

A RAD approach to deep mixture models

Laurent Dinh

Google Brain

LAURENTDINH@GOOGLE.COM

Jascha Sohl-Dickstein

Google Brain

JASCHASD@GOOGLE.COM

Hugo Larochelle

Google Brain

HUGOLAROCHELLE@GOOGLE.COM

Razvan Pascanu

DeepMind

RAZP@GOOGLE.COM

Abstract

Flow based models such as REAL NVP are an extremely powerful approach to density estimation. However, existing flow based models are restricted to transforming continuous densities over a continuous input space into similarly continuous distributions over continuous latent variables. This makes them poorly suited for modeling and representing discrete structures in data distributions, for example class membership or discrete symmetries. To address this difficulty, we present a normalizing flow architecture which relies on domain partitioning using locally invertible functions, and possesses both real and discrete valued latent variables. This Real and Discrete (RAD) approach retains the desirable normalizing flow properties of exact sampling, exact inference, and analytically computable probabilities, while at the same time allowing simultaneous modeling of both continuous and discrete structure in a data distribution.

1. Introduction

Latent generative models are one of the prevailing approaches for building expressive and tractable generative models. The generative process for a sample \mathbf{x} can be expressed as

$$\mathbf{z} \sim p_Z(\mathbf{z}) \tag{1}$$

$$\mathbf{x} = g(\mathbf{z}), \tag{2}$$

where \mathbf{z} is a noise vector, and g a parametric *generator network* (typically a deep neural network). This paradigm has several implementations, including *variational autoencoders* (Kingma and Welling, 2014; Rezende et al., 2014) and *generative adversarial networks* (Goodfellow et al., 2014). Here, we base our work on *flow based models* (Baird et al., 2005; Tabak and Turner, 2013; Dinh et al., 2015, 2017; Kingma and Dhariwal, 2018; Chen et al., 2018; Grathwohl et al., 2019) approaches.

The training process and model architecture for many existing latent generative models, and for all published flow based models, assumes a unimodal smooth distribution over latent variables \mathbf{z} . Given the parametrization of g as a neural network, the mapping to

\mathbf{x} is a continuous function. This imposed structure makes it challenging to model data distributions with discrete structure – for instance, multi-modal distributions, distributions with holes, distributions with discrete symmetries, or distributions that lie on a union of manifolds (as may approximately be true for natural images, see Tenenbaum et al., 2000). Indeed, such cases require the model to learn a generator whose input Jacobian has highly varying or infinite magnitude to separate the initial noise source into different clusters. Such variations imply a challenging optimization problem due to large changes in curvature and introduces numerical instabilities into actual computation of log-likelihood (Behrmann et al., 2020). This shortcoming can be critical as several problems of interest are hypothesized to follow a clustering structure, i.e. the distributions is concentrated along several disjoint connected sets (Eghbal-zadeh et al., 2018).

A standard way to address this issue has been to use *mixture models* (Yeung et al., 2017; Richardson and Weiss, 2018; Eghbal-zadeh et al., 2018) or structured priors (Johnson et al., 2016). In order to efficiently parametrize the model, mixture models are often formulated as a *discrete latent variable models* (Hinton and Salakhutdinov, 2006; Courville et al., 2011; Mnih and Gregor, 2014; van den Oord et al., 2017), some of which can be expressed as a *deep mixture model* (Tang et al., 2012; Van den Oord and Schrauwen, 2014; van den Oord and Dambre, 2015). Although the resulting exponential number of mixture components with depth in deep mixture models is an advantage in terms of expressivity, it is an impediment to inference, evaluation, and training of such models, often requiring as a result the use of approximate methods like *hard-EM* or variational inference (Neal and Hinton, 1998).

In this paper we combine piecewise invertible functions with discrete auxiliary variables, selecting which invertible function applies, to describe a deep mixture model. This framework enables a probabilistic model’s latent space to have both real and discrete valued units, and to capture both continuous and discrete structure in the data distribution. It achieves this added capability while preserving the exact inference, exact sampling, exact evaluation of log-likelihood, and efficient training that make standard flow based models desirable.

2. Model definition

We aim to learn a parametrized distribution $p_X(\mathbf{x})$ on the continuous input domain \mathbb{R}^d by maximizing log-likelihood. The major obstacle to training an expressive probabilistic model is typically efficiently evaluating log-likelihood.

2.1 Partitioning

If we consider a mixture model with a large number $|K|$ of components, where $|K|$ is the number of values K takes, the likelihood takes the form

$$p_X(\mathbf{x}) = \sum_{k=1}^{|K|} p_K(k) p_{X|K}(\mathbf{x} | k).$$

In general, evaluating the likelihood requires computing probabilities for all $|K|$ components. However, following a strategy similar to Rainforth et al. (2018); Cundy and Ermon (2020); Müller et al. (2019); Durkan et al. (2019); Dolatabadi et al. (2020), if we partition the domain \mathbb{R}^d into disjoint subsets \mathbb{A}_k for $1 \leq k \leq |K|$ such that $\forall i \neq j \quad \mathbb{A}_i \cap \mathbb{A}_j = \emptyset$ and

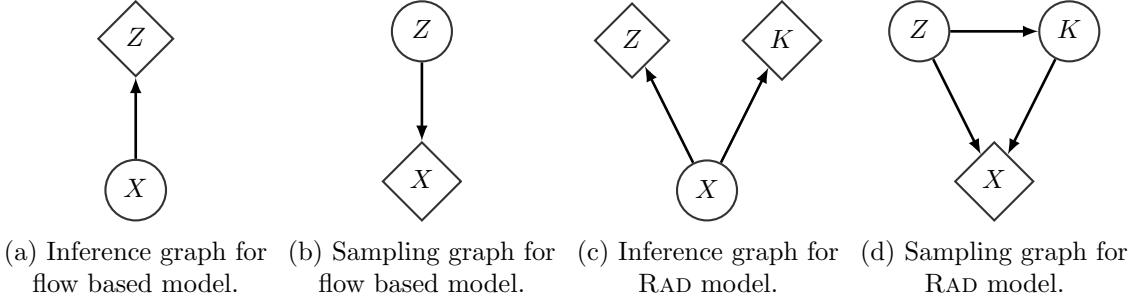


Figure 1: Stochastic computational graphs for inference and sampling for flow based models (1a, 1b) and a RAD model (1c, 1d). Note the dependency of K on Z in 1d. While this is not necessary, we will exploit this structure as highlighted later in the main text and in Figure 4.

$\bigcup_{k=1}^{|K|} \mathbb{A}_k = \mathbb{R}^d$, constrain the support of $p_{X|K}(\mathbf{x} | k)$ to \mathbb{A}_k (i.e. $\forall \mathbf{x} \notin \mathbb{A}_k, p_{X|K}(\mathbf{x} | k) = 0$), and define a set identification function $f_K(\mathbf{x})$ such that $\forall \mathbf{x} \in \mathbb{R}^d, \mathbf{x} \in \mathbb{A}_{f_K(\mathbf{x})}$ (i.e. $f_K(\mathbf{x}) = \sum_k k \cdot \mathbf{1}(\mathbf{x} \in \mathbb{A}_k)$), we can write the likelihood as

$$p_X(\mathbf{x}) = p_K(f_K(\mathbf{x})) p_{X|K}(\mathbf{x} | f_K(\mathbf{x})). \quad (3)$$

This transforms the problem of summation to a search problem $\mathbf{x} \mapsto f_K(\mathbf{x})$. This can be seen as the inferential converse of a *stratified sampling* strategy (Rubinstein and Kroese, 2016).

2.2 Change of variable formula

The proposed approach will be a direct extension of flow based models (Rippel and Adams, 2013; Dinh et al., 2015, 2017; Kingma and Dhariwal, 2018). Flow based models enable log-likelihood evaluation by relying on the *change of variable formula*

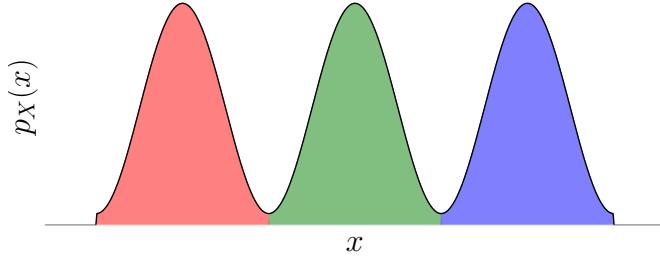
$$p_X(\mathbf{x}) = p_Z(f_Z(\mathbf{x})) \left| \frac{\partial f_Z}{\partial \mathbf{x}^T}(\mathbf{x}) \right|, \quad (4)$$

with f_Z a parametrized bijective function from \mathbb{R}^d onto \mathbb{R}^d and $\left| \frac{\partial f_Z}{\partial \mathbf{x}^T} \right|$ the absolute value of the determinant of its Jacobian.

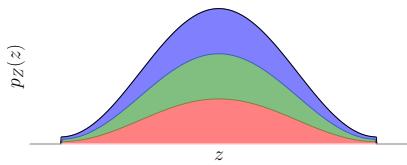
As also proposed in Falorsi et al. (2019), we relax the constraint that f_Z be bijective, and instead have it be surjective onto \mathbb{R}^d and piecewise invertible. That is, for a partition $(\mathbb{A}_k)_k$, we require $f_{Z|\mathbb{A}_k}(\mathbf{x})$ to be an invertible function, where $f_{Z|\mathbb{A}_k}(\mathbf{x})$ indicates $f_Z(\mathbf{x})$ restricted to the domain \mathbb{A}_k . Given a distribution $p_{Z,K}(\mathbf{z}, k) = p_{K|Z}(k | \mathbf{z}) p_Z(\mathbf{z})$ such that $\forall (\mathbf{z}, k), \mathbf{z} \notin f_Z(\mathbb{A}_k) \Rightarrow p_{Z,K} = 0$, we can define the following generative process:

$$\mathbf{z}, k \sim p_{Z,K}(\mathbf{z}, k) \quad (5)$$

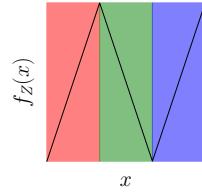
$$\mathbf{x} = (f_Z|_{\mathbb{A}_k})^{-1}(\mathbf{z}). \quad (6)$$



(a) An example of a trimodal distribution p_X , sinusoidal distribution. The different modes are colored in red, green, and blue.



(b) The resulting unimodal distribution p_Z , corresponding to the distribution of any of the initial modes in p_X .



(c) An example $f_Z(x)$ of a piecewise invertible function aiming at transforming p_Z into a unimodal distribution. The red, green, and blue zones corresponds to the different modes in input space.

Figure 2: Example of a trimodal distribution (2a) turned into a unimodal distribution (2b) using a piecewise invertible function (2c). The distribution p_X correspond to an unfolding of p_Z as $p_X(x) = p_Z(3x + 2)\mathbf{1}(x \in \mathbb{A}_1) + p_Z(-3x)\mathbf{1}(x \in \mathbb{A}_2) + p_Z(3x - 2)\mathbf{1}(x \in \mathbb{A}_3)$ where $\mathbb{A}_1 = \left[-1, -\frac{1}{3}\right]$, $\mathbb{A}_2 = \left[-\frac{1}{3}, \frac{1}{3}\right]$, and $\mathbb{A}_3 = \left[\frac{1}{3}, 1\right]$. Here $f_K(x) = 1 \cdot \mathbf{1}(x \in \mathbb{A}_1) + 2 \cdot \mathbf{1}(x \in \mathbb{A}_2) + 3 \cdot \mathbf{1}(x \in \mathbb{A}_3)$

If we use the set identification function f_K associated with \mathbb{A}_k , the distribution corresponding to this stochastic inversion can be defined by a change of variable formula

$$p_X(\mathbf{x}) = \sum_{k=1}^{|K|} p_{Z,K}(f_Z(\mathbf{x}), k) \left| \frac{\partial f_Z|_{\mathbb{A}_k}}{\partial \mathbf{x}^T} \right| \quad (7)$$

$$= p_{Z,K}(f_Z(\mathbf{x}), f_K(\mathbf{x})) \left| \frac{\partial f_Z}{\partial \mathbf{x}^T} \right|. \quad (8)$$

see Figure 2 for an example.

This contrasts with Cornish et al. (2020), which uses instead continuous indexing for k but relies as a consequence on approximate variational inference for training their resulting model. Because of the use of both *Real* and *Discrete* stochastic variables, we call this class of model RAD. The particular parametrization we use on is depicted in Figure 2. We rely on piecewise invertible functions that allow us to define a mixture model of repeated

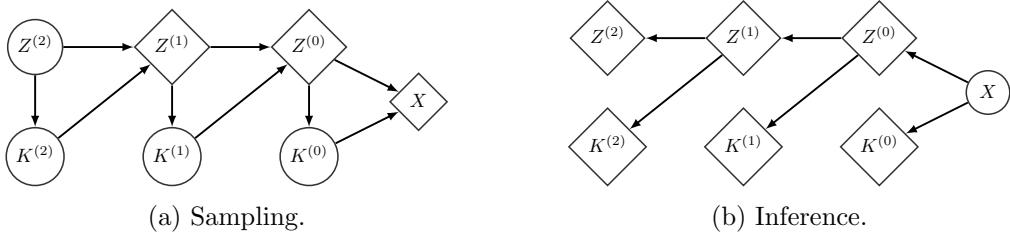


Figure 3: Stochastic computational graph in a deep RAD mixture model of $\prod_{l=1}^3 |K^{(l)}|$ components.

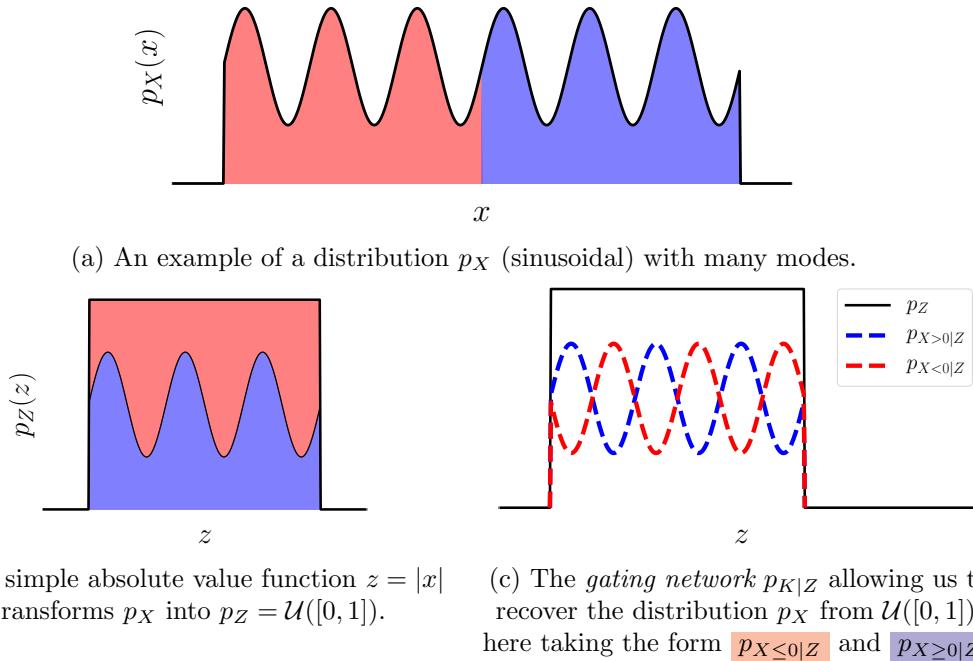


Figure 4: Illustration of the expressive power the gating distribution $p_{K|Z}$ provides. By capturing the structure of a sine wave in $p_{K|Z}$, the function $z, k \mapsto x$ can take on an extremely simple form, corresponding only to a linear function with respect to z . Therefore, p_X does not need to be a duplicated version of p_Z .

symmetrical patterns, following a method of *folding the input space*. In general, we use a mechanism similar to Montufar et al. (2014): the non-invertibility of the surjection enables the model to share statistical strength between the different pieces. Note that in this instance the function f_K is implicitly defined by f_Z , as the discrete latent corresponds to which invertible component of the piecewise function \mathbf{x} falls on.

So far, we have defined a mixture of $|K|$ components with disjoint support. We can see in Figure 2 how it allows us to obtain K different modes of the distribution. However, if we factorize $p_{Z,K}$ as $p_Z \cdot p_{K|Z}$, we can apply another piecewise invertible map to Z to define p_Z

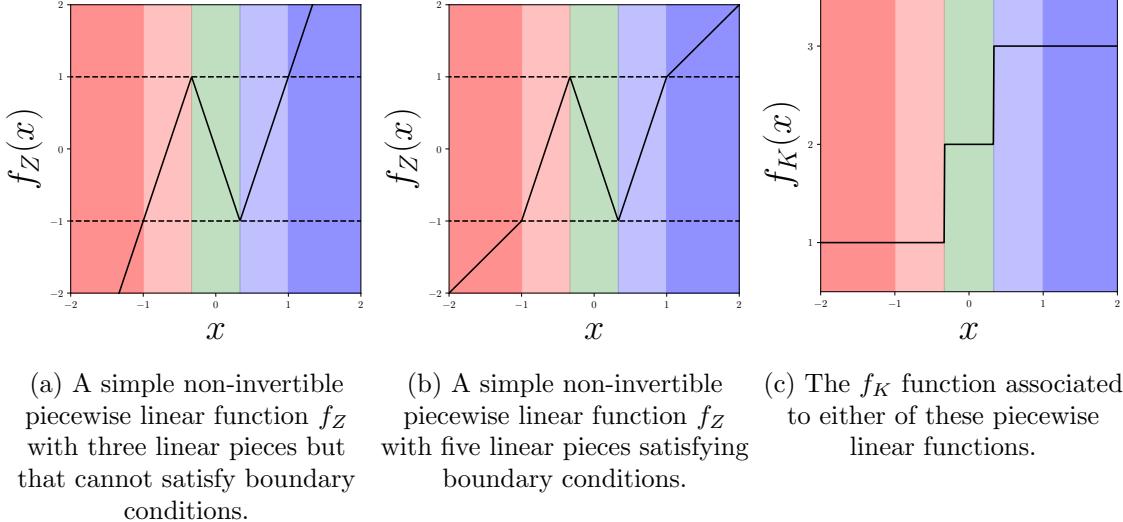


Figure 5: Simple piecewise linear scalar function f_Z before (5a) and after (5b) satisfying boundary conditions. The colored area correspond to the different indices for the mixture components, lighter color for non-invertible areas. The area between the dashed lines correspond to the non-invertible area in the output space. In 5c, we show the f_K function resulting from these nonlinearities.

as another multimodal mixture model. Recursively applying this method results in a deep mixture model (see Figure 3).

Another advantage of such factorization is in the *gating network* $p_{K|Z}$, as also designated in (van den Oord and Dambre, 2015). It provides a more constrained but less sample wasteful approach than rejection sampling (Grover et al., 2018; Azadi et al., 2019; Bauer and Mnih, 2019) of taking into account the untransformed sample \mathbf{z} before selecting the mixture component k . This allows the model to exploit the distribution p_Z in different regions \mathbb{A}_k in more complex ways than repeating it as a pattern as illustrated in Figure 4.

3. Building the surjection

3.1 A minimal example of non-invertible surjection

As a first example of a non-invertible but piecewise invertible surjection, we choose a scalar piecewise linear model defined as follow by defining boundaries $a \leq b$ (see Figure 5a),

$$f_Z(x) = \begin{cases} \alpha_1(x - a) - \alpha_2a + c, & \text{if } x \leq a \\ \alpha_3(x - b) - \alpha_2b + c, & \text{if } x \geq b \\ -\alpha_2x + c, & \text{if } x \in [a, b] \end{cases}. \quad (9)$$

This parametrized function can represent affine functions under the particular case $a = b$ and $\alpha_i = \alpha$.

Since an output can at most have 3 different inverses, we have $|K| = 3$. This is one of the simplest continuous, numerically stable, and differentiable almost everywhere surjection

we can propose. Indeed, for a uniformly continuous surjective non-invertible (and therefore not strictly monotone) scalar function, there exists a point whose preimage has at least three elements. Moreover, each inverse of an output is simply an affine function of the output.

A non-monotonic \mathcal{C}^1 scalar function would have its derivative change sign and have its log-derivative, which is ultimately a contribution to the log-likelihood, go to $-\infty$ (according the intermediate value theorem). A function which would admit $\pm\infty$ as a limit on a point might suffer from numerical instability.

The image, the set of outputs, of a parametrized non-surjective function is in general cannot be defined compactly enough to define a simple distribution over this image, the image of a deep rectified network for example. If the support of p_Z is larger than this image, the inverse is either not defined or has to be replaced by an improvised inverse function, resulting in a lower bound on the actual generative model log-likelihood. For example, there is no inverse of the absolute value on \mathbb{R}_- . However, one could use the identity function and $z \mapsto -z$ as makeshift inverses of the absolute value function on \mathbb{R}_- . This means that x have now two points of origin $z = |x|$ and $z = -|x|$, meaning that only using $z = |x|$ results in a lower bound on the log-likelihood. On other hand, taking into account all the points of origin of x and summing their contributions into the likelihood takes us back to the problem of evaluating many mixture components.

Through affine functions before and after f_Z , we can pick without loss of generality a normalized version of f_Z where at the boundaries $f_Z(a) = -f_Z(b) = 1$, $f_Z(1) = -f_Z(-1) = 1$, and $-1 < a < b < 1$ for further analyses. This implies that

$$\sum_{i=1}^3 \alpha_i^{-1} = 1 \quad (10)$$

$$a = 2\alpha_1^{-1} - 1 \quad (11)$$

$$b = 1 - 2\alpha_3^{-1} \quad (12)$$

$$c = 1 + \alpha_2(2\alpha_1^{-1} - 1) \quad (13)$$

and therefore

$$f_Z(x) = \begin{cases} \alpha_1(x+1) - 1, & \text{if } x \leq a \\ \alpha_3(x-1) + 1, & \text{if } x \geq b \\ -\alpha_2(x-a) + 1, & \text{if } x \in [a, b] \end{cases}. \quad (14)$$

3.2 Continuity

The standard approach in learning a deep probabilistic model has been stochastic gradient descent on the negative log-likelihood. Although the model enables the computation of a gradient almost everywhere, the log-likelihood is unfortunately discontinuous. Let's decompose the log-likelihood

$$\log(p_X(\mathbf{x})) = \log(p_Z(f_Z(\mathbf{x}))) + \log(p_{K|Z}(f_K(\mathbf{x}) | f_Z(\mathbf{x}))) + \log\left(\left|\frac{\partial f_Z}{\partial \mathbf{x}^T}\right|(\mathbf{x})\right). \quad (15)$$

There are two sources of discontinuity in this expression with respect to the parameters and the input \mathbf{x} (which can be the output of another parametrized function): f_K is a function

with discrete values (therefore discontinuous, see Figure 5c and 6) and $\frac{\partial f_Z}{\partial \mathbf{x}^T}$ is discontinuous because of the transitions between the subsets \mathbb{A}_k , leading to the expression of interest

$$\log(p_{K|Z}(f_K(\mathbf{x}) | f_Z(\mathbf{x}))) + \log\left(\left|\frac{\partial f_Z}{\partial \mathbf{x}^T}\right|(\mathbf{x})\right),$$

which takes on a role similar to that of the log-Jacobian determinant, a *pseudo log-Jacobian determinant*. In the scalar case $\left|\frac{\partial f_Z}{\partial \mathbf{x}^T}\right|(\mathbf{x})$ is replaced by $|f'_Z(x)|(\mathbf{x})$.

We can attempt at parametrizing the surjection such that the pseudo log-Jacobian determinant becomes continuous with respect to \mathbf{x} by expressing the boundary condition at $x = b$

$$\lim_{x \uparrow b} \log(p_{K|Z}(f_K(x) | f_Z(x))) + \log(|f'_Z|(x)) \quad (16)$$

$$= \lim_{x \downarrow b} \log(p_{K|Z}(f_K(x) | f_Z(x))) + \log(|f'_Z|(x)). \quad (17)$$

where $\lim_{x \uparrow b}$ and $\lim_{x \downarrow b}$ are the one-sided limits at b from below and above. By remembering that $\forall x > b, |f'_Z|(x) = \alpha_3$ and $f_K(x) = 3$, and $\forall x \in]a, b[, |f'_Z|(x) = \alpha_2$ and $f_K(x) = 2$, we obtain

$$\Rightarrow \log(p_{K|Z}(2 | f_Z(b))) - \log(p_{K|Z}(3 | f_Z(b))) = \log(\alpha_3) - \log(\alpha_2), \quad (18)$$

Similarly, at $x = a$

$$\log(p_{K|Z}(2 | f_Z(a))) - \log(p_{K|Z}(1 | f_Z(a))) = \log(\alpha_1) - \log(\alpha_2). \quad (19)$$

Another type of boundary condition can be found at between the non-invertible area and the invertible area, e.g. $z = 1$. When $z > 1$, $p_{K|Z}(3 | z) = 1$, therefore the boundary condition is

$$\lim_{x \uparrow 1} \log(p_{K|Z}(f_K(x) | f_Z(x))) + \log(|f'_Z|(x)) \quad (20)$$

$$= \lim_{x \downarrow 1} \log(p_{K|Z}(f_K(x) | f_Z(x))) + \log(|f'_Z|(x)) \quad (21)$$

$$\Rightarrow \lim_{x \uparrow 1} \log(p_{K|Z}(3 | f_Z(x))) = \lim_{x \downarrow 1} \log(|f'_Z|(x)) - \lim_{x \uparrow 1} \log(|f'_Z|(x)). \quad (22)$$

This condition cannot be satisfied if $\lim_{x \downarrow 1} |f'_Z|(x) = \lim_{x \uparrow 1} |f'_Z|(x)$. Therefore, we can add more linear pieces at the end and start to redefine our surjection¹ (see Figure 5b):

$$f_Z(x) = \begin{cases} \alpha_-(x+1) - 1, & \text{if } x \leq -1 \\ \alpha_+(x-1) + 1, & \text{if } x \geq 1 \\ \alpha_1(x+1) - 1, & \text{if } x \in [-1, a] \\ \alpha_3(x-1) + 1, & \text{if } x \in [b, 1] \\ -\alpha_2(x-a) + 1, & \text{if } x \in [a, b] \end{cases}. \quad (23)$$

1. In a more general setting, one could create adjustable gaps in derivatives using leaky rectifier functions (Maas et al., 2013) or modifications of spline functions (Müller et al., 2019; Durkan et al., 2019; Dolatabadi et al., 2020) that deliberately create discontinuities in the derivative.

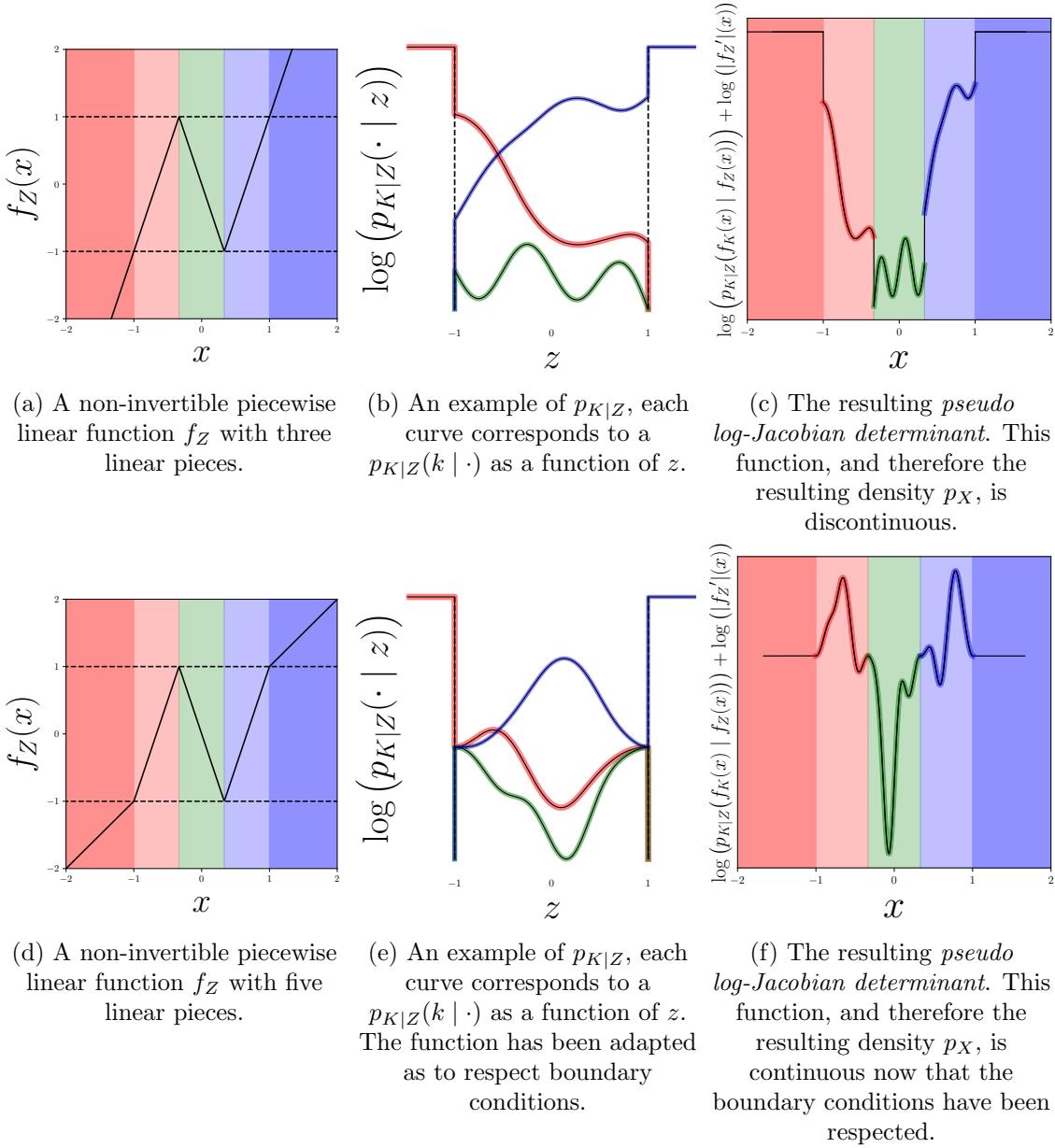


Figure 6: Illustration of the importance of boundary conditions to impose continuity on the loss function. On top, the non-invertible function f_Z has three linear pieces (6a) and cannot satisfy boundary conditions. Moreover, $p_{K|Z}$ (6b) does not satisfy the boundary conditions with f_Z . This results in a discontinuous contribution to the log-likelihood (6c). On the contrary, a non-invertible function with five pieces (6d) can satisfy the boundary conditions with an adjusted $p_{K|Z}$ (6e) to obtain a continuous contribution to the log-likelihood (6f)

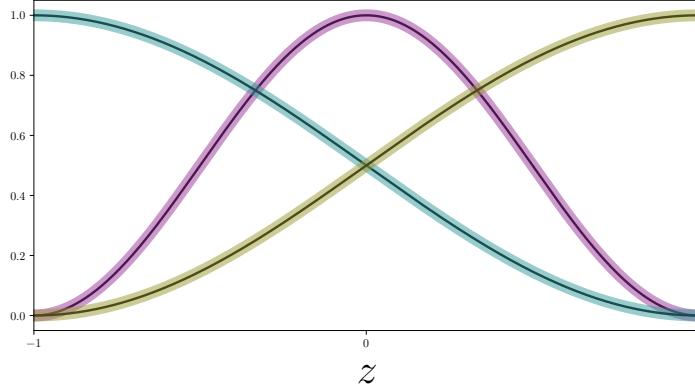


Figure 7: An example of the weightings one can put on the different additive components of $p_{K|Z}(\cdot | z)$. Using trigonometric functions, one can create weights for the middle part (which can be arbitrary defined), left part (corresponding to $p_{K|Z} \cdot (\cdot | f_Z(-1))$), and right part (corresponding to $p_{K|Z} \cdot (\cdot | f_Z(1))$).

With this redefinition, we obtain the satisfiable constraint

$$\log(p_{K|Z}(3 | f_Z(1))) = \log(\alpha_+) - \log(\alpha_3). \quad (24)$$

Similarly, at $z = -1$ we have

$$\log(p_{K|Z}(1 | f_Z(-1))) = \log(\alpha_-) - \log(\alpha_1). \quad (25)$$

Using these boundary constraints with the constraint $\sum_{i=1}^3 \alpha_i^{-1} = 1$ mentioned earlier, the solution to these constraints α_- , α_1 , α_2 , α_3 and α_+ are uniquely determined from $p_{K|Z}(\cdot | f_Z(1))$ and $p_{K|Z}(\cdot | f_Z(-1))$ (and vice-versa):

$$\alpha_2 = 1 + \frac{p_{K|Z}(1 | f_Z(a))}{p_{K|Z}(2 | f_Z(a))} + \frac{p_{K|Z}(3 | f_Z(b))}{p_{K|Z}(1 | f_Z(b))} \quad (26)$$

$$\alpha_1 = \frac{p_{K|Z}(2 | f_Z(a))}{p_{K|Z}(1 | f_Z(a))} \left(1 + \frac{p_{K|Z}(1 | f_Z(a))}{p_{K|Z}(2 | f_Z(a))} + \frac{p_{K|Z}(3 | f_Z(b))}{p_{K|Z}(2 | f_Z(b))} \right) \quad (27)$$

$$\alpha_3 = \frac{p_{K|Z}(2 | f_Z(b))}{p_{K|Z}(3 | f_Z(b))} \left(1 + \frac{p_{K|Z}(1 | f_Z(a))}{p_{K|Z}(2 | f_Z(a))} + \frac{p_{K|Z}(3 | f_Z(b))}{p_{K|Z}(2 | f_Z(b))} \right) \quad (28)$$

$$\alpha_- = \alpha_1 \cdot p_{K|Z}(1 | f_Z(b)) \quad (29)$$

$$\alpha_+ = \alpha_3 \cdot p_{K|Z}(3 | f_Z(a)). \quad (30)$$

Therefore, when using this nonlinearity in the context of a coupling layer (see Figure 10), the only parameters of interest for a normalized f_Z (such that $f_Z(a) = f_Z(1) = 1 = -f_Z(b) = -f_Z(-1)$) are $p_{K|Z}(\cdot | f_Z(1))$ and $p_{K|Z}(\cdot | f_Z(-1))$, excluding the affine functions one would like to put before and after the normalized non-invertible function.

The values of $p_{K|Z}(\cdot | f_Z(1))$ and $p_{K|Z}(\cdot | f_Z(-1))$ can be controlled by imposing the following form on $p_{K|Z}$ as a function of its desired values at the boundaries (see Figure 7)

$$p_{K|Z}(\cdot | z) \Big|_{z \in [-1,1]} = \frac{1}{2}(1 + \cos(z\pi)) \cdot s(z) \quad (31)$$

$$+ \frac{1}{2} \left(1 + \sin\left(\frac{z\pi}{2}\right)\right) \cdot p_{K|Z}(\cdot | f_Z(1)) \quad (32)$$

$$+ \frac{1}{2} \left(1 - \sin\left(\frac{z\pi}{2}\right)\right) \cdot p_{K|Z}(\cdot | f_Z(-1)), \quad (33)$$

up to an additive constant (for normalization), where $s : \mathbb{R} \mapsto \mathbb{R}^3$ is an arbitrary parametrized function. This reparametrization retains most of the flexibility of $p_{K|Z}$ while allowing us to know its exact value at $z = -1$ and $z = 1$ without evaluation, which becomes critical in a high dimensional setting (to avoid computating the $2d$ evaluations of $p_{K_i|Z}(\cdot | z_i = \pm 1, z_{-i})$). In higher dimension, f_Z and f_K can be applied element-wise while we can choose $s : \mathbb{R}^d \mapsto \mathbb{R}^{3d}$, meaning that the discrete variables K are independent conditioned on *all* dimensions of \mathbf{z} . This is the approach we use later on. Due to the discrete nature of f_K any other dependence on K , including auto-regressivity in $p_{K|Z}$ with respect to previous K (but not on previous X), would result in discontinuities in the log-likelihood function. Using a mixture of conditionally independent distributions for K is another valid approach.

Given those constraints, the model can then be reliably learned through gradient descent methods. Note that the resulting tractability of the model results from the fact that the discrete variables k is only used during inference with the distribution $p_{K|Z}$, unlike discrete variational autoencoders approaches (Mnih and Gregor, 2014; van den Oord et al., 2017) where it is fed to a deep neural network. Similar to Rolfe (2017), the learning of discrete variables is achieved by relying on the continuous component of the model, and, as opposed to other approaches (Bengio et al., 2013; Raiko et al., 2015; Jang et al., 2017; Maddison et al., 2017; Grathwohl et al., 2018; Tucker et al., 2017; Tran et al., 2019; Hoogeboom et al., 2019; van den Berg et al., 2020), this gradient signal extracted is exact and closed form.

3.3 Universal approximation

Here, we will show that this folding function f_Z previously defined, when applied repeatedly, can transform any density function so that it becomes arbitrarily close to the uniform distribution with respect to the uniform topology. Once, this is accomplished, the "inversion" of f_Z back to the density p_X can be accomplished by approximating the gating distribution $p_{K|Z}$ (a mere probabilistic classifier) arbitrarily well. Using a neural network to converge to a given function is a well explored topic (Cybenko, 1989; Hornik, 1991; Pinkus, 1999).

Theorem 1 *For any $C^1([-1, 1])$ strictly positive density p_X^* and $\epsilon > 0$, there is a series of surjections $(f_{Z,n})_{n \leq N}$ from $[-1, 1]$ onto $[-1, 1]$, composed of three linear pieces, such that*

$$\left\| p_{f_{Z,1:N}(X)}^* - \mathcal{U}([-1, 1]) \right\|_\infty < \epsilon.$$

with $f_{Z,1:N} = f_{Z,N} \circ \dots \circ f_{Z,1}$.

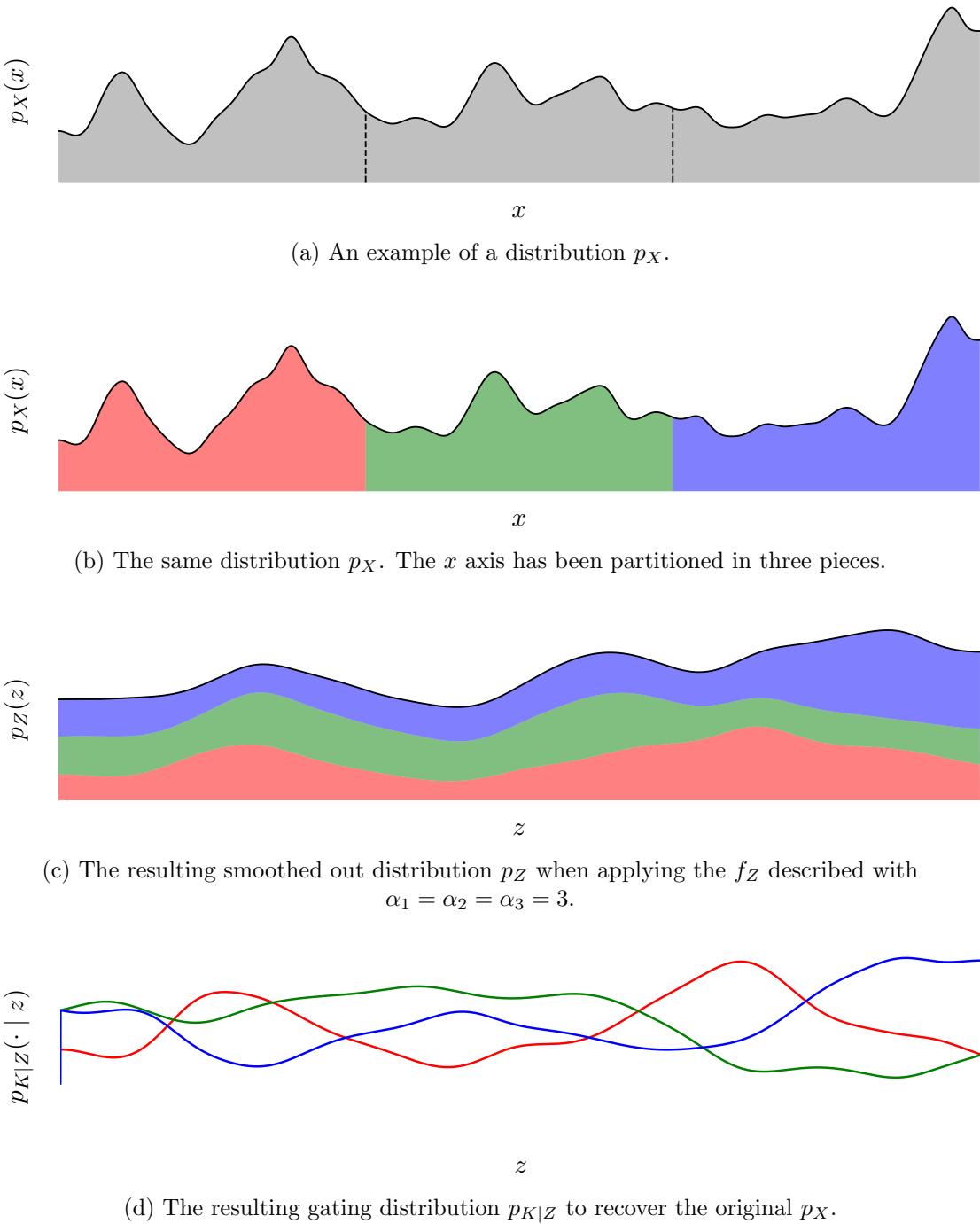


Figure 8: Illustration of the importance of the smoothing out effect of f_Z . A more rigorous explanation of conditions for this smoothing out to be guaranteed is described Subsection 3.3.

Proof $(p_X^*)'$ is continuous on $[0, 1]$ and therefore bounded (by the extreme value theorem). If we apply the previously defined surjection f_Z with $\alpha_1 = \alpha_2 = \alpha_3 = 3$ on a distribution p_X we obtain

$$p_Z(z) = \frac{1}{3} \left(p_X\left(\frac{z-2}{3}\right) + p_X\left(-\frac{z}{3}\right) + p_X\left(\frac{z+2}{3}\right) \right). \quad (34)$$

The Lipschitz bound of p_Z with respect to z is a third of $\|(p_X^*)'\|_\infty$ see Figure 8. Therefore, by repeating that process N times we have

$$N \geq \left\lceil \log_3 \left(\frac{2 \cdot \|(p_X^*)'\|_\infty}{\epsilon} \right) \right\rceil \quad (35)$$

$$\Rightarrow \left\| (p_{f_{Z,1:N}(X)}^*)' \right\|_\infty \leq \frac{1}{2} \epsilon \quad (36)$$

Using Rolle's theorem on the cumulative distribution function corresponding to $p_{f_{Z,1:N}(X)}^*$, there is a $z \in [-1, 1]$ such that $p_{f_{Z,1:N}(X)}^*(z) = \frac{1}{2}$. Therefore

$$\left\| p_{f_{Z,1:N}(X)}^* - \mathcal{U}([-1, 1]) \right\|_\infty < \epsilon. \quad (37)$$

■

To recover the original distribution p_X^* from $p_{f_{Z,1:N}(X)}^*$, one can use the universal approximation property of neural networks to parametrize gating distributions $p_{K|Z}$. Here, we use uniform pieces in f_Z for simplicity of proof, this can be of course a suboptimal choice and the use of different sizes may allow us to use a lower N .

4. Experiments

4.1 Problems

We conduct a brief comparison on six two-dimensional toy problems with REAL NVP to demonstrate the potential gain in expressivity RAD models can enable. Synthetic datasets of 10,000 points each are constructed following the *manifold hypothesis* and/or the *clustering hypothesis*. We designate these problems as: *grid Gaussian mixture*, *ring Gaussian mixture*, *two moons*, *two circles*, *spiral*, and *many moons* (see Figure 9).

4.2 Architecture

For the RAD model implementation, we use the piecewise linear activations defined in Subsection 3.2 in a coupling layer architecture (Dinh et al., 2015, 2017) for f_Z where, instead of a conditional linear transformation, the conditioning variable x_1 determines the parameters of the piecewise linear activation on x_2 to obtain z_2 and k_2 , with $z_1 = x_1$ (see Figure 10). For the gating network $p_{K|Z}$, the gating logit neural network $s(z)$ take as input $z = (z_1, z_2)$. We compare with a REAL NVP model using only affine coupling layers. p_Z is a standard Gaussian distribution.

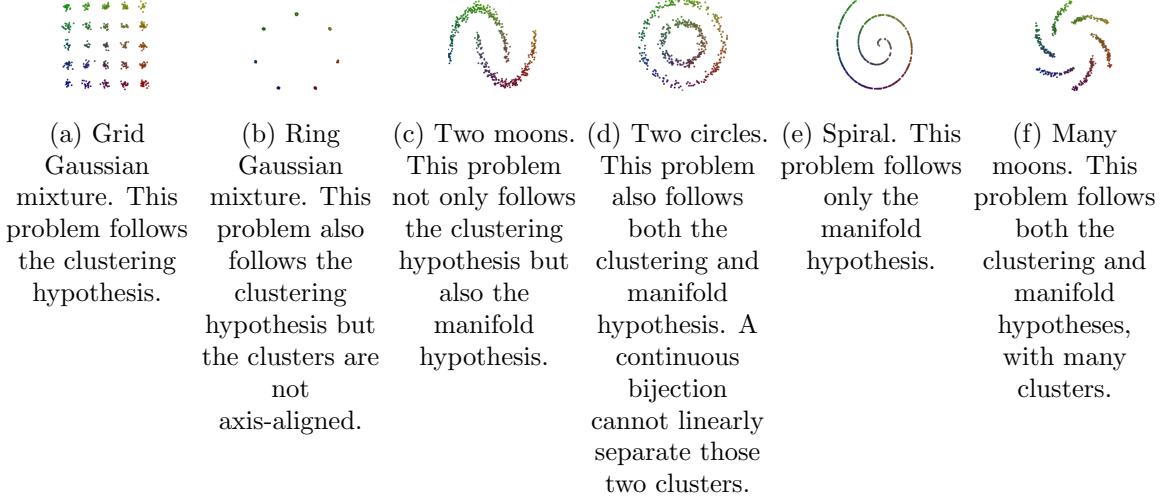


Figure 9: Samples drawn from the data distribution in each of several toy two dimensional problems.

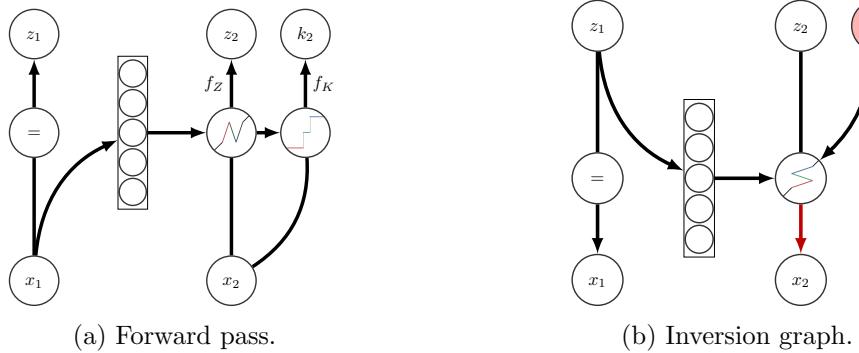


Figure 10: Computational graph of the coupling layers used in the experiments.

As both these models can easily approximately solve these generative modeling tasks provided enough capacity, we study these model in a relatively low capacity regime, where we can showcase the potential expressivity RAD may provide. Each of these models uses six coupling layers, and each coupling layer uses a one-hidden-layer rectified network with a tanh output activation scaled by a scalar parameter as described in Dinh et al. (2017). For RAD, the logit network $s(\cdot)$ also uses a one-hidden-layer rectified neural network, but with linear output. In order to fairly compare with respect to number of parameters, we provide REAL NVP seven times more hidden units per hidden layer than RAD, which uses 8 hidden units per hidden layer. For each level, $p_{K|Z}$ and f_Z are trained using stochastic gradient ascent with ADAM (Kingma and Ba, 2015) on the log-likelihood with a batch size of 500 for 50,000 steps.

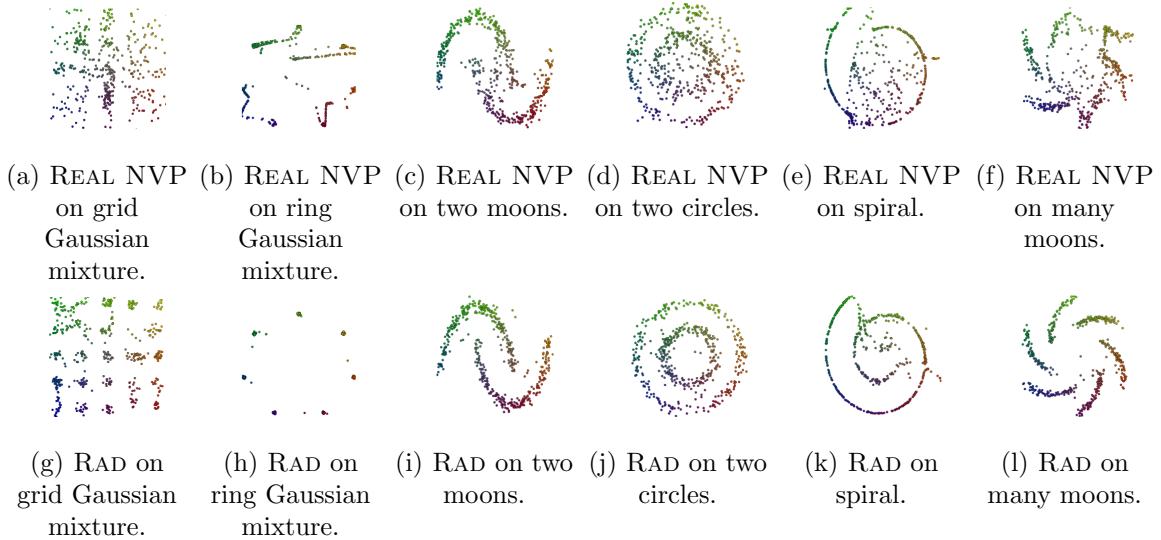


Figure 11: Comparison of samples from trained REAL NVP (top row) (a-f) and RAD (bottom row) (g-l) models. REAL NVP fails in a low capacity setting by attributing probability mass over spaces where the data distribution has low density. Here, these spaces often connect data clusters, illustrating the challenges that come with modeling multimodal data as one continuous manifold.

4.3 Results

In each of these problems, RAD is consistently able to obtain higher log-likelihood than REAL NVP.

	RAD	REAL NVP
Grid Gaussian mixture	-1.20	-2.26
Ring Gaussian mixture	3.57	1.85
Two moons	-1.21	-1.48
Two circles	-1.81	-2.17
Spiral	0.29	-0.36
Many moons	-0.83	-1.50

4.3.1 SAMPLING AND GAUSSIANIZATION

We plot the samples (Figure 11) of the described RAD and REAL NVP models trained on these problems. In the described low capacity regime, REAL NVP fails by attributing probability mass over spaces where the data distribution has low density. This is consistent with the *mode covering* behavior of maximum likelihood. However, the particular inductive bias of REAL NVP is to prefer modeling the data as one connected manifold. This results in the unwanted probability mass being distributed along the space between clusters.

Flow-based models often follow the principle of *Gaussianization* (Chen and Gopinath, 2001), i.e. transforming the data distribution into a Gaussian. The inversion of that process on a Gaussian distribution would therefore approximate the data distribution. We plot in

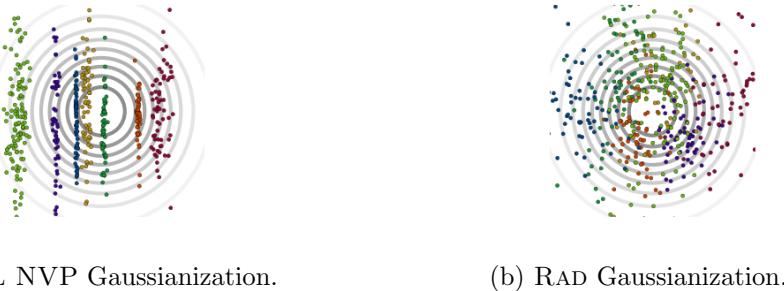


Figure 12: Comparison of the Gaussianization process for RAD and REAL NVP on the ring Gaussian mixture problem. Both plots show the image of data samples in the latent z variables, with level sets of the standard normal distribution plotted for reference. REAL NVP leaves some area of this Gaussian unpopulated, an effect which is not visually apparent for RAD.

Figure 12 the inferred Gaussianized variables $z^{(5)}$ for both models trained on the ring Gaussian mixture problem. The Gaussianization from REAL NVP leaves some area of the standard Gaussian distribution unpopulated. These unattended areas correspond to unwanted regions of probability mass in the input space. RAD suffers significantly less from this problem.

An interesting feature is that RAD seems also to outperform REAL NVP on the spiral dataset. One hypothesis is that the model successfully exploits some non-linear symmetries in this problem.

4.3.2 FOLDING

We take a deeper look at the Gaussianization process involved in both models. In Figure 13 we plot the inference process of $z^{(5)}$ from x for both models trained on the two moons problem. As a result of a folding process similar to that in Montufar et al. (2014), several points which were far apart in the input space become neighbors in $z^{(5)}$ in the case of RAD.

We further explore this folding process using the visualization described in Figure 14. We verify that the non-linear folding process induced by RAD plays at least two roles: bridging gaps in the distribution of probability mass, and exploiting symmetries in the data.

We observe that in the case of the ring Gaussian mixture (Figure 15a), RAD effectively uses foldings in order to bridge the different modes of the distribution into a single mode, primarily in the last layers of the transformation. We contrast this with REAL NVP (Figure 15b) which struggles to combine these modes under the standard Gaussian distribution using bijections.

In the spiral problem (Figure 16), RAD decomposes the spiral into three different lines to bridge (Figure 16a) instead of unrolling the manifold fully, which REAL NVP struggles to do (Figure 16b).

In both cases, the points remain generally well separated by labels, even after being pushed through a RAD layer (Figure 15a and 16a). This enables the model to maximize the conditional log-probability $\log(p_{K|Z})$.

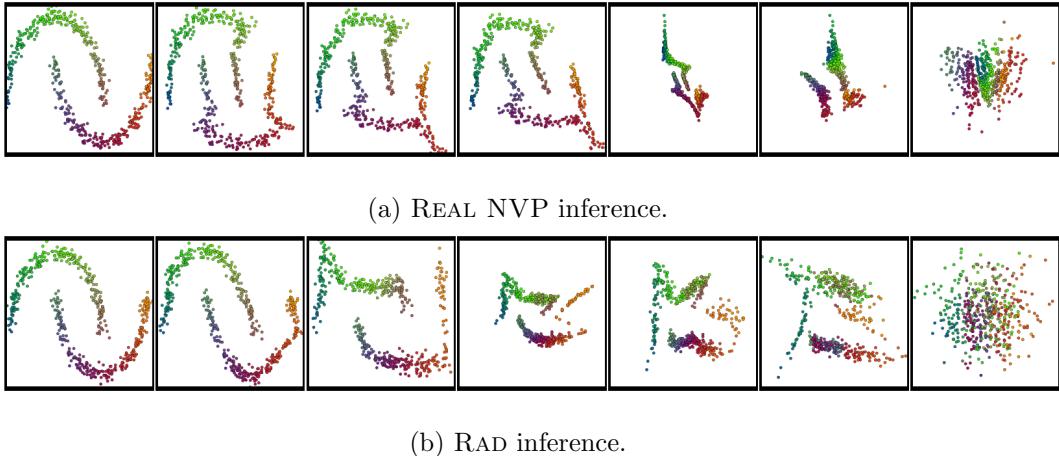


Figure 13: Comparison of the inference process for RAD and REAL NVP on the two moons problem. Each pane shows input samples embedded in different networks layers, progressing from left to right from earlier to later network layers. The points are colored according to their original position in the input space. In RAD several points which were far apart in the input space become neighbors in $z^{(5)}$. This is not the case for REAL NVP.

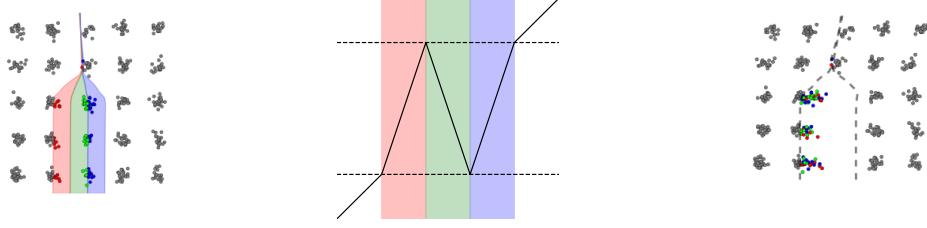
5. Conclusion

We introduced an approach to tractably evaluate and train deep mixture models using surjective piecewise invertible maps as a folding mechanism. This allows exact inference, exact generation, and exact evaluation of log-likelihood, avoiding many issues in previous discrete variables models. This method can easily be combined with other flow based architectural components (for a more comprehensive framework, read Nielsen et al., 2020), allowing flow based models to better model datasets with discrete as well as continuous structure.

ACKNOWLEDGEMENTS

The authors would like to thank Kyle Kastner, Johanna Hansen, Harm De Vries, Ben Poole, Prajit Ramachandran, Dustin Tran, Erin Grant, David Grangier, George J. Tucker, Matt D. Hoffman, Daniel Duckworth, Anna Huang, Arvind Neelakantan, Dale Schuurmans, Graham Taylor, Bart van Merriënboer, Daniel Duckworth, Vincent Dumoulin, Didrik Nielsen, Priyank Jaini, Emiel Hoogeboom, Kyunghyun Cho, Marc G. Bellemare, Ross Goroshin, the *ICLR 2019 Deep Generative Models for Highly Structured Data* reviewers, and the *AISTATS 2020* reviewers, for valuable discussion and feedbacks on this paper. We also thank Brandon Amos, Mathieu Blondel, and Misha Denil for writing advice on this section.

We would also like to thank the Python community (Van Rossum and Drake Jr, 1995; Oliphant, 2007) for developing the tools that enabled this work, including NumPy (Oliphant, 2006; Walt et al., 2011), SciPy (Jones et al., 2001), Matplotlib (Hunter, 2007), Tensorflow (Abadi et al., 2016), and JAX (Bradbury et al., 2018).



(a) Input points of a RAD layer. The red, green, and blue colors corresponds to different labels of the partition subsets ($|K|$ values), domains of \mathbb{A}_k for different k , where the function is non-invertible without knowing k (see (b)). The black points are in the invertible area, where k is not needed for the inversion.

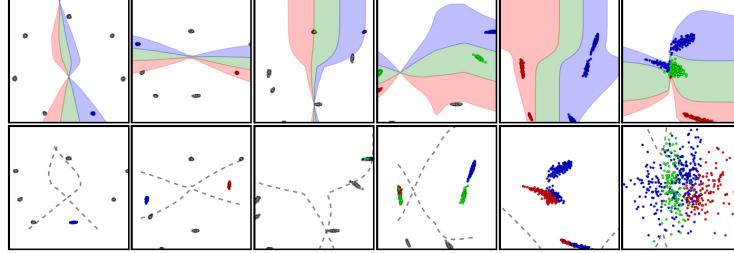
(b) An example of piecewise linear function used in a RAD layer. The red, green, and blue colors corresponds to the different labels of the partition subsets in the non-invertible area. The dashed lines correspond to the non-invertible area in output space.

(c) Output points of a RAD layer. The red, green, and blue colors corresponds to the different labels of the partition subsets in the non-invertible area of the input space, where points are folded on top of each other. The black points are in the invertible area, where k is not needed for the inversion. The dashed lines correspond to the non-invertible area in output space.

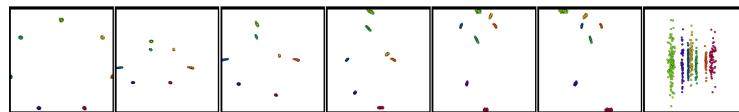
Figure 14: Understanding the folding process, and understanding other visualizations of the folding process.

References

- Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.
- Samaneh Azadi, Catherine Olsson, Trevor Darrell, Ian Goodfellow, and Augustus Odena. Discriminator rejection sampling. In *International Conference on Learning Representations*, 2019.
- Leemon Baird, David Smalenberger, and Shawn Ingkiriwang. One-step neural network inversion with pdf learning and emulation. In *International Joint Conference on Neural Networks*, volume 2, pages 966–971. IEEE, 2005.
- Matthias Bauer and Andriy Mnih. Resampled priors for variational autoencoders. In *Proceedings of the twenty-second international conference on artificial intelligence and statistics*, 2019.
- Jens Behrmann, Paul Vicol, Kuan-Chieh Wang, Roger B Grosse, and Jörn-Henrik Jacobsen. On the invertibility of invertible neural networks. In *International Conference on Learning Representations*, 2020.

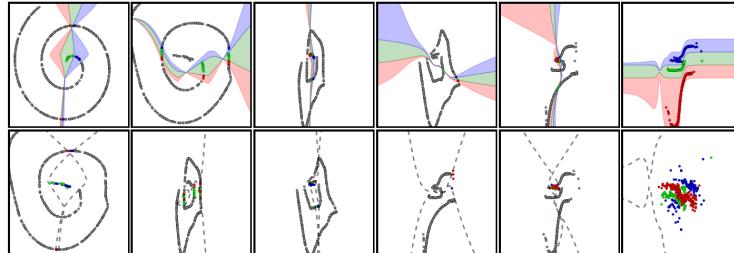


(a) RAD folding strategy on the ring Gaussian mixture problem. The top rows correspond to each RAD layer's input points, and the bottom rows to its output points, as shown in 14. The labels tends to be well separated in output space as well.



(b) REAL NVP inference strategy on the ring Gaussian mixture problem. The points are colored according to their original position in the input space.

Figure 15: RAD and REAL NVP inference processes on the ring Gaussian mixture problem. Each column correspond to a RAD or affine coupling layer. RAD effectively uses foldings in order to bridge the multiple modes of the distribution into a single mode, primarily in the last layers of the transformation, whereas REAL NVP struggles to bring together these modes under the standard Gaussian distribution using continuous bijections.



(a) RAD folding strategy on the spiral problem. The top rows correspond to each RAD layer's input points, and the bottom rows to its output points, as shown in 14.



(b) REAL NVP inference strategy on the spiral problem. The points are colored according to their original position in the input space.

Figure 16: RAD and REAL NVP inference processes on the spiral problem. Each column correspond to a RAD or affine coupling layer. Instead of unrolling the manifold as REAL NVP tries to, RAD uses a more successful strategy of decomposing the spiral into three different lines that it later bridges. While the function is surjective, RAD does not necessarily mix together points of different colors (original regions) together but can keep them separate: this is a more concrete illustration of the effect of $p_{K|Z}$ shown in Figure 4.

- Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.
- James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, and Skye Wanderman-Milne. Jax: composable transformations of python+numpy programs. *URL http://github.com/google/jax*, 2018.
- Scott Shaobing Chen and Ramesh A Gopinath. Gaussianization. In *Advances in neural information processing systems*, pages 423–429, 2001.
- Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pages 6572–6583, 2018.
- Rob Cornish, Anthony Caterini, George Deligiannidis, and Arnaud Doucet. Relaxing bijectivity constraints with continuously indexed normalising flows. In *Proceedings of Machine Learning and Systems 2020*, pages 7936–7946. 2020.
- Aaron Courville, James Bergstra, and Yoshua Bengio. A spike and slab restricted boltzmann machine. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 233–241, 2011.
- Chris Cundy and Stefano Ermon. Flexible approximate inference via stratified normalizing flows. In *Uncertainty in artificial intelligence*, 2020.
- George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, 1989.
- Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. In *International Conference on Learning Representations: Workshop Track*, 2015.
- Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *International Conference on Learning Representations*, 2017.
- Hadi Mohaghegh Dolatabadi, Sarah M. Erfani, and Christopher Leckie. Invertible generative modeling using linear rational splines. In *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020*, volume 108 of *Proceedings of Machine Learning Research*, pages 4236–4246. PMLR, 2020.
- Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In *Advances in Neural Information Processing Systems*, pages 7511–7522, 2019.
- Hamid Eghbal-zadeh, Werner Zellinger, and Gerhard Widmer. Mixture density generative adversarial networks. *Neural Information Processing Systems: Bayesian Deep Learning Workshop*, 2018.
- Luca Falorsi, Pim de Haan, Tim R. Davidson, and Patrick Forré. Reparameterizing distributions on lie groups. In *Proceedings of the twenty-second international conference on artificial intelligence and statistics*, 2019.

- Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.
- Will Grathwohl, Dami Choi, Yuhuai Wu, Geoffrey Roeder, and David Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *International Conference on Learning Representations*, 2018.
- Will Grathwohl, Ricky TQ Chen, Jesse Betterncourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. 2019.
- Aditya Grover, Ramki Gummadi, Miguel Lázaro-Gredilla, Dale Schuurmans, and Stefano Ermon. Variational rejection sampling. In *International Conference on Artificial Intelligence and Statistics, AISTATS 2018*, volume 84 of *Proceedings of Machine Learning Research*, pages 823–832. PMLR, 2018.
- Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.
- Emiel Hoogeboom, Jorn Peters, Rianne van den Berg, and Max Welling. Integer discrete flows and lossless compression. In *Advances in Neural Information Processing Systems*, pages 12134–12144, 2019.
- Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.
- John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90–95, 2007.
- Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017.
- Matthew Johnson, David K Duvenaud, Alex Wiltschko, Ryan P Adams, and Sandeep R Datta. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in neural information processing systems*, pages 2946–2954, 2016.
- Eric Jones, Travis Oliphant, Pearu Peterson, et al. Scipy: Open source scientific tools for python. 2001.
- Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.
- Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.
- Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10236–10245, 2018.

- Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.
- Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017.
- Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. In *International Conference on Machine Learning*, 2014.
- Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pages 2924–2932, 2014.
- Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. Neural importance sampling. *ACM Transactions on Graphics (TOG)*, 38(5):1–19, 2019.
- Radford M Neal and Geoffrey E Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998.
- Didrik Nielsen, Priyank Jaini, Emiel Hoogeboom, Ole Winther, and Max Welling. Survae flows: Surjections to bridge the gap between vaes and flows. *arXiv preprint arXiv:2007.02731*, 2020.
- Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.
- Travis E Oliphant. Python for scientific computing. *Computing in Science & Engineering*, 9(3):10–20, 2007.
- Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta numerica*, 8(1):143–195, 1999.
- Tapani Raiko, Mathias Berglund, Guillaume Alain, and Laurent Dinh. Techniques for learning binary stochastic feedforward neural networks. In *International Conference on Learning Representations*, 2015.
- Tom Rainforth, Yuan Zhou, Xiaoyu Lu, Yee Whye Teh, Frank Wood, Hongseok Yang, and Jan-Willem van de Meent. Inference trees: Adaptive inference with exploration. *arXiv preprint arXiv:1806.09550*, 2018.
- Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, 2014.
- Eitan Richardson and Yair Weiss. On gans and gmms. In *Advances in Neural Information Processing Systems*, pages 5852–5863, 2018.
- Oren Rippel and Ryan Prescott Adams. High-dimensional probability estimation with deep density models. *arXiv preprint arXiv:1302.5125*, 2013.

- Jason Tyler Rolfe. Discrete variational autoencoders. In *International Conference on Learning Representations*, 2017.
- Reuven Y Rubinstein and Dirk P Kroese. *Simulation and the Monte Carlo method*, volume 10. John Wiley & Sons, 2016.
- EG Tabak and Cristina V Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013.
- Yichuan Tang, Ruslan Salakhutdinov, and Geoffrey Hinton. Deep mixtures of factor analysers. In *International Conference on Machine Learning*, 2012.
- Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.
- Dustin Tran, Keyon Vafa, Kumar Agrawal, Laurent Dinh, and Ben Poole. Discrete flows: Invertible generative models of discrete data. In *Advances in Neural Information Processing Systems*, pages 14719–14728, 2019.
- George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. In *Advances in Neural Information Processing Systems*, pages 2627–2636, 2017.
- Rianne van den Berg, Alexey A Gritsenko, Mostafa Dehghani, Casper Kaae Sønderby, and Tim Salimans. Idf++: Analyzing and improving integer discrete flows for lossless compression. *arXiv preprint arXiv:2006.12459*, 2020.
- Aäron van den Oord and Joni Dambre. Locally-connected transformations for deep gmmms. In *International Conference on Machine Learning (ICML): Deep Learning Workshop*, pages 1–8, 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*, pages 3518–3526, 2014.
- Aaron van den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315, 2017.
- Guido Van Rossum and Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.
- Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in science & engineering*, 13(2):22–30, 2011.
- Serena Yeung, Anitha Kannan, Yann Dauphin, and Li Fei-Fei. Tackling over-pruning in variational autoencoders. *arXiv preprint arXiv:1706.03643*, 2017.

Appendix A. Inference processes

We plot the remaining inference processes of RAD and REAL NVP on the remaining problems not plotted previously: *grid Gaussian mixture* (Figure 18), *two circles* (Figure 19), *two moons* (Figure 20), and *many moons* (Figure 21). We also compare the final results of the Gaussianization processes on both models on the different toy problems in Figure 17.

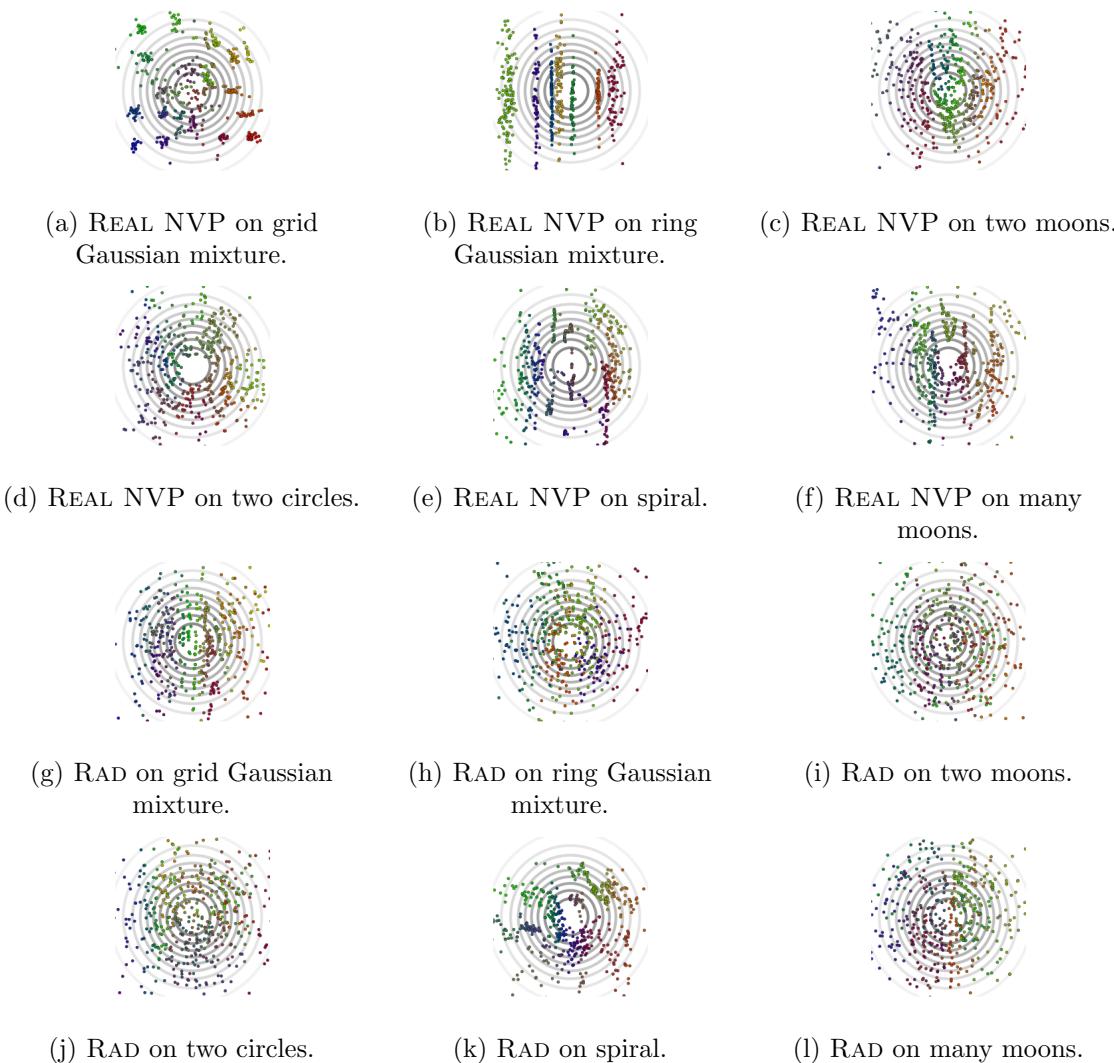
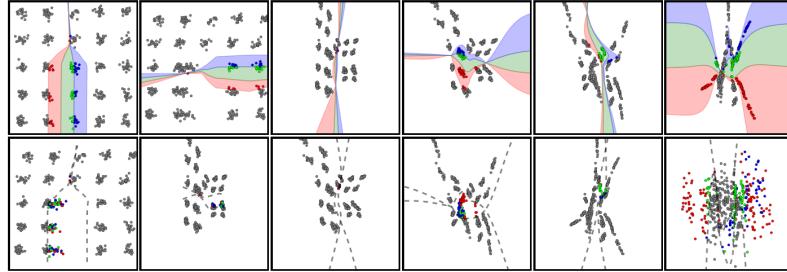
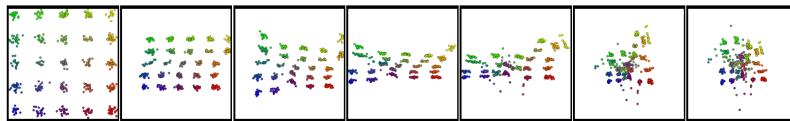


Figure 17: Comparison of the Gaussianization from the trained REAL NVP (top row) (a-f) and RAD (bottom row) (g-l). REAL NVP fails in a low capacity setting by leaving unpopulated areas where the standard Gaussian attributes probability mass. Here, these spaces as often ones separating clusters, showing the failure in modeling the data as one manifold.

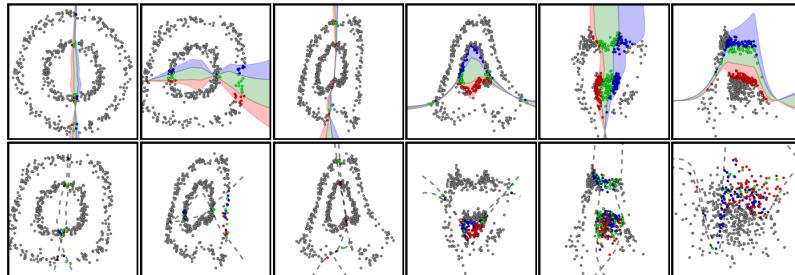


(a) RAD folding strategy on the grid Gaussian mixture problem. The top rows correspond to a RAD layer input points, and the bottom rows to its output points, as shown in 14.

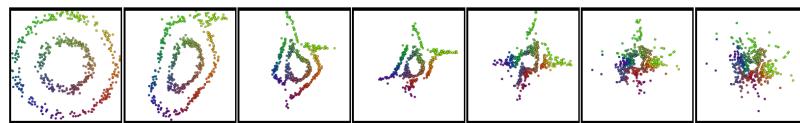


(b) REAL NVP inference strategy on the grid Gaussian mixture problem. The points are colored according to their original position in the input space.

Figure 18: RAD and REAL NVP inference process on the grid Gaussian mixture problem. Each column correspond to a RAD or affine coupling layer.

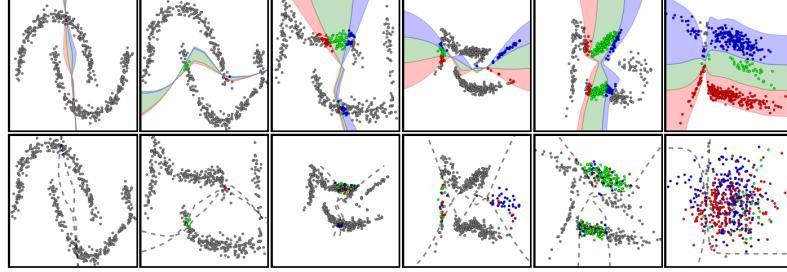


(a) RAD folding strategy on the two circles problem. The top rows correspond to a RAD layer input points, and the bottom rows to its output points, as shown in 14.

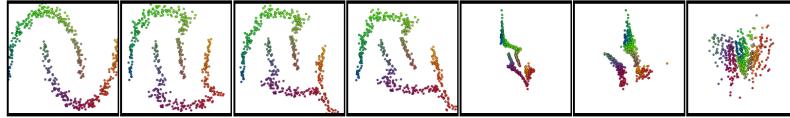


(b) REAL NVP inference strategy on the two circles problem. The points are colored according to their original position in the input space.

Figure 19: RAD and REAL NVP inference process on the two circles problem. Each column correspond to a RAD or affine coupling layer.

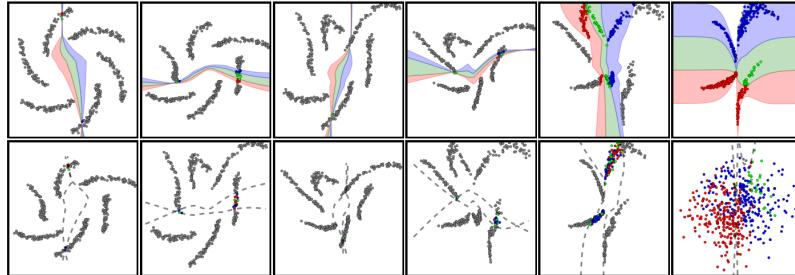


(a) RAD folding strategy on the two moons problem. The top rows correspond to a RAD layer input points, and the bottom rows to its output points, as shown in 14.

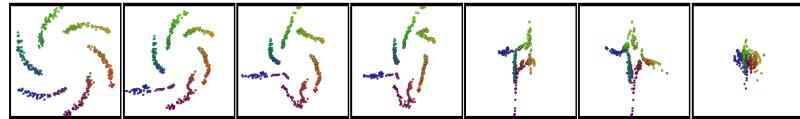


(b) REAL NVP inference strategy on the two moons problem. The points are colored according to their original position in the input space.

Figure 20: RAD and REAL NVP inference process on the two moons problem. Each column correspond to a RAD or affine coupling layer.



(a) RAD folding strategy on the many moons problem. The top rows correspond to a RAD layer input points, and the bottom rows to its output points, as shown in 14.



(b) REAL NVP inference strategy on the many moons problem. The points are colored according to their original position in the input space.

Figure 21: RAD and REAL NVP inference process on the many moons problem. Each column correspond to a RAD or affine coupling layer.