Neural Autoregressive Flows

Chin-Wei Huang [1 2 *] **David Krueger** [1 2 *] **Alexandre Lacoste** [2] **Aaron Courville** [1 3]

Abstract

Normalizing flows and autoregressive models have been successfully combined to produce state-of-the-art results in density estimation, via Masked Autoregressive Flows (MAF) (Papamakarios et al., 2017), and to accelerate state-of-the-art WaveNet-based speech synthesis to 20x faster than real-time (Oord et al., 2017), via Inverse Autoregressive Flows (IAF) (Kingma et al., 2016). We unify and generalize these approaches, replacing the (conditionally) affine univariate transformations of MAF/IAF with a more general class of invertible univariate transformations expressed as monotonic neural networks. We demonstrate that the proposed **neural autoregressive flows (NAF)** are universal approximators for continuous probability distributions, and their greater expressivity allows them to better capture multimodal target distributions. Experimentally, NAF yields state-of-the-art performance on a suite of density estimation tasks and outperforms IAF in variational autoencoders trained on binarized MNIST. [1]

1. Introduction

Invertible transformations with a tractable Jacobian, also known as **normalizing flows**, are useful tools in many machine learning problems, for example: (1) In the context of **deep generative models**, training necessitates evaluating data samples under the model's inverse transformation (Dinh et al., 2017). Tractable density is an appealing property for these models, since it allows the objective of interest to be directly optimized; whereas other mainstream methods rely on alternative losses, in the case of intractable density models (Kingma & Welling, 2014; Rezende et al., 2014), or implicit losses, in the case of adversarial models (Goodfellow et al., 2014). (2) In the context of **variational inference** (Rezende & Mohamed, 2015), they can be used to improve the variational approximation to the posterior by parameterizing more complex distributions. This is important since a poor variational approximation to the posterior can fail to reflect the right amount of *uncertainty*, and/or be biased (Turner & Sahani, 2011), resulting in inaccurate and unreliable predictions. We are thus interested in improving techniques for normalizing flows.

Recent work by Kingma et al. (2016) reinterprets autoregressive models as invertible transformations suitable for constructing normalizing flows. The inverse transformation process, unlike sampling from the autoregressive model, is not sequential and thus can be accelerated via parallel computation. This allows multiple layers of transformations to be stacked, increasing expressiveness for better variational inference (Kingma et al., 2016) or better density estimation for generative models (Papamakarios et al., 2017). Stacking also makes it possible to improve on the sequential conditional factorization assumed by autoregressive models such as PixelRNN or PixelCNN (Oord et al., 2016), and thus define a more flexible joint probability.

We note that the normalizing flow introduced by Kingma et al. (2016) only applies an affine transformation of each scalar random variable. Although this transformation is conditioned on preceding variables, the resulting flow can still be susceptible to bad local minima, and thus failure to capture the multimodal shape of a target density; see Figure 1 and 2.

1.1. Contributions of this work

We propose replacing the conditional affine transformation of Kingma et al. (2016) with a more rich family of transformations, and note the requirements for doing so. We determine that very general transformations, for instance parametrized by deep neural networks, are possible. We then propose and evaluate several specific monotonic neural network architectures which are more suited for learning multimodal distributions. Concretely, our method amounts to using an autoregressive model to output the weights of multiple independent transformer networks, each of which operates on a single random variable, replacing the affine

[*]Equal contribution [1]MILA, University of Montreal [2]Element AI [3]CIFAR fellow. Correspondence to: Chin-Wei Huang <chin-wei.huang@umontreal.ca>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

[1]Implementation can be found at https://github.com/CW-Huang/NAF/

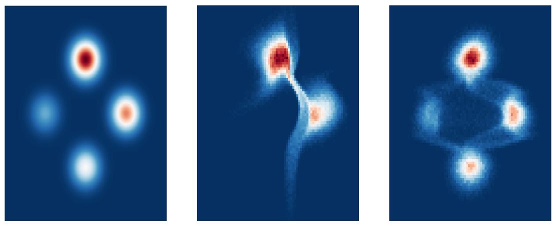


Figure 1. Energy function fitting using IAF.
Left: true distribution. Center: IAF-affine. Right: IAF-DSF.

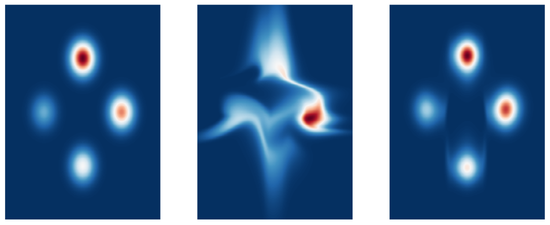


Figure 2. Density estimation using MAF.
Left: true distribution. Center: MAF-affine. Right: MAF-DSF.

transformations of previous works.

Empirically, we show that our method works better than the state-of-the-art affine autoregressive flows of Kingma et al. (2016) and Papamakarios et al. (2017), both as a sample generator which captures multimodal target densities with higher fidelity, and as a density model which more accurately evaluates the likelihood of data samples drawn from an unknown distribution.

We also demonstrate that our method is a universal approximator on proper distributions in real space, which guarantees the expressiveness of the chosen parameterization and supports our empirical findings.

2. Background

A (finite) **normalizing flow (NF)**, or **flow**, is an invertible function $f_\theta : \mathcal{X} \to \mathcal{Y}$ used to express a transformation between random variables [2]. Since f is invertible, the change of variables formula can be used to translate between densities $p_Y(\mathbf{y})$ and $p_X(\mathbf{x})$:

$$p_Y(\mathbf{y}) = \left| \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \right|^{-1} p_X(\mathbf{x}) \tag{1}$$

The determinant of f's Jacobian appears on the right hand side to account for the way in which f can (locally) expand or contract regions of X, thereby lowering or raising the resulting density in those regions' images in Y. Since the composition of invertible functions is itself invertible, complex NFs are often formed via function composition (or "stacking") of simpler NFs.

[2] We use $\mathbf{x}$ and $\mathbf{y}$ to denote inputs and outputs of a function, *not* the inputs and targets of a supervised learning problem.

Normalizing flows are most commonly trained to produce an output distribution $p_Y(\mathbf{y})$ which matches a target distribution (or, more generally, energy function) $p_{\text{target}}(\mathbf{y})$ as measured by the KL-divergence $KL(p_Y(\mathbf{y})||p_{\text{target}}(\mathbf{y}))$. When X or Y is distributed by some simple distribution, such as uniform or standard normal, we call it an unstructured noise; and we call it a structured noise when the distribution is complex and correlated. Two common settings are maximum likelihood and variational inference. Note that these two settings are typically viewed as optimizing different directions of the KL-divergence, whereas we provide a unified view in terms of different input and target distributions. A detailed derivation is presented in the appendix (See Section A).

For maximum likelihood applications (Dinh et al., 2017; Papamakarios et al., 2017), $p_{\text{target}}(\mathbf{y})$ is typically a simple prior over latent variable $\mathbf{y}$, and f attempts to disentangle the complex empirical distribution of the data, $p_X(\mathbf{x})$ into a simple latent representation $p_Y(\mathbf{y})$ matching the prior *(structured to unstructured)* [3].

In a typical application of variational inference (Rezende & Mohamed, 2015; Kingma et al., 2016), $p_{\text{target}}(\mathbf{y})$ is a complex posterior over latent variables $\mathbf{y}$, and f transforms a simple input distribution (for instance a standard normal distribution) over $\mathbf{x}$ into a complex approximate posterior $p_Y(\mathbf{y})$ *(unstructured to structured)*. In either case, since p_X does not depend on θ, the gradients of the KL-divergence are typically estimated by Monte Carlo:

$$\begin{aligned} &\nabla_\theta \mathcal{D}_{KL}(p_Y(\mathbf{y})||p_{\text{target}}(\mathbf{y})) \\ &\qquad = \nabla_\theta \int_{\mathcal{Y}} p_Y(\mathbf{y}) \log \frac{p_Y(\mathbf{y})}{p_{\text{target}}(\mathbf{y})} d\,\mathbf{y} \\ &\qquad = \int_{\mathcal{X}} p_X(\mathbf{x}) \nabla_\theta \log \frac{p_Y(\mathbf{y})}{p_{\text{target}}(\mathbf{y})} d\,\mathbf{x} \end{aligned} \tag{2}$$

Applying the change of variables formula from Equation 1 to the right hand side of Equation 2 yields:

$$\mathbb{E}_{\substack{\mathbf{x}\sim p_X(\mathbf{x}) \\ \mathbf{y}=f_\theta(\mathbf{x})}} \left[\nabla_\theta \log \left| \frac{\partial f_\theta(\mathbf{x})}{\partial \mathbf{x}} \right|^{-1} p_X(\mathbf{x}) - \nabla_\theta \log p_{\text{target}}(\mathbf{y}) \right] \tag{3}$$

[3] It may also be possible to form a generative model from such a flow, by passing samples from the prior $p_{\text{target}}(\mathbf{y})$ through f^{-1}, although the cost of doing so may vary. For example, RealNVP (Dinh et al., 2017) was devised as a generative model, and its inverse computation is as cheap as its forward computation, whereas MAF (Papamakarios et al., 2017) is designed for density estimation and is much more expensive to sample from. For the NAF architectures we employ, we do not have an analytic expression for f^{-1}, but it is possible to approximate it numerically.

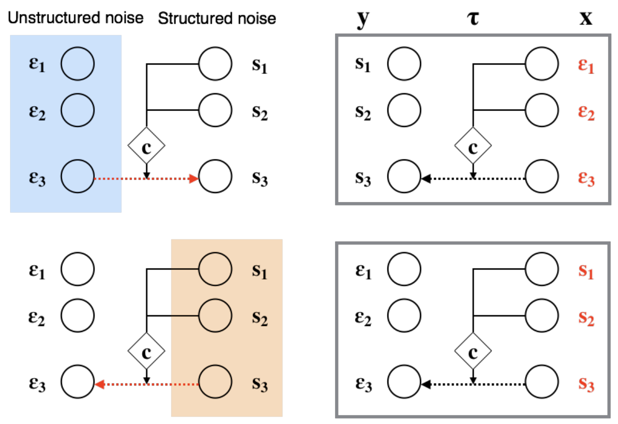


Figure 3. Difference between autoregressive and inverse autoregressive transformations (left), and difference between IAF and MAF (right). **Upper left**: sample generation of an autoregressive model. Unstructured noise is transformed into structured noise. **Lower left**: inverse autoregressive transformation of structured data. Structured variables are transformed into unstructured variables. **Upper right**: IAF-style sampling. **Lower right**: MAF-style evaluation of structured data. ϵ represents unstructured noise and s represents structured noise.

Thus for efficient training, the following operations must be tractable and cheap:

1. Sampling $\mathbf{x} \sim p_X(\mathbf{x})$
2. Computing $\mathbf{y} = f(\mathbf{x})$
3. Computing the gradient of the log-likelihood of $\mathbf{y} = f(\mathbf{x});\ \mathbf{x} \sim p_X(\mathbf{x})$ under both $p_Y(\mathbf{y})$ and $p_{\text{target}}(\mathbf{y})$
4. Computing the gradient of the log-determinant of the Jacobian of f

Research on constructing NFs, such as our work, focuses on finding ways to parametrize flows which meet the above requirements while being maximally flexible in terms of the transformations which they can represent. Note that some of the terms of of Equation 3 may be constant with respect to θ [4] and thus trivial to differentiate, such as $p_X(\mathbf{x})$ in the maximum likelihood setting.

Affine autoregressive flows (AAFs) [5], such as inverse autoregressive flows (IAF) (Kingma et al., 2016), are one particularly successful pre-existing approach. Affine autoregressive flows yield a triangular Jacobian matrix, so that the log-determinant can be computed in linear time, as the sum of the diagonal entries on log scale. In AAFs, the components of $\mathbf{x}$ and $\mathbf{y}$ are given an order (which may be chosen arbitrarily), and y_t is computed as a function of $x_{1:t}$. Specifically, this function can be decomposed via an autoregressive **conditioner**, c, and an invertible **transformer**, τ, as [6]:

$$y_t \doteq f(x_{1:t}) = \tau(c(x_{1:t-1}), x_t) \tag{4}$$

It is possible to efficiently compute the output of c for all t in a single forward pass using a model such as **MADE** (Germain et al., 2015), as pointed out by Kingma et al. (2016).

In previous work, τ is taken to be an affine transformation with parameters $\mu \in \mathbb{R}, \sigma > 0$ output from c. For instance Dinh et al. (2017) use:

$$\tau(\mu, \sigma, x_t) = \mu + \sigma x_t \tag{5}$$

with σ produced by an exponential nonlinearity. Kingma et al. (2016) use:

$$\tau(\mu, \sigma, x_t) = \sigma x_t + (1 - \sigma)\mu \tag{6}$$

with σ produced by a sigmoid nonlinearity. Such transformers are trivially invertible, but their relative simplicity also means that the expressivity of f comes entirely from the complexity of c and from stacking multiple AAFs (potentially using different orderings of the variables) [7]. However, the only requirements on τ are:

1. The transformer τ must be invertible as a function of x_t.
2. $\frac{dy_t}{dx_t}$ must be cheap to compute.

This raises the possibility of using a more powerful transformer in order to increase the expressivity of the flow.

3. Neural Autoregressive Flows

We propose replacing the affine transformer used in previous works with a neural network, yielding a more rich family of distributions with only a minor increase in computation and memory requirements. Specifically,

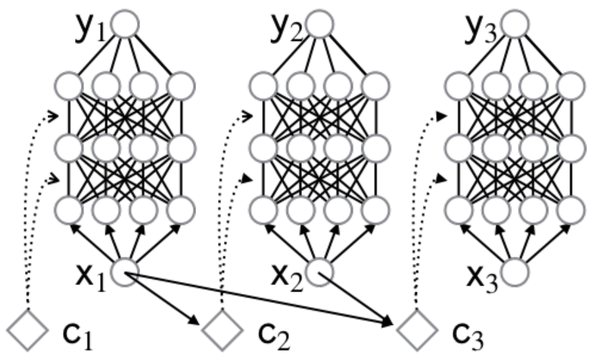
$$\tau(c(x_{1:t-1}), x_t) = \text{DNN}(x_t;\ \phi = c(x_{1:t-1})) \tag{7}$$

[4] There might be some other parameters other than θ that are learnable, such as parameters of p_X and p_{target} in the variational inference and maximum likelihood settings, respectively.

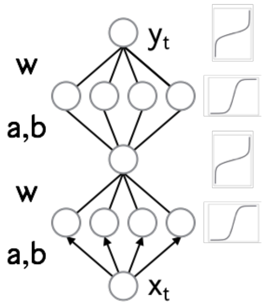
[5] Our terminology differs from previous works, and hence holds the potential for confusion, but we believe it is apt. Under our unifying perspective, NAF, IAF, AF, and MAF all make use of the same principle, which is an invertible transformer conditioned on the outputs of an autoregressive (and emphatically *not* an *inverse* autoregressive) conditioner.

[6] Dinh et al. (2014) use m and g^{-1} to denote c and τ, and refer to them as the "coupling function" and "coupling law", respectively.

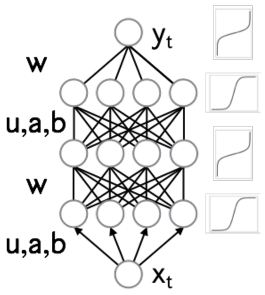
[7] Permuting the order of variables is itself a normalizing flow that does not expand or contract space and can be inverted by another permutation.



(a) Neural autoregressive flows (NAF)



(b) DSF



(c) DDSF

Figure 4. **Top:** In neural autoregressive flows, the transformation of the current input variable is performed by an MLP whose parameters are output from an autoregressive conditioner model, $c_t \doteq c(x_{1:t-1})$, which incorporates information from previous input variables. **Bottom:** The architectures we use in this work: deep sigmoidal flows (DSF) and deep dense sigmoidal flows (DDSF). See section 3.1 for details.

is a deep neural network which takes the scalar x_t as input and produces y_t as output, and its weights and biases are given by the outputs of $c(x_{1:t-1})$[8] (see Figure 4(a)). We refer to these values ϕ as **pseudo-parameters**, in order to distinguish them from the statistical parameters of the model.

We now state the condition for NAF to be strictly monotonic, and thus invertible (as per requirement 1):

Proposition 1. *Using strictly positive weights and strictly monotonic activation functions for τ_c is sufficient for the entire network to be strictly monotonic.*

Meanwhile, $\frac{dy_t}{dx_t}$ and gradients wrt the pseudo-parameters [9] can all be computed efficiently via backpropagation (as per requirement 2).

[8] We'll sometimes write τ_c for $\tau(c(x_{1:t-1}), \cdot)$.

[9] Gradients for pseudo-parameters are backpropagated through the conditioner, c, in order to train its parameters.

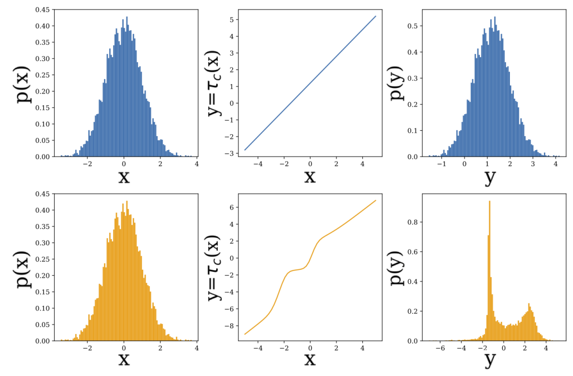


Figure 5. Illustration of the effects of traditional IAF (top), and our proposed NAF (bottom). Areas where the slope of the transformer τ_c is greater/less than 1, are compressed/expanded (respectively) in the output distribution. Inflection points in $\tau_c(x_t)$ (middle) can transform a unimodal $p(x_t)$ (left) into a multimodal $p(y_t)$ (right); NAF allows for such inflection points, whereas IAF does not.

Whereas affine transformers require information about multimodality in y_t to flow through $x_{1:t-1}$, our **neural autoregressive flows (NAFs)** are able to induce multimodality more naturally, via inflection points in τ_c, as shown in Figure 5. Intuitively, τ_c can be viewed as analogous to a cumulative distribution function (CDF), so that its derivative corresponds to a PDF, where its inflection points yield local maxima or minima.

3.1. Transformer Architectures

In this work, we use two specific architectures for τ_c, which we refer to as **deep sigmoidal flows (DSF)** and **deep dense sigmoidal flows (DDSF)** (see Figure 4(b), 4(c) for an illustration). We find that small neural network transformers of 1 or 2 hidden layers with 8 or 16 sigmoid units perform well across our experiments, although there are other possibilities worth exploring (see Section 3.3). Sigmoids contain inflection points, and so can easily induce inflection points in τ_c, and thus multimodality in $p(y_t)$. We begin by describing the DSF transformation, which is already sufficiently expressive to form a universal approximator for probability distributions, as we prove in section 4.

The DSF transformation resembles an MLP with a single hidden layer of sigmoid units. Naive use of sigmoid activation functions would restrict the range of τ_c, however, and result in a model that assigns 0 density to sufficiently large or small y_t, which is problematic when y_t can take on arbitrary real values. We address this issue by applying the inverse sigmoid (or "logit") function at the output layer. To ensure that the output's preactivation is in the domain of the logit (that is, $(0, 1)$), we combine the output of the sigmoid units via an attention-like (Bahdanau et al., 2015) softmax-weighted sums:

$$y_t = \sigma^{-1}(\underbrace{w^T}_{1\times d} \cdot \sigma(\underbrace{a}_{d\times 1} \cdot \underbrace{x_t}_{1\times 1} + \underbrace{b}_{d\times 1}))) \tag{8}$$

where $0 < w_{i,j} < 1$, $\sum_i w_{i,j} = 1$, $a_{s,t} > 0$, $b \in \mathbb{R}^d$ and d denotes the number of hidden units [10].

Since all of the sigmoid activations are bounded between 0 and 1, the final preactivation (which is their convex combination) is as well. The complete DSF transformation can be seen as mapping the original random variable to a different space through an activation function, where doing affine/linear operations is non-linear with respect to the variable in the original space, and then mapping it back to the original space through the inverse activation.

However, we realize the composition of multiple DSF layers resembles an MLP with bottleneck as shown by the bottom left of Figure 4. A more general alternative is the DDSF transformation, which takes the form of a fully connected MLP:

$$h^{(l+1)} = \\ \sigma^{-1}(\underbrace{w^{(l+1)}}_{d_{l+1}\times d_{l+1}} \cdot \sigma(\underbrace{a^{(l+1)}}_{d_{l+1}} \odot \underbrace{u^{(l+1)}}_{d_{l+1}\times d_l} \cdot \underbrace{h^{(l)}}_{d_l} + \underbrace{b^{(1+1)}}_{d_{l+1}})) \tag{9}$$

for $1 \leq l \leq L$ where $h_0 = x$ and $y = h_L$; $d_0 = d_L = 1$. We also require $\sum_j w_{ij} = 1$, $\sum_j u_{kj} = 1$ for all i, k, and all parameters except b to be positive.

We use either DSF (Equation 8) or DDSF (Equation 9) to define the transformer function τ in Equation 4. To compute the log-determinant of Jacobian in a numerically stable way, we need to apply log-sum-exp to the chain rule

$$\nabla_x y = \left[\nabla_{h^{(L-1)}} h^{(L)}\right] \left[\nabla_{h^{(L-2)}} h^{(L-1)}\right], \cdots, \left[\nabla_{h^{(0)}} h^{(1)}\right] \tag{10}$$

We elaborate more on the numerical stability in parameterization and computation of logarithmic operations in the supplementary materials.

3.2. Efficient Parametrization of Larger Transformers

Multi-layer NAFs, such as DDSF, require c to output $\mathcal{O}(d^2)$ pseudo-parameters, where d is the number of hidden units in each layer of τ. As this is impractical for large d, we propose parametrizing τ with $\mathcal{O}(d^2)$ statistical parameters, but only $\mathcal{O}(d)$ pseudo-parameters which modulate the computation on a per-unit basis, using a technique such as conditional batch-normalization (CBN) (Dumoulin et al., 2017). Such an approach also makes it possible to use minibatch-style matrix-matrix products for the forward and backwards propagation through the graph of τ_c. In particular, we use a technique similar to *conditional weight normalization (CWN)* (Krueger et al., 2017) in our experiments with DDSF; see appendix for details.

[10] Constraints on the variables are enforced via activation functions; w and a are outputs of a $\mathrm{softmax}$, and $\mathrm{softplus}$ or $\exp$, respectively.

3.3. Possibilities for Alternative Architectures

While the DSF and DDSF architectures performed well in our experiments, there are many alternatives to be explored. One possibility is using other (strictly) monotonic activation functions in τ_c, such as leaky ReLUs (LReLU) (Xu et al., 2015) or ELUs (Clevert et al., 2016). Leaky ReLUs in particular are bijections on $\mathbb{R}$ and so would not require the softmax-weighted summation and activation function inversion tricks discussed in the previous section.

Finally, we emphasize that in general, τ need not be expressed as a neural architecture; it only needs to satisfy the requirements of invertibility and differentiability given at the end of section 2.

4. NAFs are Universal Density Approximators

In this section, we prove that NAFs (specifically DSF) can be used to approximate any probability distribution over real vectors arbitrarily well, given that τ_c has enough hidden units output by generic neural networks with autoregressive conditioning. Ours is the first such result we are aware of for finite normalizing flows.

Our result builds on the work of Huang et al. (2017), who demonstrate the general universal representational capability of inverse autoregressive transformations parameterized by an autoregressive neural network (that transform uniform random variables into any random variables in reals). However, we note that their proposition is weaker than we require, as there are no constraints on the parameterization of the transformer τ, whereas we've constrained τ to have strictly positive weights and monotonic activation functions, to ensure it is invertible throughout training.

The idea of proving the universal approximation theorem for DSF (1) in the IAF direction (which transforms unstructured random variables into structured random variables) resembles the concept of the **inverse transform sampling**: we first draw a sample from a simple distribution, such as uniform distribution, and then pass the sample though DSF. If DSF converges to any inverse conditional CDF, the resulting random variable then converges in distribution to any target random variable as long as the latter has positive continuous probability density everywhere in the reals. (2) For the MAF direction, DSF serves as a solution to the non-linear independent component analysis problem (Hyvärinen & Pajunen, 1999), which disentangles structured random variables into uniformly and independently distributed random variables. (3) Combining the two, we further show that DSF can transform any structured noise variable into a random variable with any desired distribution.

We define the following notation for the pre-logit of the DSF transformation (compare equation 8):

$$\mathcal{S}(x_t, \mathcal{C}(x_{1:t-1})) = \sum_{j=1}^{n} w_j(x_{1:t-1}) \cdot \sigma\left(\frac{x_t - b_j(x_{1:t-1})}{\tau_j(x_{1:t-1})}\right) \tag{11}$$

where $\mathcal{C} = (w_j, b_j, \tau_j)_{j=1}^n$ are functions of $x_{1:1-t}$ parameterized by neural networks. Let b_j be in (r_0, r_1); τ_j be bounded and positive; $\sum_{j=1}^n w_j = 1$ and $w_j > 0$. See Appendix F and G for the proof.

Proposition 2. (DSF universally transforms uniform random variables into any desired random variables) *Let Y be a random vector in $\mathbb{R}^m$ and assume Y has a strictly positive and continuous probability density distribution. Let $X \sim$ Unif$((0,1)^m)$. Then there exists a sequence of functions $(G_n)_{n\geq 1}$ parameterized by autoregressive neural networks in the following form*

$$G(\mathbf{x})_t = \sigma^{-1}\left(\mathcal{S}\left(x_t; \mathcal{C}_t(x_{1:t-1})\right)\right) \tag{12}$$

where $\mathcal{C}_t = (a_{tj}, b_{tj}, \tau_{tj})_{j=1}^n$ are functions of $x_{1:t-1}$, such that $Y_n \doteq G_n(X)$ converges in distribution to Y.

Proposition 3. (DSF universally transforms any random variables into uniformly distributed random variables) *Let X be a random vector in an open set $\mathcal{U} \subset \mathbb{R}^m$. Assume X has a positive and continuous probability density distribution. Let $Y \sim$ Unif$((0,1)^m)$. Then there exists a sequence of functions $(H_n)_{n\geq 1}$ parameterized by autoregressive neural networks in the following form*

$$H(\mathbf{x})_t = \mathcal{S}\left(x_t; \mathcal{C}_t(x_{1:t-1})\right) \tag{13}$$

where $\mathcal{C}_t = (a_{tj}, b_{tj}, \tau_{tj})_{j=1}^n$ are functions of $x_{1:t-1}$, such that $Y_n \doteq H_n(X)$ converges in distribution to Y.

Theorem 1. (DSF universally transforms any random variables into any desired random variables) *Let X be a random vector in an open set $\mathcal{U} \subset \mathbb{R}^m$. Let Y be a random vector in $\mathbb{R}^m$. Assume both X and Y have a positive and continuous probability density distribution. Then there exists a sequence of functions $(K_n)_{n\geq 1}$ parameterized by autoregressive neural networks in the following form*

$$K(\mathbf{x})_t = \sigma^{-1}\left(\mathcal{S}\left(x_t; \mathcal{C}_t(x_{1:t-1})\right)\right) \tag{14}$$

where $\mathcal{C}_t = (a_{tj}, b_{tj}, \tau_{tj})_{j=1}^n$ are functions of $x_{1:t-1}$, such that $Y_n \doteq K_n(X)$ converges in distribution to Y.

5. Related work

Neural autoregressive flows are a generalization of the affine autoregressive flows introduced by Kingma et al. (2016) as **inverse autoregressive flows (IAF)** and further developed by Chen et al. (2017) and Papamakarios et al. (2017) as **autoregressive flows (AF)** and **masked autoregressive flows (MAF)**, respectively; for details on their relationship to our work see Sections 2 and 3. While Dinh et al. (2014) draw a particular connection between their **NICE** model and the **Neural Autoregressive Density Estimator (NADE)** (Larochelle & Murray, 2011), (Kingma et al., 2016) were the first to highlight the general approach of using autoregressive models to construct normalizing flows. Chen et al. (2017) and then Papamakarios et al. (2017) subsequently noticed that this same approach could be used efficiently in reverse when the key operation is evaluating, as opposed to sampling from, the flow's learned output density. Our method increases the expressivity of these previous approaches by using a neural net to output pseudo-parameters of another network, thus falling into the hypernetwork framework (Ha et al., 2017; Bertinetto et al., 2016; Jia et al., 2016).

There has been a growing interest in normalizing flows (NFs) in the deep learning community, driven by successful applications and structural advantages they have over alternatives. Rippel & Adams (2013), Rezende & Mohamed (2015) and Dinh et al. (2014) first introduced normalizing flows to the deep learning community as density models, variational posteriors and generative models, respectively. In contrast to traditional variational posteriors, NFs can represent a richer family of distributions without requiring approximations (beyond Monte Carlo estimation of the KL-divergence). The NF-based **RealNVP**-style generative models (Dinh et al., 2017; 2014) also have qualitative advantages over alternative approaches. Unlike **generative adversarial networks (GANs)** (Goodfellow et al., 2014) and **varational autoencoders (VAEs)** (Kingma & Welling, 2014; Rezende et al., 2014), computing likelihood is cheap. Unlike autoregressive generative models, such as **pixelCNNs** (Oord et al., 2016), sampling is also cheap. Unfortunately, in practice RealNVP-style models are not currently competitive with autoregressive models in terms of likelihood, perhaps due to the more restricted nature of the transformations they employ.

Several promising recent works expand the capabilities of NFs for generative modeling and density estimation, however. Perhaps the most exciting example is Oord et al. (2017), who propose the **probability density distillation** technique to train an IAF (Kingma et al., 2016) based on the autoregressive **WaveNet** (van den Oord et al., 2016) as a generative model using another pretrained WaveNet model to express the target density, thus overcoming the slow sequential sampling procedure required by the original WaveNet (and characteristic of autoregressive models in general), and reaching super-real-time speeds suitable for production. The previously mentioned MAF technique (Pa-

Table 1. Using DSF to improve variational inference. We report the number of affine IAF with our implementation. We note that the negative log likelihood reported by Kingma et al. (2016) is 78.88. The average and standard deviation are carried out with 5 trials of experiments with different random seeds.

Model	-ELBO	$\log p(x)$
VAE	85.00 ± 0.03	81.66 ± 0.05
IAF-affine	82.25 ± 0.05	80.05 ± 0.04
IAF-DSF	81.92 ± 0.04	79.86 ± 0.01

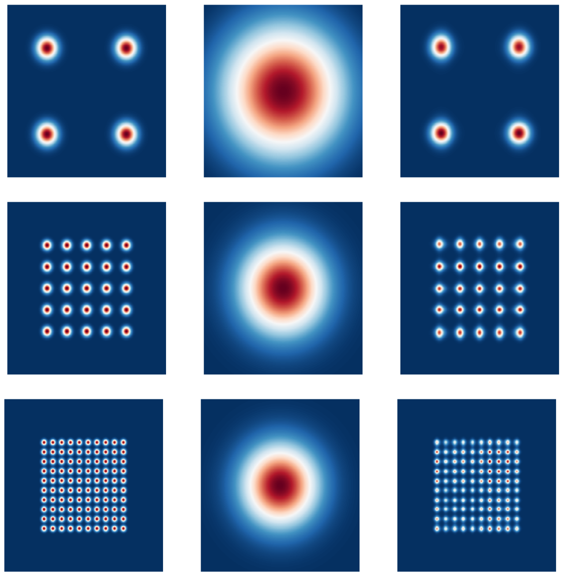


Figure 6. Fitting grid of Gaussian distributions using maximum likelihood. Left: true distribution. Center: affine autoregressive flow (AAF). Right: neural autoregressive flow (NAF)

pamakarios et al., 2017) further demonstrates the potential of NFs to improve on state-of-the-art autoregressive density estimation models; such highly performant MAF models could also be "distilled" for rapid sampling using the same procedure as in Oord et al. (2017).

Other recent works also find novel applications of NFs, demonstrating their broad utility. Loaiza-Ganem et al. (2017) use NFs to solve maximum entropy problems, rather than match a target distribution. Louizos & Welling (2017) and Krueger et al. (2017) apply NFs to express approximate posteriors over parameters of neural networks. Song et al. (2017) use NFs as a proposal distribution in a novel Metropolis-Hastings MCMC algorithm.

Finally, there are also several works which develop new techniques for constructing NFs that are orthogonal to ours (Tomczak & Welling, 2017; 2016; Gemici et al., 2016; Duvenaud et al., 2016; Berg et al., 2018).

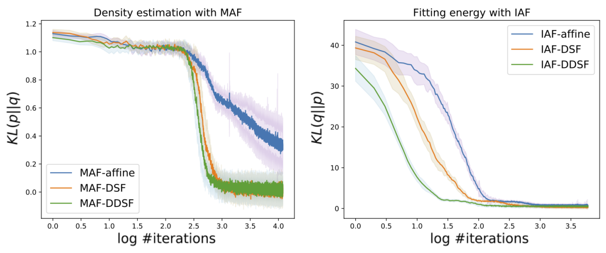


Figure 7. Learning curve of MAF-style and IAF-style training. q denotes our trained model, and p denotes the target.

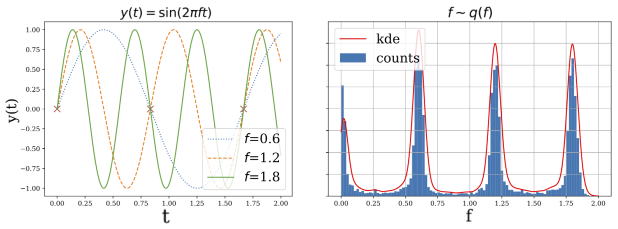


Figure 8. The DSF model effectively captures the true posterior distribution over the frequency of a sine wave. Left: The three observations (marked with red x's) are compatible with sine waves of frequency $f \in 0.0, 0.6, 1.2, 1.8$. Right: a histogram of samples from the DSF approximate posterior ("counts") and a Kernel Density Estimate of the distribution it represents (KDE).

6. Experiments

Our experiments evaluate NAFs on the classic applications of variational inference and density estimation, where we outperform IAF and MAF baselines. We first demonstrate the qualitative advantage NAFs have over AAFs in energy function fitting and density estimation (Section 6.1). We then demonstrate the capability of NAFs to capture a multimodal Bayesian posterior in a limited data setting (Section 6.2). For larger-scale experiments, we show that using NAF instead of IAF to approximate the posterior distribution of latent variables in a variational autoencoder (Kingma & Welling, 2014; Rezende et al., 2014) yields better likelihood results on binarized MNIST (Larochelle & Murray, 2011) (Section 6.3). Finally, we report our experimental results on density estimation of a suite of UCI datasets (Section 6.4).

6.1. Toy energy fitting and density estimation

6.1.1. Expressiveness

First, we demonstrate that, in the case of marginally independent distributions, affine transformation can fail to fit the true distribution. We consider a mixture of Gaussian density estimation task. We define the modes of the Gaussians to be laid out on a 2D meshgrid within the range [-5,5], and consider 2, 5 and 10 modes on each dimension. While the affine flow only produces a single mode, the neural flow

Table 2. Test log-likelihood and error bars of 2 standard deviations on the 5 datasets (5 trials of experiments). Neural autoregressive flows (NAFs) produce state-of-the-art density estimation results on all 5 datasets. The numbers (5 or 10) in parantheses indicate the number of transformations which were stacked; for TAN (Oliva et al., 2018), we include their best results, achieved using different architectures on different datasets. We also include validation results to give future researchers a fair way of comparing their methods with ours during development.

Model	POWER	GAS	HEPMASS	MINIBOONE	BSDS300
MADE MoG	0.40 ± 0.01	8.47 ± 0.02	-15.15 ± 0.02	-12.27 ± 0.47	153.71 ± 0.28
MAF-affine (5)	0.14 ± 0.01	9.07 ± 0.02	-17.70 ± 0.02	-11.75 ± 0.44	155.69 ± 0.28
MAF-affine (10)	0.24 ± 0.01	10.08 ± 0.02	-17.73 ± 0.02	-12.24 ± 0.45	154.93 ± 0.28
MAF-affine MoG (5)	0.30 ± 0.01	9.59 ± 0.02	-17.39 ± 0.02	-11.68 ± 0.44	156.36 ± 0.28
TAN (various architectures)	0.48 ± 0.01	11.19 ± 0.02	-15.12 ± 0.02	-11.01 ± 0.48	157.03 ± 0.07
MAF-DDSF (5)	$\mathbf{0.62 \pm 0.01}$	11.91 ± 0.13	$\mathbf{-15.09 \pm 0.40}$	$\mathbf{-8.86 \pm 0.15}$	$\mathbf{157.73 \pm 0.04}$
MAF-DDSF (10)	0.60 ± 0.02	$\mathbf{11.96 \pm 0.33}$	-15.32 ± 0.23	-9.01 ± 0.01	157.43 ± 0.30
MAF-DDSF (5) valid	0.63 ± 0.01	11.91 ± 0.13	15.10 ± 0.42	-8.38 ± 0.13	172.89 ± 0.04
MAF-DDSF (10) valid	0.60 ± 0.02	11.95 ± 0.33	15.34 ± 0.24	-8.50 ± 0.03	172.58 ± 0.32

matches the target distribution quite well even up to a 10x10 grid with 100 modes (see Figure 5).

6.1.2. Convergence

We then repeat the experiment that produces Figure 1 and 2 16 times, smooth out the learning curve and present average convergence result of each model with its corresponding standard deviation. For affine flow, we stack 6 layers of transformation with reversed ordering. For DSF and DDSF we used one transformation. We set $d = 16$ for both, $L = 2$ for DDSF.

6.2. Sine Wave experiment

Here we demonstrate the ability of DSF to capture multimodal posterior distributions. To do so, we create a toy experiment where the goal is to infer the posterior over the frequency of a sine wave, given only 3 datapoints. We fix the form of the function as $y(t) = \sin(2\pi f \cdot t)$ and specify a Uniform prior over the frequency: $p(f) = U([0, 2])$. The task is to infer the posterior distribution $p(f|T, Y)$ given the dataset $(T, Y) = ((0, 5/6, 10/6), (0, 0, 0))$, as represented by the red crosses of Figure 8 (left). We assume the data likelihood given the frequency parameter to be $p(y_i|t_i, f) = \mathcal{N}(y_i; y_f(t_i), 0.125)$, where the variance $\sigma^2 = 0.125$ represents the inherent uncertainty of the data. Figure 8 (right) shows that DSF learns a good posterior in this task.

6.3. Amortized Approximate Posterior

We evaluate NAF's ability to improve variational inference, in the context of the binarized MNIST (Larochelle & Murray, 2011) benchmark using the well-known variational autoencoder (Kingma & Welling, 2014; Rezende et al., 2014) (Table 1). Here again the DSF architecture outperforms both standard IAF and the traditional independent Gaussian posterior by a statistically significant margin.

6.4. Density Estimation with Masked Autoregressive Flows

We replicate the density estimation experiments of Papamakarios et al. (2017), which compare MADE (Germain et al., 2015) and RealNVP (Dinh et al., 2017) to their proposed MAF model (using either 5 or 10 layers of MAF) on BSDS300 (Martin et al., 2001) as well as 4 UCI datasets (Lichman, 2013) processed as in Uria et al. (2013). Simply replacing the affine transformer with our DDSF architecture in their best performing architecture for each task (keeping all other settings fixed) results in substantial performance gains, and also outperforms the more recent Transformation Autoregressive Networks (TAN) Oliva et al. (2018), setting a new state-of-the-art for these tasks. Results are presented in Table 2.

7. Conclusion

In this work we introduce the neural autoregressive flow (NAF), a flexible method of tractably approximating rich families of distributions. In particular, our experiments show that NAF is able to model multimodal distributions and outperform related methods such as inverse autoregressive flow in density estimation and variational inference. Our work emphasizes the difficulty and importance of capturing multimodality, as previous methods fail even on simple toy tasks, whereas our method yields significant improvements in performance.

Acknowledgements

We would like to thank Tegan Maharaj, Ahmed Touati, Shawn Tan and Giancarlo Kerg for helpful comments and advice. We also thank George Papamakarios for providing details on density estimation task's setup.

References

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, 2015.

Berg, R. v. d., Hasenclever, L., Tomczak, J. M., and Welling, M. Sylvester normalizing flows for variational inference. *arXiv preprint arXiv:1803.05649*, 2018.

Bertinetto, L., Henriques, J. F., Valmadre, J., Torr, P., and Vedaldi, A. Learning feed-forward one-shot learners. In *Advances in Neural Information Processing Systems*, 2016.

Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., and Abbeel, P. Variational lossy autoencoder. In *International Conference on Learning Representations*, 2017.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). *International Conference on Learning Representations*, 2016.

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4), 1989.

Dinh, L., Krueger, D., and Bengio, Y. NICE: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real nvp. *International Conference on Learning Representations*, 2017.

Dumoulin, V., Shlens, J., and Kudlur, M. A learned representation for artistic style. In *International Conference on Learning Representations*, volume 2, 2017.

Duvenaud, D., Maclaurin, D., and Adams, R. Early stopping as nonparametric variational inference. In *Artificial Intelligence and Statistics*, 2016.

Gemici, M. C., Rezende, D., and Mohamed, S. Normalizing flows on riemannian manifolds. *arXiv preprint arXiv:1611.02304*, 2016.

Germain, M., Gregor, K., Murray, I., and Larochelle, H. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, 2015.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, 2014.

Ha, D., Dai, A., and Le, Q. V. Hypernetworks. *International Conference on Learning Representations*, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

Huang, C.-W., Touati, A., Dinh, L., Drozdzal, M., Havaei, M., Charlin, L., and Courville, A. Learnable explicit density for continuous latent space and variational inference. *arXiv preprint arXiv:1710.02248*, 2017.

Hyvärinen, A. and Pajunen, P. Nonlinear independent component analysis: Existence and uniqueness results. *Neural Networks*, 12(3), 1999.

Jia, X., De Brabandere, B., Tuytelaars, T., and Gool, L. V. Dynamic filter networks. In *Advances in Neural Information Processing Systems*, 2016.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2014.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, 2016.

Krueger, D., Huang, C.-W., Islam, R., Turner, R., Lacoste, A., and Courville, A. Bayesian hypernetworks. *arXiv preprint arXiv:1710.04759*, 2017.

Larochelle, H. and Murray, I. The neural autoregressive distribution estimator. In *The Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, volume 15 of *JMLR: W&CP*, 2011.

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

Loaiza-Ganem, G., Gao, Y., and Cunningham, J. P. Maximum entropy flow networks. In *International Conference on Learning Representations*, 2017.

Louizos, C. and Welling, M. Multiplicative normalizing flows for variational bayesian neural networks. In *International Conference on Machine Learning*, 2017.

Martin, D., Fowlkes, C., Tal, D., and Malik, J. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2. IEEE, 2001.

Oliva, J. B., Dubey, A., Póczos, B., Schneider, J., and Xing, E. P. Transformation autoregressive networks. *arXiv preprint arXiv:1801.09819*, 2018.

Oord, A. v. d., Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural networks. In *International Conference on Machine Learning*, 2016.

Oord, A. v. d., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., Driessche, G. v. d., Lockhart, E., Cobo, L. C., Stimberg, F., et al. Parallel wavenet: Fast high-fidelity speech synthesis. *arXiv preprint arXiv:1711.10433*, 2017.

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

Polyak, B. T. and Juditsky, A. B. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4), 1992.

Reddi, S. J., Kale, S., and Kumar, S. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. In *International Conference on Machine Learning*, 2015.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, 2014.

Rippel, O. and Adams, R. P. High-dimensional probability estimation with deep density models. *arXiv preprint arXiv:1302.5125*, 2013.

Salimans, T. and Kingma, D. P. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, 2016.

Song, J., Zhao, S., and Ermon, S. A-nice-mc: Adversarial training for mcmc. In *Advances in Neural Information Processing Systems*, 2017.

Tomczak, J. M. and Welling, M. Improving variational auto-encoders using householder flow. *arXiv preprint arXiv:1611.09630*, 2016.

Tomczak, J. M. and Welling, M. Improving variational auto-encoders using convex combination linear inverse autoregressive flow. In *Benelearn*, 2017.

Turner, R. E. and Sahani, M. Two problems with variational expectation maximisation for time-series models. *Bayesian Time series models*, 1(3.1), 2011.

Uria, B., Murray, I., and Larochelle, H. Rnade: The real-valued neural autoregressive density-estimator. In *Advances in Neural Information Processing Systems*, 2013.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. WaveNet: A Generative Model for Raw Audio. *ArXiv e-prints*, September 2016.

Xu, B., Wang, N., Chen, T., and Li, M. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.