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

Jakob Kruse[*1], **Gianluca Detommaso**[*2], **Ullrich Köthe** [1], **Robert Scheichl** [1]
[1] Heidelberg University, [2] Amazon.com
jakob.kruse@iwr.uni-heidelberg.de

Abstract

Many recent invertible neural architectures are based on coupling block designs where variables are divided in two subsets which serve as inputs of an easily invertible (usually affine) triangular transformation. While such a transformation is invertible, its Jacobian is very sparse and thus may lack expressiveness. This work presents a simple remedy by noting that subdivision and (affine) coupling can be repeated recursively within the resulting subsets, 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 evaluate our method on some standard data sets and benchmark its full power 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.

Introduction

Invertible neural networks based on the normalizing flow principle have recently gained increasing attention for generative modeling, in particular networks built on a coupling block design (Dinh, Sohl-Dickstein, and Bengio 2017). Their success is due to a number of useful properties: *(a)* they can tractably model complex high-dimensional probability densities without suffering from the curse-of-dimensionality, *(b)* training via the maximum likelihood objective is generally very stable, *(c)* their latent space opens up opportunities for model interpretation and manipulation, and *(d)* the same trained model can be used for both efficient data generation and efficient density calculation.

While autoregressive models can also be trained as normalizing flows and share properties *(a)* and *(b)*, they sacrifice efficient invertibility for expressive power and thus lose properties *(c)* and *(d)*. In contrast, lack of expressive power of a single invertible block is a core limitation of invertible networks, which needs to be compensated by extremely deep models with dozens or hundreds of blocks, e.g., the GLOW architecture (Kingma and Dhariwal 2018). While invertibility allows to back-propagate through very deep networks with minimal memory footprint (Gomez et al. 2017), more expressive invertible building blocks are still of great interest. The superior performance of autoregressive approaches such as (Van den Oord et al. 2016) is due to the stronger interaction between variables, reflected in a dense triangular Jacobian matrix, at the expense of cheap inversion. The theory of transport maps (Villani 2008) provides certain guarantees of universality for triangular maps, which do not hold for the standard coupling block design with a comparatively sparse Jacobian (figure 1, *left*).

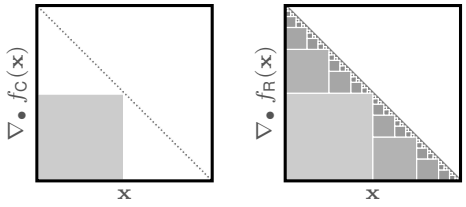

Figure 1: Sparse *(left)* and dense *(right)* triangular Jacobian of a standard coupling block and of our recursive design, respectively. Nonzero parts of the Jacobian in gray.

Here, we propose an extension to the coupling block design that recursively fills in the previously unused portions of the Jacobian using smaller coupling blocks. This allows for dense triangular maps (figure 1, *right*), or any intermediate design if the recursion is stopped before, while retaining the advantages of the original coupling block architecture. Furthermore, the recursive structure of this mapping can be used for efficient conditional sampling and Bayesian inference. Splitting the variables of interest into two subsets $\mathbf{x}$ and $\mathbf{y}$, a single normalizing flow model can be built that allows efficient sampling from both the joint distribution $p(\mathbf{x}, \mathbf{y})$ and the conditional $p(\mathbf{x} \mid \mathbf{y})$. It should be noted that our extension would also work for convolutional architectures like GLOW.

Finally, we introduce a new family of data sets based on Fourier parameterizations of two-dimensional curves. 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 output. However, the sparsity of the basic coupling block only becomes an issue beyond two dimensions where it is challenging to visualize the distribution or individual samples. Pixel-based image data sets, on the other hand, quickly are too high dimensional for a mean-

[*]equal contribution.

ingful assessment of the quality of the estimated densities.

A step towards visualizable data sets of intermediate size has been made in (Kruse et al. 2019), but their four-dimensional problems are still too 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 dimension, where each data point parameterizes a closed curve in 2D space that is easy to visualize. Increasing the input data dimension allows the representation of distributions of more and more complex curves.

To summarize, the contributions of this paper are: *(a)* a simple, efficiently invertible flow model with dense, triangular Jacobian; *(b)* a hierarchical architecture to model joint as well as conditional distributions; *(c)* a novel family of data sets allowing easy visualization for arbitrary dimensions.

The remainder of this work consists of a literature review, some mathematical background, a description of our method and supporting numerical experiments, followed by closing remarks.

Related Work

Normalizing flows were popularized in the context of deep learning chiefly by the work of (Rezende and Mohamed 2015) and (Dinh, Krueger, and Bengio 2015). By now, a large variety of architectures exist to realize normalizing flows. The majority falls into one of two groups: coupling block architectures and autoregressive models. For a comprehensive overview and background information on invertible neural networks and normalizing flows see (Kobyzev, Prince, and Brubaker 2019) or (Papamakarios et al. 2019).

Additive and then affine coupling blocks were first introduced by (Dinh, Krueger, and Bengio 2015; Dinh, Sohl-Dickstein, and Bengio 2017), while (Kingma and Dhariwal 2018) went on to generalize the permutation of variables between blocks by learning the corresponding matrices, besides demonstrating the power of flow networks as generators. Subsequent works have focused on replacing the (componentwise) affine transformation at the heart of such networks, which limits expressiveness, e.g., by replacing affine couplings with more expressive monotonous splines (Durkan et al. 2019), albeit at the cost of evaluation speed.

On the other hand, there is lso a rich body of work on autoregressive (flow) networks (Huang et al. 2018; Kingma et al. 2016; Van den Oord et al. 2016; Van den Oord, Kalchbrenner, and Kavukcuoglu 2016; Papamakarios, Pavlakou, and Murray 2017). More recently, (Jaini, Selby, and Yu 2019) applied second-order polynomials to improve expressive power over typical autoregressive models and proved that their model is a universal density approximator. While such models provide excellent density estimation compared to coupling architectures (Liao, He, and Shu 2019; Ma et al. 2019), generating samples is often not a priority and can be prohibitively slow.

There are other approaches, outside those two subfields, that also seek 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, Meng, and Ermon 2019) uses lower triangular weight matrices that can be inverted via fixed-point iteration. The normalizing flow principle is formulated continuously as a differential equation (DE) by (Grathwohl et al. 2019), which allows free-form Jacobians but requires integrating a DE for each network pass. (Karami et al. 2019) introduce another method with dense Jacobian, based on invertible convolutions 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 a latent vector, training the flow part with a *maximum mean discrepancy* objective (MMD, Gretton et al. 2012) instead of maximum likelihood. Later (Ardizzone et al. 2019b) introduced a simple conditional coupling block to construct a conditional normalizing flow.

Mathematical Background

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

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

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

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

For affine coupling blocks (Dinh, Sohl-Dickstein, and Bengio 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 the Jacobian determinant of such a block can be computed very efficiently as

$$\log\Big|\det \mathbf{J}_{f_{\mathrm{C}}}(\mathbf{x})\Big| = \log\left|\det \frac{\partial f_{\mathrm{C}}(\mathbf{x})}{\partial \mathbf{x}}\right| = \mathrm{sum}(s(\mathbf{x}_1)). \tag{3}$$

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

Normalizing Flows and Transport Maps

To create a normalizing flow, this 'pipeline' $T = f_{\mathrm{C}1} \circ f_{\mathbf{Q}1} \circ f_{\mathrm{C}2} \circ f_{\mathbf{Q}2} \circ \ldots$ is trained via maximum likelihood loss

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

to transport the data distribution p_X to a standard normal latent distribution $p_Z = \mathcal{N}(\mathbf{0}, \mathbf{I})$. The map T can then be used to sample from p_X by drawing a sample $\mathbf{z}^{(i)}$ from p_Z in the latent space and by passing it through the inverse model $S = T^{-1}$ to obtain $\mathbf{x}^{(i)} = S(\mathbf{z}^{(i)})$.

Using the change-of-variables formula, the density at a given data point $\mathbf{x}$ can also be calculated as $p_X(\mathbf{x}) = p_Z(T(\mathbf{x})) \cdot |\det \mathbf{J}_T(\mathbf{x})|$. The mathematical basis of this procedure is 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. the data distribution, (Marzouk et al. 2016)). In fact, up to a constant, namely the (typically inaccessible) fixed entropy $H(p_X)$ of the data distribution, the expected value of the objective in equation (4) is the *Kullback-Leibler (KL) divergence* between the data distribution p_X and the push-forward of the latent density $S_{\#}p_Z$:

$$D_{\mathrm{KL}}(p_X \parallel S_{\#}p_Z) = \int p_X(\mathbf{x}) \log \frac{p_X(\mathbf{x})}{S_{\#}p_Z(\mathbf{x})} \, \mathrm{d}\mathbf{x} = \mathbb{E}_{\mathbf{x}\sim p_X}[\mathcal{L}(\mathbf{x})] + H(p_X). \quad (5)$$

Normalizing flows represent one parametrised family of maps over which equation (4) can be minimized. Other examples include polynomial (Marzouk et al. 2016), kernel-based (Liu and Wang 2016) or low-rank tensor (Dolgov et al. 2020) approximations.

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 can be interpreted as a non-linear generalization of the classic QR decomposition (Stoer and 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.

Bayesian Inference with Conditional Flows

Inverse problems arise when one possesses a well-understood model for the *forward mapping* $\mathbf{x} \to \mathbf{y}$ from hidden parameters $\mathbf{x}$ to observable outcomes $\mathbf{y}$, e.g. in the form of an explicit likelihood $p(\mathbf{y} \mid \mathbf{x})$ or a Monte-Carlo simulation. However, the actual object of interest is the *inverse mapping* $\mathbf{y} \to \mathbf{x}$ from observations to parameters. According to Bayes' theorem, this requires estimation of the posterior conditional density $p(\mathbf{x} \mid \mathbf{y})$. Such Bayesian inference problems arise frequently in the sciences and are generally very hard.

Normalizing flows can be used in several ways to estimate conditional densities. The approach in this paper is inspired by (Marzouk et al. 2016) and exploits the link to Knothe-Rosenblatt maps highlighted above. As described below, see figure 3 (left), it suffices to constrain the possible rearrangements of variables in the coupling blocks, i.e. the choice of the orthogonal blocks $f_{\mathbf{Q}i}$, to enable conditional sampling. This was first noted in (Detommaso et al. 2019).

Independently, (Ardizzone et al. 2019b) and (Winkler et al. 2019) 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 $\mathbf{z} = T(\mathbf{x} \mid \mathbf{y})$, its inverse $T^{-1}(\mathbf{z} \mid \mathbf{y})$ can be used to turn the latent distribution $p_Z(\mathbf{z})$ into an approximation of the posterior $p(\mathbf{x} \mid \mathbf{y})$.

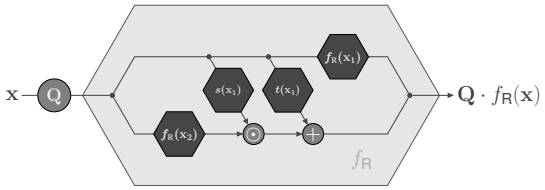

Figure 2: A recursive affine coupling block. The inner functions $f_{\mathrm{R}}(\mathbf{x}_i)$ take again the form of the outer gray block, repeated until the maximum hierarchy depth is reached. Each such coupling block in itself has a triangular Jacobian.

Method

We extend the basic coupling block design in two ways.

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 and Pajunen 1999) states that a single triangular transformation can, in theory, already represent arbitrary distributions.

The following recursive coupling scheme f_{R} makes use of this potential and fills the empty areas below the diagonal: Given $\mathbf{x} \in \mathbb{R}^N$ and a hierarchy depth $K \in \mathbb{N}$, we define recursively, for $k = K, K-1, \ldots, 1$:

$$\mathbf{x}' = f_{\mathrm{R},k}(\mathbf{x}) = \begin{cases} f_{\mathrm{C}}(\mathbf{x}), & \text{if } N_k \leq 3, \\ \begin{bmatrix} f_{\mathrm{R},k-1}(\mathbf{x}_1) \\ C_k\left(f_{\mathrm{R},k-1}(\mathbf{x}_2) \mid \mathbf{x}_1\right) \end{bmatrix}, & \text{else,} \end{cases} \quad (6)$$

where for each k, $\mathbf{x}_1 = \mathbf{x}_{0:\lfloor N_k/2 \rfloor}$ and $\mathbf{x}_2 = \mathbf{x}_{\lfloor N_k/2 \rfloor : N_k}$, N_k is the size of the current input vector and $N_K = N$. Note that each sub-coupling has its own coupling function C_k with independent parameters. The inverse transform is

$$\mathbf{x} = f_{\mathrm{R},k}^{-1}(\mathbf{x}') = \begin{cases} f_{\mathrm{C}}^{-1}(\mathbf{x}'), & \text{if } N_k \leq 3, \\ \begin{bmatrix} f_{\mathrm{R},k-1}^{-1}(\mathbf{x}_1') \\ f_{\mathrm{R},k-1}^{-1}\left(C_k^{-1}\left(\mathbf{x}_2' \,\middle|\, f_{\mathrm{R},k-1}^{-1}(\mathbf{x}_1')\right)\right) \end{bmatrix}, & \text{else.} \end{cases} \quad (7)$$

For $K = \lceil \log_2 N \rceil$, 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_k. A visual representation of the architecture can be seen in figure 2.

However, since the (sub-)coupling blocks are affine, $f_{\mathrm{R},K}$ can only represent an approximation of the exact Knothe-Rosenblatt map and it is still necessary, as for standard coupling blocks, to create a normalizing flow by composing several recursive coupling blocks interspersed with orthogonal transformations $f_{\mathbf{Q}}$. Thus, in practice, it is also more economical to limit the depth of the hierarchy to 2 or 3. This already increases the amount of interaction between individual variables considerably, while limiting the computational overhead, but it allows to use much shallower networks. The trade-off between number of blocks and hierarchy depth will be studied for the Fourier shapes data set in this work.

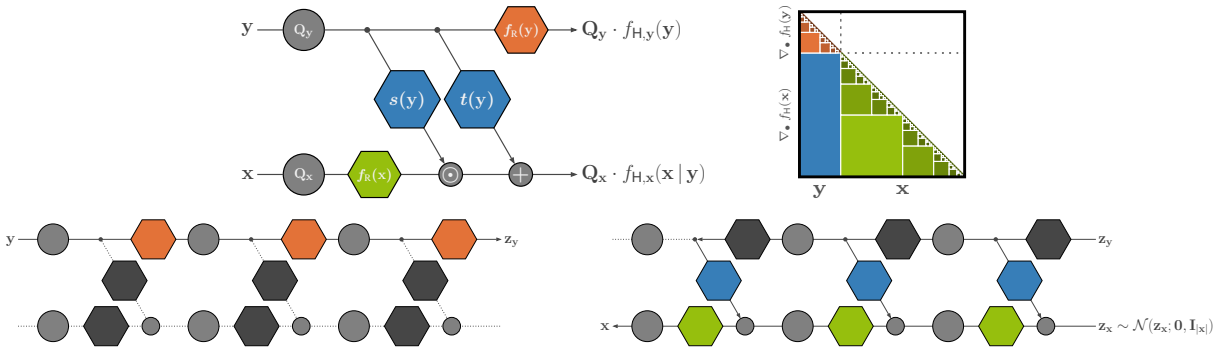

Figure 3: *Top:* Single HINT block with recursive coupling, and its Jacobian matrix. Transformation of $\mathbf{x}$ is influenced by $\mathbf{y}$, but not vice-versa, imposing a hierarchy on variables. *Bottom:* Using HINT flows for conditional sampling/Bayesian inference.

Hierarchical Invertible Neural Transport

While the recursive coupling block defined above is motivated by the search for a more expressive architecture, it is also ideally suited for estimating conditional flows and thus for Bayesian inference.

Specifically, in a setting with paired data $(\mathbf{x}_i, \mathbf{y}_i)$, where subsequently we want a sampler for $\mathbf{x}$ conditioned on $\mathbf{y}$, we can provide both variables as input to the flow, separating them in the first hierarchy level for further transformation at the next recursion level. Crucially, $\mathbf{x}$ and $\mathbf{y}$ variables are never permuted between lanes, thus only feeding forward information from the $\mathbf{y}$-lane to the $\mathbf{x}$-lane as shown in figure 3 *(top left)*. Instead of one large permutation operation over all variables, as in the hierarchical coupling block design in figure 2, we apply individual permutations $\mathbf{Q_y}$ and $\mathbf{Q_x}$ to each respective lane at the beginning of the block. 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, $\mathbf{z} = [\mathbf{z_y}, \mathbf{z_x}]^\top = T(\mathbf{y}, \mathbf{x})$, 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:

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

where $S = T^{-1}$. But because the $\mathbf{y}$-lane in HINT can be evaluated independently of the $\mathbf{x}$-lane, we can determine the partial latent code $\mathbf{z_y}$ for a given $\mathbf{y}$ and hold it fixed (figure 3, *bottom left*), while drawing $\mathbf{z_x}$ from the $\mathbf{x}$-part of the latent distribution *(bottom right)*. This yields samples from the conditional density:

$$\mathbf{x} = S^{\mathbf{x}}([\mathbf{z_y}, \mathbf{z}_x]) \sim p_T(\mathbf{x} \mid \mathbf{y}) \quad \text{with } \mathbf{z_y} = T^{\mathbf{y}}(\mathbf{y}), \qquad (10)$$

where superscripts $\mathbf{x}$ and $\mathbf{y}$ respectively denote $\mathbf{x}$- and $\mathbf{y}$-lanes of the transformations. This means HINT gives access to both the joint density of $\mathbf{x}$ and $\mathbf{y}$, as well as the conditional density of $\mathbf{x}$ given $\mathbf{y}$, e.g. for Bayesian inference.

Computational Complexity

The number of couplings doubles in every recursion level, whereas the workload per coupling decreases exponentially, so that the total order of complexity of HINT and RealNVP is the same. All sub-networks s and t within one level are independent of each other and can be processed in a single parallel pass on the GPU (see appendix). Only the final affine transformations and some bookkeeping operations must be executed sequentially, but at negligible cost compared to the other tensor operations. Our first, non-parallel implementation of HINT is 2-10 times slower than RealNVP.

Experiments

We perform experiments on classical UCI data sets (Dua and Graff 2017) and a new data set called "Fourier shapes". We introduce this new data set to balance four conflicting goals:

1. The dimension of the data should be high enough for HINT's hierarchical decomposition to make a difference.
2. The dimension should be low enough to allow for accurate quantitative evaluation and comparison of results.
3. Learning the joint distribution should be challenging due to complex interactions between variables.
4. Visualizations should allow intuitive qualitative comparison of the differences between alternative approaches.

Our new data set represents families of 2-dimensional contours in terms of the probability density of their Fourier coefficients and fulfills the above requirements: The dimension of the problem can be easily adjusted by controlling the complexity of the shapes under consideration and the number of Fourier coefficients (1,2). Shapes with sharp corners and long-range symmetries require accurate alignment of many Fourier coefficients (specifically, of their phases, 3 – see appendix). Humans can readily recognize the quality of a shape representation in a picture (4).

Our experiments show considerable improvements of HINT over RealNVP. To clearly demonstrate these advantages, we heavily restrict the networks' parameter budgets – larger networks would be much more accurate, but exhibit less meaningful differences. Models were trained on an *RTX 2080 Ti* GPU. Hyper-parameters are listed in the appendix.[1]

[1]Code + data at https://github.com/VLL-HD/HINT.

BLOCKS		DIM	REAL-NVP	RECURSIVE
4	POWER	6	-0.054 ± 0.017	$\mathbf{-0.027 \pm 0.018}$
	GAS	8	7.620 ± 0.136	$\mathbf{7.662 \pm 0.094}$
	MINIBOONE	42	-19.296 ± 0.395	$\mathbf{-14.547 \pm 0.164}$
8	POWER	6	$\mathbf{0.093 \pm 0.002}$	0.080 ± 0.007
	GAS	8	8.062 ± 0.177	$\mathbf{8.137 \pm 0.055}$
	MINIBOONE	42	-16.625 ± 0.119	$\mathbf{-14.117 \pm 0.163}$

Table 1: Normal and recursive coupling compared on UCI benchmarks in terms of average log-likelihood (mean $\pm$ std over 3 training runs; higher is better $\uparrow$).

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

UCI Density Estimation Benchmarks

The tabular UCI data sets (Dua and Graff 2017) are popular for comparing density models. Using public code for preprocessing[2], we compare several flow models with REAL-NVP and RECURSIVE coupling blocks in terms of the average log-likelihood on the test set. To tease out shortcomings, each model is "handicapped" to a budget of 500k (POWER, GAS) or 250k (MINIBOONE) trainable parameters.

Table 1 shows that the recursive design achieves similar or better test likelihood in all cases, even when the low dimensionality (DIM) of the data allows little recursion.

Fourier Shapes Data Set

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

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

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

Vice-versa, we can calculate the Fourier coefficients

$$\mathbf{a}_m = \frac{1}{L} \sum_{l=0}^{L-1} \mathbf{p}_l \cdot e^{-2\pi \cdot i \cdot m \cdot l / L}, \text{ for } m \in [-M, M], \tag{12}$$

to approximately fit a curve through a sequence of L points $\mathbf{p}_l \in \mathbb{R}^2$, $l = 0, \ldots, L-1$. By increasing M, higher order terms are added to the parameterization in equation (11) and the shape is 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 $\mathbf{a}_m$ is represented by four real numbers.

We perform experiments on two specific data sets, first using curves of order $M = 2$, i.e. $|\mathbf{x}| = 20$, to represent a distribution of simple shapes that arise from the intersection of two randomly placed circles with a fixed ratio of radii and a fixed distance. The resulting *Lens* shapes can be seen in figure 5 *(left)*, together with the highly structured correlation matrix of Fourier parameters $\mathbf{x}_i$ that yield such shapes.

The second data set uses $M = 12$, i.e. $|\mathbf{x}| = 100$, to represent *Cross* shapes which are generated by crossing two bars of random length, width and lateral shift at a right angle, oriented randomly, but positioned close to the origin. This results in a variation of Xs, Ls and Ts, some of which are shown in figure 5 *(right)* together with the even more complicated (100×100) parameter correlation matrix.

Density estimation. For density estimation, a single-block and a two-block network are trained on the *Lens*-shapes data, once with standard coupling blocks and once with the new recursive design. All networks have the same total parameter budget – details in the appendix.

Samples from the two-block models and absolute differences to the true parameter correlation matrices are shown in figure 6 *(left)*. Qualitatively, samples from the recursive model are visually more faithful and have smaller errors in the correlation matrices. Quantitatively, we compare over three training runs per model using the following metrics:

- **Maximum mean discrepancy** (MMD, Gretton et al. 2012) 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 over 100 batches from the data prior and from each trained model. Lower is better.
- Average **log-likelihood** (LL) of the test data under the model, i.e. $-\frac{1}{2}T(\mathbf{x})^2 + \log|\mathbf{J}_T(\mathbf{x})| - \log(2\pi\frac{N}{2})$ where N is the data dimensionality. Higher is better.
- Average **intersection-over-union** (IOU) between generated shapes and the best fitting shape that follows the construction rules of the data set. See appendix for details on the fitting procedure. Higher is better.
- Average **Hausdorff distance** (H-DIST) between the contours of the generated shapes and those of the best fitting shapes, as above. Lower is better.

Table 2 shows how recursive coupling blocks outperform the conventional design. The difference is especially striking for the single-block network, as a non-recursive coupling block leaves half the variables untouched and is thus inherently unable to model the data distribution properly.

We also trained standard and recursive networks with 4 and 8 coupling blocks on the larger *Cross*-shapes data set. Representative samples from the 4-block models are shown in figure 6 *(right)* together with the best fitting, actual *Cross* shape. Here, it is even more clearly visible that the recursive

[2] https://github.com/LukasRinder/normalizing-flows

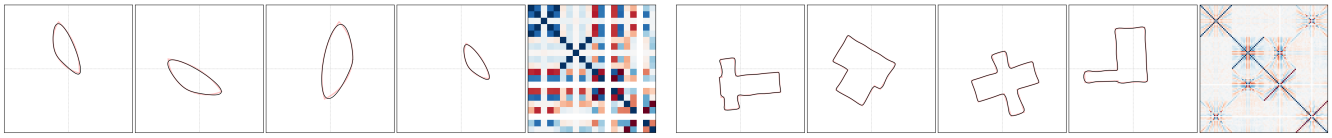

Figure 5: *Left:* Samples from the *Lens* shapes data set, and true correlation matrix of Fourier coefficients for a large batch of *Lens* shapes. *Right:* The same for *Cross* shapes. Red lines behind the Fourier curves show the original geometry they approximate.

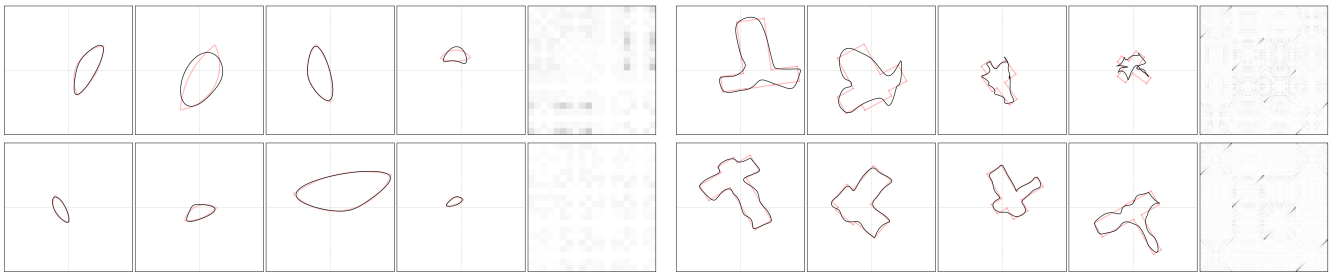

Figure 6: Samples from REAL-NVP *(top)* and RECURSIVE *(bottom)* coupling block networks, trained on the *Lens* shapes *(left)* and the *Cross* shapes *(right)* from figure 5. The closest fitting shapes from the true distributions are depicted in red for reference. Here and in subsequent figures, the fifth panel shows the absolute differences between true and sampling correlation matrices.

BLOCKS		REAL-NVP	RECURSIVE
1	MMD ↓	0.370 ± 0.000	$\mathbf{0.140 \pm 0.002}$
	LL ↑	2.021 ± 0.001	$\mathbf{2.861 \pm 0.006}$
	IOU ↑	0.449 ± 0.010	$\mathbf{0.688 \pm 0.014}$
	H-DIST ↓	0.519 ± 0.005	$\mathbf{0.109 \pm 0.005}$
2	MMD ↓	0.012 ± 0.001	$\mathbf{0.009 \pm 0.004}$
	LL ↑	3.141 ± 0.036	$\mathbf{3.219 \pm 0.005}$
	IOU ↑	0.789 ± 0.024	$\mathbf{0.819 \pm 0.006}$
	H-DIST ↓	0.063 ± 0.007	$\mathbf{0.057 \pm 0.000}$

Table 2: Comparing REAL-NVP and RECURSIVE coupling for sampling and density estimation for the *Lens*-shapes data (plotting mean $\pm$ standard deviation over 3 training runs).

model produces samples with better geometry, i.e right angles, straight lines and symmetries in the expected locations.

A quantitative comparison in terms of LL, IOU and H-DIST is presented in table 3, with recursive coupling consistently outperforming standard REAL-NVP blocks.

Bayesian Inference on Fourier Shapes

To set-up Bayesian inference tasks, we formulate forward mappings $\mathbf{x} \rightarrow \mathbf{y}$ from $\mathbf{x}$ to observable features $\mathbf{y}$. Since these features are incomplete shape descriptors, the inverse $\mathbf{y} \rightarrow \mathbf{x}$ is ambiguous, and $p(\mathbf{x} \mid \mathbf{y})$ is learned with HINT.

Given a *Lens* shape, our forward mapping locates its two tips and returns their horizontal and vertical distances d_h and d_v. The tips' absolute positions and the side of the lens' "bulge" remain undetermined by these features.

The forward mapping for *Cross* shapes returns four geometrical features. These are the 2d coordinates of the center, i.e where the bars cross, plus the angle of and thickness ratio between the two bars. What remains free, are the absolute thickness, as well as the length and lateral shift of the bars.

BLOCKS	4× RNVP	4× REC	8× RNVP	8× REC
LL ↑	3.419	**3.627**	3.329	**3.637**
IOU ↑	0.594	**0.823**	0.588	**0.823**
H-DIST ↓	0.134	**0.077**	0.138	**0.073**

Table 3: Comparison of flow models for *Cross* shapes with normal (RNVP) and recursive coupling (REC), respectively.

Finally, noise $\sigma \sim \mathcal{N}(\mathbf{0}, \frac{1}{20}\mathbf{I})$ is added to the output of each forward mapping to obtain observed data vectors $\mathbf{y}$.

We trained a conditional flow model (cINN) and HINT with 1, 2, 4 and 8 blocks for Bayesian inference on the *Lens* shapes. A quantitative comparison, in terms of MMD, IOU and H-DIST, is given in table 4. Here, however, MMD does not compare to samples from the prior $p_X(\mathbf{x})$, but to samples from an estimate of the true posterior $p(\mathbf{x} \mid \mathbf{y})$, estimated via *Approximate Bayesian Computation* (ABC, Csilléry et al. 2010), as in (Ardizzone et al. 2019a), see appendix.

In table 4 we see that HINT consistently produces better shapes (measured by IOU and H-DIST), especially in the case of a single block, and it exhibits better conditioning (as evidenced by MMD) for all but the 8-block model, most likely due to the limited parameter budget, which leaves some of the sub-networks underparameterized. A similar effect can be observed in the LL (not shown in table 4), when only measuring the log-likelihood of the $\mathbf{x}$-lane in HINT and simply excluding contributions from the $\mathbf{y}$-lane.

Qualitative results in figures 7 and 8, for the *Lens* and for the *Cross* shapes, respectively, confirm the superior performance of HINT also visually, in particular producing significantly better angles and symmetries for the *Cross* shapes. This is quantitatively supported in the metrics in table 5.

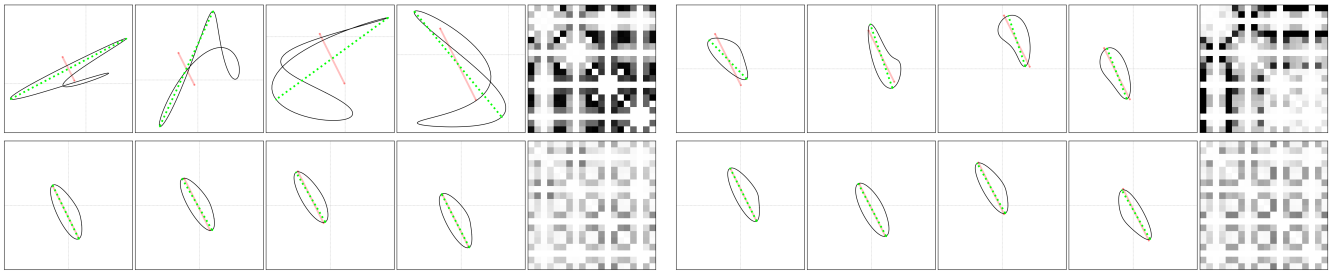

Figure 7: Samples from a conditional coupling net *(*CINN*, left)* and from HINT *(right)*, trained *Lens* shapes. Green dotted lines mark the largest diameter of each shape; red lines show how it should look according to the data $\mathbf{y}$. Both models do well with 4 blocks *(bottom)*, but only HINT generates reasonable samples with a single coupling block *(top)*. Last panels as in figure 6.

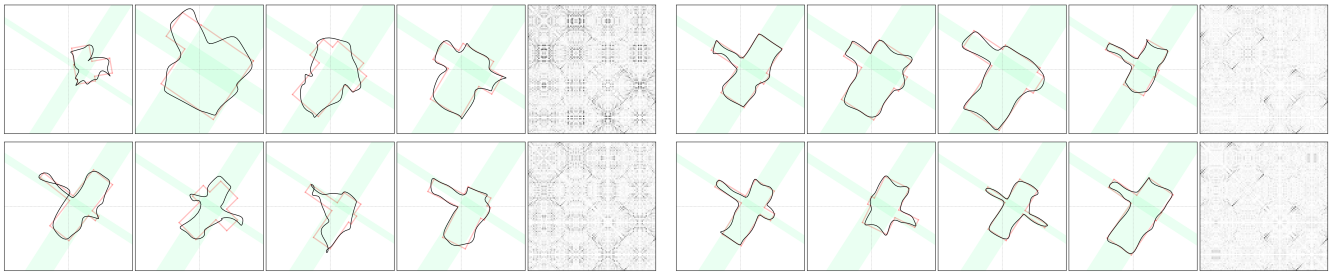

Figure 8: Samples from a conditional coupling net *(*CINN*, left)* and from HINT *(right)* with 4 *(top)* or 8 *(bottom)* blocks, trained on *Cross* shapes. The expected center, angle and thickness ratio of the *Cross* according to the data $\mathbf{y}$ are shown in green, the best fitting *Cross* shape in red. Visually, HINT reproduces shapes from the data set much better. Last panels as in figure 6.

BLOCKS		CINN	HINT
1	MMD ↓	0.746 ± 0.000	$\mathbf{0.030 \pm 0.001}$
	IOU ↑	0.456 ± 0.001	$\mathbf{0.849 \pm 0.003}$
	H-DIST ↓	0.521 ± 0.006	$\mathbf{0.051 \pm 0.001}$
2	MMD ↓	0.048 ± 0.012	$\mathbf{0.016 \pm 0.001}$
	IOU ↑	0.839 ± 0.011	$\mathbf{0.869 \pm 0.005}$
	H-DIST ↓	0.054 ± 0.002	$\mathbf{0.044 \pm 0.002}$
4	MMD ↓	0.020 ± 0.003	$\mathbf{0.010 \pm 0.001}$
	IOU ↑	0.865 ± 0.006	$\mathbf{0.875 \pm 0.007}$
	H-DIST ↓	0.046 ± 0.002	$\mathbf{0.043 \pm 0.002}$
8	MMD ↓	$\mathbf{0.007 \pm 0.002}$	0.011 ± 0.003
	IOU ↑	0.864 ± 0.003	$\mathbf{0.876 \pm 0.003}$
	H-DIST ↓	0.046 ± 0.001	$\mathbf{0.043 \pm 0.001}$

Table 4: Conditional (CINN) vs. hierarchical (HINT) coupling for Bayesian inference on *Lens* shapes (mean $\pm$ standard deviation over 3 training runs). See text for details.

Recursion Depth vs. Number of Blocks

Seeing that recursive coupling blocks outperform standard ones, we also tested if using more standard coupling blocks closes the gap. We looked at several combinations of number of blocks, recursion depth and parameter budget on the *Cross*-shapes data. In summary, the first two recursion levels improve performance more than additional coupling blocks. Beyond that, we see diminishing returns, as the limited parameter budget gets distributed over too many subnetworks.

BLOCKS	4× CINN	4× HINT	8× CINN	8× HINT
LL ↑	3.625	**3.724**	3.609	**3.766**
IOU ↑	0.654	**0.843**	0.590	**0.859**
H-DIST ↓	0.116	**0.073**	0.119	**0.066**

Table 5: Fidelity of *Cross* shapes generated by conditional (CINN) and hierarchical (HINT) flows, respectively.

Full results are in the appendix.

Conclusion

We presented recursive coupling blocks and HINT, a new invertible architecture for normalizing flow, improving on the traditional coupling block in terms of expressive power by densifying the triangular Jacobian, while keeping the advantages of an accessible latent space. This keeps the efficient sampling and density estimation of RealNVP, which is often compromised by other approaches to denser Jacobians, e.g. auto-regressive flows. To evaluate the model, we introduced a versatile family of data sets based on Fourier decompositions of simple 2D shapes that can be visualized easily, independent of the chosen dimension. In terms of future improvements, we expect that our formulation can be made more computationally efficient through the use of e.g. masking operations, enabling more advanced parallelization.

APPENDIX Supplementary material and most up-to-date paper version found under https://arxiv.org/abs/1905.10687

Acknowledgments

Jakob Kruse was supported by Informatics for Life funded by the Klaus Tschira Foundation. Gianluca Detommaso was supported by the EPSRC Centre for Doctoral Training in Statistical Applied Mathematics at Bath (EP/L015684/1).

References

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

Ardizzone, L.; Lüth, C.; Kruse, J.; Rother, C.; and Köthe, U. 2019b. Guided image generation with conditional invertible neural networks. *arXiv:1907.02392* .

Behrmann, J.; Grathwohl, W.; Chen, R. T. Q.; Duvenaud, D.; and Jacobsen, J.-H. 2019. Invertible residual networks. In *Intl. Conf. Machine Learning*.

Chen, T. Q.; Behrmann, J.; Duvenaud, D. K.; and Jacobsen, J.-H. 2019. Residual flows for invertible generative modeling. In *Adv. Neural Information Process. Syst.*, 9913–9923.

Csilléry, K.; Blum, M.; Gaggiotti, O.; and François, O. 2010. Approximate Bayesian Computation (ABC) in practice. *Trends in Ecology & Evolution* 25: 410–8. doi: 10.1016/j.tree.2010.04.001.

Detommaso, G.; Kruse, J.; Ardizzone, L.; Rother, C.; Köthe, U.; and Scheichl, R. 2019. HINT: Hierarchical Invertible Neural Transport for general and sequential Bayesian inference. *arXiv:1905.10687v1* .

Dinh, L.; Krueger, D.; and Bengio, Y. 2015. NICE: Nonlinear Independent Components Estimation. In *Intl. Conf. Learning Representations*.

Dinh, L.; Sohl-Dickstein, J.; and Bengio, S. 2017. Density estimation using Real NVP. In *Intl. Conf. Learning Representations*.

Dolgov, S.; Anaya-Izquierdo, K.; Fox, C.; and Scheichl, R. 2020. Approximation and sampling of multivariate probability distributions in the tensor train decomposition. *Stat. Comput.* 30: 603–625.

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository. URL http://archive.ics.uci.edu/ml. Last accessed on 2021-03-08.

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

Gomez, A. N.; Ren, M.; Urtasun, R.; and Grosse, R. B. 2017. The reversible residual network: Backpropagation without storing activations. In *Adv. Neural Information Processing Systems*, 2214–2224.

Grathwohl, W.; Chen, R. T. Q.; Bettencourt, J.; and Duvenaud, D. 2019. Scalable reversible generative models with free-form continuous dynamics. In *Intl. Conf. Learning Representations*.

Gretton, A.; Borgwardt, K. M.; Rasch, M. J.; Schölkopf, B.; and Smola, A. 2012. A kernel two-sample test. *J. Mach. Learn. Res.* 13(Mar): 723–773.

Huang, C.-W.; Krueger, D.; Lacoste, A.; and Courville, A. 2018. Neural autoregressive flows. In *Intl. Conf. Machine Learning*, 2078–2087.

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

Jaini, P.; Selby, K. A.; and Yu, Y. 2019. Sum-of-squares polynomial flow. In *Intl. Conf. Machine Learn.*, 3009–3018.

Karami, M.; Schuurmans, D.; Sohl-Dickstein, J.; Dinh, L.; and Duckworth, D. 2019. Invertible Convolutional Flow. In *Adv. Neural Information Processing Systems*, 5636–5646.

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

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

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

Kruse, J.; Ardizzone, L.; Rother, C.; and Köthe, U. 2019. Benchmarking invertible architectures on inverse problems. *ICML Workshop on Invertible Neural Nets and Normalizing Flows (INNF'19)* .

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

Liu, Q.; and Wang, D. 2016. Stein variational gradient descent: A general purpose Bayesian inference algorithm. In *Adv. Neural Information Processing Systems*, 2378–2386.

Ma, X.; Kong, X.; Zhang, S.; and Hovy, E. 2019. MaCow: Masked convolutional generative flow. *Adv. Neural Information Processing Systems* 5891–5900.

Marzouk, Y.; Moselhy, T.; Parno, M.; and Spantini, A. 2016. Sampling via measure transport: An introduction. In Ghanem, R.; Higdon, D.; and Owhadi, H., eds., *Handbook of Uncertainty Quantification*, 1–41. Springer.

McGarva, J.; and Mullineux, G. 1993. Harmonic representation of closed curves. *Appl. Math. Model.* 17(4): 213–218.

Papamakarios, G.; Nalisnick, E.; Rezende, D. J.; Mohamed, S.; and Lakshminarayanan, B. 2019. Normalizing flows for probabilistic modeling and inference. *arXiv:1912.02762* .

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

Rezende, D.; and Mohamed, S. 2015. Variational inference with normalizing flows. In *Intl. Conf. Machine Learning*, 1530–1538.

Song, Y.; Meng, C.; and Ermon, S. 2019. MintNet: Building invertible neural networks with masked convolutions. In *Adv. Neural Information Processing Systems*, 11002–11012.

Stoer, J.; and Bulirsch, R. 2013. *Introduction to Numerical Analysis*, volume 12. Springer Science & Business Media.

Van den Oord, A.; Kalchbrenner, N.; Espeholt, L.; Vinyals, O.; Graves, A.; et al. 2016. Conditional image generation with PixelCNN decoders. In *Adv. Neural Information Processing Systems*, 4790–4798.

Van den Oord, A.; Kalchbrenner, N.; and Kavukcuoglu, K. 2016. Pixel recurrent neural networks. In *Intl. Conf. Machine Learning*, 1747–1756.

Villani, C. 2008. *Optimal Transport: Old and New*, volume 338. Springer Science & Business Media.

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