LOCALISED GENERATIVE FLOWS

Anonymous authors
Paper under double-blind review

ABSTRACT

We argue that flow-based density models based on continuous bijections are limited in their ability to learn target distributions with complicated topologies, and propose *localised generative flows* (LGFs) to address this problem. LGFs are composed of stacked continuous *mixtures* of bijections, which enables each bijection to learn a local region of the target rather than its entirety. Our method is a generalisation of existing flow-based methods, which can be used without modification as the basis for an LGF model. Unlike normalising flows, LGFs do not permit exact computation of log likelihoods, but we propose a simple variational scheme that performs well in practice. We show empirically that LGFs yield improved performance across a variety of common density estimation tasks.

1 INTRODUCTION

Flow-based generative models, often referred to as *normalising flows*, have become popular methods for density estimation because of their flexibility, expressiveness, and their tractable likelihoods. Given the problem of learning an unknown target density $p_X^\star$ on a data space $\mathcal{X}$, normalising flows model $p_X^\star$ as the marginal in X of the following generative process:

$$Z \sim p_Z, \quad X := g^{-1}(Z), \tag{1}$$

where p_Z is a prior density on a space $\mathcal{Z}$, and $g : \mathcal{X} \to \mathcal{Z}$ is a bijection.[1] Assuming sufficient regularity, it follows that X has density $p_X(x) = p_Z(g(x))|\det Dg(x)|$ (see e.g. Billingsley (2008)). The parameters of g can be learned via maximum likelihood given i.i.d. samples from $p_X^\star$.

To be effective, a normalising flow model must specify an expressive family of bijections with tractable Jacobians. Affine coupling layers (Dinh et al., 2014; 2016), stacked autoregressive transformations (Papamakarios et al., 2017), and invertible ResNet blocks (Behrmann et al., 2019) are all examples of such bijections that can be composed to produce complicated flows. These models have demonstrated significant promise in their ability to model complex datasets (Papamakarios et al., 2017) and to synthesise novel data points (Kingma & Dhariwal, 2018).

However, in all these cases, g is continuous in x. We believe this is a significant limitation of these models since it imposes a *global* constraint on g^{-1}, which must learn to match the topology of $\mathcal{Z}$ (which is usually quite simple) to the topology of $\mathcal{X}$ (which we expect to be very complicated). We argue that this constraint makes maximum likelihood estimation extremely difficult in general, leading to training instabilities and artefacts in the learned density landscape.

To address this problem we introduce *localised generative flows* (LGFs), which generalise equation 1 by replacing the single bijection g with a hierarchical continuous mixture of bijections G. Intuitively, LGFs allows these bijections each to focus on modelling only a *local* component of the target, which may have a much simpler topology than the full density. LGFs do not stipulate the form of G, and indeed any standard choice of g can be used as the basis for its construction. We pay a price for these benefits in that we can no longer compute the likelihood of our model exactly, and must instead resort to a variational approximation, with our training objective replaced by the evidence lower bound (ELBO). However, in practice we find this is not a significant limitation, as the bijection structure of LGFs permits learning a high-quality variational distribution suitable for large-scale training, and we show empirically that LGFs outperform their counterpart normalising flows across a variety of density estimation tasks.

[1]We assume throughout that $\mathcal{X}, \mathcal{Z} \subseteq \mathbb{R}^d$, and that all densities are with respect to the Lebesgue measure.

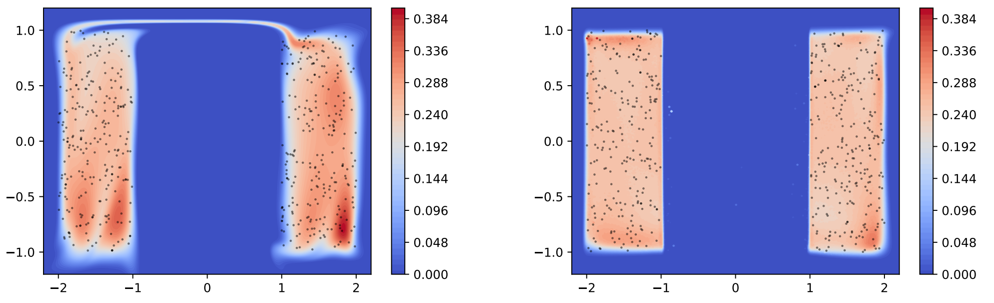

Figure 1: Density models learned after 300 epochs by a standard ten-layer MAF (left) and by a five-layer LGF-MAF (right). Both models have roughly 80,000 parameters and use a standard Gaussian prior distribution. Samples from the target distribution are shown in black. Details of the experimental setup are given in subsection C.1.

2 Limitations of Normalising Flows

Consider a normalising flow model defined by a parameterised family of bijections g_θ, where θ denotes our parameters. Suppose we are in the typical case that each g_θ is continuous in x. Intuitively, this seems to pose a problem when $p_X^\star$ and p_Z are effectively supported[2] on domains with different topologies, since continuous functions necessarily preserve topology. In practice, suggestive pathologies along these lines are readily apparent in simple 2-D experiments as shown in Figure 1. In this case, the density model (a Masked Autoregressive Flow (MAF) (Papamakarios et al., 2017)) is unable to continuously transform the support of the prior (a standard 2-D Gaussian) into the support of $p_X^\star$, which is the union of two disjoint rectangles and hence clearly has a different topology.

The standard maximum likelihood objective is asymptotically equivalent to

$$\arg\min_\theta D_{\mathrm{KL}}(p_X^\star || p_X^\theta), \tag{2}$$

where $p_X^\theta(x) = p_Z(g_\theta(x))|\det Dg_\theta(x)|$ and D_{KL} denotes the Kullback-Leibler divergence. We conjecture that the pathologies in Figure 1 are the result of sensitivities of this KL, which are greatly exacerbated when modelling $p_X^\star$ as the pushforward of p_Z by a *single* continuous g_θ^{-1}. In particular, observe that $D_{\mathrm{KL}}(p_X^\star || p_X) = \infty$ unless the support $\operatorname{supp} p_X$ contains $\operatorname{supp} p_X^\star$. However, at the same time, Pinsker's inequality (Massart, 2007) means that the KL is bounded below like

$$D_{\mathrm{KL}}(p_X^\star || p_X^\theta) \geq 2\left(1 - \int_{\operatorname{supp} p_X^\star} p_X^\theta(x)dx\right)^2 = 2\left(1 - \int_{\operatorname{supp}(p_X^\star)\cap \operatorname{supp} p_X^\theta} p_X^\theta(x)dx\right)^2$$

and is hence strictly positive unless $\operatorname{supp} p_X^\star = \operatorname{supp} p_X^\theta$. Thus the objective of equation 2 encourages the support of $p_X^\star$ to approximate that of p_X^θ, from above, as closely as possible, but any overshoot – however small – carries an immediate infinite penalty.

We believe this behaviour significantly limits the ability of normalising flows to properly learn any target distribution whose support has a complicated topology. Note that in practice our parameterisation usually satisfies $\det Dg_\theta(x) \neq 0$ for all θ and x. It then follows that no θ will yield $\operatorname{supp} p_X^\theta = \operatorname{supp} p_X^\star$ when $\operatorname{supp} p_X^\star$ and $\operatorname{supp} p_Z$ are not homeomorphic, even if our parameterisation contains *all* such continuous bijections. When this holds, we conjecture that the optimum of equation 2 will typically be pathological, since, in order to drive the KL to zero, g_θ must necessarily approximate some mapping that is not a continuous bijection. For gradient-based methods of solving equation 2, we further conjecture that this problem is compounded by the discontinuous behaviour of the KL described above. It seems plausible that perturbations of θ might entail small perturbations of parts of $\operatorname{supp} p_X^\theta$. When g_θ is close to optimal, so that $\operatorname{supp} p_X^\theta \approx \operatorname{supp} p_X^\star$, these perturbations

[2]Strictly speaking, the *support* of a density p technically refers to the set on which p is positive. However, here and elsewhere, our statements are also approximately true when $\operatorname{supp} p$ is interpreted as the region of p which is not smaller than some threshold. This is relevant in practice since, even if both are highly concentrated on some small region, it is common to assume that $p_X^\star$ and p_Z have full support, in which case the supports of $p_X^\star$ and p_Z would be trivially homeomorphic.

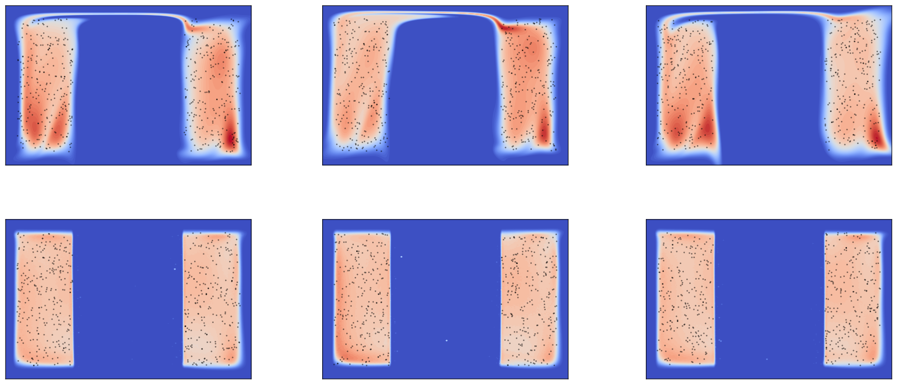

Figure 2: Three subsequent epochs of training for the models shown in Figure 1. The support of the standard MAF fluctuates significantly, while LGF-MAF is stable.

may potentially push *some* region of $\operatorname{supp} p_X^\theta$ outside the boundary of $\operatorname{supp} p_X^\star$ and hence drive the KL to infinity. We conjecture this makes the landscape around the optimal θ extremely difficult to navigate for gradient-based methods, leading to increasing fluctuations and instabilities in the KL in equation 2, degrading the quality of the final g_θ produced. Figure 2 illustrates this behaviour: observe that, even after 500 training epochs, the support of the standard MAF is unstable.

A simple way to fix these problems would be to use a complicated p_Z more closely matched to the structure of $p_X^\star$. For instance, taking p_Z to be a mixture model has previously been found to improve the performance of normalising flows in some cases (Papamakarios et al., 2017). However, this approach requires prior knowledge of the topology of $\operatorname{supp} p_X^\star$ that might be difficult to obtain: e.g. using a Gaussian mixture p_Z seems to require us to know the number of connected components of $\operatorname{supp} p_X^\star$ beforehand, and even then would not be sufficient unless these components are each homeomorphic to a hypersphere. Ideally, we would like our model to be flexible enough to learn the structure of the target on its own, with minimal explicit design choices required on our part.

An alternative approach would be to try a more expressive family g_θ in the hope that this better conditions the optimisation problem in equation 2. Several works have considered families of g_θ that are (in principle) universal approximators of any continuous probability densities (Huang et al., 2018; Jaini et al., 2019). While we have not performed a thorough empirical evaluation of these methods, we suspect that this approach can at best mitigate the problems described above, since the assumption of continuity of g_θ holds for universal approximators also. Moreover, the method we propose below can be used in conjunction with any standard flow-based method, so that we expect even further improvement when an expressive g_θ is combined with our approach.

Finally, we note that the shortcomings of the KL divergence for generative modelling are described at length by Arjovsky et al. (2017). There, the authors suggest instead using the Wasserstein distance to measure the discrepancy between $p_X^\star$ and p_Z, since under typical assumptions this will yield a continuous loss function suitable for gradient-based training. However, the Wasserstein distance is difficult to estimate in high dimensions, and its performance can be sensitive to the choice of ground metric used (Peyré et al., 2019). Our proposal here is to keep the KL objective in equation 2 and to modify the model instead, so that we are not required to meet the stringent requirement of mapping $\operatorname{supp} p_Z$ onto $\operatorname{supp} p_X^\star$ using a single continuous bijection. We describe our method in full now.

3 Localised generative flows

3.1 Model

The essence of our idea is to replace the single g used in equation 2 with an *indexed family* $\{G(\cdot; u)\}_{u \in \mathcal{U}}$ such that each $G(\cdot; u)$ is a bijection from $\mathcal{X}$ to $\mathcal{Z}$. Intuitively, our aim is for each $G(\cdot; u)$ only to push the prior onto a *local* region of $\operatorname{supp} p_X^\star$, thereby relaxing the constraints posed by standard normalising flows as described above. To do so, we now define p_X as the marginal density of X obtained via the following generative process:

$$Z \sim p_Z, \quad U \sim p_{U|Z}(\cdot|Z), \quad X := G^{-1}(Z;U). \tag{3}$$

Here $p_{U|X}$ is an additional term that we must specify. In all our experiments we take this to be a mean field Gaussian, so that $p_{U|X}(u|z) = \text{Gaussian}(u; \mu(z), \sigma(z)^2 I_{d_u})$ for some functions $\mu, \sigma : \mathcal{Z} \to \mathcal{U} \subseteq \mathbb{R}^{d_u}$, where d_u is the dimensionality of $\mathcal{U}$, and I_{d_u} is the $d_u \times d_u$ identity matrix. Other possibilities, such as the Concrete distribution, (Maddison et al., 2016) might also be useful.

Informally,[3] this yields the joint model $p_{X,U,Z}(x,u,z) = p_Z(z)\, p_{U|Z}(u|z)\, \delta(x - G^{-1}(z;u))$, where δ is the Dirac delta. To obtain a proper density for (X, U) we can marginalise out the dependence on z by making the change of variable $z = G(x'; u)$, yielding $dz = |\det DG(x';u)|\, dx'$.[4] Thus

$$\begin{aligned}
p_{X,U}(x,u) &= \int p_{X,U,Z}(x,u,z)\, dz \\
&= \int p_Z(G(x';u)) p_{U|Z}(u|G(x';u)) \delta(x - x')\, |\det DG(x';u)|\ dx' \\
&= p_Z(G(x;u))\, p_{U|Z}(u|G(x;u))\ |\det DG(x;u)|\,.
\end{aligned}$$

We then obtain our density model p_X by marginalising out u:

$$p_X(x) = \int p_Z(G(x;u))\, p_{U|Z}(u|G(x;u))\ |\det DG(x;u)|\ du. \tag{4}$$

In other words, our $p_X(x)$ is a mixture (in general, infinite) of individual normalising flows $G(x;u)$, each weighted by $p_{U|Z}(u|G(x;u))$.

We can also stack this architecture by taking p_Z itself to be a density of the form of equation 4. Doing so with L layers of stacking corresponds to the marginal of $X \equiv Z_L$ obtained via:

$$\begin{aligned}
&Z_0 \sim p_{Z_0}, \\
&U_\ell \sim p_{U_\ell|Z_{\ell-1}}(\cdot|Z_{\ell-1}), && \text{for } \ell \in \{1, \ldots, L\}, \\
&Z_\ell = G_\ell^{-1}(Z_{\ell-1}; U_\ell), && \text{for } \ell \in \{1, \ldots, L\},
\end{aligned}$$

where now each $G_\ell(\cdot;u) : \mathcal{X} \to \mathcal{Z}$ is a bijection for all u. The stochastic computation graph corresponding to this model is shown in Figure 3a. In this case, the same argument yields

$$p_{Z_\ell,U_{1:\ell}}(z_\ell, u_{1:\ell}) = p_{Z_{\ell-1},U_{1:\ell-1}}(G_\ell(z_\ell;u_\ell), u_{1:\ell-1}) p_{U_\ell|Z_{\ell-1}}(u_\ell|G_\ell(z_\ell;u_\ell)) |\det DG_\ell(z_\ell;u_\ell)| \tag{5}$$

where $p_{Z_0,U_{1:0}} \equiv p_{Z_0}$. This approach is in keeping with the standard practice of constructing normalising flows as the composition of simpler bijections, which can indeed be recovered here by taking each $p_{U_\ell|Z_{\ell-1}}(\cdot|Z_{\ell-1})$ to be Dirac. We have found stacking to improve significantly the overall expressiveness of our models, and make extensive use of it in our experiments below.

3.2 Benefits

Intuitively, we believe our model allows each $G(\cdot;u)$ to learn a *local* region of $p_X^\star$, thereby greatly relaxing the global constraints on existing flow-based models described above. To ensure a finite KL, we no longer require the density $p_Z(G(x;u))\, |\det DG(x;u)|$ to have support covering the entirety of $\operatorname{supp} p_X^\star$ for any given u: all that matters is that every region of $\operatorname{supp} p_X^\star$ is covered for *some* choice of u. Thus, our model can achieve good performance with each bijection $G(\cdot;u)$ faithfully capturing only a potentially very small component of $p_X^\star$. This seems significantly more achievable than the previous case, wherein a single bijection is required to capture the entire target.

This argument is potentially clearest if u is discrete. For example, even if $u \in \{u_1, u_2\}$ can take only two possible values, it immediately becomes simpler to represent the target shown in Figure 1 using a Gaussian prior: we simply require $G(x;u_1)$ to map onto one component, and $G(x;u_2)$ to map onto the other. In practice, we could easily implement such a G using two separate normalising flows that are trained jointly. The discrete case is moreover appealing since in principle it allows exact evaluation of the integral in equation 4, which becomes a summation. Unfortunately this approach also has significant drawbacks that we discuss at greater length in Appendix B.

We therefore instead focus here on a continuous u. In this case, for example, we can recover $p_X^\star$ from Figure 1 by partitioning $\mathcal{U}$ into disjoint regions $\mathcal{U}_{-1}$ and $\mathcal{U}_1$, and having $G^{-1}(\cdot;u)$ map onto

[3]This argument can be made precise using disintegrations (Chang & Pollard, 1997), but since the proof is mainly a matter of measure-theoretic formalities we omit it.

[4]Note that $DG(x;u)$ refers to the Jacobian with respect to x only

the left component of $p^\star_X$ for $u \in \mathcal{U}_{-1}$, and the right component for $u \in \mathcal{U}_1$. Observe that in this scenario we do not require any given $G^{-1}(\cdot; u)$ to map onto *both* components of the target, which is in keeping with our goal of localising the model of $p^\star_X$ that is learned by our method.

In practice G will invariably be continuous in both its arguments, in which case it will not be possible to partition $\mathcal{U}$ disjointly in this way. Instead we will necessarily obtain some additional intermediate region $\mathcal{U}_0$ on which $G^{-1}(\cdot; u)$ maps part of $\operatorname{supp} p_Z$ outside of $\operatorname{supp} p^\star_X$, so that $p_X(x)$ will be strictly positive there. This might appear to return us to the situation shown for the standard MAF shown in Figure 1. However, if only a relatively small region of $\mathcal{U}_0$ maps to any such x, then the overall value of the integral in equation 4 will be small here. Moreover, if $p_{U|X}$ is sufficiently flexible, it might additionally learn to downweight such (u, x) pairs, further reducing the value of $p_X(x)$. Empirically, we find these conditions are indeed sufficient to learn an accurate density estimator without the pathologies of standard flows. Observe in Figure 1 that our model is able to separate cleanly the two distinct components of the target density. Moreover, in Figure 2, we see that the learned density avoids the instabilities we observed previously.

We believe a similar story holds when our model is applied on more complicated targets also. At a heuristic level, we expect our model will learn to (softly) partition $\operatorname{supp} p^\star_X$ (via u) into separate components, and that each $G^{-1}(\cdot; u)$ will learn to map $\operatorname{supp} p_Z$ onto only one of these components. We conjecture that learning many such localised bijections is easier than learning a single global bijection, and we demonstrate this empirically for several density estimation problems of interest.

3.3 Inference

Even in the single layer case ($L = 1$), if u is continuous, then the integral in equation 4 is intractable. In order to train our model, we resort to a variational approximation: we introduce an approximate posterior $q_{U_{1:L}|X} \approx p_{U_{1:L}|X}$, and consider the evidence lower bound (ELBO) of $\log p_X(x)$:

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

It is straightforward to show that $\mathcal{L}(x) \leq \log p_X(x)$. This bound is tight when $q_{U_{1:L}|X}$ is the exact posterior $p_{U_{1:L}|X}$, which allows learning an approximation to $p^\star_X$ by maximising $n^{-1} \sum_{i=1}^n \mathcal{L}(x_i)$ jointly in $p_{X,U_{1:L}}$ and $q_{U_{1:L}|X}$, where we assume a dataset of n i.i.d. samples $x_i \sim p^\star_X$.

It can be shown (see Appendix A) that the exact posterior factors as $p_{U_{1:L}|X}(u_{1:L}|x) = \prod_{\ell=1}^L p_{U_\ell|Z_\ell}(u_\ell|z_\ell)$, where $z_L = x$, and $z_{\ell-1} := G_\ell(z_\ell; u_\ell)$ for $\ell \leq L$. We thus endow $q_{U_{1:L}|X}$ with the same form:

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

The stochastic computation graph for this inference model is shown in Figure 3b. In conjunction with equation 5, this allows writing the ELBO recursively as

$$\mathcal{L}_\ell(z_\ell) = \mathbb{E}_{q_{U_\ell|Z_\ell}(u_\ell|z_\ell)} \left[\mathcal{L}_{\ell-1}(G_\ell(z_\ell; u_\ell)) + \log p_{U_\ell|Z_\ell}(u_\ell|z_\ell) + \log |\det DG_\ell(z_\ell; u_\ell)| - \log q_{U_\ell|X}(u_\ell|z_\ell) \right]$$

for $\ell \geq 1$, with the base case $\mathcal{L}_0(z_0) = p_{Z_0}(z_0)$. Here we recover $\mathcal{L}(x) \equiv \mathcal{L}_L(z_L)$.

Now let θ denote all the parameters of both $p_{X,U_{1:L}}$ and $q_{U_{1:L}|X}$. Suppose each q_ℓ can be suitably reparametrised (Kingma & Welling, 2013; Rezende et al., 2014) so that $h_\ell(\epsilon_\ell, z_\ell) \sim q_\ell(\cdot|z_\ell)$ when $\epsilon_\ell \sim \eta_\ell$ for some density η_ℓ not depending on θ. In all our experiments we give $q_{U_\ell|Z_\ell}$ the same mean field form as in equation 3, in which case this holds immediately as described e.g. by Kingma & Welling (2013). We can then obtain unbiased estimates of $\nabla_\theta \mathcal{L}(x)$ straightforwardly using Algorithm 1, which in turn allows training our model objective via stochastic gradient descent. Note that although this algorithm is specified in terms of a single value of z_ℓ, it is trivial to obtain an unbiased estimate of $\nabla_\theta m^{-1} \sum_{j=1}^m \mathcal{L}(x_j)$ for a minibatch of points x_j by averaging over the batch index at each layer of recursion.

3.3.1 Performance

A major reason for the popularity of normalising flows is the tractability of their exact log likelihoods. In contrast, the variational scheme described here can produce at best an approximation to this value, which we might expect reduces performance of the final density estimator learned. Moreover, particularly when the number of layers L is large, it might seem that the variance of gradient estimators obtained from Algorithm 1 would be impractically high.

Algorithm 1 Recursive calculation of an unbiased estimator of $\nabla_\theta \mathcal{L}_\ell(z_\ell)$

```
function GRADELBO($z_\ell$, $\ell$)
    if $\ell = 0$ then
        return $p_{Z_0}(z_\ell)$
    else
        $\epsilon_\ell \sim \eta_\ell$
        $u_\ell \leftarrow h_\ell(\epsilon_\ell, z_\ell)$
        $z_{\ell-1} \leftarrow G_\ell(z_\ell; u_\ell)$
        $\Delta_\ell \leftarrow \log p_{U_\ell|Z_{\ell-1}}(u_\ell|z_{\ell-1}) + \log|\det DG_\ell(z_\ell; u_\ell)| - \log q_{U_\ell|Z_\ell}(u_\ell|z_\ell)$
        return GRADELBO($z_{\ell-1}, \ell - 1$) $+\nabla_\theta \Delta_\ell$
```

However, in practice we have not found either of these problems to be a significant limitation, as our experimental results in section 5 show. Empirically we find that importance sampling is sufficient for obtaining good, low-variance (if slightly biased) estimates of $\log p_X(x)$ (Rezende et al., 2014, Appendix E). Likewise, even when we take L to be large, we do not find that the variance of Algorithm 1 becomes correspondingly large. We conjecture that this occurs because, as the number of layers of bijections G^{-1} in our generative model grows, the complexity required of each individual bijection in order to map p_Z to $p_X^\star$ naturally decreases. We therefore have reason to think that, as L grows, learning each $q_{U_\ell|Z_\ell}$ will become easier, so that the variance at each layer will decrease, and the *overall* variance may remain fairly stable.

3.4 Choice of indexed bijection family

We now consider the choice of G, for which there are many possibilities. In our experiments, we focus on the simple case in which

$$G(x; u) = s(u) \odot g(x) + t(u) \tag{6}$$

where $s, t : \mathcal{U} \to \mathcal{Z}$ are unrestricted mappings, $g : \mathcal{X} \to \mathcal{Z}$ is a bijection, and $\odot$ denotes elementwise multiplication. In this case, $\det DG(x; u) = (\det Dg(x)) \sum_{i=1}^{d} [s(u)]_i$, where $[s(u)]_i$ denotes the i^{th} component of $s(u)$, and $\mathcal{Z} \subseteq \mathbb{R}^d$ as before. This has the advantage of working out-of-the-box with all pre-existing normalising flow methods for which a tractable Jacobian of g is available.

Equation 6 also has an appealing similarity with the common practice of applying affine transformations between flow steps for normalisation purposes, which has been found empirically to improve stability, convergence time, and overall performance (Dinh et al., 2016; Papamakarios et al., 2017; Kingma & Dhariwal, 2018). In prior work, s and t have been simple parameters that are learned either directly as part of the model, or updated according to running batch statistics. Our approach may be understood as a generalisation of this family of techniques.

Other choices of G are certainly possible here. For instance, we have had preliminary experimental success on small problems by simply taking g to be the identity, in which case the model is greatly simplified by not requiring any Jacobians at all. Alternatively, it is also frequently possible to modify the architecture of standard choices of g to obtain an appropriate G. For instance, affine coupling layers, a key component of models such as RealNVP (Dinh et al., 2016), make use of neural networks that take as input a subset of the dimensions of x. By concatenating u to this input, we straightforwardly obtain a family of bijections $G(\cdot; u)$ for each value of u. This requires more work to implement than our suggested method, but has the advantage of no longer requiring a choice of s and t. We have again had preliminary empirical success with this approach. We leave a more thorough exploration of these alternative possibilites for future work.

4 Related Work

4.1 Mixture Methods

There are other approaches which seek to model mixtures of normalising flows; however all of the examples below rely on discrete mixing variables, which are not amenable to stacking or variational inference (cf. Appendix B). A closely-related approach to ours is the RAD model (Dinh et al., 2019) which briefly notes similar topological concerns to us and proposes a superficially similar generative scheme. However, there are some limitations, notably that RAD requires an awkward parametrisation to ensure continuity of the flow and does not demonstrate the effectiveness of stacking the model architecture. Duan (2019) proposes a single-layer instance of our model with discrete

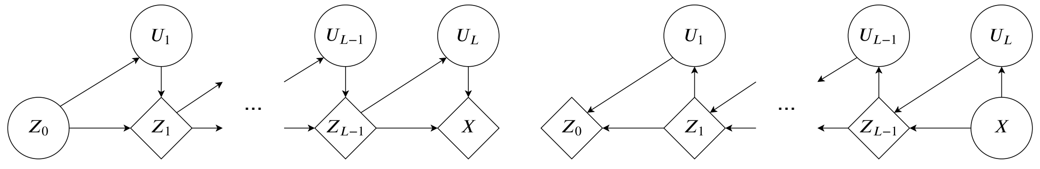

Figure 3: Multi-layer model schematic with L layers. Given their parents, circular nodes are random, and diamond nodes are deterministic.

mixing variable designed for Monte Carlo estimation. Ziegler & Rush (2019) introduce a normalising flow model with an additional latent variable indicating sequence length, but this model is only applicable to sequences. Our method could also be considered an addition to the class of *Deep Mixture Models* (Tang et al., 2012; Van den Oord & Schrauwen, 2014), but again our use of continuous mixing variables avoids computational cost scaling exponentially in the number of layers, and the additional structure imposed by the bijections allows for coherent inference across layers.

4.2 Methods Combining Variational Inference and Normalising Flows

It should be noted that the recent popularisation of normalising flows began with their use in variational inference models (Rezende & Mohamed, 2015). Models such as IAF (Kingma et al., 2016) and Sylvester flows (Berg et al., 2018) sought to improve the mean-field variational posterior q by additionally transforming samples from q through a normalising flow to obtain a new approximate posterior. However, these models do not include any sort of flow in the generative process, generally sticking to the VAE (Kingma & Welling, 2013; Rezende et al., 2014) encoder. Furthermore, the goal of these models is to *use normalising flows to improve variational inference*, which is somewhat tangential to the goal of *using variational inference to improve normalising flows* in the context of density estimation, which sets these methods apart from ours.

However, there are indeed some methods augmenting normalising flows with variational inference, but in all cases below the variational structure is not stacked to obtain extra expressiveness. Ho et al. (2019) use a variational scheme to improve upon the standard dequantisation method for deep generative modelling of images (Theis et al., 2015); this approach is orthogonal to ours and could indeed be integrated into LGFs. Gritsenko et al. (2019) also generalise normalising flows using variational methods, but they incorporate the extra latent noise into the model in a much more restrictive way. Das et al. (2019) only learn a low-dimensional prior over the noise space variationally.

4.3 Purely Variational Methods

Finally, we can contrast our approach with purely variational methods which are not flow-based, but still involve some type of stacking architecture. The main difference between these methods and LGFs is that the bijections we use provide us with a generative model with far more structure, which allows us to use the theory of D-separation to build appropriate inference models. Contrast this with, for example, Rezende et al. (2014), in which the layers are independently inferred, or Sønderby et al. (2016), which requires a complicated parameter-sharing scheme to reliably perform inference. A single-layer instance of our model also shares some similarity with the fully-unsupervised case in Maaløe et al. (2016), but the generative process there conditions the auxiliary variable on the data.

5 Experiments

We now describe various experiments we ran that demonstrate the effectiveness of LGFs. In each subsection, we compare a base flow with a flow that has been extended as an LGF model. We extend the base flows by inserting a mixing variable U between every component layer of the base flow: for example, we insert U after each MADE layer in MAF (Papamakarios et al., 2017), and after each affine coupling layer in RealNVP (Dinh et al., 2016). In all cases we train our models to maximise either the log probability (for the base flow) or the ELBO (for the extended flow), using the ADAM optimiser (Kingma & Ba, 2014) with default hyperparameters and no weight decay. We insert batch normalisation (Ioffe & Szegedy, 2015) between flow steps for the base flows as suggested by Dinh et al. (2016), but *omit* it for the LGFs, since our choice of G is a generalisation of batch normalisation as described above. To evaluate the models, we calculate the average log-likelihood over the test set for the base flows, and use the consistent estimator of the average log-likelihood over the test set

Table 1: Average plus/minus standard error of the best test-set negative log-likelihood.

	POWER	GAS	HEPMASS	MINIBOONE
MAF	0.19 ± 0.02	9.23 ± 0.07	-18.33 ± 0.10	-10.98 ± 0.03
LGF-MAF	$\mathbf{0.48 \pm 0.01}$	$\mathbf{12.02 \pm 0.10}$	$\mathbf{-16.63 \pm 0.09}$	$\mathbf{-9.93 \pm 0.04}$

with $S = 1000$ importance samples for LGFs, as in Rezende et al. (2014, Appendix E). We used early stopping during training, halting after 30 epochs of no validation improvement[5] for the UCI experiments, and after 50 epochs of no improvement for the image experiments.

Our code is available at `https://github.com/anonsubmission974/lgf`.

5.1 UCI Datasets

Following Papamakarios et al. (2017), we tested the performance of our method on the POWER, GAS, HEPMASS, and MINIBOONE datasets from the UCI repository (Bache & Lichman, 2013). We preprocessed these datasets identically to Papamakarios et al. (2017), including the same train, validation, and test splits. To increase the speed of training, we increased the batch size and correspondingly the learning rate by a factor of 10 from the configuration used by Papamakarios et al. (2017). We note that our results slightly differ from theirs perhaps because of this change.

We focused on MAF as our baseline normalising flow because of its improved performance over alternatives such as RealNVP for general-purpose density estimation (Papamakarios et al., 2017). A given MAF model is defined primarily by how many MADE (Germain et al., 2015) layers it uses, as well as the sizes of the autoregressive networks at each layer. For an LGF-MAF, we must additionally define the neural networks used in the generative and inference process, for which we use multi-layer perceptrons – the full details are in subsection C.2. For 3 separate random seeds, we trained MAFs and LGF-MAFs using a variety of choices of hyperparameters. Each instance of LGF-MAF has a corresponding MAF configuration, but in order to compensate for the parameters introduced by our additional neural networks, we also explore deeper and wider MAF models. We then chose the best performing model across all parameter configurations using validation performance for a particular seed, and we report the test set log-likelihood of this model. Table 1 shows these log likelihoods averaged across the different seeds. It is clear that extending MAFs with LGFs produces superior results in this case when measured on test log-likelihood. We also note that for each of the datasets considered, the range of test log-likelihoods achieved by LGF-MAF models *strictly does not overlap the range of test log-likelihoods achieved by MAF models*.

5.2 Image datasets

In this section we use RealNVP as the base flow and compare its performance to a RealNVP-based LGF model on the Fashion-MNIST (Xiao et al., 2017) and CIFAR-10 (Krizhevsky et al., 2009) datasets. We use the standard dequantisation step from Theis et al. (2015) to pre-process the data, and a learning rate of 10^{-4} across all experiments.

We match the original RealNVP architecture given in Dinh et al. (2016), using a residual network (He et al., 2016) with 8 residual blocks of 64 channels (i.e. 8×64) each for the coupling networks, 10 layers of coupling networks (with prescribed scale and squeeze operations), and batch normalisation in between all layers. This method has 5.94M parameters for Fashion-MNIST and 6.01M parameters for CIFAR-10. We also consider a RealNVP model with coupler networks of size 4×64 to match those used below for LGF-RealNVP for completeness; this model has 2.99M and 3.05M parameters for Fashion-MNIST and CIFAR-10, respectively.

For the LGF-RealNVP, we use a only 4 residual blocks of 64 channels each in the coupling networks and no batch normalisation between the layers. Additionally, we specify the mean and variance of both $p_{U_\ell|Z_{\ell-1}}$ and $q_{U_\ell|Z_\ell}$ at each layer ℓ using residual networks of size 2×64. We also specify s and t as the outputs of a residual network of size 2×8. We use a single-channel image-like u at each layer, whose height and width are the same size as the outputs of the coupling network (which

[5]Validation performance for LGFs was again calculated using the importance sampling estimator of the log-likelihood from Rezende et al. (2014) over the validation set, but with only $S = 10$ importance samples.

may not be the same size as the original image because of the scale and squeeze operations). This model has 5.99M parameters for Fashion-MNIST and 6.07M parameters for CIFAR-10.

As noted in Table 2, LGFs consistently outperform the base models for the same number of parameters, and tend to train faster even without the batch normalisation layers: e.g. the average epoch of top validation loss on CIFAR-10 is 458 for LGF-RealNVP, and 723 for RealNVP (8). Samples synthesised from all models can be found in subsection C.3.

Table 2: Average plus/minus the standard error of test-set bits per dimension. RealNVP (m) refers to a RealNVP model with m residual blocks in the coupling networks.

	Fashion-MNIST	**CIFAR-10**
RealNVP (4)	2.944 ± 0.003	3.565 ± 0.001
RealNVP (8)	2.946 ± 0.002	3.554 ± 0.001
LGF-RealNVP (4)	$\mathbf{2.823 \pm 0.003}$	$\mathbf{3.477 \pm 0.019}$

We can dig deeper into our experimental results to justify that using the ELBO as a surrogate objective instead of the log-likelihood does not penalise our method, even in high dimension. First note that the gap between the estimated test set log-likelihood and the average test set ELBO is not very large for the LGF models. In particular, the average relative error in estimated log probability when estimating with the ELBO across the test set is 8.98×10^{-3} for Fashion-MNIST and 6.88×10^{-3} for CIFAR-10. Next, we note that the importance-sampling-based log-likelihood estimator itself has incredibly low variance with $S = 1000$ importance samples. For each LGF model, we computed the relative standard deviation of this estimator across three runs: the average of this across models was 8.34×10^{-4} for Fashion-MNIST, and 2.07×10^{-5} for CIFAR-10.

6 Conclusion

In this paper, we have proposed localised generative flows for density estimation, which generalise existing normalising flow methods and address the limitations of these models in expressing complicated target distributions. Our method obtains promising empirical results on a variety of tasks. Many extensions appear possible, and we believe localised generative flows therefore show promise as a means for improving the performance of density estimators in practice.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*, 2017.

K. Bache and M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Joern-Henrik Jacobsen. Invertible residual networks. In *International Conference on Machine Learning*, pp. 573–582, 2019.

Rianne van den Berg, Leonard Hasenclever, Jakub M Tomczak, and Max Welling. Sylvester normalizing flows for variational inference. *arXiv preprint arXiv:1803.05649*, 2018.

Patrick Billingsley. *Probability and measure*. John Wiley & Sons, 2008.

Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

Joseph T Chang and David Pollard. Conditioning as disintegration. *Statistica Neerlandica*, 51(3): 287–317, 1997.

Hari Prasanna Das, Pieter Abbeel, and Costas J Spanos. Dimensionality reduction flows. *arXiv preprint arXiv:1908.01686*, 2019.

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. *arXiv preprint arXiv:1605.08803*, 2016.

Laurent Dinh, Jascha Sohl-Dickstein, Razvan Pascanu, and Hugo Larochelle. A RAD approach to deep mixture models. *arXiv preprint arXiv:1903.07714*, 2019.

Leo L Duan. Transport Monte Carlo. *arXiv preprint arXiv:1907.10448*, 2019.

Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pp. 881–889, 2015.

Alexey A Gritsenko, Jasper Snoek, and Tim Salimans. On the relationship between normalising flows and variational-and denoising autoencoders. 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning*, pp. 2722–2730, 2019.

Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural autoregressive flows. In *International Conference on Machine Learning*, pp. 2083–2092, 2018.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456, 2015.

Priyank Jaini, Kira A Selby, and Yaoliang Yu. Sum-of-squares polynomial flow. In *International Conference on Machine Learning*, pp. 3009–3018, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pp. 10215–10224, 2018.

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

Durk 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*, pp. 4743–4751, 2016.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary deep generative models. In *International Conference on Machine Learning*, pp. 1445–1453, 2016.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Pascal Massart. *Concentration Inequalities and Model Selection*. Springer, 2007.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pp. 2338–2347, 2017.

Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pp. 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*, pp. 1278–1286, 2014.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *Advances in neural information processing systems*, pp. 3738–3746, 2016.

Yichuan Tang, Ruslan Salakhutdinov, and Geoffrey Hinton. Deep mixtures of factor analysers. In *Proceedings of the 29th International Conference on International Conference on Machine Learning*, pp. 1123–1130. Omnipress, 2012.

Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

Aaron Van den Oord and Benjamin Schrauwen. Factoring variations in natural images with deep Gaussian mixture models. In *Advances in Neural Information Processing Systems*, pp. 3518–3526, 2014.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Zachary M Ziegler and Alexander M Rush. Latent normalizing flows for discrete sequences. *arXiv preprint arXiv:1901.10548*, 2019.

A Correctness of Posterior Factorisation

We want to prove that the exact posterior factorises according to

$$p_{U_{1:L}|X}(u_{1:L}|x) = \prod_{\ell=1}^{L} p_{U_\ell|Z_\ell}(u_\ell|z_\ell), \tag{7}$$

where $z_L = x$, and $z_{\ell-1} := G_\ell(z_\ell; u_\ell)$ for $\ell \leq L$. We first note that we can always write this posterior autoregressively as

$$p_{U_{1:L}|X}(u_{1:L}|x) = \prod_{\ell=1}^{L} p_{U_\ell|U_{>\ell},X}(u_\ell|u_{>\ell}, x). \tag{8}$$

Now, consider the graph of the full generative model, including the random variables $\{Z_\ell\}_\ell$. Using the theory of D-separation (Bishop, 2006), we can prove that, for any $\ell \in \{1, \dots, L-1\}$, U_ℓ is conditionally independent of all random variables in the set $H_\ell := \{U_{>\ell}, Z_{>\ell}\}$, given Z_ℓ. This can be seen by noting that all paths from U_ℓ to a node $i \in H_\ell$ contain a chain that is "blocked" by Z_ℓ; in other words, all information about H_ℓ that can be gleaned from U_ℓ flows through Z_ℓ. Thus, we know that

$$p_{U_\ell|U_{>\ell},X,Z_\ell}(u_\ell|u_{>\ell}, x, z_\ell) = p_{U_\ell|Z_\ell}(u_\ell|z_\ell).$$

We can then (somewhat informally – refer to footnote on pg. 4) write the following for any $\ell \in \{1, \dots, L-1\}$:

$$\begin{aligned} p_{U_\ell|U_{>\ell},X}(u_\ell|u_{>\ell}, x) &= \int p_{U_\ell,Z_\ell|U_{>\ell},X}(u_\ell, z_\ell|u_{>\ell}, x) dz_\ell \\ &= \int p_{U_\ell|Z_\ell,U_{>\ell},X}(u_\ell|z_\ell, u_{>\ell}, x) p_{Z_\ell|U_{>\ell},X}(z_\ell|u_{>\ell}, x) dz_\ell \\ &= \int p_{U_\ell|Z_\ell}(u_\ell|z_\ell) \delta\left(z_\ell - \left(G_{\ell+1,u_{\ell+1}} \circ \cdots \circ G_{L,u_L}\right)(x)\right) dz_\ell \\ &= p_{U_\ell|Z_\ell}\left(u_\ell| \left(G_{\ell+1,u_{\ell+1}} \circ \cdots \circ G_{L,u_L}\right)(x)\right) = p_{U_\ell|Z_\ell}(u_\ell|z_\ell) \end{aligned}$$

by the definition of z_ℓ, where we write $G_\ell(\cdot; u) \equiv G_{\ell,u}(\cdot)$ to remove any ambiguities when composing functions. We can substitute this result into equation 8 to obtain equation 7.

B Discrete Case

In this section, we briefly discuss the drawbacks of using a discrete u. First, observe that the choice of number of discrete values taken by u has immediate implications for the number of disconnected components of $\operatorname{supp} p_X^\star$ that G can separate, which therefore seems to require making fairly concrete assumptions (perhaps implicitly) about the topology of the target of interest. To mitigate this, we might try taking the number of u values to be very large, but then in turn the time required to evaluate equation 4 (now as a sum over the components instead of an integral) necessarily increases. This is particularly true when using a stacked architecture, since to evaluate $p_X(x)$ with L layers each having K possible u-values takes $\Theta(K^L)$ complexity. Dinh et al. (2019) propose a model that partitions $\mathcal{X}$ so that only one component in each summation is nonzero for any given x, which reduces this cost to $\Theta(L)$. However, this partitioning means that their p_X is not continuous as a function of x, which is reported to make the optimisation problem in equation 2 difficult. We thus cannot rely on the otherwise seemingly convenient exact form of the log-likelihood in the discrete case.

Not having an exact log-likelihood was not prohibitive in the continuous case though, as we were able to introduce an approximate posterior and perform variational inference to instead maximise a lower bound on the marginal log-likelihood. Unfortunately, we will not be able to use variational methods when u is exactly discrete. Since $p_{U_\ell|Z_{\ell-1}}$ defines a discrete distribution at each layer, we would require our variational distribution to also be a discrete distribution to avoid high-variance estimates of the ELBO, but the parameters of a discrete distribution *cannot* be optimised using the reparametrisation trick.

As mentioned earlier, we could have alternatively used the CONCRETE distribution (Maddison et al., 2016) to relax the discreteness of u and still apply variational methods – this is indeed a topic for future work but would still require specifying an appropriate number of mixture components.

C Further Experimental Details

C.1 2-D experiments

To gain intuitions about our model, we ran several experiments on the simple 2-D datasets shown in Figure 1 and Figure 4. Specifically, we compared the performance of a baseline MAF against an LGF-MAF. For the LGF-MAF, s_ℓ and t_ℓ were obtained as the joint output of a multi-layer perceptron (MLP), and the means and standard deviations of each $p_{U_\ell|Z_{\ell-1}}$ and $q_{U_\ell|Z_\ell}$ were similarly each obtained as the joint output of an MLP.

For the dataset shown in Figure 1, the baseline MAF had 10 autoregressive layers consisting of 4 hidden layers with 50 hidden units. The LGF-MAF had 5 autoregressive layers consisting of 2 hidden layers with 50 hidden units. Additionally, the s_ℓ/t_ℓ network consisted of 2 hidden layers of 10 hidden units, and the mean/standard deviation networks consisted of 4 hidden layers of 50 hidden units. In total the baseline MAF had 80080 parameters, while our model had 80810 parameters. We trained both models for 300 epochs.

We used more parameters for the datasets shown in Figure 4, since these targets have more complicated topologies. In particular, the baseline MAF had 20 autoregressive layers, each with the same structure as before. The LGF-MAF had 5 autoregressive layers, now with 4 hidden layers of 50 hidden units. The s_ℓ/t_ℓ network and the mean/standard deviation networks were the same as before. In total the baseline MAF had 160160 parameters, while our model had 80810 parameters. We trained both models now for 500 epochs.

The results of this experiment are shown in Figure 1 and Figure 4. Observe that LGF-MAF consistently produces a more faithful representation of the target distribution than the baseline. A failure mode of our approach is exhibited in the spiral dataset, where our model 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 the mean/standard deviation networks to 8 hidden layers of 50 hidden units (and keeping all other parameters fixed), we were able to obtain the result shown in Figure 5. This model had a total of 221910 parameters. For the sake of a fair comparison, we also tried increasing the complexity of the MAF model, by the size of its autoregressive networks to 8 hidden layers of 50 hidden units (obtaining 364160 parameters total). This model diverged after approximately 160 epochs. The result after 150 epochs is shown in Figure 5.

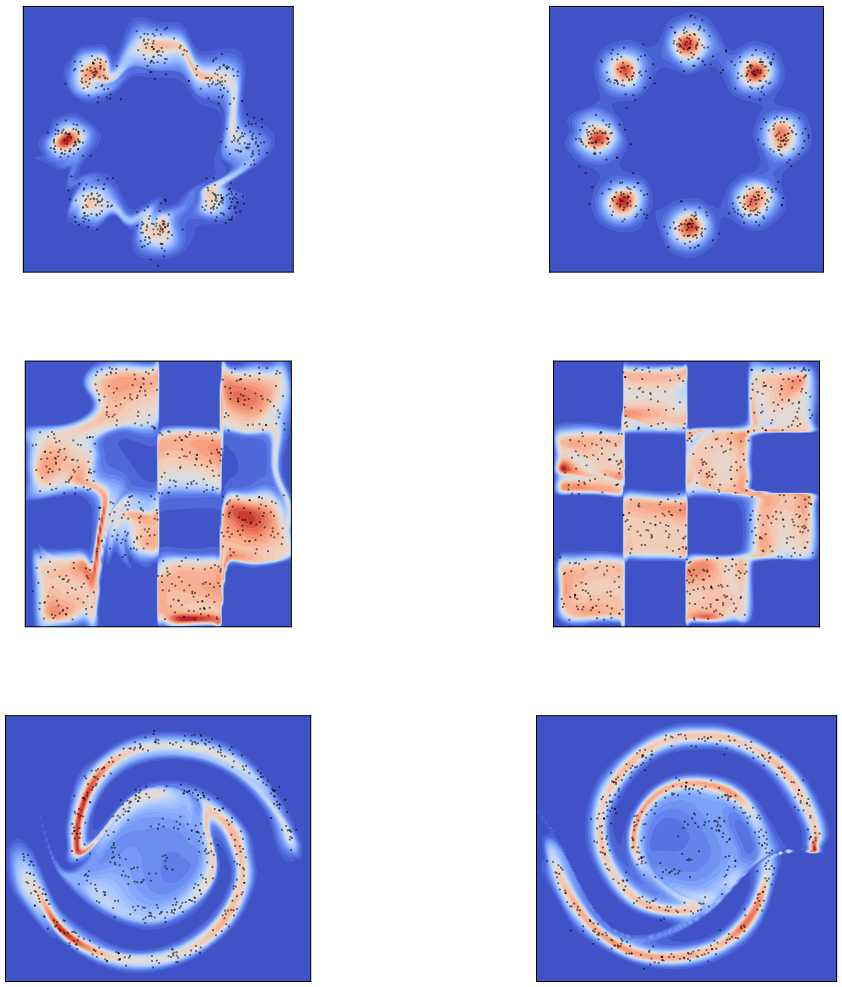

Figure 4: Density models learned by a standard 20 layer MAF (left) and by a 5 layer LGF-MAF (right) for a variety of 2-D target distributions. Samples from the target are shown in black.

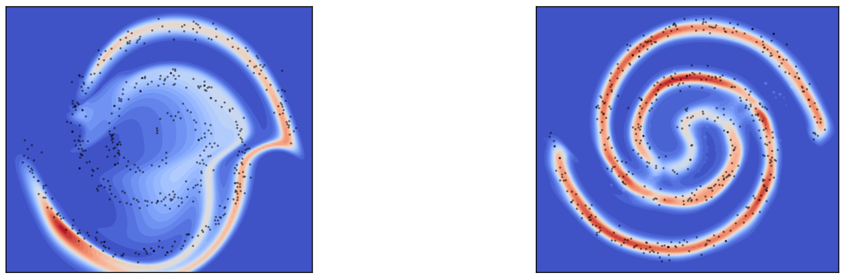

Figure 5: Density models learned by a larger 20 layer MAF (left) and a larger 5 layer LGF-MAF (right) for the spirals dataset.

C.2 UCI EXPERIMENTS

C.2.1 EXTENDED MAF

To turn a MAF model into an LGF-MAF model here, at each layer we must specify the conditional generative distribution $p_{U_\ell|Z_{\ell-1}}$, the one-layer approximate posterior $q_{U_\ell|Z_\ell}$, and the affine transformation networks s_ℓ and t_ℓ. We use multilayer perceptrons (MLPs) for all of these, in particular:

- We set $p_{U_\ell|Z_{\ell-1}} = \mathcal{N}\left(\mu_p(Z_{\ell-1}), \sigma_p(Z_{\ell-1})^2\right)$, where μ_p and σ_p are two separate outputs of the same MLP
- We set $q_{U_\ell|Z_\ell} = \mathcal{N}\left(\mu_q(Z_\ell), \sigma_q(Z_\ell)^2\right)$, where μ_q and σ_q are two separate outputs of the same MLP
- We set s_ℓ and t_ℓ to be two separate outputs of the same MLP

C.2.2 PARAMETER CONFIGURATIONS

In tables 3 and 4, we list the choices of parameters for MAF and LGF-MAF. In all cases, we allowed the base MAF to have more layers and deeper coupler networks to compensate for the additional parameters added by p, q, s, and t. Note that neural networks are listed as size $L \times K$, where L denotes the number of hidden layers and K denotes the size of the hidden layers. All combinations of parameters in list form in the table below were considered. In each case, there are 9 configurations for MAF and 8 configurations for LGF-MAF.

Table 3: Parameter configurations for HEPMASS and MINIBOONE.

	Layers	**Coupler size**	u **size**	p**,** q **size**	s, t **size**
MAF	5, 10, 20	2×128, 2×512, 2×1024	N/A	N/A	N/A
LGF-MAF	5, 10	2×128	10	$2 \times 128, 2 \times 512$	2×128

Table 4: Parameter configurations for POWER and GAS.

	Layers	**Coupler size**	u **size**	p**,** q **size**	s, t **size**
MAF	5, 10, 20	2×100, 2×200, 2×400	N/A	N/A	N/A
LGF-MAF	5, 10	2×128	10	$2 \times 100, 2 \times 200$	2×128

C.3 IMAGE EXPERIMENTS

In figures 6 to 11, we present some samples synthesised from the density models trained on Fashion-MNIST and CIFAR-10.

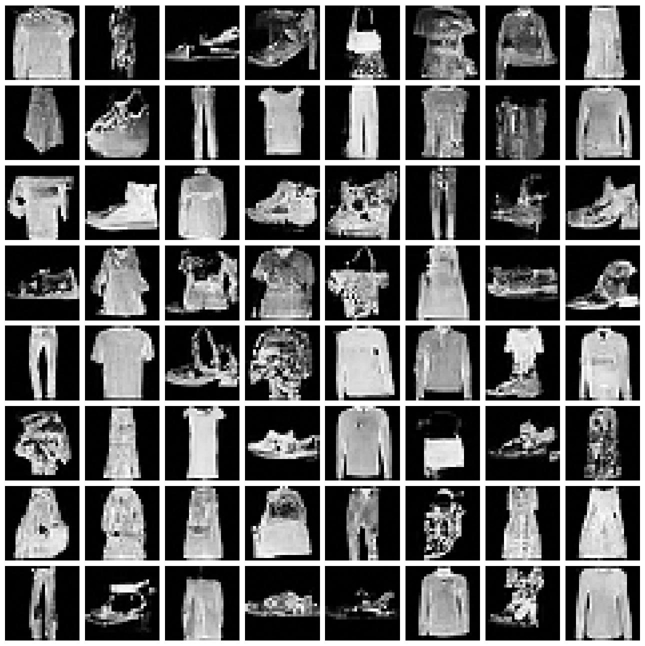

Figure 6: Synthetic samples from Fashion-MNIST generated by RealNVP (4)

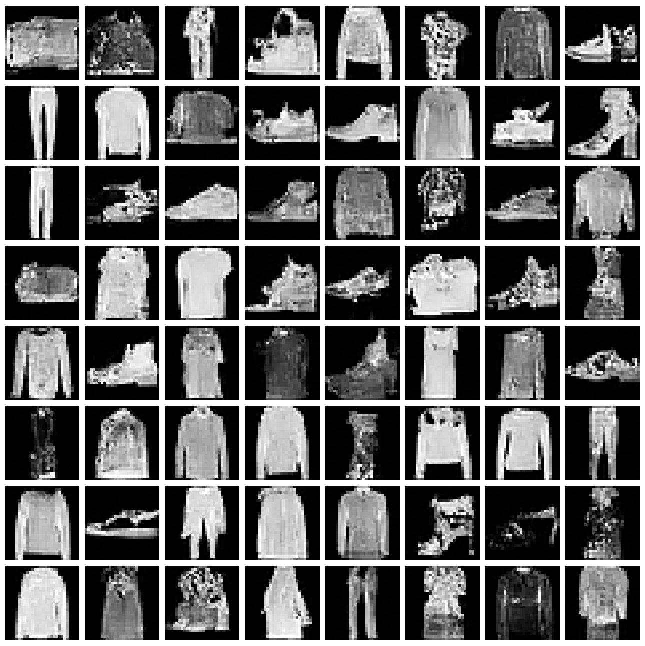

Figure 7: Synthetic samples from Fashion-MNIST generated by RealNVP (8)

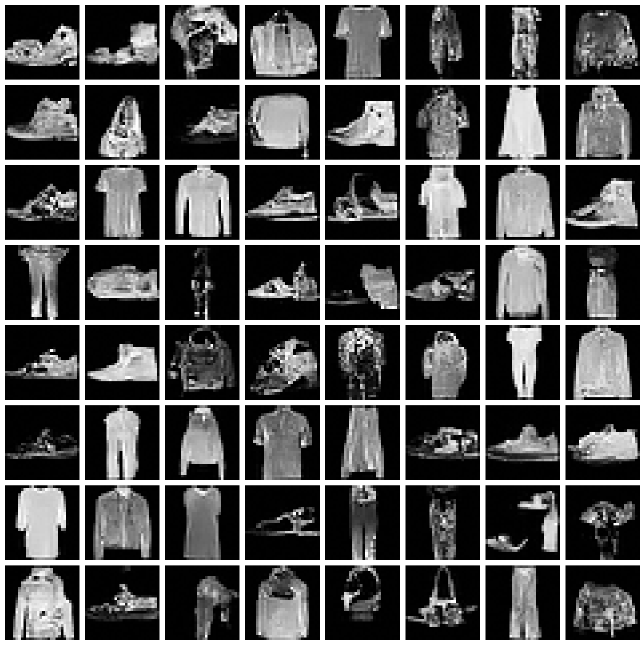

Figure 8: Synthetic samples from Fashion-MNIST generated by LGF-RealNVP (4)

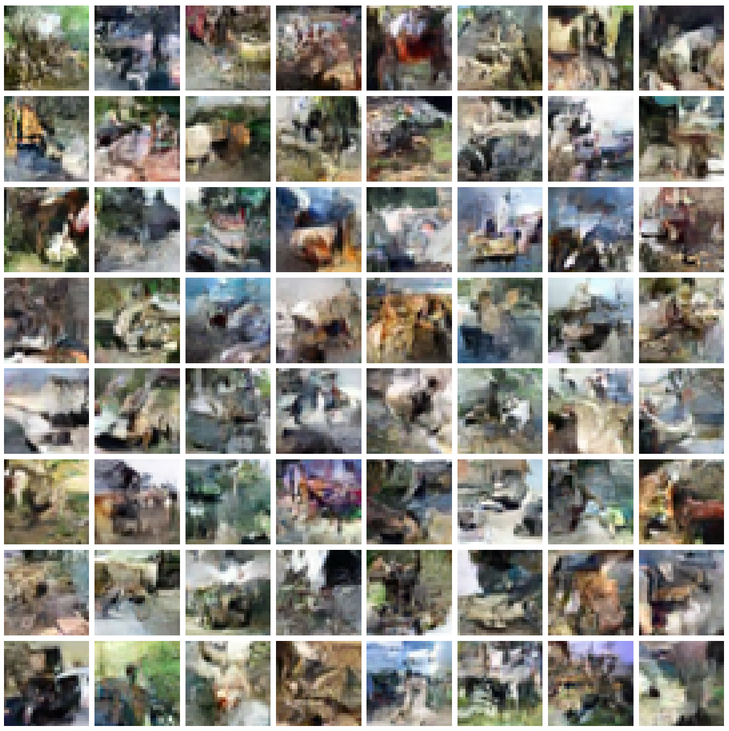

Figure 9: Synthetic samples from CIFAR-10 generated by RealNVP (4)

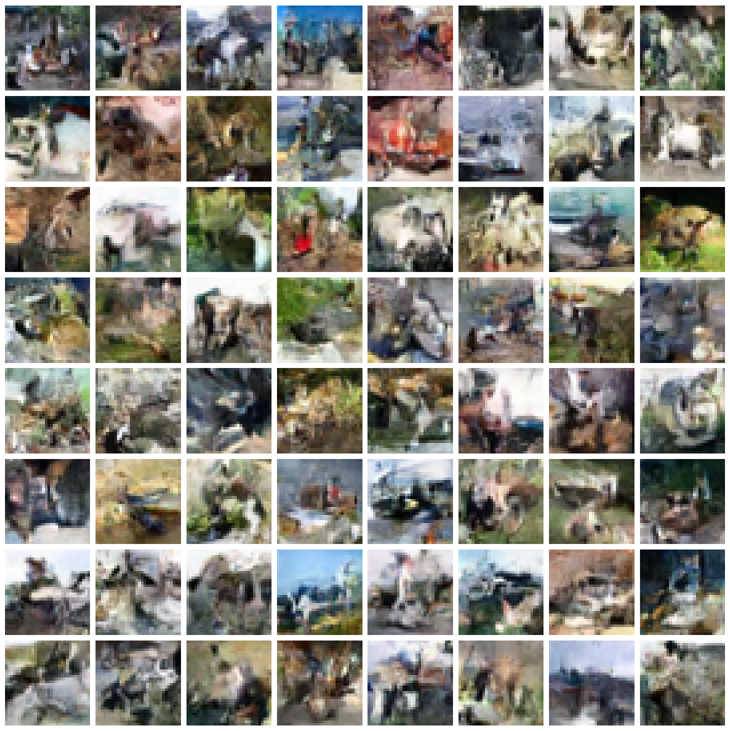
Figure 10: Synthetic samples from CIFAR-10 generated by RealNVP (8)

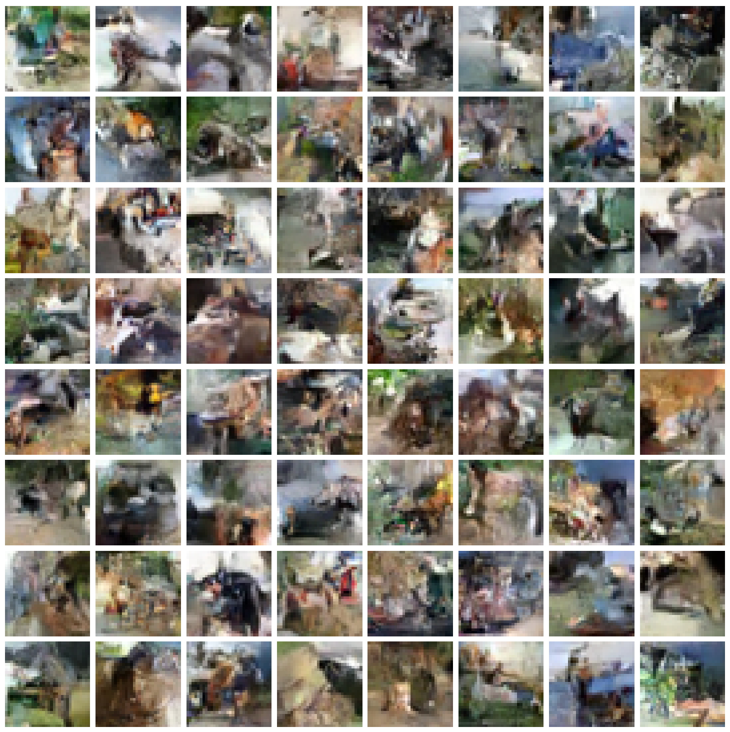
Figure 11: Synthetic samples from CIFAR-10 generated by LGF-RealNVP (4)