A. Exclusive KL View of the MLE

Lets assume a change-of-variable model $p_Z(\mathbf{z})$ on the random variable $Z \in \mathbb{R}^m$, such as the one used in Dinh et al. (2017): $\mathbf{z_0} \sim p_0(\mathbf{z_0})$ and $\mathbf{z} = \psi(\mathbf{z_0})$, where ψ is an invertible function and density evaluation of $\mathbf{z_0} \in \mathbb{R}^m$ is tractable under p_0. The resulting density function can be written

$$p_Z(\mathbf{z}) = p_0(\mathbf{z_0}) \left| \frac{\partial \psi(\mathbf{z_0})}{\partial \mathbf{z_0}} \right|^{-1}$$

The maximum likelihood principle requires us to minimize the following metric:

$$\begin{aligned}
&\mathcal{D}_{\mathrm{KL}}\big(p_{data}(\mathbf{z}) || p_Z(\mathbf{z})\big) \\
&\quad = \mathbb{E}_{p_{data}}\left[\log p_{data}(\mathbf{z}) - \log p_Z(\mathbf{z})\right] \\
&\quad = \mathbb{E}_{p_{data}}\left[\log p_{data}(\mathbf{z}) - \log p_0(\mathbf{z_0}) \left| \frac{\partial \psi^{-1}(\mathbf{z})}{\partial \mathbf{z}} \right| \right] \\
&\quad = \mathbb{E}_{p_{data}}\left[\log p_{data}(\mathbf{z}) \left| \frac{\partial \psi^{-1}(\mathbf{z})}{\partial \mathbf{z}} \right|^{-1} - \log p_0(\mathbf{z_0}) \right]
\end{aligned}$$

which coincides with exclusive KL divergence; to see this, we take $X = Z$, $Y = Z_0$, $f = \psi^{-1}$, $p_X = p_{data}$, and $p_{target} = p_0$

$$\begin{aligned}
&= \mathbb{E}_{p_X}\left[\log p_X(\mathbf{x}) \left| \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \right|^{-1} - \log p_{target}(\mathbf{y}) \right] \\
&= \mathcal{D}_{\mathrm{KL}}\left(p_Y(\mathbf{y}) || p_{target}(\mathbf{y})\right)
\end{aligned}$$

This means we want to transform the empirical distribution p_{data}, or p_X, to fit the target density (the usually unstructured, base distribution p_0), as explained in Section 2.

B. Monotonicity of NAF

Here we show that using *strictly positive weights* and *strictly monotonically increasing activation functions* is sufficient to ensure strict monotonicity of NAF. For brevity, we write *monotonic*, or *monotonicity* to represent the strictly monotonically increasing behavior of a function. Also, note that a continuous function is strictly monotonically increasing exactly when its derivative is greater than 0.

Proof. (Proposition 1)

Suppose we have an MLP with $L + 1$ layers: $h_0, h_1, h_2, ..., h_L$, $x = h_0$ and $y = h_L$, where x and y are scalar, and $h_{l,j}$ denotes the j-th node of the l-th layer.For $1 \leq l \leq L$, we have

$$p_{l,j} = w_{l,j}^T h_{l-1} + b_{l,j} \tag{15}$$

$$h_{l,j} = A_l(p_{l,j}) \tag{16}$$

for some monotonic activation function A_l, positive weight vector $w_{l,j}$, and bias $b_{l,j}$. Differentiating this yields

$$\frac{\mathrm{d}h_{l,j}}{\mathrm{d}h_{l-1,k}} = \frac{\mathrm{d}A_l(p_{l,j})}{\mathrm{d}p_{l,j}} \cdot w_{l,j,k}, \tag{17}$$

which is greater than 0 since A is monotonic (and hence has positive derivative), and $w_{l,j,k}$ is positive by construction. Thus any unit of layer l is monotonic with respect to any unit of layer $l-1$, for $1 \leq l \leq L$.

Now, suppose we have J monotonic functions f_j of the input x. Then the weighted sum of these functions $\sum_{j=1}^{J} u_j f_j$ with $u_j > 0$ is also monotonic with respect to x, since

$$\frac{\mathrm{d}}{\mathrm{d}x} \sum_{j=1}^{J} u_j f_j = \sum_{j=1}^{J} u_j \frac{\mathrm{d}f_j}{\mathrm{d}x} > 0 \tag{18}$$

Finally, we use induction to show that all $h_{l,j}$ (including y) are monotonic with respect to x.

1. The base case ($l = 1$) is given by Equation 17.

2. Suppose the inductive hypothesis holds, which means $h_{l,j}$ is monotonic with respect to x for all j of layer l. Then by Equation 18, $h_{l+1,k}$ is also monotonic with respect to x for all k.

Thus by mathematical induction, monotonicity of $h_{l,j}$ holds for all l and j. □

C. Log Determinant of Jacobian

As we mention at the end of Section 3.1, to compute the log-determinant of the Jacobian as part of the objective function, we need to handle the numerical stability. We first derive the Jacobian of the DDSF (note that DSF is a special case of DDSF), and then summarize the numerically stable operations that were utilized in this work.

C.1. Jacobian of DDSF

Again defining $x = h_0$ and $y = h_L$, the Jacobian of each DDSF transformation can be written as a sequence of dot products due to the chain rule:

$$\nabla_x y = \left[\nabla_{h^{(L-1)}} h^{(L)}\right] \left[\nabla_{h^{(L-2)}} h^{(L-1)}\right], \cdots, \left[\nabla_{h^{(0)}} h^{(1)}\right] \tag{19}$$

For notational convenience, we define a few more interme-

diate variables. For each layer of DDSF, we have

$$\begin{aligned}\underbrace{C^{(l+1)}}_{d_{l+1}} &= \underbrace{a^{(l+1)}}_{d_{l+1}} \odot (\underbrace{u^{(l+1)}}_{d_{l+1}\times d_l} \cdot \underbrace{h^{(l)}}_{d_l}) + \underbrace{b^{(1+1)}}_{d_{l+1}} \\ \underbrace{D^{(l+1)}}_{d_{l+1}} &= \underbrace{w^{(l+1)}}_{d_{l+1}\times d_{l+1}} \cdot \sigma(\underbrace{C^{(l+1)}}_{d_{l+1}}) \\ \underbrace{h^{(l+1)}}_{d_{l+1}}) &= \sigma^{-1}(\underbrace{D^{(l+1)}}_{d_{l+1}})\end{aligned}$$

The gradient can be expanded as

$$\begin{aligned}\nabla_{h^{(l)}}\left(h^{(l+1)}\right) = \Bigg(& \nabla_D \left(\sigma^{-1}(D^{(l+1)})\right)_{[:,\bullet]} \odot \\ & \nabla_{\sigma(C)}\left(D^{(l+1)}\right) \odot \\ & \nabla_C \left(\sigma(C^{(l+1)})\right)_{[\bullet,:]} \odot \\ & \nabla_{(u^{(l+1)}h^{(l)})}\left(C^{(l+1)}\right)_{[\bullet,:]}\Bigg)_{[:,:,\bullet]} \times_{-1} \\ & \nabla_{h^{(l)}}\left(u^{(l+1)}h^{(l)}\right)_{[\bullet,:,:]}\end{aligned}$$

where the bullet $\bullet$ in the subscript indicates the dimension is broadcasted, $\odot$ denotes element-wise multiplication, and $\times_{-1}$ denotes summation over the last dimension after element-wise product,

$$\begin{aligned}= \Bigg(& \left(\frac{1}{D^{(l+1)}(1-D^{(l+1)})}\right)_{[:,\bullet]} \odot \\ & \left(w^{(l+1)}\right) \odot \\ & \left(\sigma(C^{(l+1)}) \odot (1-\sigma(C^{(l+1)}))\right)_{[\bullet,:]} \odot \\ & \left(a^{(l+1)}\right)_{[\bullet,:]}\Bigg)_{[:,:,\bullet]} \times_{-1} \\ & \left(u^{(l+1)}\right)_{[\bullet,:,:]} \end{aligned} \tag{20}$$

C.2. Numerically Stable Operations

Since the Jacobian of each DDSF transformation is chain of dot products (Equation 19), with some nested multiplicative operations (Equation 20), we calculate everything in the log-scale (where multiplication is replaced with addition) to avoid unstable gradient signal.

C.2.1. Log Activation

To ensure the summing-to-one and positivity constraints of u and w, we let the autoregressive conditioner output pre-activation $u_{\text{-}}$ and $w_{\text{-}}$, and apply softmax to them. We do the same for a by having the conditioner output $a_{\text{-}}$ and apply softplus to ensure positivity. In Equation 20, we have

$$\begin{aligned}\log w &= \text{logsoftmax}(w_{\text{-}}) \\ \log u &= \text{logsoftmax}(u_{\text{-}}) \\ \log \sigma(C) &= \text{logsigmoid}(C) \\ \log 1-\sigma(C) &= \text{logsigmoid}(-C)\end{aligned}$$

where

$$\begin{aligned}\text{logsoftmax}(x) &= x - \text{logsumexp}(x) \\ \text{logsigmoid}(x) &= -\text{softplus}(-x) \\ \text{logsumexp}(x) &= \log(\sum_i \exp(x_i - x^*)) + x^* \\ \text{softplus}(x) &= \log(1+\exp(x)) + \delta\end{aligned}$$

where $x^* = \max_i\{x_i\}$ and δ is a small value such as 10^{-6}.

C.2.2. Logarithmic dot product

In both Equation 19 and 20, we encounter matrix/tensor product, which is achived by summing over one dimension after doing element-wise multiplication. Let $\tilde{M}_1 = \log M_1$ and $\tilde{M}_2 = \log M_2$ be $d_0 \times d_1$ and $d_1 \times d_2$, respectively. The logarithmic matrix dot product $\star$ can be written as:

$$\begin{aligned}&\tilde{M}_1 \star \tilde{M}_2 = \\ &\quad \text{logsumexp}_{\text{dim}=1}\left(\left(\tilde{M}_1\right)_{[:,:,\bullet]} + \left(\tilde{M}_2\right)_{[\bullet,:,:]}\right)\end{aligned}$$

where the subscript of logsumexp indicates the dimension (index starting from 0) over which the elements are to be summed up. Note that $\tilde{M}_1 \star \tilde{M}_2 = \log(M_1 \cdot M_2)$.

D. Scalability and Parameter Sharing

As discussed in Section 3.3, a multi-layer NAF such as DDSF requires the autoregressive conditioner c to output many pseudo-parameters, on the order of $\mathcal{O}(Ld^2)$, where L is the number of layers of the transformer network (τ), and d is the average number of hidden units per layer. In practice, we reduce the number of outputs (and thus the computation and memory requirements) of DDSF by instead endowing τ with some learned (*non-conditional*) statistical parameters. Specifically, we decompose $w_{\text{-}}$ and $u_{\text{-}}$ (the preactivations of τ's weights, see section C.2.1) into pseudo-parameters and statistical parameters. Take $u_{\text{-}}$ for example:

$$u^{(l+1)} = \text{softmax}_{\text{dim}=1}(v^{(l+1)} + \eta^{(l+1)}_{[\bullet,:]})$$

where $v^{(l+1)}$ is a $d_{l+1} \times d_l$ matrix of statistical parameters, and η is output by c. See figure D for a depiction.

The linear transformation before applying sigmoid resembles conditional weight normalization (CWN) (Krueger

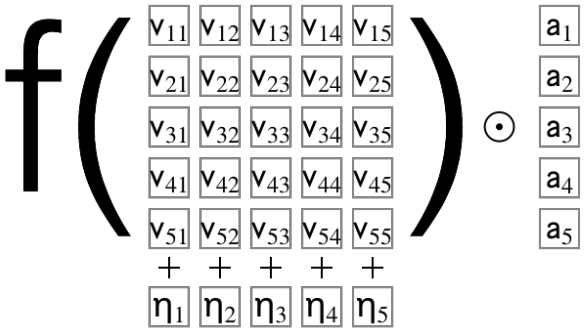

Figure 9. Factorized weight of DDSF. The v_i, j are learned parameters of the model; only the pseudo-parameters η and a are output by the conditioner. The activation function f is softmax, so adding η yields an element-wise rescaling of the inputs to this layer of the transformer by $\exp(\eta)$.

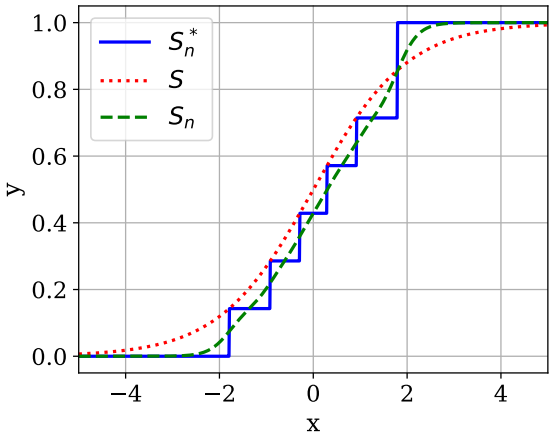

Figure 10. Visualization of how sigmoidal functions can universally approximate an monotonic function in $[0, 1]$. The red dotted curve is the target monotonic function (S), and blue solid curve is the intermediate superposition of step functions (S_n^*) with the parameters chosen in the proof. In this case, n is 6 and $|S_n^* - S| \leq \frac{1}{7}$. The green dashed curve is the resulting superposition of sigmoids (S_n), that intercepts with each step of S_n^* with the additionally chosen τ.

et al., 2017). While CWN rescales the weight vectors normalized to have unit L2 norm, here we rescale the weight vector normalized by softmax such that it sums to one and is positive. We call this *conditional normalized weight exponentiation*. This reduces the number of pseudo-parameters to $\mathcal{O}(Ld)$.

E. Identity Flow Initialization

In many cases, initializing the transformation to have a minimal effect is believed to help with training, as it can be thought of as a warm start with a simpler distribution. For instance, for variational inference, when we initialize the normalizing flow to be an identity flow, the approximate posterior is at least as good as the input distribution (usually a fully factorized Gaussian distribution) before the transformation. To this end, for DSF and DDSF, we initialize the pseudo-weights a to be close to 1, the pseudo-biases b to be close to 0.

This is achieved by initializing the conditioner (whose outputs are the pseudo-parameters) to have small weights and the appropriate output biases. Specifically, we initialize the output biases of the last layer of our MADE (Germain et al., 2015) conditioner to be zero, and add $\text{softplus}^{-1}(1) \approx 0.5413$ to the outputs of which correspond to a before applying the softplus activation function. We initialize all conditioner's weights by sampling from from $\text{Unif}(-0.001, 0.001)$. We note that there might be better ways to initialize the weights to account for the different numbers of active incoming units.

F. Lemmas: Uniform Convergence of DSF

We want to show the convergence result of Equation 11. To this end, we first show that DSF can be used to universally approximate any strictly monotonic function. The is the case where $x_{1:t-1}$ are fixed, which means $\mathcal{C}(x_{1:t-1})$ are simply constants. We demonstrate it using the following two lemmas.

Lemma 1. (Step functions universally approximate monotonic functions) *Define:*

$$S_n^*(x) = \sum_{j=1}^{n} w_j \cdot s\left(x - b_j\right)$$

where $s(z)$ is defined as a step function that is 1 when $z \geq 0$ and 0 otherwise. For any continuous, strictly monotonically increasing $S : [r_0, r_1] \to [0, 1]$ where $S(r_0) = 0$, $S(r_1) = 1$ and $r_0, r_1 \in \mathbb{R}$; and given any $\epsilon > 0$, there exists a positive integer n, real constants w_j and b_j for $j = 1, ..., n$, where $\sum_j^n w_j = 1$, $w_j > 0$ and $b_j \in [r_0, r_1]$ for all j, such that $|S_n^*(x) - S(x)| < \epsilon \quad \forall x \in [r_0, r_1]$.

Proof. (Lemma 1)

For brevity, we write $s_j(x) = s(x - b_j)$. For any $\epsilon > 0$, we choose $n = \lceil \frac{1}{\epsilon} \rceil$, and divide the range $(0, 1)$ into $n + 1$ evenly spaced intervals: $(0, y_1)$, (y_1, y_2), ..., $(y_n, 1)$. For each y_j, there is a corresponding inverse value since S is strictly monotonic, $x_j = S^{-1}(y_j)$. We want to set

$S_n^*(x_j) = y_j$ for $1 \leq j \leq n-1$ and $S_n^*(x_n) = 1$. To do so, we set the bias terms b_j to be x_j. Then we just need to solve a system of n linear equations $\sum_{j'=1}^{n} w_{j'} \cdot s_{j'}(x_j) = t_j$, where $t_j = y_j$ for $1 \leq j < n$, $t_0 = 0$ and $t_n = 1$. We can express this system of equations in the matrix form as $\mathbf{Sw} = \mathbf{t}$, where:

$$\mathbf{S}_{j,j'} = s_{j'}(x_j) = \delta_{x_j \geq b_{j'}} = \delta_{j \geq j'} \,,$$
$$\mathbf{w}_j = w_j \,, \quad \mathbf{t}_j = t_j$$

where $\delta_{\omega \geq \eta} = 1$ whenever $\omega \geq \eta$ and $\delta_{\omega \geq \eta} = 0$ otherwise. Then we have $\mathbf{w} = \mathbf{S}^{-1}\mathbf{t}$. Note that $\mathbf{S}$ is a lower triangular matrix, and its inverse takes the form of a Jordan matrix: $(\mathbf{S}^{-1})_{i,i} = 1$ and $(\mathbf{S}^{-1})_{i+1,i} = -1$. Additionally, $t_j - t_{j-1} = \frac{1}{n+1}$ for $j = 1, ..., n-1$ and is equal to $\frac{2}{n+2}$ for $j = n$. We then have $S_n^*(x) = \mathbf{t}^T \mathbf{S}^{-T} \mathbf{s}(x)$, where $\mathbf{s}(x)_j = s_j(x)$; thus

$$\begin{aligned}
|S_n^*(x) - S(x)| = \; & |\sum_{j=1}^{n} s_j(x)(t_j - t_{j-1}) - S(x)| \\
= \; & |\frac{1}{n+1} \sum_{j=1}^{n-1} s_j(x) + \frac{2 s_n(x)}{n+1} - S(x)| \\
= \; & |\frac{C_{y_{1:n-1}}(x)}{n+1} + \frac{2\delta_{x \geq y_n}}{n+1} - S(x)| \\
\leq \; & \frac{1}{n+1} < \frac{1}{\lceil 1/\epsilon \rceil} \leq \epsilon \qquad (21)
\end{aligned}$$

where $C_v(z) = \sum_k \delta_{z \geq v_k}$ is the count of elements in a vector that z is no smaller than. □

Note that the additional constraint that $\mathbf{w}$ lies on an $n-1$ dimensional simplex is always satisfied, because

$$\sum_j w_j = \sum_j t_j - t_{j-1} = t_n - t_0 = 1$$

See Figure 10 for a visual illustration of the proof. Using this result, we now turn to the case of using sigmoid functions instead of step functions.

Lemma 2. (Superimposed sigmoids universally approximate monotonic functions) *Define:*

$$S_n(x) = \sum_{j=1}^{n} w_j \cdot \sigma\left(\frac{x - b_j}{\tau_j}\right)$$

With the same constraints and definition in Lemma 1, given any $\epsilon > 0$, there exists a positive integer n, real constants w_j, τ_j and b_j for $j = 1, ..., n$, where additionally τ_j are bounded and positive, such that $|S_n(x) - S(x)| < \epsilon \quad \forall x \in (r_0, r_1)$.

Proof. (Lemma 2)

Let $\epsilon_1 = \frac{1}{3}\epsilon$ and $\epsilon_2 = \frac{2}{3}\epsilon$. We know that for this ϵ_1, there exists an n such that $|S_n^* - S| < \epsilon_1$.

We chose the same w_j, b_j for $j = 1, ..., n$ as the ones used in the proof of Lemma 1, and let $\tau_1, ..., \tau_n$ all be the same value denoted by τ.

Take $\kappa = \min_{j \neq j'} |b_j - b_{j'}|$ and $\tau = \frac{\kappa}{\sigma^{-1}(1-\epsilon_0)}$ for some $\epsilon_0 > 0$. Take Γ to be a lower triangular matrix with values of 0.5 on the diagonal and 1 below the diagonal.

$$\begin{aligned}
& \max_{j=1,...,n} |S_n(b_j) - \Gamma_j \cdot w| \\
& = \max \left| \sum_{j'} w_{j'} \sigma\left(\frac{b_j - b_{j'}}{\tau}\right) - \sum_{j'} w_{j'} \Gamma_{jj'} \right| \\
& = \max \left| \sum_{j'} w_{j'} \left(\sigma\left(\frac{b_j - b_{j'}}{\tau}\right) - \Gamma_{jj'} \right) \right| \\
& < \max \sum_{j'} w_{j'} \epsilon_0 = \epsilon_0
\end{aligned}$$

The inequality is due to

$$\sigma\left(\frac{b_j - b_{j'}}{\gamma}\right) = \sigma\left(\frac{b_j - b_{j'}}{\min_{k \neq k'} b_k - b_{k'}} \sigma^{-1}(1-\epsilon_0)\right)$$
$$\begin{cases} = 0.5 & \text{if } j = j' \\ \geq 1 - \epsilon_0 & \text{if } j > j' \\ \leq \epsilon_0 & \text{if } j < j' \end{cases}$$

Since the product $\Gamma \cdot w$ represents the half step points of S_n^* at $x = b_j$'s, the result above entails $|S_n(x) - S_n^*(x)| < \epsilon_2 = 2\epsilon_1$ for all x. To see this, we choose $\epsilon_0 = \frac{1}{2(n+1)}$. Then S_n intercepts with all segments of S_n^* except for the ends. We choose $\epsilon_2 = 2\epsilon_1$ since the last step of S_n^* is of size $\frac{2}{n+1}$, and thus the bound also holds true in the vicinity where S_n intercepts with the last step of S_n^*.

Hence,

$$\begin{aligned}
& |S_n(x) - S(x)| \\
& \quad \leq |S_n(x) - S_n^*(x)| + |S_n^*(x) - S(x)| < \epsilon_1 + \epsilon_2 = \epsilon
\end{aligned}$$

□

Now we show that (the pre-logit) DSF (Equation 11) can universally approximate monotonic functions. We do this by showing that the well-known universal function approximation properties of neural networks (Cybenko, 1989) allow us to produce parameters which are sufficiently close to those required by Lemma 2.

Lemma 3. *Let $x_{1:m} \in [r_0, r_1]^m$ where $r_0, r_1 \in \mathbb{R}$. Given any $\epsilon > 0$ and any multivariate continuously differentiable*

function [11] $S(x_{1:m})_t = S_t(x_t, x_{1:t-1})$ *for* $t \in [1, m]$ *that is strictly monotonic with respect to the first argument when the second argument is fixed, where the boundary values are* $S_t(r_0, x_{1:t-1}) = 0$ *and* $S_t(r_1, x_{1:t-1}) = 1$ *for all* $x_{1:t-1}$ *and* t*, then there exists a multivariate function* $\mathcal{S}$ *such that* $\| \mathcal{S}(x_{1:m}) - S(x_{1:m}) \|_\infty < \epsilon$ *for all* $x_{1:m}$*, of the following form:*

$$\begin{aligned}\mathcal{S}(x_{1:m})_t &= \mathcal{S}_t(x_t, \mathcal{C}_t(x_{1:t-1})) \\ &= \sum_{j=1}^{n} w_{tj}(x_{1:t-1}) \cdot \sigma\left(\frac{x_t - b_{tj}(x_{1:t-1})}{\tau_{tj}(x_{1:t-1})}\right)\end{aligned}$$

where $t \in [1, m]$*, and* $\mathcal{C}_t = (w_{tj}, b_{tj}, \tau_{tj})_{j=1}^n$ *are functions of* $x_{1:1-t}$ *parameterized by neural networks, with* τ_{tj} *bounded and positive,* $b_{tj} \in [r_0, r_1]$*,* $\sum_{j=1}^n w_{tj} = 1$*, and* $w_{tj} > 0$

Proof. (Lemma 3)

First we deal with the univariate case (for any t) and drop the subscript t of the functions. We write $\mathcal{S}_n$ and $\mathcal{C}_k$ to denote the sequences of univariate functions. We want to show that for any $\epsilon > 0$, there exist (1) a sequence of functions $\mathcal{S}_n(x_t, \mathcal{C}_k(x_{1:t-1}))$ in the given form, and (2) large enough N and K such that when $n \geq N$ and $k \geq K$, $|\mathcal{S}_n(x_t, \mathcal{C}_k(x_{1:t-1})) - S(x_t, x_{1:t-1})| \leq \epsilon$ for all $x_{1:t} \in [r_0, r_1]^t$.

The idea is first to show that we can find a sequence of parameters, $\mathcal{C}_n(x_{1:t-1})$, that yield a good approximation of the target function, $S(x_t, x_{1:t-1})$. We then show that these parameters can be arbitrarily well approximated by the outputs of a neural network, $\mathcal{C}_k(x_{1:t-1})$, which in turn yield a good approximation of S.

From Lemma 2, we know that such a sequence $\mathcal{C}_n(x_{1:t-1})$ exists, and furthermore that we can, for any ϵ, and independently of S and $x_{1:t-1}$ choose an N large enough so that:

$$|\mathcal{S}_n(x_t, \mathcal{C}_n(x_{1:t-1})) - S(x_t, x_{1:t-1})| < \frac{\epsilon}{2} \tag{22}$$

To see that we can further approximate a given $\mathcal{C}_n(x_{1:t-1})$ well by $\mathcal{C}_k(x_{1:t-1})$ [12], we apply the classic result of Cybenko (1989), which states that a multilayer perceptron can approximate any continuous function on a compact subset of $\mathbb{R}^m$. Note that specification of $\mathcal{C}_n(x_{1:t-1}) = (w_{tj}, b_{tj}, \tau_{tj})_{j=1}^n$ in Lemma 2 depends on the quantiles of $S_t(x_t, \cdot)$ as a function of $x_{1:t-1}$; since the quantiles are continuous functions of $x_{1:t-1}$, so is $\mathcal{C}_n(x_{1:t-1})$, and the theorem applies.

Now, $\mathcal{S}$ has bounded derivative wrt $\mathcal{C}$, and is thus uniformly continuous, so long as τ is greater than some positive constant, which is always the case for any fixed $\mathcal{C}_n$, and thus can be guaranteed for $\mathcal{C}_k$ as well (for large enough k). Uniform continuity allows us into translate the convergence of $\mathcal{C}_k \to \mathcal{C}_n$ to convergence of $\mathcal{S}_n(x_t, \mathcal{C}_k(x_{1:t-1})) \to \mathcal{S}_n(x_t, \mathcal{C}_n(x_{1:t-1}))$, since for any ϵ, there exists a $\delta > 0$ such that

$$\begin{aligned}&\| \mathcal{C}_k(x_{1:t-1}) - \mathcal{C}_n(x_{1:t-1}) \|_\infty < \delta \\ &\implies |\mathcal{S}_n(x_t, \mathcal{C}_k(x_{1:t-1})) - \mathcal{S}_n(x_t, \mathcal{C}_n(x_{1:t-1}))| < \frac{\epsilon}{2}\end{aligned} \tag{23}$$

Combining this with Equation 22, we have for all x_t and $x_{1:t-1}$, and for all $n \geq N$ and $k \geq K$

$$\begin{aligned}&\left|\mathcal{S}_n\left(x_t, \mathcal{C}_k(x_{1:t-1})\right) - S(x_t, x_{x_{1:t-1}})\right| \\ &\quad \leq |\mathcal{S}_n\left(x_t, \mathcal{C}_k(x_{1:t-1})\right) - \mathcal{S}_n(x_t, \mathcal{C}_n(x_{1:t-1}))| + \\ &\quad\quad \left|\mathcal{S}_n(x_t, \mathcal{C}_n(x_{1:t-1})) - S(x_t, x_{x_{1:t-1}})\right| \\ &\quad < \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon\end{aligned}$$

Having proved the univariate case, we add back the subscript t to denote the index of the function. From the above, we know that given any $\epsilon > 0$ for each t, there exist $N(t)$ and a sequence of univariate functions $\mathcal{S}_{n,t}$ such that for all $n \geq N(t)$, $|\mathcal{S}_{n,t} - S_t| < \epsilon$ for all $x_{1:t}$. Choosing $N = \max_{t \in [1,m]} N(t)$, we have that there exists a sequence of multivariate functions $\mathcal{S}_n$ in the given form such that for all $n \geq N$, $\| \mathcal{S}_n - S \|_\infty < \epsilon$ for all $x_{1:m}$.

□

G. Proof of Universal Approximation of DSF

Lemma 4. *Let* $X \in \mathcal{X}$ *be a random variable, and* $\mathcal{X} \subseteq \mathbb{R}^m$ *and* $\mathcal{Y} \subseteq \mathbb{R}^m$*. Given any function* $J : \mathcal{X} \to \mathcal{Y}$ *and a sequences of functions* J_n *that converges pointwise to* J*, the sequence of random variables induced by the transformations* $Y_n \doteq J_n(X)$ *converges in distribution to* $Y \doteq J(X)$*.*

Proof. (Lemma 4)

Let h be any bounded, continuous function on $\mathbb{R}^m$, so that $h \circ J_n$ converges pointwise to $h \circ J$ by continuity of h. Since h is bounded, then by the **dominated convergence theorem**, $\mathbb{E}[h(Y_n)] = \mathbb{E}[h(J_n(X))]$ converges to $\mathbb{E}[h(J(X)] = \mathbb{E}[h(Y)]$. As this result holds for any bounded continuous function h, by the **Portmanteau's lemma**, we have $Y_n \xrightarrow{d} Y$. □

[11] $S(\cdot) : [r_0, r_1]^m \to [0, 1]^m$ is a multivariate-multivariable function, where $S(\cdot)_t$ is its t-th component, which is a univariate-multivariable function, written as $S_t(\cdot, \cdot) : [r_0, r_1] \times [r_0, r_1]^{t-1} \to [0, 1]$.

[12] Note that $\mathcal{C}_n$ is a chosen function (we are not assuming its parameterization; i.e. not necessarily a neural network) that we seek to approximate using $\mathcal{C}_k$, which is the output of a neural network.

Proof. (Proposition 2)

Given an arbitrary ordering, let F be the CDFs of Y, defined as $F_t(y_t, x_{1:t-1}) = \Pr(Y_t \leq y_t | x_{1:t-1})$. According to Theorem 1 of Hyvärinen & Pajunen (1999), $F(Y)$ is uniformly distributed in the cube $[0,1]^m$. F has an upper triangular Jacobian matrix, whose diagonal entries are conditional densities which are positive by assumption. Let G be a multivariate and multivariable function where G_t is the inverse of the CDF of Y_t: $G_t(F_t(y_t, x_{1:t-1}), x_{1:t-1}) = y_t$.

According to Lemma 3, there exists a sequence of functions in the given form $(\mathcal{S}_n)_{n\geq 1}$ that converge uniformly to $\sigma \circ G$. Since uniform convergence implies pointwise convergence, $G_n = \sigma^{-1} \circ S_n$ converges pointwise to G, by continuity of σ^{-1}. Since G_n converges pointwise to G and $G(X) = Y$, by Lemma 4, we have $Y_n \xrightarrow{d} Y$

□

Proof. (Proposition 3)

Given an arbitrary ordering, let H be the CDFs of X:

$$\begin{aligned} y_1 &\doteq H_1(x_1, \emptyset) = F_1(x_1, \emptyset) = \Pr(X_1 \leq x_1 | \emptyset) \\ y_t &\doteq H_t(x_t, x_{1:t-1}) \\ &= F_t\big(x_t, \{H_{t-t'}(x_{t-t'}, x_{1:t-t'-1})\}_{t'=1}^{t-1}\big) \\ &= \Pr(X_t \leq x_t | y_{1:t-1}) \quad \text{for } 2 \leq t \leq m \end{aligned}$$

Due to Hyvärinen & Pajunen (1999), $y_1, ...y_m$ are independently and uniformly distributed in $(0,1)^m$.

According to Lemma 3, there exists a sequence of functions in the given form $(\mathcal{S}_n)_{n\geq 1}$ that converge uniformly to H. Since $H_n = \mathcal{S}_n$ converges pointwise to H and $H(X) = Y$, by Lemma 4, we have $Y_n \xrightarrow{d} Y$

□

Proof. (Theorem 1)

Given an arbitrary ordering, let H be the CDFs of X defined the same way in the proof for Proposition 3, and let G be the inverse of the CDFs of Y defined the same way in the proof for Proposition 2. Due to Hyvärinen & Pajunen (1999), $H(X)$ is uniformly distributed in $(0,1)^m$, so $G(H(X)) = Y$. Since $H_t(x_t, x_{1:t-1})$ is monotonic wrt x_t given $x_{1:t-1}$, and $G_t(H_t, H_{1:t-1})$ is monotonic wrt H_t given $H_{1:t-1}$, G_t is also monotonic wrt x_t given $x_{1:t-1}$, as

$$\frac{\partial G_t(H_t, H_{1:t-1})}{\partial x_t} = \frac{\partial G_t(H_t, H_{1:t-1})}{\partial H_t} \frac{\partial H_t(x_t, x_{1:t-1})}{\partial x_t}$$

is always positive.

According to Lemma 3, there exists a sequence of functions in the given form $(\mathcal{S}_n)_{n\geq 1}$ that converge uniformly to $\sigma \circ G \circ H$. Since uniform convergence implies pointwise convergence, $K_n = \sigma^{-1} \circ S_n$ converges pointwise to $G \circ H$, by continuity of σ^{-1}. Since K_n converges pointwise to $G \circ H$ and $G(H(X)) = Y$, by Lemma 4, we have $Y_n \xrightarrow{d} Y$

□

H. Experimental Details

For the experiment of amortized variational inference, we implement the Variational Autoencoder (Kingma & Welling, 2014). Specifically, we follow the architecture used in Kingma et al. (2016): the encoder has three layers with $[16, 32, 32]$ feature maps. We use resnet blocks (He et al., 2016) with 3×3 convolution filters and a stride of 2 to downsize the feature maps. The convolution layers are followed by a fully connected layer of size 450 as a context for the flow layers that transform the noise sampled from a standard normal distribution of dimension 32. The decoder is symmetrical with the encoder, with the strided convolution replaced by a combination of bilinear upsampling and regular resnet convolution to double the feature map size. We used the ELUs activation function (Clevert et al., 2016) and weight normalization (Salimans & Kingma, 2016) in the encoder and decoder. In terms of optimization, Adam (Kingma & Ba, 2014) is used with learning rate fined tuned for each inference setting, and Polyak averaging (Polyak & Juditsky, 1992) was used for evaluation with $\alpha = 0.998$ which stands for the proportion of the past at each time step. We also consider a variant of Adam known as Amsgrad (Reddi et al., 2018) as a hyperparameter. For vanilla VAE, we simply apply a resnet dot product with the context vector to output the mean and the pre-softplus standard deviation, and transform each dimension of the noise vector independently. We call this linear flow. For IAF-affine and IAF-DSF, we employ MADE (Germain et al., 2015) as the conditioner $c(x_{1:t-1})$, and we apply dot product on the context vector to output a scale vector and a bias vector to conditionally rescale and shift the preactivation of each layer of the MADE. Each MADE has one hidden layer with 1920 hidden units. The IAF experiments all start with a linear flow layer followed by IAF-affine or IAF-DSF transformations. For DSF, we choose $d = 16$.

For the experiment of density estimation with MAF, we followed the implementation of Papamakarios et al. (2017). Specifically for each dataset, we experimented with both 5 and 10 flow layers, followed by one linear flow layer. The following table specifies the number of hidden layers and the number of hidden units per hidden layer for MADE:

Table 3. Architecture specification of MADE in the MAF experiment. Number of hidden layers and number of hidden units.

POWER	GAS	HEPMASS	MINIBOONE	BSDS300
2×100	2×100	2×512	1×512	2×1024