HINT: Hierarchical Invertible Neural Transport
for Density Estimation and Bayesian Inference

Jakob Kruse [* 1]   Gianluca Detommaso [* 2]   Robert Scheichl [1]   Ullrich Köthe [1]

Abstract

A large proportion of recent invertible neural architectures is based on a coupling block design. It operates by dividing incoming variables into two sub-spaces, one of which parameterizes an easily invertible (usually affine) transformation that is applied to the other. While the Jacobian of such a transformation is triangular, it is very sparse and thus may lack expressiveness. This work presents a simple remedy by noting that (affine) coupling can be repeated recursively within the resulting sub-spaces, leading to an efficiently invertible block with dense triangular Jacobian. By formulating our recursive coupling scheme via a hierarchical architecture, HINT allows sampling from a joint distribution $p(\mathbf{y}, \mathbf{x})$ and the corresponding posterior $p(\mathbf{x} \mid \mathbf{y})$ using a single invertible network. We demonstrate the power of our method for density estimation and Bayesian inference on a novel data set of 2D shapes in Fourier parameterization, which enables consistent visualization of samples for different dimensionalities.

1. Introduction

Invertible neural networks based on the normalizing flow principle have recently gained increasing attention for generative modeling tasks. Arguably the most popular group of such networks are those built on the coupling block design. Their success is due to a number of useful properties setting them apart from other generative approaches:

(a) they offer tractable likelihood computation for any sample or data point, *(b)* training via the maximum likelihood criterion is generally very stable, *(c)* their interpretable and easily accessible latent space opens up possibilities for e.g. style transfer and *(d)* the same trained model can be
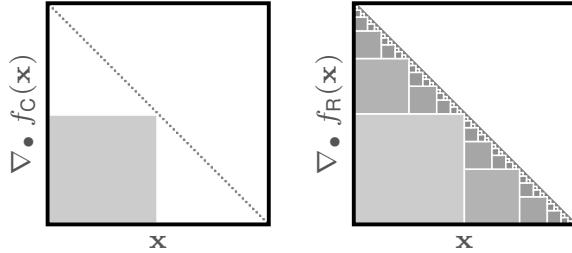

Figure 1. Sparse triangular Jacobian of a single coupling *(left)* and dense triangular Jacobian of hierarchical coupling block *(right)*. Gray marks populated areas of the matrix, the remainder is empty.

used for efficient data generation as well as efficient density estimation.

While autoregressive models can also be trained as normalizing flows and share the former two properties, they sacrifice efficient invertibility for expressive power and thus lose the latter two properties. In contrast, the expressive power of a single invertible block is a core limitation of invertible networks, leading to impractically deep models with dozen or hundreds of blocks, as is exemplified by the GLOW architecture of Kingma & Dhariwal (2018). While invertibility actually allows to backpropagate through very deep networks with minimal memory footprint (Gomez et al., 2017), the search for more expressive invertible building blocks remains an important question.

The superior performance of autoregressive approaches such as (Van den Oord et al., 2016a) stems from the fact that they model a larger range of interactions between variables, as reflected in the dense triangular Jacobian matrix of the mapping they represent. From the theory of transport maps we know that certain guarantees of universality exist for triangular maps. These do not hold for the standard coupling block design, which has a comparatively sparse Jacobian as shown in figure 1 *(left)*.

In this work we propose an extension to the coupling block design that recursively fills out the previously unused portions of the Jacobian until we obtain a dense triangular map (figure 1, *right*), or any intermediate design if we choose to stop early. The advantages of the original coupling block architecture are retained in the process.

*Equal contribution   [1]Heidelberg University, Germany   [2]University of Bath, United Kingdom. Correspondence to: Jakob Kruse <jakob.kruse@iwr.uni-heidelberg.de>.

We furthermore show that the recursive structure of this mapping can be used to split the variables of interest into two subsets $\mathbf{x}$ and $\mathbf{y}$, such that the transformation of $\mathbf{x}$ is modeled conditionally on $\mathbf{y}$. This property can be exploited for a new sampling scheme, where a single normalizing flow model efficiently generates samples from both the joint distribution $p(\mathbf{x}, \mathbf{y})$ and the conditional $p(\mathbf{x} \mid \mathbf{y})$ of the factorized variables.

Finally, we introduce a new family of data sets based on 2d Fourier curve parameterizations. In the normalizing flow literature, there is an abundance of two-dimensional toy densities that provide an easy visual check for correctness of the model's output. Beyond two dimensions however, it is challenging to visualize the distribution or even individual samples produced by a model. Image data sets solve this problem, but are quickly of high enough dimensionality that it becomes infeasible to evaluate statistical baselines like Approximate Bayesian Computation (ABC) to compare against.

A step towards visualizable data sets of intermediate size has been made in (Kruse et al., 2019), but their four-dimensional problems are still two simple to demonstrate the advantages of the recursive coupling approach described above. To fill the gap, we describe a way to generate data sets of arbitrary dimensionality such that each data point parameterizes a closed curve in 2d space which is easy to visualize. With more data dimensions, i.e. more degrees of freedom, distributions of more detailed curves can be represented.

To summarize, the contributions of this paper to the field of normalizing flow research are as follows:

- A simple, efficiently invertible flow module with dense lower triangular Jacobian

- A hierarchical coupling architecture that models joint and conditional distributions simultaneously

- A novel family of data sets allowing 2d visualization for a flexible number of dimensions

The remainder of this work is consists of a literature review (section 2), some mathematical background (section 3), a description of our method (section 4) and numerical experiments (section 5), followed by some closing remarks (section 6).

2. Related Work

Normalizing flows were popularized in the context of deep learning chiefly by the work of Rezende & Mohamed (2015) and Dinh et al. (2015). At present, a large variety of different architectures exist to realize normalizing flows in practice. A comprehensive overview of many of these, as well as general background on invertible neural networks and normalizing flows, can be found in Kobyzev et al. (2019) and in the simultaneous work of Papamakarios et al. (2019).

Many existing networks fall into one of two groups, namely coupling block architectures and autoregressive models. First additive and then affine coupling blocks were introduced by Dinh et al. (2015; 2017). Kingma & Dhariwal (2018), besides demonstrating the power of flow networks as generators, went on to generalize the permutation of variables between blocks by learning the corresponding matrices. There have been a number of works that focus on improving on the limiting nature of the affine transformation at the core of most coupling block networks. E.g. Durkan et al. (2019) replace affine couplings with more expressive monotonous splines, albeit at the cost of evaluation speed.

On the other hand, there is a rich body of research into autoregressive (flow) networks for various purposes, ranging from Van den Oord et al. (2016b;a) over Kingma et al. (2016) and Papamakarios et al. (2017) to Huang et al. (2018). More recently, Jaini et al. (2019) applied second-order polynomials to improve expressive power compared to typical autoregressive models, and proved that their model is a universal density approximator. While such models are known for excellent density estimation results compared to coupling architectures (Ma et al., 2019; Liao et al., 2019), generating samples is often not a priority and can be prohibitively slow.

Outside of these two subfields, there are other approaches towards a favorable trade-off between expressive power and efficient invertibility.

Residual Flows (Behrmann et al., 2019; Chen et al., 2019) impose Lipschitz constraints on a standard residual block, which guarantees invertibility with a full Jacobian and enables approximate maximum likelihood training but requires an iterative procedure for sampling. Similarly, Song et al. (2019) use lower triangular weight matrices which can be inverted via fixed-point iteration. The normalizing flow principle is formulated continuously as a differential equation by Grathwohl et al. (2019), which allows free-form Jacobians but requires integrating an ODE for each network pass. Karami et al. (2019) introduce another method with dense Jacobian based on invertible convolutions performed in the Fourier domain.

In terms of modeling conditional densities with invertible neural networks, Ardizzone et al. (2019a) proposed an approach that divides the network output into conditioning variables and latent vector, training the flow part with an MMD (Gretton et al., 2012) objective instead of maximum likelihood. Later Ardizzone et al. (2019b) introduced a simple conditional coupling block from which a conditional normalizing flow can be constructed.

3. Background

For an input vector $\mathbf{x} \in \mathbb{R}^N$, the function defined by a standard invertible coupling block is

$$f_{\mathrm{C}}(\mathbf{x}) = \begin{bmatrix} \mathbf{x}_1 \\ C(\mathbf{x}_2 \,|\, \mathbf{x}_1) \end{bmatrix} \qquad (1)$$

with $\mathbf{x}_1 = \mathbf{x}_{0:\lfloor N/2 \rfloor}$ and $\mathbf{x}_2 = \mathbf{x}_{\lfloor N/2 \rfloor:N}$ being the first and second half of the input vector and $C(\mathbf{x}_2 \,|\, \mathbf{x}_1)$ an easily invertible transform of $\mathbf{x}_2$ conditioned on $\mathbf{x}_1$. Its inverse is then simply given as

$$f_{\mathrm{C}}^{-1}(\mathbf{x}) = \begin{bmatrix} \mathbf{x}_1 \\ C^{-1}(\mathbf{x}_2 \,|\, \mathbf{x}_1) \end{bmatrix}. \qquad (2)$$

In the case of an affine coupling block (Dinh et al., 2017), C takes the form $C(\mathbf{u} \,|\, \mathbf{v}) = \mathbf{u} \odot \exp\big(s(\mathbf{v})\big) + t(\mathbf{v})$ with s and t unconstrained feed-forward networks. The logarithm of this block's Jacobian determinant can be computed very efficiently as

$$\log \big| \det \mathbf{J}_{f_C}(\mathbf{x}) \big| = \log \left| \det \frac{\partial f_C(\mathbf{x})}{\partial \mathbf{x}} \right| = \sum s(\mathbf{x}_1). \quad (3)$$

To ensure that all entries of $\mathbf{x}$ are transformed, and interact with each other in different combinations, they construct a pipeline that alternates between coupling blocks and random orthogonal matrices $\mathbf{Q}$, where $f_{\mathbf{Q}}(\mathbf{x}) = \mathbf{Q}\mathbf{x}$ can trivially be inverted as $f_{\mathbf{Q}}^{-1}(\mathbf{x}) = \mathbf{Q}^\top \mathbf{x}$ and has Jacobian log-determinant $\log \big| \det \mathbf{J}_{f_{\mathbf{Q}}}(\mathbf{x}) \big| = 0$.

3.1. Normalizing flows and transport maps

To create a normalizing flow, this pipeline $T = f_{\mathrm{C}1} \circ f_{\mathbf{Q}1} \circ f_{\mathrm{C}2} \circ f_{\mathbf{Q}2} \circ \dots$ is trained via maximum likelihood loss

$$\mathcal{L}(\mathbf{x}) = \tfrac{1}{2} \|T(\mathbf{x})\|_2^2 - \log |\mathbf{J}_T(\mathbf{x})| \qquad (4)$$

to transport the data distribution $p_X(\mathbf{x})$ to a standard normal latent distribution $p_Z(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$.

We can draw samples from $p_X(\mathbf{x})$ via T by drawing latent samples $\mathbf{z}$ from $p_Z(\mathbf{z})$ and passing them through the inverse model: $\mathbf{x} = T^{-1}(\mathbf{z})$. Using the change-of-variables formula, we can also evaluate the density $p_X(\mathbf{x})$ of a given data point $\mathbf{x}$ as $p_X(\mathbf{x}) = p_Z(T(\mathbf{x})) \cdot |\det \mathbf{J}_T(\mathbf{x})|$.

This procedure is founded on the theory of transport maps (Villani, 2008), which are employed in exactly the same way to push a reference density (e.g. Gaussian) to a target density (e.g. data distribution, Marzouk et al. (2016)).

The objective in equation (4) is in fact the KL divergence between the data distribution $p_X(\mathbf{x})$ and the pull-back $T_{\#}^{-1}$ of the latent density $p_Z(\mathbf{z})$, with $H(p_X)$ the unknown fixed entropy of the data distribution:

$$\begin{aligned} D_{\mathrm{KL}}(p_X \,\|\, T_{\#}^{-1} p_Z) &= \int p_X(\mathbf{x}) \log \frac{p_X(\mathbf{x})}{T_{\#}^{-1} p_Z(\mathbf{x})} \, \mathrm{d}\mathbf{x} \\ &= - \int p_X(\mathbf{x}) \log\big(p_Z(T(\mathbf{x})) \mathbf{J}_T(\mathbf{x})\big) \, \mathrm{d}\mathbf{x} + H(p_X) \\ &= \mathbb{E}_{\mathbf{x} \sim p_X}[\mathcal{L}(\mathbf{x})] + \mathrm{const.}. \end{aligned}$$

Note also that each pair $f_{\mathrm{C}i} \circ f_{\mathbf{Q}i}$ in T is a composition of an orthogonal transformation and a triangular map, where the latter is better known in the field of transport maps as a *Knothe-Rosenblatt rearrangement* (Marzouk et al., 2016). This factorization can be seen as a non-linear generalization of the classic QR decomposition (Stoer & Bulirsch, 2013). Whereas the triangular part encodes the possibility to represent non-linear transformations, the orthogonal part reshuffles variables to foster dependence of each part of the input to the final output, thereby drastically increasing the representational power of the map T.

3.2. Bayesian inference with conditional flows

When a data set does not just represent a distribution of data points $\mathbf{x}$, but consists of tuples $(\mathbf{x}, \mathbf{y})$, the labels/features/observations $\mathbf{y}$ don't have a clear place in the normalizing flow framework described above. It is of course possible to just concatenate $\mathbf{x}$ and $\mathbf{y}$ into a single vector and let the transport $T([\mathbf{x}, \mathbf{y}]^\top)$ model their joint distribution. But in many cases the paired data stems from a known (probabilistic) forward process $\mathbf{x} \to \mathbf{y}$ and what we are interested in is the inverse process, in particular evaluating and sampling from the conditional density $p(\mathbf{x} \,|\, \mathbf{y})$. This task is called Bayesian inference.

Ardizzone et al. (2019b) and Winkler et al. (2019) independently introduced *conditional* coupling blocks that allow an entire normalizing flow to be conditioned on external variables. By conditioning the transport T between $p_X(\mathbf{x})$ and $p_Z(\mathbf{z})$ on the corresponding values of $\mathbf{y}$ as $\mathbf{z} = T(\mathbf{x} \,|\, \mathbf{y})$, its inverse $T^{-1}(\mathbf{z} \,|\, \mathbf{y})$ can be used to turn the latent distribution $p_Z(\mathbf{z})$ into an approximation of the posterior $p(\mathbf{x} \,|\, \mathbf{y})$. A similar idea can be found in Marzouk et al. (2016).

The maximum likelihood objective from equation (4) is readily adjusted to the conditional setting as

$$\mathcal{L}(\mathbf{x}, \mathbf{y}) = \tfrac{1}{2} \|T(\mathbf{x} \,|\, \mathbf{y})\|_2^2 - \log |\mathbf{J}_T(\mathbf{x} \,|\, \mathbf{y})|, \qquad (5)$$

where the Jacobian is w.r.t. $\mathbf{x}$, i.e. $\mathbf{J}_T(\mathbf{x} \,|\, \mathbf{y}) = \nabla_{\mathbf{x}} T(\mathbf{x} \,|\, \mathbf{y})$. This is equivalent to minimizing the KL divergence between $p_{X,Y}(\mathbf{x}, \mathbf{y})$ and the pull-back of a standard normal distribution through the inverse of T:

$$\begin{aligned} &D_{\mathrm{KL}}(p_{X,Y} \,\|\, T(\cdot \,|\, Y)_{\#}^{-1} p_Z) \\ &= \mathbb{E}_{\mathbf{x}, \mathbf{y} \sim p_{X,Y}}[\mathcal{L}(\mathbf{x}, \mathbf{y})] + H(p_{X,Y}). \end{aligned}$$
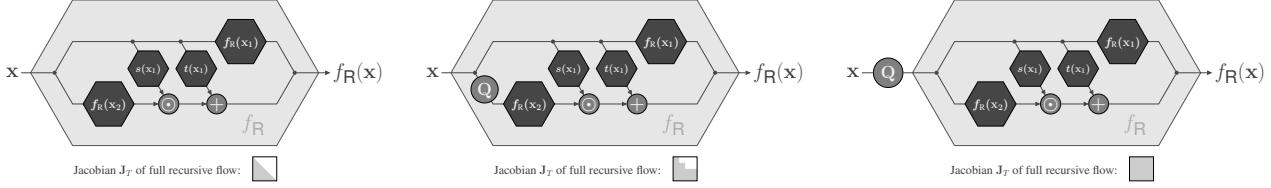
Figure 2. Recursive affine coupling blocks. In each case, the inner functions $f_R(\mathbf{x}_i)$ again take the form of the outer gray block, repeated until the vectors $\mathbf{x}_i$ can not be split any further. Each block in itself has triangular Jacobian. Placement of the orthogonal permutation matrix $\mathbf{Q}$ determines how the Jacobian $\mathbf{J}_T$ of a composition T of many blocks is populated. *Left:* No permutation means T is a strict Knothe-Rosenblatt rearrangement. *Middle:* Permutation in the lower branch only induces a recursive conditional structure where variables in the lower branches can never influence variables in the upper branches. *Right:* Permutation between coupling blocks yields a map with full Jacobian. Expressive power of the map increases from left to right while interpretability decreases.

4. Method

We will now extend the basic coupling block architecture described above in two ways.

4.1. The recursive coupling block

As visualized in figure 1 *(left)*, the Jacobian $\mathbf{J}_f$ of a simple coupling block is very sparse, i.e. many possible interactions between variables are not modelled. However the efficient determinant computation in equation (3) works for any lower triangular $\mathbf{J}_f$, and indeed Theorem 1 of Hyvärinen & Pajunen (1999) states that a single triangular transformation can, in theory, already represent arbitrary distributions.

To make use of this potential and fill the empty areas below the diagonal in $\mathbf{J}_{f_C}$, we propose a recursive coupling scheme

$$f_R(\mathbf{x}) = \begin{cases} f_C(\mathbf{x}), & \text{if } \mathbf{x} \in \mathbb{R}^{N \leq 3} \\ \begin{bmatrix} f_R(\mathbf{x}_1) \\ C\big(f_R(\mathbf{x}_2) \,\big|\, \mathbf{x}_1\big) \end{bmatrix}, & \text{otherwise} \end{cases} \tag{6}$$

where each sub-space $\mathbf{x}_1, \mathbf{x}_2$ is again split and transformed until we end up with single dimensions (or stop early). Note that each sub-coupling has its own – e.g. affine – coupling function C with independent parameters. The inverse transform is

$$f_R^{-1}(\mathbf{x}) = \begin{cases} f_C^{-1}(\mathbf{x}), & \text{if } \mathbf{x} \in \mathbb{R}^{N \leq 3} \\ \begin{bmatrix} f_R^{-1}(\mathbf{x}_1) \\ f_R^{-1}\big(C^{-1}\big(\mathbf{x}_2 \,\big|\, f_R^{-1}(\mathbf{x}_1)\big)\big) \end{bmatrix}, & \text{otherwise} \end{cases}$$
$$\tag{7}$$

This procedure leads to the dense lower triangular Jacobian visualized in figure 1 *(right)*, the log-determinant of which is simply the sum of the log-determinants of all sub-couplings C. If N is a power of two and hence the dimensions of all splits work out, f_R becomes a full Knothe-Rosenblatt map.

Having defined the recursive coupling block, we find that the way we integrate it with the permutation matrix $\mathbf{Q}$ has a profound effect on the type of transport T we get when compositing many blocks.

If we omit the permutations entirely, the composition of lower triangular maps gives us another dense lower triangular map (figure 2, *left*) and thus, in fact, an autoregressive model.

Placing a permutation matrix at the beginning of the lower branch in each recursive split (figure 2, *middle*) interestingly produces a conditional structure: The sub-transport performed by the lower branch at each level is informed by the upper branch, but has no influence on the latter itself. Because variables never get permuted between the outermost branches, we end up with a lower "lane" that performs a transport conditioned on intermediate representations from the upper lane. The Jacobian of T in this case exhibits a block triangular structure.

Finally, if we simply apply a permutation over all variables outside of each recursive block (figure 2, *right*), we end up with a transport T with full Jacobian and, presumably, the greatest expressive power among the options considered here.

4.2. Hierarchical Invertible Neural Transport

While the recursive coupling block defined above is motivated by the search for a more expressive architecture, we have seen that its structure also lends itself well to a hierarchical treatment where splits of the variables at different levels carry different meaning.

Specifically, in a setting with paired data $(\mathbf{x}_i, \mathbf{y}_i)$, we can provide both variables as input to the flow and use the top level split to transform them separately at the next recursion level. To this end we take inspiration from figure 2 *(middle)* to design a hierarchical affine coupling block. Instead of relying on the recursive permutation structure implied there, however, we only apply a single permutation $\mathbf{Q}_{\mathbf{y}}$ and $\mathbf{Q}_{\mathbf{x}}$ to each respective lane at the beginning of the block. This preserves the hierarchical setup while letting variables within each lane interact more freely and saving on expensive tensor operations.

Figure 3 shows a hierarchical affine coupling block of this

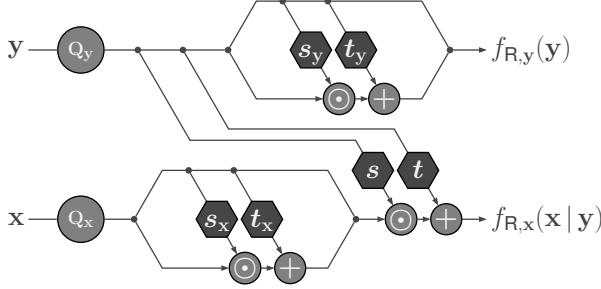Figure 3. Hierarchical affine coupling block with one level of recursion. Note how the transformation in the $\mathbf{x}$-lane is influenced by $\mathbf{y}$, but not vice-versa, imposing a hierarchy on variables.
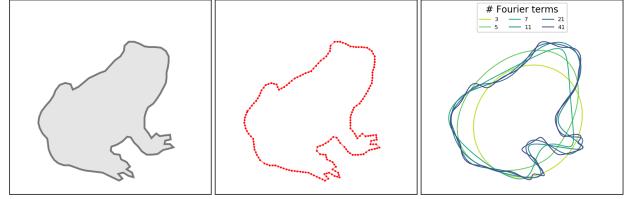

Figure 4. Left: A 2d polygon obtained from the segmentation of a natural image. *Middle:* The vertices $\mathbf{p}_n$ of the polygon, which are the basis for computing the Fourier coefficients in equation (12). *Right:* Tracing the parameterized 2d curve $\mathbf{f}(t)$ according to equation (11) for different numbers M of Fourier terms $\mathbf{a}_m$.

type, with one level of recursion for simplicity. A normalizing flow model constructed in this way performs *hierarchical invertible neural transport*, or HINT for short.

The output of a HINT model is a latent code with two components, $\mathbf{z} = [\mathbf{z}_\mathbf{y}, \mathbf{z}_\mathbf{x}]^\top = T(\mathbf{y}, \mathbf{x})$, but the training objective stays the same as in equation (4):

$$\mathcal{L}(\mathbf{y}, \mathbf{x}) = \tfrac{1}{2}\|T(\mathbf{y}, \mathbf{x})\|_2^2 - \log|\mathbf{J}_T(\mathbf{y}, \mathbf{x})| \qquad (8)$$

As with a standard normalizing flow, the joint density of input variables is the pull-back of the latent density via T:

$$p_T(\mathbf{y}, \mathbf{x}) = T_\#^{-1} p_Z(\mathbf{z}) = T_\#^{-1} \mathcal{N}(\mathbf{0}, \mathbf{I}_{|\mathbf{y}|+|\mathbf{x}|}). \qquad (9)$$

But because the $\mathbf{y}$-lane in HINT can be evaluated independently of the $\mathbf{x}$-lane, we can determine the partial latent code $\mathbf{z}_\mathbf{y}$ for a given $\mathbf{y}$ and hold it fixed, while drawing $\mathbf{z}_\mathbf{x}$ from the $\mathbf{x}$-part of the latent distribution. Doing so yields samples from the conditional density

$$p_T(\mathbf{x}\,|\,\mathbf{y}) = T_\#^{-1}\begin{bmatrix} T_\mathbf{y}(\mathbf{y}) \\ \mathcal{N}(\mathbf{0}, \mathbf{I}_{|\mathbf{x}|}) \end{bmatrix}. \qquad (10)$$

This means HINT gives access to both the joint and the conditional density of the two variables $\mathbf{x}$ and $\mathbf{y}$.

5. Experiments

All experiments were carried out on an artificial data set of 2d shapes that allows visualizing samples regardless of the chosen data dimensionality. All flow models were trained on a single NVIDIA GeForce RTX 2080 Ti.

5.1. Fourier shapes data set

A curve $\mathbf{f}(t) \in \mathbb{R}^2$ parameterized by $2M+1$ complex 2d Fourier coefficients $\mathbf{a}_m \in \mathbb{C}^2$ can be traced as

$$\mathbf{f}(t) = \sum_{m=-M}^{M} \mathbf{a}_m \cdot e^{2\pi \cdot i \cdot m \cdot t} \qquad (11)$$

with parameter t running from 0 to 1. This parameterization will always yield a closed, possibly self-intersecting curve (McGarva & Mullineux, 1993).

Vice-versa, given a sequence of N 2d points $\mathbf{p}_n \in \mathbb{R}^2$ we can calculate the Fourier coefficients of a curve approximating this sequence as

$$\mathbf{a}_m = \frac{1}{N} \sum_{n=0}^{N-1} \mathbf{p}_n \cdot e^{-2\pi \cdot i \cdot m \cdot n / N} \text{ for } m \in [-M, M] \quad (12)$$

When we increase the limit M, higher order terms are added to the parameterization in equation (11) and the shape can be approximated in greater detail. An example of this effect for a natural shape is shown in figure 4 *(right)*. Note that the actual dimensionality of the parameterization in our data set is $|\mathbf{x}| = 4 \cdot (2M + 1)$, as each complex 2d coefficient $\mathbf{a}_m$ needs four real numbers to represent it.

In this paper we perform experiments on two specific Fourier shapes data sets. The first one uses $M = 2$, i.e. $|\mathbf{x}| = 20$, and we choose the prior to be an arbitrary Gaussian mixture model with five components. Figure 5 *(left)* shows a large sample from this shape prior and four single representatives. Note that these shapes carry little geometric meaning, often self-intersect and are best understood as a way to visualize samples $\mathbf{x} \in \mathbb{R}^{20}$ in two dimensions.

The second one uses $M = 12$, i.e. $|\mathbf{x}| = 100$, to represent a distribution of simple shapes that are generated geometrically by crossing two bars of randomized length and width at a right angle. This results in a variation of X, L and T shapes, some of which are shown in figure 5 *(right)*.

In both instances, our training set has 10^6 samples and the test set has 10^5 samples.

5.2. Forward process

Our forward operator in the Bayesian inference task returns three simple features of the 2d shape parameterized by $\mathbf{x}$. We determine the largest diameter as $d_{\max}$ and the total width of the shape orthogonal to this direction as $d_{\min}$. The
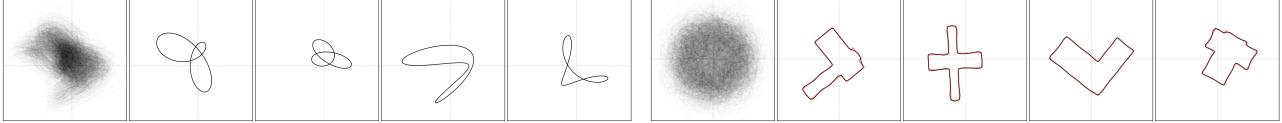
Figure 5. Left: Large batch and individual samples drawn from the prior of the Gaussian mixture data set. *Right:* The same for the geometrical shapes data set. Red lines behind the individual Fourier curves show the original shapes they approximate.
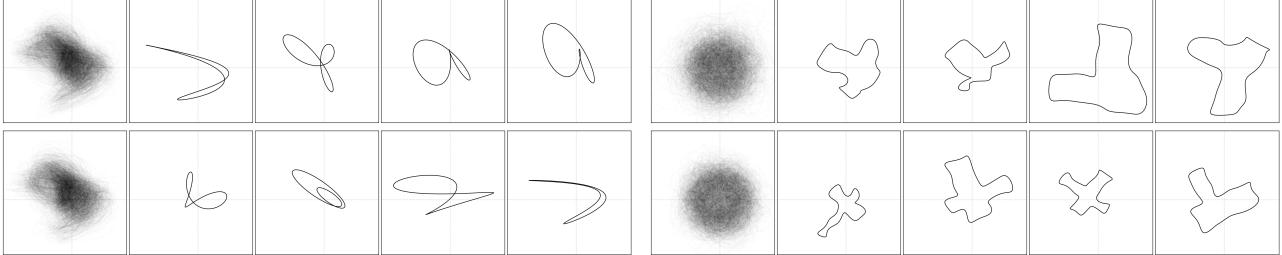

Figure 6. Samples from standard coupling block networks *(top row)* and recursive coupling block networks *(bottom row)*, trained on the two data sets presented in figure 5. Each network has four blocks and either 10^5 *(left)* or $2 \cdot 10^6$ *(right)* parameters. For the geometrical shapes it is clearly visible that recursive blocks better capture the details of the data distribution.

Table 1. Comparison of normal and recursive coupling for sampling and density estimation on Gaussian mixture prior. MMD (Gretton et al., 2012) compares the model's sample distribution to test data, bits/dim is the log-likelihood of test data under the model. Mean and standard deviation from five independent training runs.

BLOCK TYPE	#	MMD $\downarrow$	BITS/DIM $\uparrow$
NORMAL	1	0.0936 ± 0.0038	0.8380 ± 0.0582
NORMAL	2	0.0034 ± 0.0010	1.5106 ± 0.0062
RECURSIVE	1	0.0345 ± 0.0004	1.4195 ± 0.0007
RECURSIVE	2	0.0016 ± 0.0001	1.5288 ± 0.0012

angle α of the former is one feature, the *aspect ratio* $a = \frac{d_{\min}}{d_{\max}}$ another. The third feature is *circularity*, defined as $c = \frac{4\pi \cdot \text{area}}{\text{perimeter}^2}$.

For ambiguity we add some noise $\sigma \sim \mathcal{N}(\mathbf{0}, \frac{1}{20}\mathbf{I})$ to these three values to obtain our vector $\mathbf{y}$.

5.3. Density estimation

On the Gaussian mixture data set, we trained a single-block and a two-block network for density estimation, once with standard coupling blocks and once with our recursive design. Each internal sub-network s and t consist of three fully connected layers. Sub-networks further down in the recursion use progressively fewer parameters, and all four normalizing flows are scaled to have a total of 10^5 trainable parameters.

For these and all following experiments, we trained each network for 50 epochs with Adam and a learning rate decaying exponentially from 10^{-2} to 10^{-4}. With a small amount

of weight regularization and clamping gradients to $[-5, 5]$, we observed very stable training for all networks.

Qualitative samples from these models are shown in figure 6 *(left)*, with the left-most panel being an empirical estimate of the prior $p_X(\mathbf{x})$. Quantitative results can be found in table 1. We compare across several training runs per model in terms of maximum mean discrepancy (Gretton et al., 2012) and bits per dimension.

The former measures the dissimilarity of two distributions using only samples from both. Following Ardizzone et al. (2019a), we use MMD with an inverse multi-quadratic kernel and average the results from 100 batches from the data prior and from each trained model.

Bits per dimension are a representation of the log-likelihood of the test data under a model, calculated as

$$\text{bits/dim}_T(\mathbf{x}) = -\frac{\frac{1}{2}\|T(\mathbf{x})\|_2^2 - \log|\mathbf{J}_T(\mathbf{x})|}{|\mathbf{x}| \cdot \ln 2}.$$

For both metrics, we see that a single standard coupling block performs very badly, which is unsurprising since it leaves half the variables untouched. A single recursive coupling block on the other hand is reasonably successful and achieves bits/dimension almost on par with two standard coupling blocks. Out of the tested arrangements, a recursive two-block network performs the best.

We also trained networks with standard and recursive coupling blocks on the larger geometrical shape data set, where we afforded each network a total of $2 \cdot 10^6$ parameters spread over four coupling blocks. Qualitative samples from these models are shown in figure 6 *(right)*, with those from the normal coupling network in the top row and those from the

Table 2. Comparison of normal (conditional) and hierarchical coupling for Bayesian inference on Gaussian mixture prior. Columns as in table 1. Mean and standard deviation over three training runs.

BLOCK TYPE	#	MMD $\downarrow$	BITS/DIM $\uparrow$
NORMAL	1	0.3718 ± 0.0011	0.9663 ± 0.0073
NORMAL	2	0.1263 ± 0.0332	1.7072 ± 0.0276
NORMAL	4	0.0273 ± 0.0038	1.7868 ± 0.0094
NORMAL	8	0.0158 ± 0.0018	1.7939 ± 0.0070
HINT	1	0.0358 ± 0.0009	1.6465 ± 0.0004
HINT	2	0.2028 ± 0.0460	1.6695 ± 0.0012
HINT	4	0.0588 ± 0.0542	1.7553 ± 0.0162
HINT	8	0.0241 ± 0.0003	1.7379 ± 0.0129

recursive network in the bottom row.

While it is difficult to judge how well the entire distribution of shapes matches the prior (see figure 5, *(right)*), it is very clear that the network based on recursive coupling blocks produces individual samples that are more faithful to the data set.

5.4. Bayesian inference

For the Bayesian inference task on the Gaussian mixture data set, we trained a range of conditional flow models and HINT models with the following properties:

- with 1 block and $2 \cdot 10^5$ trainable parameters

- with 2 blocks and $2 \cdot 10^5$ trainable parameters

- with 4 blocks and $4 \cdot 10^5$ trainable parameters

- with 8 blocks and $4 \cdot 10^5$ trainable parameters

We train multiple instances of each model as described above and again evaluate in terms of MMD and bits per dimension.

In this case however the MMD comparison is not with samples from the prior $p_X(\mathbf{x})$, but with samples from an estimate of the posterior $p(\mathbf{x} \mid \mathbf{y})$ for 10^3 values of $\mathbf{y}$ obtained via the prior and forward process. We create these estimated ground truth posterior samples via quantitative approximate Bayesian computation as explained in Ardizzone et al. (2019a).

The bits per dimension measure also requires extra attention in this setting, since we have to compute the log-likelihood of only the $\mathbf{x}$-lane of our HINT models. Fortunately, this is easily done by excluding the contributions from the $\mathbf{y}$-lane when adding up the log-determinants of the Jacobians in each sub-coupling.

The quantitative results of these experiments are summarized in table 2, where we can see that the standard coupling

model with eight blocks achieves the best score. As in the unconditional case, however, the recursive design vastly outperforms the standard one when only a single block is compared.

Qualitative results are presented in figure 7. Each individual sample has a red bounding box that displays its true aspect ratio and angle of greatest diameter, and a green box displaying the conditioning target $\mathbf{y}$. The jagged rings around the samples are chosen to have the exact circularity measured from the sample *(red)* or required by $\mathbf{y}$ *(green)*, respectively. Visually, we find both approaches to be on equal footing.

In figure 8, we also show qualitative results from models with four and eight blocks and $4 \cdot 10^6$ parameters each trained on the geometrical shape data set. Adherence to the condition is visualized just like in figure 7.

As in the unconditional case, it is clear that the Fourier coefficients produced by the HINT models parameterize shapes that are much closer to those in the data set. Both conditional coupling block models have trouble recovering the rectangular nature of the shapes. While all four models stay true to the circularity they are conditioned on, they all struggle to place the greatest diameter of the shapes at the required angle. The HINT models appear slightly more faithful to the aspect ratio even when the rotation is off.

5.5. Proof of Concept: Automatic Posterior Transformation using HINT

In the following, we use HINT to implement the Automatic Posterior Transformation (APT) algorithm proposed by Greenberg et al. (2019), and compare the implementation to a standard coupling block INN. In short, given some observations, APT estimates the posterior of unobserved variables. The posterior is then used as a prior for the next step, and subsequently updated by newly arriving observations. This can be repeated many times. To prevent errors from accumulating over many timesteps, the density models have to be very accurate, making this an interesting application for HINT.

As a task, we use the general predator-prey model, also termed *competitive Lotka-Volterra equations*, which typically describes the interaction of d species in a biological system over time:

$$\frac{\partial}{\partial t} x_i = \beta_i x_i \left(1 - \sum_{j=1}^{d} \alpha_{ij} x_j \right) \qquad (13)$$

The undisturbed growth rate of species i is given by β_i (can be $<$ or > 0, growing or shrinking naturally), and is further affected by the other species in a positive way ($\alpha_{ij} > 0$, predator), or in a negative way ($\alpha_{ij} < 0$, prey). The solutions to this system of equations can not be expressed
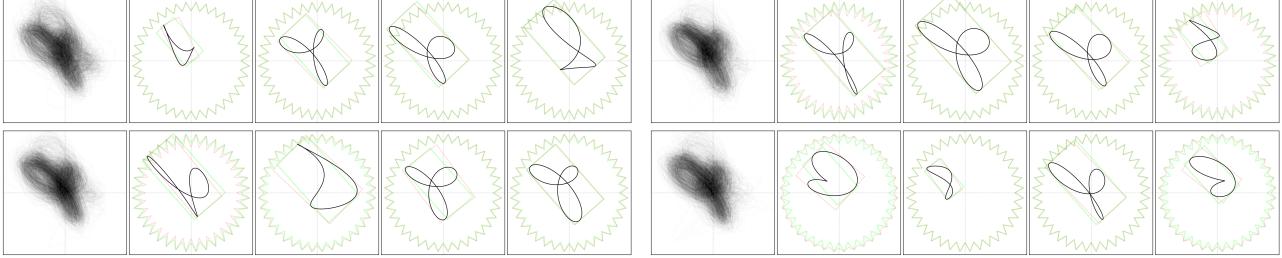
Figure 7. Samples from conditional flow *(left)* and HINT *(right)* with 4 *(top)* and 8 blocks *(bottom)* trained on Gaussian mixture data.
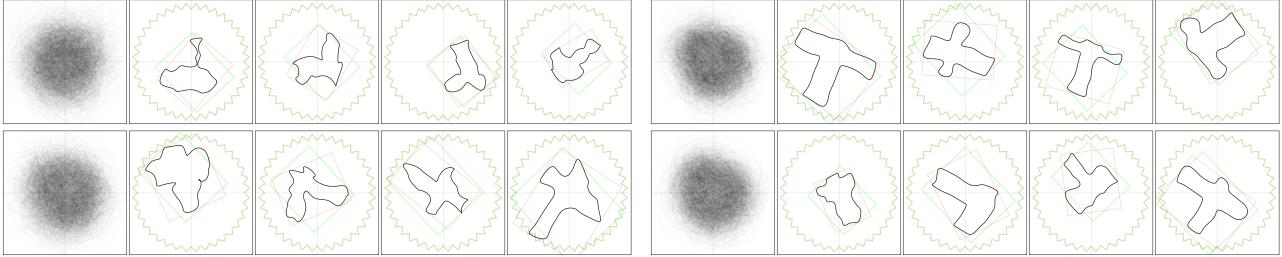

Figure 8. Samples from conditional flow *(left)* and HINT *(right)* with 4 *(top)* and 8 blocks *(bottom)* trained on geometrical shape data.

analytically, which makes their prediction a challenging task. Additionally, we make the process stochastic, by adding random noise at each time step. Therefore, at each time step, noisy measurements of x_1, x_2, x_3 are observed, with the task to predict the remaining population x_4, given the current and past observations. The exact parameters of the data model are found in appendix section C.

We compare this to an implementation using a standard coupling block INN with the training procedure proposed by Ardizzone et al. (2019a). The results for a single example time-series are shown in figure 9. We find that the standard INN does not make meaningful predictions in the short- or mid-term, only correctly predicting the final dying off of population x_4. HINT on the other hand is able to correctly model the entire sequence. Importantly, the modeled posterior distribution at each step correctly contracts over time, as more measurements are accumulated. Experimental details are found in appendix section C.

6. Conclusion

In this work, we presented HINT, a new architecture with the primary use as a normalizing flow. It improves on the traditional coupling block design in terms of expressive power by making the Jacobian densely triangular. Hereby, HINT keeps the advantages of interpretable latent space and equally efficient sampling and density estimation, which are typically not present in other models with densely triangular Jacobians, specifically autoregressive flows. To evaluate the model, we present a new data set based on Fourier decompositions of 2D shapes, which can be easily visualized while still posing a reasonable level of challenge to compare
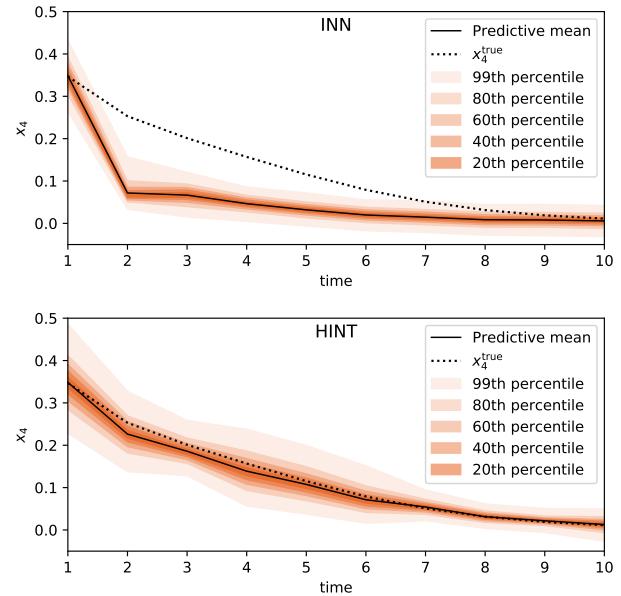

Figure 9. Contracting posterior for sequential Lotka-Volterra.

different methods. In terms of future improvements, we expect that our formulation can be made even more computationally efficient through the use of masking operations enabling more advanced parallelization.

Acknowledgments

References

Ardizzone, L., Kruse, J., Rother, C., and Köthe, U. Analyzing inverse problems with invertible neural networks. In *Intl. Conf. on Learning Representations*, 2019a.

Ardizzone, L., Lüth, C., Kruse, J., Rother, C., and Köthe, U. Guided image generation with conditional invertible neural networks. *arXiv:1907.02392*, 2019b. URL http://arxiv.org/abs/1907.02392.

Behrmann, J., Grathwohl, W., Chen, R. T. Q., Duvenaud, D., and Jacobsen, J.-H. Invertible residual networks. In *International Conference on Machine Learning*, 2019. URL http://proceedings.mlr.press/v97/behrmann19a.html.

Chen, T. Q., Behrmann, J., Duvenaud, D. K., and Jacobsen, J.-H. Residual flows for invertible generative modeling. In *Advances in Neural Information Processing Systems 32*, pp. 9913–9923, 2019.

Dinh, L., Krueger, D., and Bengio, Y. NICE: non-linear independent components estimation. In *International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1410.8516.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real NVP. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=HkpbnH9lx.

Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. Neural spline flows. In *Advances in Neural Information Processing Systems*, pp. 7509–7520, 2019.

Gomez, A. N., Ren, M., Urtasun, R., and Grosse, R. B. The reversible residual network: Backpropagation without storing activations. In *Advances in neural information processing systems*, pp. 2214–2224, 2017.

Grathwohl, W., Chen, R. T. Q., Bettencourt, J., and Duvenaud, D. Scalable reversible generative models with free-form continuous dynamics. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJxgknCcK7.

Greenberg, D. S., Nonnenmacher, M., and Macke, J. H. Automatic posterior transformation for likelihood-free inference. *arXiv:1905.07488*, 2019. URL http://arxiv.org/abs/1905.07488.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.

Huang, C.-W., Krueger, D., Lacoste, A., and Courville, A. Neural autoregressive flows. In *International Conference on Machine Learning*, pp. 2078–2087, 2018.

Hyvärinen, A. and Pajunen, P. Nonlinear independent component analysis: Existence and uniqueness results. *Neural Networks*, 12(3):429–439, 1999.

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

Karami, M., Schuurmans, D., Sohl-Dickstein, J., Dinh, L., and Duckworth, D. Invertible convolutional flow. In *Advances in Neural Information Processing Systems*, pp. 5636–5646, 2019.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. *arXiv:1807.03039*, 2018.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pp. 4743–4751, 2016.

Kobyzev, I., Prince, S., and Brubaker, M. A. Normalizing flows: An introduction and review of current methods. *arXiv:1908.09257*, 2019.

Kruse, J., Ardizzone, L., Rother, C., and Köthe, U. Benchmarking invertible architectures on inverse problems. *Workshop on Invertible Neural Networks and Normalizing Flows, International Conference on Machine Learning*, 2019.

Liao, H., He, J., and Shu, K. Generative model with dynamic linear flow. *IEEE Access*, 7:150175–150183, 2019. ISSN 2169-3536. doi: 10.1109/ACCESS.2019.2947567.

Ma, X., Kong, X., Zhang, S., and Hovy, E. Macow: Masked convolutional generative flow. *Advances in Neural Information Processing Systems 32*, pp. 5891–5900, 2019. URL http://papers.nips.cc/paper/8824-macow-masked-convolutional-generative-flow.pdf.

Marzouk, Y., Moselhy, T., Parno, M., and Spantini, A. Sampling via measure transport: An introduction. *Handbook of Uncertainty Quantification*, pp. 1–41, 2016.

McGarva, J. and Mullineux, G. Harmonic representation of closed curves. *Applied Mathematical Modelling*, 17(4):213 – 218, 1993. ISSN 0307-904X. URL http://www.sciencedirect.com/science/article/pii/0307904X9390109T.

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

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. Normalizing flows for probabilistic modeling and inference. *arXiv:1912.02762*, 2019. URL http://arxiv.org/abs/1912.02762.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pp. 1530–1538, 2015.

Song, Y., Meng, C., and Ermon, S. Mintnet: Building invertible neural networks with masked convolutions. In *Advances in Neural Information Processing Systems*, pp. 11002–11012, 2019.

Stoer, J. and Bulirsch, R. *Introduction to numerical analysis*, volume 12. Springer Science & Business Media, 2013.

Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al. Conditional image generation with PixelCNN decoders. In *Advances in neural information processing systems*, pp. 4790–4798, 2016a.

Van den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pp. 1747–1756, 2016b.

Villani, C. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

Winkler, C., Worrall, D., Hoogeboom, E., and Welling, M. Learning likelihoods with conditional normalizing flows. *arXiv preprint arXiv:1912.00042*, 2019.

– Appendix –

A. Details on Gaussian mixture data set

The prior for our smaller data set instance is a Gaussian mixture model with five components in 20 dimensions. The parameters μ and σ for this distribution can be generated with the following Python snippet:

```python
1 rng = np.random.RandomState(seed=123)
2 coeffs_shape = (2,5,2)
3 n_components = 5
4 component_weights = (.5 + rng.rand(n_components))
5 component_weights /= np.sum(component_weights)
6 mus = [.5 * rng.randn(*coeffs_shape)
7     for i in range(n_components)]
8 sigmas = [.1 + .2 * rng.rand(*coeffs_shape)
9     for i in range(n_components)]
```

B. Details on geometrical shape data set

The shapes for the larger data set instance are generated by taking the union of two oblong rectangles crossing each other at a right angle. For both rectangles, the longer side length is drawn uniformly from $[3, 5]$ and the other from $[\frac{1}{2}, 2]$. We shift both rectangles along their longer side by a uniformly random amount from $[-\frac{3}{2}, \frac{3}{2}]$.

Then we form the union and insert equally spaced points along the resulting polygon's sides such that no line segment is longer than $\frac{1}{5}$. This is necessary to obtain point sequences which are approximated more faithfully by Fourier curves.

Finally, we center the shape at the origin, rotate it by a random angle and shift it in either direction by a distance drawn from $\mathcal{N}(0, \frac{1}{2})$.

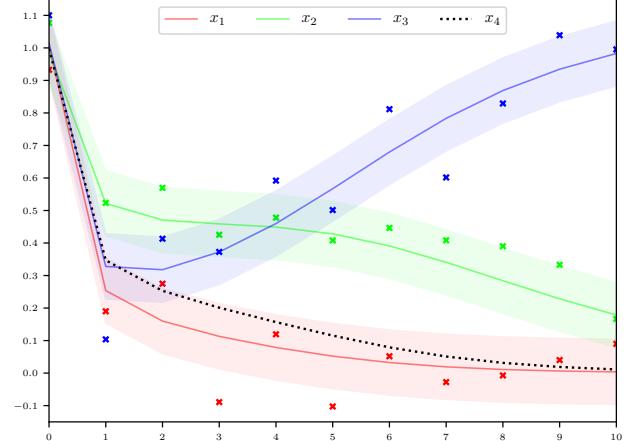Python code for creating both data set variants can be found under https://github.com/VLL-HD/HINT.

Figure 10. Population development in competitive Lotka-Volterra model. x_1, x_2 and x_3 are only acessible via noisy measurements *(small crosses)*, the colored bands show the standard deviation of this noise. The task is to predict x_4 as the sequence developes.

C. Details on extra experiment in 5.5.

Parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ for the 4d competitive Lotka-Volterra model in equation (13) were picked randomly as

$$\boldsymbol{\beta} = \begin{bmatrix} 1.14143055 \\ 0.64270729 \\ 1.42981209 \\ 0.90620443 \end{bmatrix} \text{ and }$$

$$\boldsymbol{\alpha} = \begin{bmatrix} 0.78382338 & 1.26614888 & 1.25787652 & 0.80904295 \\ 1.00470891 & 0.32719451 & 1.34501072 & 1.29758381 \\ 1.28599724 & 0.39362355 & 0.89977679 & 1.00063551 \\ 1.12163602 & 1.08672758 & 1.39634746 & 0.53592833 \end{bmatrix},$$

and the true initial population values $\mathbf{x}_0^*$ set to

$$\mathbf{x}_0^* = \begin{bmatrix} 1.00471435 \\ 0.98809024 \\ 1.01432707 \\ 0.99687348 \end{bmatrix}.$$

Figure 10 shows the resulting development of the four populations over ten unit time steps, including noise on the observed values for x_1, x_2 and x_3.

Both the INN and the HINT network we trained consist of ten (non-recursive) coupling blocks with a total of 10^6 trainable weights. We used Adam to train both for 50 epochs per time step with 64000 training samples and a batch size of 500. The learning rate started at 10^{-2} and exponentially decayed to 10^{-3} (HINT) and 10^{-4} (INN), respectively. Inference on x_4 begins with an initial guess at $t = 0$ drawn from $\mathcal{N}(1, \frac{1}{10})$.