\phi^\dagger(x))$, $\mathbf{J}[f_\theta]^\top (f_\theta^\dagger(x))\mathbf{J}[f_\theta](f_\theta^\dagger(x))$, and $\mathbf{J}[h_\eta](g_\phi^\dagger(x))$ are all square $d \times d$ matrices; and the third equality follows because determinants are invariant to transpositions. The observation that the three involved matrices are square is the reason behind why we can decompose the change-of-variable formula for g_ϕ as applying first the change-of-variable formula for h_η , and then applying it for f_θ .

This property, unlike in the case of square flows, does not always hold. That is, the change-of-variable formula for a composition of injective transformations is not necessarily equivalent to applying the injective change-of-variable formula twice. To see this, consider the case where $g_1 : \mathbb{R}^d \rightarrow \mathbb{R}^{d_2}$ and $g_2 : \mathbb{R}^{d_2} \rightarrow \mathbb{R}^D$ are injective, where $d < d_2 < D$ and let $g = g_2 \circ g_1$. Clearly g is injective by construction, and thus the determinant from its change-of-variable formula at a point $z \in \mathbb{R}^d$ is given by:

$$\det \mathbf{J}[g]^\top(z) \mathbf{J}[g](z) = \det \mathbf{J}[g_1]^\top(z) \mathbf{J}[g_2]^\top(g_1(z)) \mathbf{J}[g_2](g_1(z)) \mathbf{J}[g_1](z), \quad (15)$$

where now $\mathbf{J}[g_1](z) \in \mathbb{R}^{d_2 \times d}$ and $\mathbf{J}[g_2](g_1(z)) \in \mathbb{R}^{D \times d_2}$. Unlike the determinant from (14), this determinant cannot be easily decomposed into a product of determinants since the involved matrices are not all square. In particular, (15) need not match:

$$\det \mathbf{J}[g_1]^\top(z) \mathbf{J}[g_1](z) \cdot \det \mathbf{J}[g_2]^\top(g_1(z)) \mathbf{J}[g_2](g_1(z)), \quad (16)$$

which would be the determinant terms from applying the change-of-variable formula twice. Note that this observation does not imply that a flow like g could not be trained with our method, it simply implies that the $\det \mathbf{J}[g]^\top(z) \mathbf{J}[g](z)$ term has to be considered as a whole, and not decomposed into separate terms. It is easy to verify that in general, only an initial d -dimensional square flow can be separated from the overall Jacobian-transpose-Jacobian determinant.

B Conjugate Gradients

We outline the CG algorithm in Algorithm 1, whose output we write as $\text{CG}(A; \epsilon)$ in the main manuscript (we omit the dependance on the tolerance δ for notational simplicity). Note that CG does not need access to A , just a matrix-vector product routine against A , $\text{mvp}_A(\cdot)$. If A is symmetric positive definite, then CG converges in at most d steps, i.e. its output matches $A^{-1}\epsilon$ and the corresponding residual is 0, and CG uses thus at most d calls to $\text{mvp}_A(\cdot)$. This convergence holds mathematically, but can be violated numerically if A is ill-conditioned, which is why the $\tau < d$ condition is added in the while loop.

Algorithm 1: CG

Input : $\text{mvp_A}(\cdot)$, function for matrix-vector products against $A \in \mathbb{R}^{d \times d}$

$\epsilon \in \mathbb{R}^d$

$\delta \geq 0$, tolerance

Output : $A^{-1}\epsilon$

$u_0 \leftarrow \mathbf{0} \in \mathbb{R}^d$ // current solution

$r_0 \leftarrow -\epsilon$ // current residual

$q_0 \leftarrow r_0$

$\tau \leftarrow 0$

while $\|r_\tau\|_2 > \delta$ **and** $\tau < d$ **do**

$v_\tau \leftarrow \text{mvp_A}(q_\tau)$

$\alpha_\tau \leftarrow (r_\tau^\top r_\tau) / (q_\tau^\top v_\tau)$

$u_{\tau+1} \leftarrow u_\tau + \alpha_\tau q_\tau$

$r_{\tau+1} \leftarrow r_\tau - \alpha_\tau v_\tau$

$\beta_\tau \leftarrow (r_{\tau+1}^\top r_{\tau+1}) / (r_\tau^\top r_\tau)$

$q_{\tau+1} \leftarrow r_{\tau+1} + \beta_\tau q_\tau$

$\tau \leftarrow \tau + 1$

end

return u_τ

C Automatic Differentiation

Here we summarize the relevant properties from forward- and backward-mode automatic differentiation (AD) which we use in the main manuscript. Let f be the composition of smooth functions $f_1, \dots, f_L$, i.e. $f = f_L \circ f_{L-1} \circ \dots \circ f_1$. For example, in our setting this function could be f_θ , so that $f_1 = \text{pad}$, and the rest of the functions could be coupling layers from a D -dimensional square flow (or the functions whose compositions results in the coupling layers). By the chain rule, the Jacobian of f is given by:

$$\mathbf{J}[f](z) = \mathbf{J}[f_L](\bar{f}_{L-1}(z)) \cdots \mathbf{J}[f_2](\bar{f}_1(z)) \mathbf{J}[f_1](z), \quad (17)$$

where $\bar{f}_l := f_l \circ f_{l-1} \circ \dots \circ f_1$ for $l = 1, 2, \dots, L-1$. Forward-mode AD computes products from right to left, and is thus efficient for computing jvp operations. Computing $\mathbf{J}[f](z)\epsilon$ is thus obtained by performing L matrix-vector multiplications, one against each of the Jacobians on the right hand side of (17). Backward-mode AD computes products from left to right, and would thus result in significantly more inefficient jvp evaluations involving $L-1$ matrix-matrix products, and a single matrix-vector product. Analogously, backward-mode AD computes vjps of the form $v^\top \mathbf{J}[f](z)$ efficiently, using L vector-matrix products, while forward-mode AD would require $L-1$ matrix-matrix products and a single vector-matrix product.

Typically, the cost of evaluating a matrix-vector or vector-matrix product against $\mathbf{J}[f_{l+1}](\bar{f}_l)$ (or $\mathbf{J}[f_1](z)$) is the same as computing $\bar{f}_{l+1}(z)$ from $\bar{f}_l(z)$, i.e. the cost of evaluating f_{l+1} (or the cost of evaluating f_1 in the case of $\mathbf{J}[f_1](z)$) [3]. jvp and vjp computations thus not only have the same computational cost, but this cost is also equivalent to a forward pass, i.e. computing f .

When computing f , obtaining a jvp with forward-mode AD adds the same memory cost as another computation of f since intermediate results do not have to be stored. That is, in order to compute $\mathbf{J}[f_l](\bar{f}_{l-1}(z)) \cdots \mathbf{J}[f_1](z)\epsilon$, we only need to store $\mathbf{J}[f_{l-1}](\bar{f}_{l-2}(z)) \cdots \mathbf{J}[f_1](z)\epsilon$ and $\bar{f}_{l-1}(z)$ (which has to be stored anyway for computing f) in memory. On the other hand, computing a vjp with backward-mode AD has a higher memory cost: One has to first compute $\bar{f}$ and store all the intermediate $\bar{f}_l(z)$ (along with z), since computing $v^\top \mathbf{J}[f_L](\bar{f}_{L-1}(z)) \cdots \mathbf{J}[f_l](\bar{f}_{l-1}(z))$ from $v^\top \mathbf{J}[f_L](\bar{f}_{L-1}(z)) \cdots \mathbf{J}[f_{l+1}](\bar{f}_l(z))$ requires having $\bar{f}_{l-1}(z)$ in memory.

In practice, we use PyTorch's implementation of backpropagation to compute vjps, and as mentioned in the main manuscript, we use our own implementation of forward-mode AD for jvps. We achieve this by having every layer and non-linearity f_l in our networks not only take an input x , but also a vector ϵ of the same length as x ; and not just output the usual output $f_l(x)$, but also $\mathbf{J}[f_l](x)\epsilon$ (for linear layers, this is equivalent to applying the layer without the bias to ϵ , and for element-wise non-linearities $\mathbf{J}[f_l](x)$ is a straightforward-to-compute diagonal matrix and so $\mathbf{J}[f_l](x)\epsilon$ can be obtained though element-wise products).

D Summary of our Proposed Methods

We summarize our methods for computing/estimating the gradient of the log determinant arising in maximum likelihood training of rectangular flows. Algorithm 2 shows the exact method, where $\text{jvp}(f, z, \epsilon)$ denotes computing $\mathbf{J}[f](z)\epsilon$ using forward-mode AD, and $\epsilon_i \in \mathbb{R}^d$ is the i -th standard basis vector, i.e. a one-hot vector with a 1 on its i -th coordinate. Note that $\partial/\partial\theta \log \det A_\theta$ is computed using backpropagation. The `for` loop is easily parallelized in practice. For density evaluation, rather than returning $\partial/\partial\theta \log \det A_\theta$, the output of Algorithm 2 becomes $\log \det A_\theta$.

Algorithm 3 shows our stochastic method, where $\text{vjp}(f, z, v)$ denotes $v^\top \mathbf{J}[f](z)$ computed through backward-mode AD. As mentioned in the main manuscript, the ϵ_k vectors can be sampled from any zero-mean, identity-covariance distribution and not just a Gaussian. For added clarity, we change the CG notation and use $\text{CG}(\text{mvp_A}(\cdot), \epsilon, \delta)$ to denote the output of the conjugate gradients method. Backpropagation is once again used to compute $\partial/\partial\theta(s_\theta/K)$, and the `for` loop is again parallelized. Note that, unlike Algorithm 2, s_θ/K is not a valid log determinant estimate, and Algorithm 3 should only be used for gradient estimates during training, and not density evaluation at test time.

Algorithm 2: Exact method

Input : $f_\theta : \mathbb{R}^d \rightarrow \mathbb{R}^D$
 $x \in \mathbb{R}^D$
Output : $\partial/\partial\theta \log \det J_\theta^\top J_\theta$
 $z \leftarrow f_\theta^\dagger(x)$
for $i = 1, \dots, d$ **do**
 $v_i \leftarrow \text{jvp}(f_\theta, z, \epsilon_i)$
end
 $J_\theta \leftarrow (v_1 | \dots | v_d)$
 $A_\theta \leftarrow J_\theta^\top J_\theta$
return $\partial/\partial\theta \log \det A_\theta$

Algorithm 3: Stochastic method

Input : $f_\theta : \mathbb{R}^d \rightarrow \mathbb{R}^D$
 $x \in \mathbb{R}^D$
 $K \in \mathbb{N}_+$
 $\delta \geq 0$, CG tolerance
Output : Unbiased stochastic approximation of $\partial/\partial\theta \log \det J_\theta^\top J_\theta$
 $z \leftarrow f_\theta^\dagger(x)$
 $\text{mvp_A}(\cdot) \leftarrow \text{vjp}(f_\theta, z, \text{jvp}(f_\theta, z, \cdot))^\top$
 $s_\theta \leftarrow 0$
for $k = 1, \dots, K$ **do**
 $\epsilon_k \sim \mathcal{N}(0, I_d)$
 $s_\theta \leftarrow s_\theta + \text{stop_gradient}(\text{CG}(\text{mvp_A}(\cdot), \epsilon_k, \delta)^\top) \cdot \text{mvp_A}(\epsilon_k)$
end
return $\partial/\partial\theta(s_\theta/K)$

E Batch Normalization

We now explain the issues that arise when combining batch normalization with vjps. These issues arise not only in our setting, but every time backward-mode AD has to be called to compute or approximate the gradient of the determinant term. We consider the case with a batch of size 2, x_1 and x_2 , as it exemplifies the issue and the notation becomes simpler. Consider applying f_θ (without batch normalization) to each element in the batch, which we denote with the batch function F_θ :

$$F_\theta(x_1, x_2) := (f_\theta(x_1), f_\theta(x_2)). \quad (18)$$

The Jacobian of F_θ clearly has a block-diagonal structure:

$$\mathbf{J}[F_\theta](x_1, x_2) = \begin{pmatrix} \mathbf{J}[f_\theta](x_1) & \mathbf{0} \\ \mathbf{0} & \mathbf{J}[f_\theta](x_2) \end{pmatrix}. \quad (19)$$

This structure implies that relevant computations such as `vjps`, `jvps`, and determinants parallelize over the batch:

$$(v_1, v_2)^\top \mathbf{J}[F_\theta](x_1, x_2) = (v_1^\top \mathbf{J}[f_\theta](x_1), v_2^\top \mathbf{J}[f_\theta](x_2)) \quad (20)$$

$$\mathbf{J}[F_\theta](x_1, x_2) \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \end{pmatrix} = \begin{pmatrix} \mathbf{J}[f_\theta](x_1)\epsilon_1 \\ \mathbf{J}[f_\theta](x_2)\epsilon_2 \end{pmatrix}$$

$$\det \mathbf{J}[F_\theta]^\top(x_1, x_2) \mathbf{J}[F_\theta](x_1, x_2) = \det \mathbf{J}[f_\theta]^\top(x_1) \mathbf{J}[f_\theta](x_1) \det \mathbf{J}[f_\theta]^\top(x_2) \mathbf{J}[f_\theta](x_2).$$

In contrast, when using batch normalization, the resulting computation $F_\theta^{BN}(x_1, x_2)$ does not have a block-diagonal Jacobian, and thus this parallelism over the batch breaks down, in other words:

$$(v_1, v_2)^\top \mathbf{J} \left[F_\theta^{BN} \right] (x_1, x_2) \neq (v_1^\top \mathbf{J}[f_\theta](x_1), v_2^\top \mathbf{J}[f_\theta](x_2)) \quad (21)$$

$$\mathbf{J} \left[F_\theta^{BN} \right] (x_1, x_2) \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \end{pmatrix} \neq \begin{pmatrix} \mathbf{J}[f_\theta](x_1)\epsilon_1 \\ \mathbf{J}[f_\theta](x_2)\epsilon_2 \end{pmatrix}$$

$$\det \mathbf{J} \left[F_\theta^{BN} \right]^\top (x_1, x_2) \mathbf{J} \left[F_\theta^{BN} \right] (x_1, x_2) \neq \det \mathbf{J}[f_\theta]^\top(x_1) \mathbf{J}[f_\theta](x_1) \det \mathbf{J}[f_\theta]^\top(x_2) \mathbf{J}[f_\theta](x_2),$$

where the above $\neq$ signs should be interpreted as “not generally equal to” rather than always not equal to, as equalities could hold coincidentally in rare cases.

In square flow implementations, AD is never used to obtain any of these quantities, and the Jacobian log determinants are explicitly computed for each element in the batch. In other words, this batch dependence is ignored in square flows, both in the log determinant computation, and when back-propagating through it. Elaborating on this point, AD is only used to backpropagate (with respect to θ) over this explicit computation. If AD was used on F_θ^{BN} to construct the matrices and we then computed the corresponding log determinants, the results would not match with the explicitly computed log determinants: The latter would be equivalent to using batch normalization with a `stop_gradient` operation *with respect to* (x_1, x_2) *but not with respect to* θ , while the former would use no `stop_gradient` whatsoever. Unfortunately, this partial `stop_gradient` operation only with respect to inputs but not parameters is not available in commonly used AD libraries. While our custom implementation of `jvps` can be easily “hard-coded” to have this behaviour, doing so for `vjps` would require significant modifications to PyTorch. We note that this is *not* a fundamental limitation and that these modifications could be done to obtain `vjps` that behave as expected with a low-level re-implementation of batch normalization, but these fall outside of the scope of our paper. Thus, in the interest of performing computations in a manner that remains consistent with what is commonly done for square flows and that allows fair comparisons of our exact and stochastic methods, we avoid using batch normalization.

F FID and FID-like Scores

For a given dataset $\{x_1, \dots, x_n\} \subset \mathbb{R}^D$ and a set of samples generated by a model $\{x_1^{(g)}, \dots, x_m^{(g)}\} \subset \mathbb{R}^D$, along with a statistic $T : \mathbb{R}^D \rightarrow \mathbb{R}^r$, the empirical means and covariances are given by:

$$\hat{\mu} := \frac{1}{n} \sum_{i=1}^n T(x_i), \quad \hat{\Sigma} := \frac{1}{n-1} \sum_{i=1}^n (T(x_i) - \hat{\mu})(T(x_i) - \hat{\mu})^\top \quad (22)$$

$$\hat{\mu}^{(g)} := \frac{1}{m} \sum_{i=1}^m T(x_i^{(g)}), \quad \hat{\Sigma}^{(g)} := \frac{1}{m-1} \sum_{i=1}^m (T(x_i^{(g)}) - \hat{\mu}^{(g)})(T(x_i^{(g)}) - \hat{\mu}^{(g)})^\top. \quad (23)$$

The FID score takes T as the last hidden layer of a pre-trained inception network, and evaluates generated sample quality by comparing generated moments against data moments. This comparison is done with the squared Wasserstein-2 distance between Gaussians with corresponding moments, which is given by:

$$\left\| \hat{\mu} - \hat{\mu}^{(g)} \right\|_2^2 + \text{tr} \left(\hat{\Sigma} + \hat{\Sigma}^{(g)} - 2 \left(\hat{\Sigma} \hat{\Sigma}^{(g)} \right)^{1/2} \right), \quad (24)$$

which is 0 if and only if the moments match. Our proposed FID-like score for tabular data is computed the exact same way, except no inception network is used. Instead, we simply take T to be the identity, $T(x) = x$.

Table 4: Training times in seconds, “ $K > 1$ ” means $K = 10$ for tabular data and $K = 4$ for images.

Dataset	RNFs-ML (exact)		RNFs-ML ($K = 1$)		RNFs-ML ($K > 1$)		RNFs-TS	
	EPOCH	TOTAL	EPOCH	TOTAL	EPOCH	TOTAL	EPOCH	TOTAL
POWER	53.8	4.13e3	67.4	6.76e3	136	1.14e4	45.1	3.83e3
GAS	37.3	2.51e3	62.7	4.51e3	80.1	5.24e3	43.2	3.49e3
HEPMASS	143	1.01e4	146	8.28e3	159	1.20e4	29.1	2.42e3
MINIBOONE	49.3	4.16e3	26.3	2.01e3	29.8	2.94e3	4.61	481
MNIST	2.40e3	2.59e5	1.71e3	1.57e5	3.03e3	3.20e5	2.13e2	3.90e4
FMNIST	2.34e3	2.59e5	1.72e3	1.50e5	3.15e3	2.10e5	1.04e2	1.11e4

G Experimental Details

First we will comment on hyperparameters/architectural choices shared across experiments. The D -dimensional square flow that we use, as mentioned in the main manuscript, is a RealNVP network [15]. In all cases, we use the ADAM [27] optimizer and train with early stopping against some validation criterion specified for each experiment separately and discussed further in each of the relevant subsections below. We use no weight decay. We also do not use batch normalization in any experiments for the reasons mentioned above in Appendix E. We use a standard Gaussian on d dimensions as p_Z in all experiments.

Compute We ran our two-dimensional experiments on a Lenovo T530 laptop with an Intel i5 processor, with negligible training time per epoch. We ran the tabular data experiments on a variety of NVIDIA GeForce GTX GPUs on a shared cluster: we had, at varying times, access to 1080, 1080 Ti, and 2080 Ti models, but never access to more than six cards in total at once. For the image experiments, we had access to a 32GB-configuration NVIDIA Tesla v100 GPU. We ran each of the tabular and image experiments on a single card at a time, except for the image experiments for the RNFs-ML (exact) and ($K = 10$) models which we parallelized over four cards.

Table 4 includes training times for all of our experiments. Since we used FID-like and FID scores for early stopping, we include both per-epoch and total times. Per epoch times of RNFs-ML exclude epochs where the Jacobian-transpose-Jacobian log determinant is annealed with a 0 weight, although we include time added from this portion of training into the total time cost. Note throughout this section we also consider one epoch of the two-step baseline procedure to be one full pass through the data training the likelihood term, and then one full pass through the data training the reconstruction term.

G.1 Simulated Data

The data for this experiment is simulated from a von Mises distribution centred at $\frac{\pi}{2}$ projected onto a circle of radius 1. We randomly generate 10,000 training data points and train with batch sizes of 1,000. We use 1,000 points for validation, performing early stopping using the value of the full objective and halting training when we do not see any validation improvement for 50 epochs. We create visualizations in Figure 1 by taking 1,000 grid points equally-spaced between -3 and 3 as the low-dimensional space, project these into higher dimensions by applying the flow g_ϕ , and then assign density to these points using the injective change-of-variable formula (2). In this low-dimensional example, we use the full Jacobian-transpose-Jacobian which ends up just being a scalar as $d = 1$. We commence likelihood annealing (when active) on the 500-th training epoch and end up with a full likelihood term by the 1000-th.

For the D -dimensional square flow f_θ , we used a 5-layer RealNVP model, with each layer having a fully-connected coupler network of size 2×10 , i.e. 2 hidden layers each of size 10, outputting the shift and (log) scale values. The baseline additionally uses a simple shift-and-scale transformation in d -dimensional space as h_η ; we simply use the identity map for h_η in this simple example.

We perform slightly different parameter sweeps for the two methods based on preliminary exploration. For the baseline two-step procedure, we perform runs over the following grid:

- Learning rate: 10^{-3} , 10^{-4} .

- Regularization parameter (β): 10, 50, 100, 200, 1,000, 10,000 (which for this method is equivalent to having a separate learning rate for the regularization objective).
- Likelihood annealing: True or False.

For our method, we search over the following, although noting that our method was stable at the higher learning rate of 10^{-3} :

- Learning rate: 10^{-3} , 10^{-4} .
- Regularization parameter (β): 10, 50, 200.
- Likelihood annealing: True or False.

Empirically we found the two-step baseline performed better with the higher regularization, which also agrees with the hyperparameter settings from their paper. Note that we have searched over 2 times as many runs for the baseline and still obtain better runs with our approach.

Divergences on RNFs-TS between our codebase and the implementation of Brehmer and Cranmer [8] Although we were able to replicate the baseline RNF-TS method, there were some different choices made in the codebase of the baseline method (available here: <https://github.com/johannbrehmer/manifold-flow>), which we outline below:

- The baseline was trained for 120 epochs and then selects the model with best validation score, whereas we use early stopping over an (essentially) unlimited number of epochs.
- The baseline weights the reconstruction term with a factor of 100 and the likelihood term with a factor of 0.1. This is equivalent in our codebase to setting $\beta = 1,000$, and lowering the learning rate by a factor of 10.
- The baseline uses cosine annealing of the learning rate, which we do not use.
- The baseline includes a sharp Normal base distribution on the pulled-back padded coordinates. We neglected to include this as it isn't mentioned in the paper and can end up resulting in essentially a square flow construction.
- The baseline uses the ADAMW optimizer [38] to fix issues with weight decay within ADAM (which they also use). We stick with standard ADAM as we do not use weight decay.
- The baseline flow reparametrizes the scale s of the RealNVP network as $s = \sigma(\tilde{s} + 2) + 10^{-3}$, where $\tilde{s}$ is the unconstrained scale and σ is the sigmoid function, but this constrains the scale to be less than $1 + 10^{-3}$. This appears to be done for stability of the transformation (cf. the ResNets below). We instead use the standard parametrization of $s = \exp(\tilde{s})$ as the fully-connected networks appear to be adequately stable.
- The Baseline uses ResNets with ReLU activation of size 2×100 as the affine coupling networks. We use MLPs with tanh activation function instead.
- The baseline uses a dataset which is not strictly on a manifold. The radius of a point on the circle is sampled from $\mathcal{N}(1, 0.01^2)$. We use a strictly one-dimensional distribution instead with a von Mises distribution on the angle as noted above.

In general, we favoured more standard and simpler choices for modelling the circle, outside of the likelihood annealing which is non-standard.

Densities of all runs We note that, while the results reported in the main manuscript are representative of common runs, both for RNFs-ML (exact) and RNFs-TS; not every single run of RNFs-ML (exact) obtained results as good as the ones from the main manuscript. Similarly, some runs of RNFs-TS recovered better likelihoods than the one from the main manuscript. We emphasize again that the results reported on the main manuscript are the most common ones: most RNFs-ML (exact) runs correctly recovered both the manifold and the distribution on it, and most RNFs-TS runs recovered only the manifold correctly. For completeness, we include in Figures 3, 4, and 5 all the runs we obtained, where it becomes evident that RNFs-ML consistently outperforms RNFs-TS across runs and hyperparameter values. However we do note in Figure 3b that we get the perpendicular effect that Brehmer and Cranmer [8] predicted might happen if optimizing the full objective, although this is far from typical of our results.

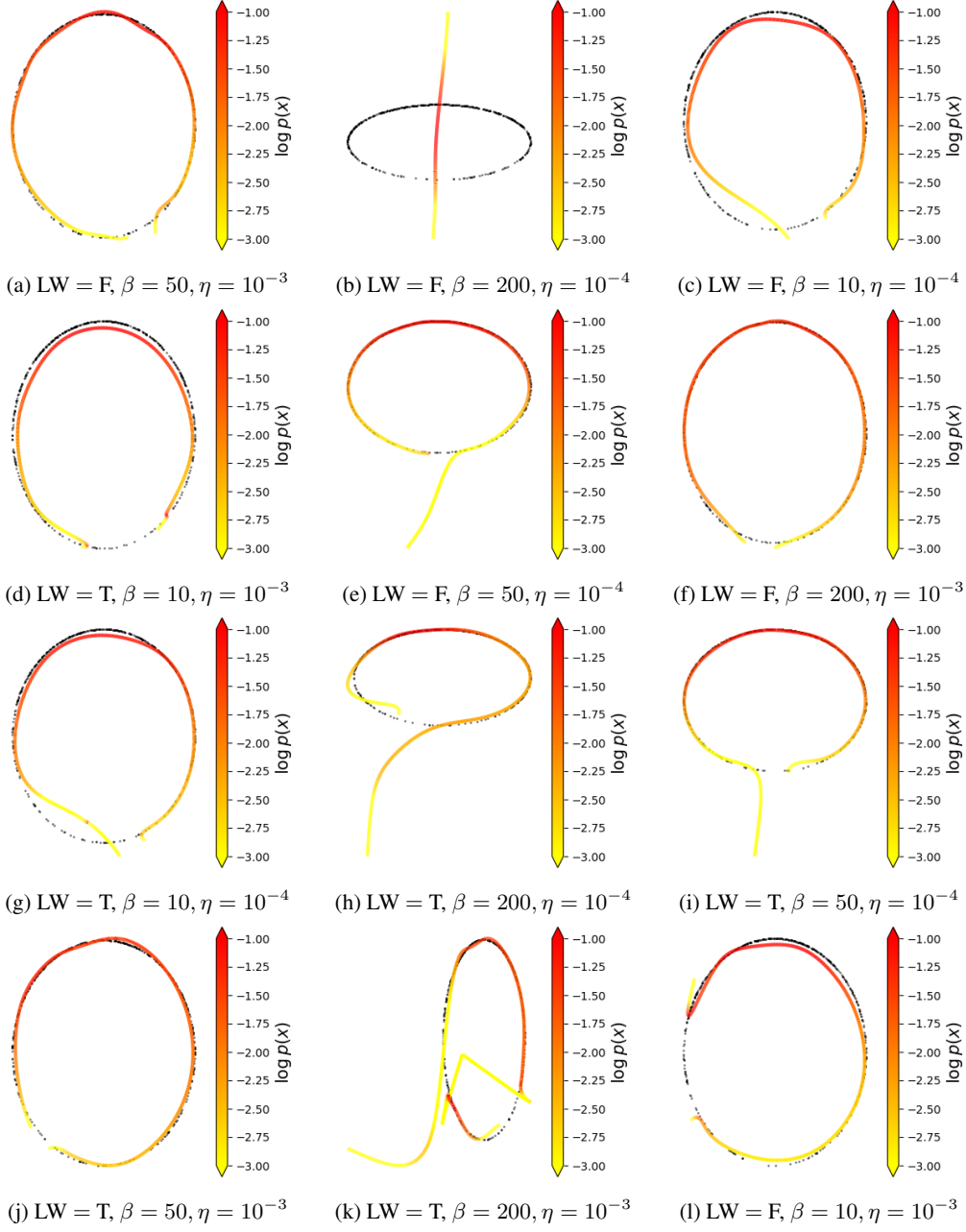


Figure 3: Runs of RNFs-ML (exact), swept over the hyperparameter combinations $\{\text{Likelihood Warmup} \in \{\text{True}, \text{False}\}\} \times \{\beta \in \{10, 50, 200\}\} \times \{\eta \in \{10^{-3}, 10^{-4}\}\}$

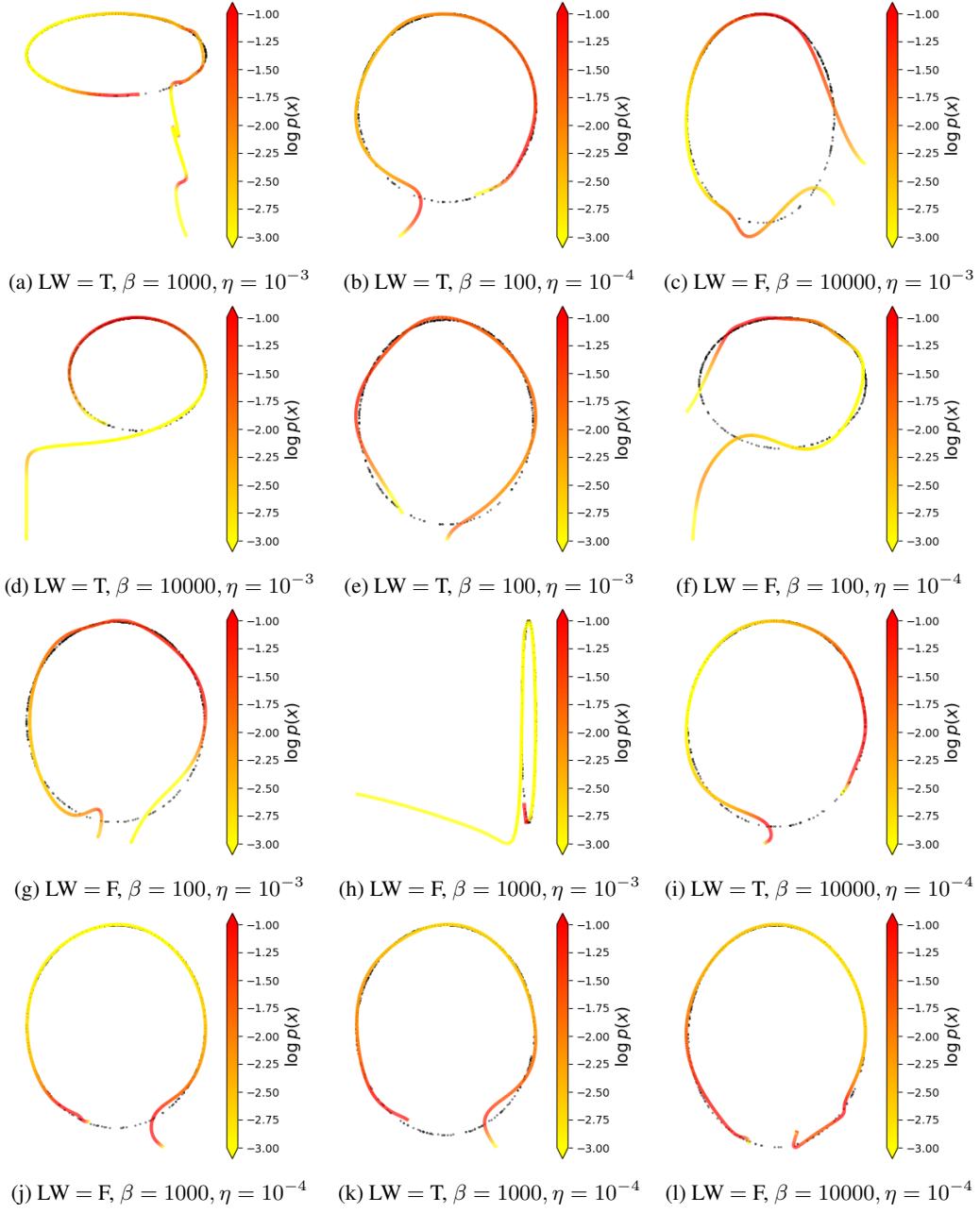


Figure 4: Runs of RNFs-TS, swept over the hyperparameter combinations $\{\text{Likelihood Warmup} \in \{\text{True}, \text{False}\}\} \times \{\beta \in \{100, 1000, 10000\}\} \times \{\eta \in \{10^{-3}, 10^{-4}\}\}$

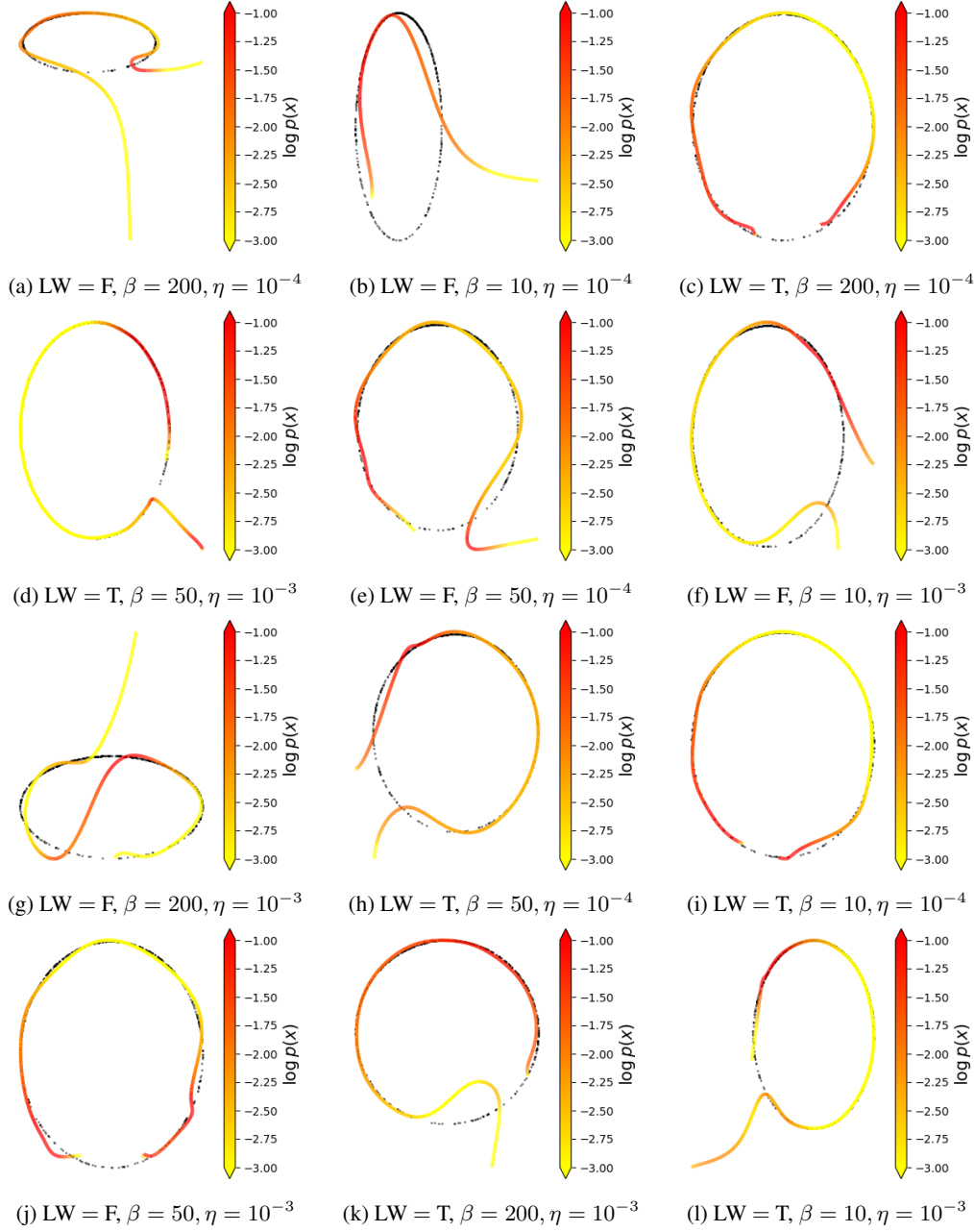


Figure 5: Runs of RNFs-TS, swept over the hyperparameter combinations $\{\text{Likelihood Warmup} \in \{\text{True}, \text{False}\}\} \times \{\beta \in \{10, 50, 200\}\} \times \{\eta \in \{10^{-3}, 10^{-4}\}\}$

G.2 Tabular Data

For the tabular data, we use the GAS, POWER, HEPMASS, and MINIBOONE datasets, preprocessed as in Papamakarios et al. [45]. We did not observe problems with overfitting in practice for any of the methods. We use the FID-like metric with the first and second moments of the generated and observed data as described in Appendix F for early stopping, halting training after 20 epochs of no improvement.

We again use a RealNVP flow in D dimensions but now with 10 layers, with each layer having a fully-connected coupler network of hidden dimension 4×128 . The d -dimensional flow here is also a RealNVP, but just a 5-layer network with couplers of size 2×32 .

In all methods, we use a regularization parameter of $\beta = 50$. We introduce the likelihood term with low weight after 25 epochs, linearly increasing its contribution to the objective until it is set to its full weight after 50 epochs. We select d as $\lfloor \frac{D}{2} \rfloor$, except for ML methods on $D = 8$ GAS which use $d = 2$ (noted below). We use a learning rate of 10^{-4} . For the methods involving the Hutchinson estimator, we use a standard Gaussian as the estimating distribution. We also experimented with a Rademacher distribution here but found the Gaussian to be superior.

Results reported on the main manuscript are the mean of 5 runs (with different seeds) plus/minus standard error. Occasionally, both RNFs-ML and RNFs-TS resulted in failed runs with FID-like scores at least an order of magnitude larger than other runs. In these rare instances, we did another run and ignored the outlier. We did this for both methods, and we do point out that RNFs-ML did not have a higher number of failed runs.

As mentioned in the main manuscript, GAS required slightly more tuning as RNFs-ML did not outperform RNFs-TS when using $d = 4$. We instead use latent dimension $d = 2$, where this time RNFs-ML did outperform. Since RNFs-TS did better with $d = 4$, we report those numbers in the main manuscript. Otherwise, our methods outperformed the baseline out-of-the-box, using parameter configurations gleaned from the image and circle experiments.

We also include the batch sizes here for completeness, which were set to be reasonably large for the purposes of speeding up the runs:

- POWER - 5,000
- GAS - 2,500
- HEPMASS - 750
- MINIBOONE - 400

G.3 Image Data and Out-of-Distribution Detection

In this set of experiments, we mostly tuned the RNFs-ML methods on MNIST for $K = 1$ – applying any applicable settings to RNFs-TS on MNIST as well – which is likely one of the main reasons that RNFs-ML perform so well for $K = 1$ vs. the exact method or $K = 4$. The reason why we spent so much time on $K = 1$ is that it was the fastest experiment to run and thus the easiest to iterate on. Our general strategy for tuning was to stick to a base set of parameters that performed reasonably well and then try various things to improve performance. A full grid search of all the parameters we might have wanted to try was quite prohibitive on the compute that we had available. Some specific details on settings follow below.

For the D -dimensional square flow, we mainly used the 10-layer RealNVP model which exactly mirrors the setup that Dinh et al. [15] used on image data, except we neglect to include batch normalization (as discussed in Appendix E) and we also tried reducing the size of the ResNet coupling networks from 8×64 to 4×64 for computational purposes. For further computational savings, we additionally attempted to use a RealNVP with fewer layers as the D -dimensional square flow, but this performed extremely poorly and we did not revisit it. For the d -dimensional square component, we used another RealNVP with either 5 or 10 layers, and fully-connected coupler networks of size 4×32 . We also looked into modifying the flow here to be a neural spline flow [16], but this, like the smaller D -dimensional RealNVP, performed very poorly as well. This may be because we did not constrain the norm of the gradients, although further investigation is required. We also looked into using no d -dimensional flow for our methods as in the circle experiment, but this did not work well at all.

For padding, we first randomly (although this is fixed once the run begins) permute the d -dimensional input, pad to get to the appropriate length of vector, and then reshape to put into image dimension. We also pad with zeros when performing the inverse of the density split operation (cf. the z to x direction of Dinh et al. [15, Figure 4(b)]), so that the input is actually padded twice at various steps of the flow.

When we used likelihood annealing, we did the same thing as for the tabular data: optimize only the reconstruction term for 25 epochs, then slowly and linearly introduce the likelihood term up until it has a weight of 1 in the objective function after epoch 50.

We summarize our attempted parameters in Table 5. For some choices of parameters, such as likelihood annealing set to `False`, $d = 15, 30$, $\beta = 10,000$, and CG tolerance set to 1, we had very few runs because of computational reasons. However, we note that the run with low CG tolerance ends up being the most successful run on MNIST. We have included “SHORT NAMES” in the table for ease of listing hyperparameter values for the runs in Table 3, which we now provide for MNIST and FMNIST in Table 6 and Table 7 respectively. We also include batch sizes in the table. Note that * indicates that the run was launched on 2 GPU cards simultaneously, whereas ** indicates that the run was launched on 4 GPU cards. For the CIFAR-10 parameters, we attempt several runs of the best configurations below: we sent 1 run for RNFs-ML (exact), 2 runs for RNFs-ML ($K = 1$), and 4 runs for RNFs-TS.

Table 5: Parameter combinations investigated for MNIST runs. Note that the final two rows are irrelevant for RNF-ML (exact) and RNF-TS. We include "short names" for ease of listing parameters for the runs in Table 3.

PARAMETER	SHORT NAME	MAIN VALUE	ALTERNATIVES
Likelihood Annealing	LA	True	False
Reconstruction parameter	β	50	5, 500, 10000
Low dimension	d	20	10, 15, 30
D -dim flow coupler	D NET	8×64	4×64
d -dim flow layers	d LAYERS	5	10
Hutchinson distribution	HUTCH	Gaussian	Rademacher
CG tolerance (normalized)	tol	1	0.001

Table 6: Parameter choices for the MNIST runs reported in Table 3.

METHOD	LA	β	d	D NET	d LAYERS	HUTCH	tol	BATCH
RNFs-ML (exact)	True	5	20	8×64	10	N/A	N/A	100**
RNFs-ML ($K = 1$)	True	5	20	8×64	10	Gaussian	0.001	200
RNFs-ML ($K = 4$)	True	50	20	8×64	5	Gaussian	1	100*
RNFs-TS	True	50	20	8×64	5	N/A	N/A	200

Table 7: Parameter choices for the FMNIST runs reported in Table 3.

METHOD	LA	β	d	D NET	d LAYERS	HUTCH	tol	BATCH
RNFs-ML (exact)	True	50	20	8×64	10	N/A	N/A	100**
RNFs-ML ($K = 1$)	True	50	20	8×64	5	Rademacher	1	200
RNFs-ML ($K = 4$)	True	50	20	8×64	10	Rademacher	1	200**
RNFs-TS	False	5	20	4×64	10	N/A	N/A	200

Visualizations As an attempt to visualize the learned manifold, we also trained our model with $d = 2$, and show the samples obtained for different values of $z \in \mathbb{R}^2$ in Figure 6, where the spatial location of each sample is given by the Cartesian coordinates of the corresponding z value. While there are some abrupt changes, which we believe are to be expected since the true manifold likely consists of several connected components, we can see that for the most part similar-looking images have nearby latent representations.

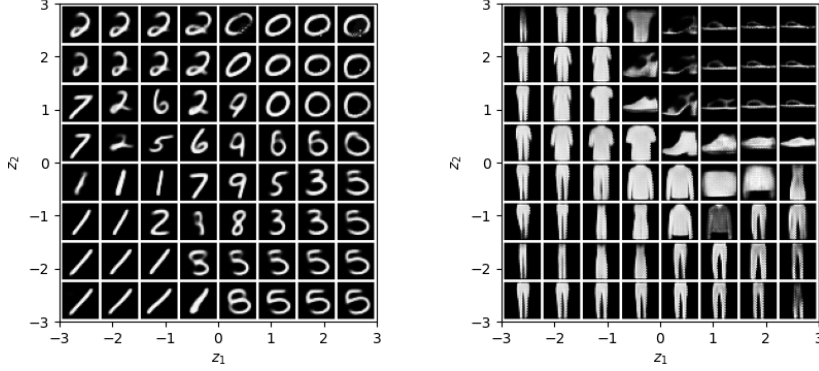


Figure 6: MNIST (left) and FMNIST (right) samples from RNFs-ML (exact) with $d = 2$. The subindices in z_1 and z_2 index coordinates, not datapoint number.

Further Out-of-Distribution Detection Results Figure 7 shows RNFs-ML log-likelihoods for models trained on MNIST (**left panel**), and we can see that indeed MNIST is assigned higher likelihoods than FMNIST. We also include OoD detection results when using reconstruction error instead of log-likelihoods, for models trained on FMNIST (**middle panel**) and MNIST (**right panel**). We observed similar results with RNFs-TS. Surprisingly, it is now the reconstruction error which exhibits puzzling behaviour: it is *always* lower on FMNIST, regardless of whether the model was trained on FMNIST or MNIST. Once again, this behaviour also happens for RNFs-TS, where the reconstruction error is optimized separately. We thus hypothesize that this behaviour is not due to maximum likelihood training, and rather is a consequence of inductive biases of the architecture.

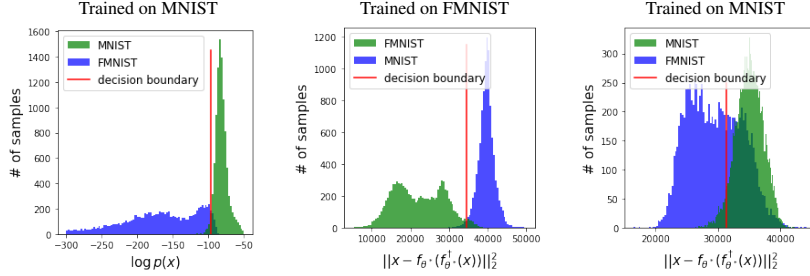


Figure 7: OoD log-likelihood histograms trained on MNIST (left), and OoD reconstruction error histograms trained on FMNIST (middle) and MNIST (right). Log-likelihood results (left) are RNFs-ML (exact), and reconstruction results (middle and right) are RNFs-ML ($K = 1$). Note that green denotes in-distribution data, and blue OoD data; and colors *do not* correspond to datasets.

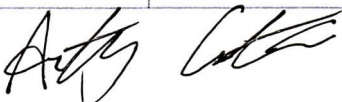
Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor (**only required where there isn't already a statement of contribution within the paper itself**).

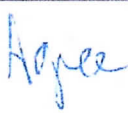
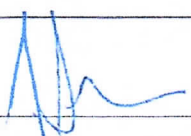
Title of Paper	Rectangular Flows for Manifold Learning
Publication Status	☐ Published ☐ Accepted for Publication ☒ Submitted for Publication ☐ Unpublished and unsubmitted work written in a manuscript style
Publication Details	Caterini, A. L., Loaiza-Ganem, G., Pleiss, G., & Cunningham, J. P. (2021). Rectangular flows for manifold learning. <i>arXiv preprint arXiv:2106.01413</i> . Submitted for publication at <i>Neural Information Processing Systems 2021</i> .

Student Confirmation

Student Name:	Anthony Caterini		
Contribution to the Paper	Developed idea from co-author Gabriel Loaiza-Ganem, with third author Geoff Pleiss contributing the suggestion about conjugate gradients. Collaboratively designed model structure and objective function, formulated motivation, derived details of gradient tricks, and designed experiments. Individually designed the entire codebase and ran almost all experiments, along with hyperparameter searches. Revised paper with Gabriel, iterating on the first draft which he wrote.		
Signature		Date	22 June 2021

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof. Arnaud Doucet			
Supervisor comments 			
Signature		Date	27/06/21

This completed form should be included in the thesis, at the end of the relevant chapter

6

Conclusion

This thesis has analyzed some shortcomings of normalizing flows and proposed various remedies, all of which involve modifying the assumption of strict bijectivity over the ambient space of interest. We focused on the settings of variational inference and deep generative modelling, two places where normalizing flows research has been quite active in the past several years. In this section we reflect on the contributions made in this thesis and provide directions for future work.

6.1 Contributions

The first contribution, in chapter 2, centres around improving the inference procedure in a variational auto-encoder, which can alternatively be seen as building a normalizing flow guided by the target distribution for general variational Bayesian inference. We use discretized Hamiltonian dynamics to augment a baseline approximate posterior, which can be viewed as parametrizing a normalizing flow on a higher-dimensional position-momentum space. We include tempering in the model – learned via variational parameters – to add time-inhomogeneity to the flow and thus increase expressiveness. We then demonstrate the viability of our method, entitled the Hamiltonian Variational Auto-Encoder (HVAE), for modelling large-scale datasets, achieving improvements over the baselines while adding very few parameters.

Next, in chapter 3, we show that normalizing flows for density estimation *must* become non-invertible to exactly model topologically non-trivial distributions. We propose a solution – namely, the *continuously indexed flow* (CIF) model – which relaxes the bijectivity assumption and indexes each flow step with additional trainable randomness, sacrificing exactness of the density computation for expressiveness of the model. By introducing a distribution to infer the indexing variables, we can train the model using the ELBO as a surrogate for the likelihood, obtaining improved results over baseline flows in a variety of density estimation benchmarks.

The work in chapter 4 then builds off this by considering how to apply CIFs to the setting of variational inference. CIFs admit intractable marginal densities over the latent variables of interest, but admit a tractable *joint* density over those and the additional indexing variables, and thus can be trained for use in variational inference using auxiliary methods. We can now use the inference distribution over indexing variables from the context of density estimation as the auxiliary inference distribution required in such a scheme, making it natural to apply CIFs to this setting. Theoretically, we show that CIF models trained using the auxiliary bound outperform baseline normalizing flows trained with the standard ELBO, and then demonstrate this outperformance empirically on two standard, but challenging, variational inference problems.

Then, in chapter 5 – the final body chapter of this thesis – we return to density estimation but now with *injective* flows mapping from low- to high-dimensional space. This approach, which we call Rectangular Normalizing Flows (RNFs), is motivated by the manifold hypothesis, directly embedding a low-dimensional manifold in high-dimensional space. This injective construction diverges from the typical bijective structure of flow-based models. As far as we know, our approach is the first method which learns a generic manifold while also including the injective volume-change term in optimization, thus not losing important information about the flow while still scaling to high-dimensional settings. We show that our method improves over relevant baselines in density estimation tasks.

6.2 Challenges and Future Work

In this section we briefly explore some of the challenges and drawbacks of the works in this thesis, with an eye on possible directions for future work stemming from these.

6.2.1 Hamiltonian Variational Auto-Encoder

We have already noted that Hamiltonian methods for variational inference, such as the HVAE, are guided by the true posterior. This presents an interesting trade-off: since the flow is guided by the target, we require fewer parameters to get a comparable improvement in inference versus a standard normalizing flow, but we also incur greatly increasing memory costs as the likelihood model increases in size since its gradient is required in forward steps of the flow. With the decoder architectures in variational auto-encoders continuing to grow larger (e.g. [Vahdat and Kautz, 2020]), devising methods to somehow limit the cost of evaluating steps of the flow – and gradients thereof – presents an interesting direction for future work. One way to approach this may be to consider more efficient automatic differentiation methods for evaluating the required “gradient of the gradient” either exactly or approximately, much like the analysis provided in chapter 5 for RNFs. Alternatively, it may be best to only consider Hamiltonian-based approaches for likelihood models which are either more modestly-sized or admit particularly tractable gradients; pinpointing these settings will provide guidance on when we would expect Hamiltonian methods to outperform other flows.

Another issue with the HVAE may be its expressiveness. Notably, without momentum refreshes or Metropolis accept-reject steps, Hamiltonian dynamics may not be powerful enough to represent the true posterior. Even with additional time-inhomogeneity from the learned tempering scheme and variationally-estimated leapfrog step sizes, the method has far fewer parameters than standard normalizing flows. Although parameter efficiency is an advantage of this method, there may be some middle ground where we achieve sufficient expressiveness through richer parametrizations while maintaining a low overall number of parameters and keeping log-determinant evaluations tractable. Recent work extending the HVAE has

included the mass matrix as a variational parameter [Chadebec et al., 2020]; another direction would be using a more complicated volume-preserving dynamics such as the Nosé-Hoover thermostat [Evans and Holian, 1985]. However, it remains to be seen if such methods would substantially improve empirical performance. A final thought on improving expressiveness is replacing the standard normal target for the terminal momentum distribution of the HVAE with one that explicitly depends on the terminal position variable, and possibly the data too, if amortized. Originally, the standard normal distribution was motivated as the maximum-entropy distribution at the target temperature of 1, but in the view of HVAE as an auxiliary variational scheme we can reduce the bias of the objective (cf. (1.4)) by instead making the parametrized replacement. This is perhaps the most natural and impactful way to increase the representational power of the HVAE.

6.2.2 Continuously-Indexed Normalizing Flows

We have shown that continuously indexed flows (CIFs) have the ability to outperform standard normalizing flows in the settings of density estimation and variational inference, despite parametrizing an intractable marginal distribution. In both cases, we were required to optimize surrogate objectives instead of the more standard objectives used by the baselines, but this ultimately did not limit the performance of our methods. The most obvious next step for CIFs then is to see which other applications of normalizing flows can be improved by continuous indexing. We can look at specific examples of flow-based models, such as those used in approximate Bayesian computation [Papamakarios et al., 2019b] or reinforcement learning [Singh et al., 2020], and consider if it is possible to formulate a surrogate objective which allows training of CIFs while not introducing too much bias. As one example, Golinski and Caterini [2021] have recently applied this idea to augment a pre-existing flow-based lossless compression algorithm [Ho et al., 2019], but it remains unclear whether or not this approach truly provides practical benefits. Nevertheless, this general line of research – wherein CIFs are used as stand-in replacements for

normalizing flow models in various applications – remains a promising direction and ought to be further pursued.

Now, specific to the setting of variational inference, we have previously noted that standard normalizing flow models do not incorporate information about the true posterior when parametrizing the variational model. Unfortunately CIF models in VI are not immune from this drawback, and overcoming this is an important direction for future research. One way to accomplish this is to continuously index Hamiltonian flows – which, as we know, are themselves guided by the true posterior – but it is not entirely clear if the benefits of relaxing bijectivity constraints translate to bijectivity on an *extended* space. Another approach might be to somehow incorporate the true posterior into the indexing step of CIFs, but this is even less clear and would require much more development to become effective.

Finally, although CIFs generalize and outperform normalizing flows in the applications studied here, they introduce additional hyperparameters and design choices that could be better understood. Empirically, we tried a variety of hyperparameters defining, for example, the structure of the indexing distribution and the shift-and-scale networks. Although we eventually found some defaults which worked well across applications, further understanding of the functions of each of the added components may be able to provide additional performance improvements in specific applications. Our choice of indexed bijection as well is fairly simple; surely there exists a more expressive function that could be used. More generally, it may be worthwhile to do a thorough empirical study of all the hyperparameters and design choices to gain further understanding on the best possible performance of the method, including arguably the most important choice: which baseline flow to continuously index.

6.2.3 Rectangular Normalizing Flows

Rectangular normalizing flows (RNFs) are more compatible with the manifold hypothesis than their standard counterparts. However, despite providing a framework for overcoming misspecification based on incorrect dimensionality, RNFs may still

suffer from *topological* misspecification as the resulting embedded manifold is still homeomorphic to low-dimensional Euclidean space. The pitfalls of topological misspecification in normalizing flows are well-documented in chapter 3; this chapter also fortunately presents CIFs as a remedy. Thus, we can break the topological restrictions here by using CIFs as the low-dimensional base distribution of an RNF model, encouraging improved regularity of the injective flow. This idea is closely related to recent work on parametrizing an embedded manifold with multiple charts [Kalatzis et al., 2021], and it would be interesting to draw connections between these two approaches especially since this method uses CIFs as a reverse inference model.

Another issue with RNFs is that the high-dimensional flow component is required to be easily computable in both the forward and inverse directions, thus limiting the expressiveness of the high-dimensional flow component. This property is often not satisfied by flow architectures, as only one direction generally needs to be computed during training time in most applications. We were thus restricted to using a RealNVP-style architecture [Dinh et al., 2017] here, and not a more expressive network such as an autoregressive [Papamakarios et al., 2019a] or residual flow [Chen et al., 2019] since neither of these have quickly-computable inverses. Future research then to improve RNFs might focus on devising more powerful flow-based architectures with fast forward and inverse maps. A complementary idea to improve expressiveness here is to replace the padding step with a parametrized injective step, such as those used in the very recent works Kothari et al. [2021] or Ross and Cresswell [2021] which were not available when chapter 5 was written.

Lastly, we note that the empirical results in the images section of the RNFs paper are quite counter-intuitive: we obtain the best performance with the approximate objective, and the number of samples in Hutchinson’s estimator K set to just 1. The reason for this is simply that we focused most of our tuning efforts on this case since it was the fastest model to train, importing our optimal settings from $K = 1$ into both the exact objective and the $K = 4$ case. Future work comparing all three methods across the same swathe of hyperparameters would be illuminating. In this same vein, somehow controlling the memory cost – perhaps through smaller

networks with more expressiveness on a per-layer basis – would allow for more rapid comparisons between the competing models. A more thorough hyperparameter sweep across architecture choices may also provide further insight into the crucial components of RNFs, much like CIFs in the previous section.

6.3 Final Thoughts and Broader Context

Although this thesis has concentrated on the challenges and opportunities in using normalizing flows to model high-dimensional data probabilistically, we are more generally interested in rethinking the default assumptions made to formulate certain classes of machine learning models, especially in settings of high dimension. Normalizing flows, for example, are an attractive framework for density modelling, but can struggle to stably learn distributions supported on complicated topologies and therefore are likely to become increasingly brittle when faced with data of increasing complexity. Flow models are also incompatible with the *manifold hypothesis*, which characterizes long-standing and intuitive views about how high-dimensional data behaves. Indeed, this problem is not unique to normalizing flows, as any full-support deep generative model – such as a variational auto-encoder or an energy-based model – is plagued by this same pathology. Thus, further work towards encoding more expressive topological or manifold structure in deep generative models, and, more broadly, carefully considering the implications of the particular modelling assumptions made to construct the model, is an important direction to continue expanding the practical capabilities of deep probabilistic models.

Bibliography

- Felix V Agakov and David Barber. An auxiliary variational method. In *International Conference on Neural Information Processing*, pages 561–566. Springer, 2004.
- Michael Arbel, Liang Zhou, and Arthur Gretton. Generalized energy based models. In *International Conference on Learning Representations*, 2020.
- Jens Behrmann, Paul Vicol, Kuan-Chieh Wang, Roger Grosse, and Jörn-Henrik Jacobsen. Understanding and mitigating exploding inverses in invertible neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 1792–1800. PMLR, 2021.
- Lucien Birgé and Yves Rozenholc. How many bins should be put in a regular histogram. *ESAIM: Probability and Statistics*, 10:24–45, 2006.
- David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.
- Johann Brehmer and Kyle Cranmer. Flows for simultaneous manifold learning and density estimation. *Advances in Neural Information Processing Systems*, 33, 2020.
- Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019.
- Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of markov chain monte carlo*. CRC press, 2011.
- Yuri Burda, Roger B. Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. In *International Conference on Learning Representations*, 2016.
- Olivier Cappé and Eric Moulines. On-line expectation–maximization algorithm for latent data models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):593–613, 2009.
- Anthony L Caterini, Arnaud Doucet, and Dino Sejdinovic. Hamiltonian variational auto-encoder. *Advances in Neural Information Processing Systems*, 32, 2018.
- Anthony L Caterini, Rob Cornish, Dino Sejdinovic, and Arnaud Doucet. Variational inference with continuously-indexed normalizing flows. In *37th Conference on Uncertainty in Artificial Intelligence (UAI)*. Association for Uncertainty in Artificial Intelligence (AUAI), 2021a.
- Anthony L Caterini, Gabriel Loaiza-Ganem, Geoff Pleiss, and John P Cunningham. Rectangular flows for manifold learning. *arXiv preprint arXiv:2106.01413*, 2021b.

- Clément Chadebec, Clément Mantoux, and Stéphanie Allasonnière. Geometry-aware hamiltonian variational auto-encoder. *arXiv preprint arXiv:2010.11518*, 2020.
- Edward Challis and David Barber. Affine independent variational inference. *Advances in Neural Information Processing Systems*, 25:2186–2194, 2012.
- Jianfei Chen, Jun Zhu, Yee Whye Teh, and Tong Zhang. Stochastic expectation maximization with variance reduction. *Advances in Neural Information Processing Systems*, 31:7967–7977, 2018.
- Jianfei Chen, Cheng Lu, Biqi Chenli, Jun Zhu, and Tian Tian. Vflow: More expressive generative flows with variational data augmentation. In *International Conference on Machine Learning*, pages 1660–1669. PMLR, 2020.
- Ricky TQ Chen, Jens Behrmann, David K Duvenaud, and Joern-Henrik Jacobsen. Residual flows for invertible generative modeling. In *Advances in Neural Information Processing Systems*, volume 32, 2019.
- Rob Cornish, Anthony L Caterini, George Deligiannidis, and Arnaud Doucet. Relaxing bijectivity constraints with continuously indexed normalising flows. In *International Conference on Machine Learning*, pages 2133–2143. PMLR, 2020.
- Chris Cremer, Xuechen Li, and David Duvenaud. Inference suboptimality in variational autoencoders. In *International Conference on Machine Learning*, pages 1078–1086. PMLR, 2018.
- Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.
- Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *International Conference on Learning Representations*, 2017.
- Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. In *Advances in Neural Information Processing Systems*, volume 32, 2019.
- Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 3140–3150, 2019.
- Denis J Evans and Brad Lee Holian. The nose–hoover thermostat. *The Journal of chemical physics*, 83(8):4069–4074, 1985.
- Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. CRC press, 2013.
- Charles Geyer. Introduction to markov chain monte carlo. In Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng, editors, *Handbook of markov chain monte carlo*, chapter 1, pages 3–48. CRC press, 2011.
- Adam Golinski and Anthony L Caterini. Lossless compression using continuously-indexed normalizing flows. In *Neural Compression: From Information Theory to Applications–Workshop@ ICLR 2021*, 2021.

- Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.
- Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.
- Jonathan Ho, Evan Lohn, and Pieter Abbeel. Compression with flows via local bits-back coding. *Advances in Neural Information Processing Systems*, 32:3879–3888, 2019.
- Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14(5), 2013.
- Dimitris Kalatzis, Johan Ziruo Ye, Jesper Wohler, and Søren Hauberg. Multi-chart flows. *arXiv preprint arXiv:2106.03500*, 2021.
- Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.
- Diederick P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29:4743–4751, 2016.
- Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.
- Ivan Kobyzev, Simon Prince, and Marcus Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.
- Konik Kothari, AmirEhsan Khorashadizadeh, Maarten de Hoop, and Ivan Dokmanić. Trumpets: Injective flows for inference and inverse problems. In *37th Conference on Uncertainty in Artificial Intelligence (UAI)*. Association for Uncertainty in Artificial Intelligence (AUAI), 2021.
- Steven G Krantz and Harold R Parks. *Geometric integration theory*. Springer Science & Business Media, 2008.
- Dieterich Lawson, George Tucker, Bo Dai, and Rajesh Ranganath. Energy-inspired models: Learning with sampler-induced distributions. *Advances in Neural Information Processing Systems*, 32, 2019.
- Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.
- Emile Mathieu and Maximilian Nickel. Riemannian continuous normalizing flows. *Advances in Neural Information Processing Systems*, 33, 2020.
- Thomas Nagler and Claudia Czado. Evading the curse of dimensionality in nonparametric density estimation with simplified vine copulas. *Journal of Multivariate Analysis*, 151:69–89, 2016.

- Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? In *International Conference on Learning Representations*, 2018.
- Radford M. Neal. Hamiltonian importance sampling. Talk presented at the Banff International Research Station (BIRS) workshop on Mathematical Issues in Molecular Dynamics, 2005.
- Radford M Neal. Mcmc using hamiltonian dynamics. In Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng, editors, *Handbook of markov chain monte carlo*, chapter 5, pages 113–162. CRC press, 2011.
- Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Learning non-convergent non-persistent short-run mcmc toward energy-based model. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, volume 32, 2019.
- George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*, 2019a.
- George Papamakarios, David Sterratt, and Iain Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 837–848. PMLR, 2019b.
- Rajesh Ranganath, Dustin Tran, and David Blei. Hierarchical variational models. In *International Conference on Machine Learning*, pages 324–333. PMLR, 2016.
- Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR, 2015.
- Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286, 2014.
- Danilo Jimenez Rezende, George Papamakarios, Sébastien Racanière, Michael Albergo, Gurtej Kanwar, Phiala Shanahan, and Kyle Cranmer. Normalizing flows on tori and spheres. In *International Conference on Machine Learning*, pages 8083–8092. PMLR, 2020.
- Brendan Leigh Ross and Jesse C Cresswell. Tractable density estimation on learned manifolds with conformal embedding flows. *arXiv preprint arXiv:2106.05275*, 2021.
- Lars Ruthotto and Eldad Haber. An introduction to deep generative modelling. *arXiv preprint arXiv:2103.05180*, 2021.
- Tim Salimans, Diederik Kingma, and Max Welling. Markov chain monte carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, pages 1218–1226. PMLR, 2015.
- Avi Singh, Huihan Liu, Gaoyue Zhou, Albert Yu, Nicholas Rhinehart, and Sergey Levine. Parrot: Data-driven behavioral priors for reinforcement learning. In *International Conference on Learning Representations*, 2020.

- Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.
- Esteban G 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.
- Yee Whye Teh, Max Welling, Simon Osindero, and Geoffrey E Hinton. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4(Dec):1235–1260, 2003.
- Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. In *Advances in Neural Information Processing Systems*, volume 33, pages 19667–19679, 2020.
- Rianne Van Den Berg, Leonard Hasenclever, Jakub M Tomczak, and Max Welling. Sylvester normalizing flows for variational inference. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pages 393–402. Association For Uncertainty in Artificial Intelligence (AUAI), 2018.
- Martin J Wainwright and Michael I Jordan. *Graphical models, exponential families, and variational inference*. Now Publishers Inc, 2008.
- Zhisheng Xiao, Karsten Kreis, Jan Kautz, and Arash Vahdat. Vaebm: A symbiosis between variational autoencoders and energy-based models. In *International Conference on Learning Representations*, 2020.
- Mingzhang Yin and Mingyuan Zhou. Semi-implicit variational inference. In *International Conference on Machine Learning*, pages 5660–5669. PMLR, 2018.
- Adriano Z Zambom and Dias Ronaldo. A review of kernel density estimation with applications to econometrics. *International Econometric Review*, 5(1):20–42, 2013.
- Han Zhang, Xi Gao, Jacob Unterman, and Tom Arodz. Approximation capabilities of neural ordinary differential equations. *arXiv preprint arXiv:1907.12998*, 2(4):3–1, 2019.