Multi-Resolution Continuous Normalizing Flows

Vikram Voleti

vikram.voleti@gmail.com

University of Montreal

Chris Finlay

Deep Render

Adam Oberman

McGill University

Christopher Pal

Polytechnique Montréal

Research Article

Keywords: Continuous Normalizing Flows, Image generation, wavelet

Posted Date: June 16th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-3027011/v1

Additional Declarations: No competing interests reported.

Version of Record: A version of this preprint was published at Annals of Mathematics and Artificial Intelligence on March 21st, 2024. See the published version at https://doi.org/10.1007/s10472-024-09939-5.

Multi-Resolution Continuous Normalizing Flows

Vikram Voleti[1,2*], Chris Finlay[1,3,5], Adam Oberman[1,3], Christopher Pal[1,4]

[1]Mila, 6666 Rue St. Urbain, Montreal, H2S 3H1, QC, Canada.
[2]DIRO, University of Montreal, 2900 Bd Édouard-Montpetit, Montreal, H3T 1J4, QC, Canada.
[3]McGill University, 845 Rue Sherbrooke O, Montreal, H3A 0G4, QC, Canada.
[4]École Polytechnique de Montréal, 2500 Chem. de Polytechnique, Montreal, H3T 1J4, QC, Canada.
[5]DeepRender.

*Corresponding author(s). E-mail(s): vikram.voleti@gmail.com;

Abstract

Recent work has shown that Neural Ordinary Differential Equations (ODEs) can serve as generative models of images using the perspective of Continuous Normalizing Flows (CNFs). Such models offer exact likelihood calculation, and invertible generation/density estimation. In this work we introduce a Multi-Resolution variant of such models (MRCNF), by characterizing the conditional distribution over the additional information required to generate a fine image that is consistent with the coarse image. We introduce a transformation between resolutions that allows for no change in the log likelihood. We show that this approach yields comparable likelihood values for various image datasets, with improved performance at higher resolutions, with fewer parameters, using only one GPU. Further, we examine the out-of-distribution properties of MRCNFs, and find that they are similar to those of other likelihood-based generative models.

Keywords: Continuous Normalizing Flows, Image generation, wavelet

1 Introduction

Reversible generative models derived through the use of the change of variables technique [1–4] are growing in interest as alternatives to generative models based on Generative Adversarial Networks (GANs) [5] and Variational Autoencoders (VAEs) [6]. While GANs and VAEs have been able to produce visually impressive samples of images, they have a number of limitations. A change of variables approach facilitates the transformation of a simple base probability distribution into a more complex model distribution. Reversible generative models using this technique are attractive because they enable efficient density estimation, efficient sampling, and computation of exact likelihoods.

A promising variation of the change-of-variable approach is based on the use of a continuous time variant of normalizing flows [7–9], which uses an integral over continuous time dynamics to transform a base distribution into the model distribution, called Continuous Normalizing Flows (CNF). This approach uses ordinary differential equations (ODEs) specified by a neural network, or Neural ODEs. CNFs have been shown to be capable of modelling complex distributions such as those associated with images.

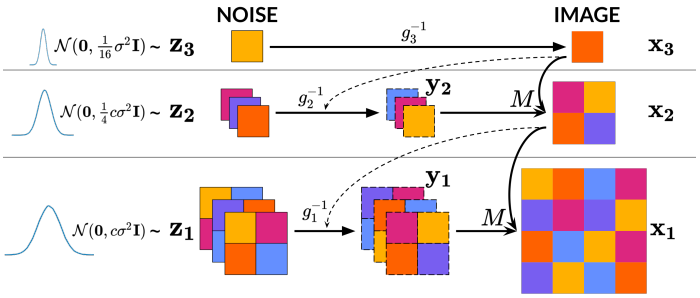


Fig. 1: The architecture of our Multi-Resolution Continuous Normalizing Flow (MRCNF) method (best viewed in color). Continuous normalizing flows (CNFs) g_s are used to generate images $\mathbf{x}_s$ from noise $\mathbf{z}_s$ at each resolution, with those at finer resolutions conditioned (dashed lines) on the coarser image one level above $\mathbf{x}_{s+1}$, except at the coarsest level where it is unconditional. Every finer CNF produces an intermediate image $\mathbf{y}_s$, which is then combined with the immediate coarser image $\mathbf{x}_{s+1}$ using a linear map M from Equation 11 to form $\mathbf{x}_s$. The multiscale maps are defined by Equation 20.

While this new paradigm for the generative modelling of images is not as mature as GANs or VAEs in terms of the generated image quality, it is a promising direction of research as it does not have some key shortcomings associated with GANs and VAEs. Specifically, GANs are known to suffer from mode-collapse [10], and are notoriously difficult to train [11] compared to feed forward networks because their adversarial loss seeks a saddle point instead of a local minimum [12]. CNFs are trained by mapping images to noise, and their reversible architecture allows images to be generated by going in reverse, from noise to images. This leads to fewer issues related to mode

collapse, since any input example in the dataset can be recovered from the flow using the reverse of the transformation learned during training. VAEs only provide a lower bound on the marginal likelihood whereas CNFs provide exact likelihoods. Despite the many advantages of reversible generative models built with CNFs, quantitatively such methods still do not match the widely used Fréchet Inception Distance (FID) scores of GANs or VAEs. However their other advantages motivate us to explore them further.

Furthermore, state-of-the art GANs and VAEs exploit the multi-resolution properties of images, and recent top-performing methods also inject noise at each resolution [13–16]. While shaping noise is fundamental to normalizing flows, only recently have normalizing flows exploited the multi-resolution properties of images. For example, WaveletFlow [4] splits an image into multiple resolutions using the Discrete Wavelet Transform, and models the average image at each resolution using a normalizing flow. While this method has advantages, it suffers from many issues such as high parameter count and long training time.

In this work, we consider a non-trivial multi-resolution approach to continuous normalizing flows, which fixes many of these issues. A high-level view of our approach is shown in Figure 1. Our main contributions are:

1. We propose a multi-resolution transformation that does not add cost in terms of likelihood.
2. We introduce **Multi-Resolution Continuous Normalizing Flows (MRCNF)**.
3. We achieve comparable Bits-per-dimension (BPD) (negative log likelihood per pixel) on image datasets using fewer model parameters and significantly less training time with only one GPU.
4. We explore the out-of-distribution properties of (MR)CNF, and find that they are similar to non-continuous normalizing flows.

2 Background

2.1 Normalizing Flows

Normalizing flows [1, 17–20] are generative models that map a complex data distribution, such as real images, to a known noise distribution. They are trained by maximizing the log likelihood of their input images. Suppose a normalizing flow g produces output $\mathbf{z}$ from an input $\mathbf{x}$ i.e. $\mathbf{z} = g(\mathbf{x})$. The change-of-variables formula provides the likelihood of the image under this transformation as:

$$\log p(\mathbf{x}) = \log \left| \det \frac{\mathrm{d}g}{\mathrm{d}\mathbf{x}} \right| + \log p(\mathbf{z}) \tag{1}$$

The first term on the right (log determinant of the Jacobian) is often intractable, however, previous works on normalizing flows have found ways to estimate this efficiently. The second term, $\log p(\mathbf{z})$, is computed as the log probability of $\mathbf{z}$ under a known noise distribution, typically the standard Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The normalizing flow is trained by maximizing the log-likelilhood of the data $\mathbf{x}$ in the real distribution i.e. $\log p(\mathbf{x})$, using Equation 1.

2.2 Continuous Normalizing Flows

Continuous Normalizing Flows (CNF) [7–9] are a variant of normalizing flows that operate in the continuous domain. A CNF creates a geometric flow between the input and target (noise) distributions, by assuming that the state transition is governed by an Ordinary Differential Equation (ODE). It further assumes that the differential function is parameterized by a neural network, this model is called a Neural ODE [7]. Suppose CNF g transforms its state $\mathbf{v}(t)$ using a Neural ODE, with the differential defined by the neural network f parameterized by θ. $\mathbf{v}(t_0) = \mathbf{x}$ is, say, an image, and at the final time step $\mathbf{v}(t_1) = \mathbf{z}$ is a sample from a known noise distribution.

$$\frac{\mathrm{d}\mathbf{v}(t)}{\mathrm{d}t} = f(\mathbf{v}(t), t, \theta) \implies \mathbf{v}(t_1) = g(\mathbf{v}(t_0)) = \mathbf{v}(t_0) + \int_{t_0}^{t_1} f(\mathbf{v}(t), t, \theta)\ \mathrm{d}t \tag{2}$$

This integration is typically performed by an ODE solver. Since this integration can be run backwards as well to obtain the same $\mathbf{v}(t_0)$ from $\mathbf{v}(t_1)$, a CNF is a reversible model. Equation 1 can be used to compute the change in log-probability induced by the CNF. However, [7] and [8] proposed a more efficient variant in the CNF context, the instantaneous change-of-variables formula:

$$\frac{\partial \log p(\mathbf{v}(t))}{\partial t} = -\mathrm{Tr}\left(\frac{\partial f_\theta}{\partial \mathbf{v}(t)}\right) \tag{3}$$

$$\implies \Delta \log p_{\mathbf{v}(t_0)\to\mathbf{v}(t_1)} = -\int_{t_0}^{t_1} \mathrm{Tr}\left(\frac{\partial f_\theta}{\partial \mathbf{v}(t)}\right) \mathrm{d}t \tag{4}$$

Hence, the change in log-probability of the state of the Neural ODE i.e. $\Delta \log p_{\mathbf{v}}$ is expressed as another differential equation. The ODE solver now solves both differential equations Equation 2 and Equation 4 by augmenting the original state with the above. Thus, a CNF provides both the final state $\mathbf{v}(t_1)$ as well as the change in log probability $\Delta \log p_{\mathbf{v}(t_0)\to\mathbf{v}(t_1)}$ together.

Prior works [8, 9, 21–23] have trained CNFs as reversible generative models of images by maximizing the image likelihood:

$$\mathbf{z} = g(\mathbf{x}) \quad ; \quad \log p(\mathbf{x}) = \Delta \log p_{\mathbf{x}\to\mathbf{z}} + \log p(\mathbf{z}) \tag{5}$$

where $\mathbf{x}$ is an image, $\mathbf{z}$ and $\Delta \log p_{\mathbf{x}\to\mathbf{z}}$ are computed by the CNF using Equation 2 and Equation 4, and $\log p(\mathbf{z})$ is the likelihood of $\mathbf{z}$ under a known noise distribution, typically the standard Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Novel images are generated by sampling $\mathbf{z}$ from the noise distribution, and running the CNF in reverse.

3 Our method

Our method is a reversible generative model of images that builds on top of CNFs. We introduce the notion of multiple resolutions in images, and connect the different resolutions in an autoregressive fashion. This helps generate images faster, with better

likelihood values at higher resolutions, using only one GPU in all our experiments. We call this model Multi-Resolution Continuous Normalizing Flow (MRCNF).

3.1 Multi-Resolution image representation

Multi-resolution representations of images have been explored in computer vision for decades [24–29]. Much of the content of an image at a resolution is a composition of low-level information captured at coarser resolutions, and high-level information not present in the coarser images. We take advantage of this by first decomposing an image in *resolution space* i.e. by expressing it as a series of S images at decreasing resolutions: $\mathbf{x} \to (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_S)$, where $\mathbf{x}_1 = \mathbf{x}$ is the finest image, $\mathbf{x}_S$ is the coarsest, and every $\mathbf{x}_{s+1}$ is the average image of $\mathbf{x}_s$. This called an image pyramid [24, 26, 27, 29, 30]. In this work, we obtain a coarser image simply by averaging pixels in every 2x2 patch, thereby halving the width and height. We then express $\mathbf{x}$ as a series of high-level information $\mathbf{y}_s$ not present in the immediate coarser images $\mathbf{x}_{s+1}$, and a final coarse image $\mathbf{x}_S$, and our overall method is to map these S terms to S noise samples using S CNFs.:

$$\mathbf{x} \to (\mathbf{y}_1, \mathbf{x}_2) \to (\mathbf{y}_1, \mathbf{y}_2, \mathbf{x}_3) \to \cdots \to (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{S-1}, \mathbf{x}_S) \tag{6}$$

3.2 Defining the high-level information $\mathbf{y}_s$

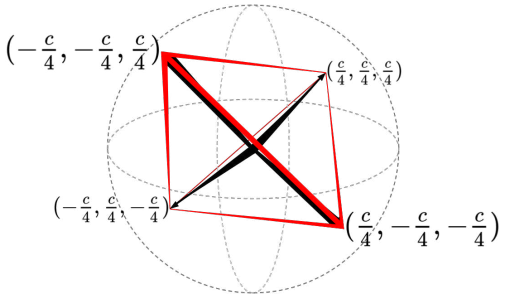


Fig. 2: Tetrahedron in 3D space with 4 corners. $c = 2^{2/3}$

We choose to design a linear transformation with the following properties: 1) invertible i.e. it should be possible to deterministically obtain $\mathbf{x}_s$ from $\mathbf{y}_s$ and $\mathbf{x}_{s+1}$, and vice versa ; 2) volume preserving i.e. determinant is 1, change in log-likelihood is 0 ; 3) angle preserving ; and 4) range preserving.

Consider the simplest case of 2 resolutions where $\mathbf{x}_1$ is a 2x2 image with pixel values x_1, x_2, x_3, x_4, and $\mathbf{x}_2$ is a 1x1 image with pixel value $\bar{x} = \frac{1}{4}(x_1 + x_2 + x_3 + x_4)$. We require three values $(y_1, y_2, y_3) = \mathbf{y}_1$ that contain information not present in $\mathbf{x}_2$, such that $\mathbf{x}_1$ is obtained when $\mathbf{y}_1$ and $\mathbf{x}_2$ are combined.

This could be viewed as a problem of finding a matrix $\mathbf{M}$ such that: $[x_1, x_2, x_3, x_4]^\top = \mathbf{M}\ [y_1, y_2, y_3, \bar{x}]^\top$. We fix the last column of $\mathbf{M}$ as $[1, 1, 1, 1]^\top$, since every pixel value in $\mathbf{x}_1$ depends on $\bar{x}$. Finding the rest of the parameters can be viewed as requiring four 3D vectors that are spaced such that they do not degenerate the

number of dimensions of their span. These can be considered as the four corners of a tetrahedron in 3D space, under any configuration (rotated in 3D space), and any scaling of the vectors (see Figure 2).

Out of the many possibilities for this tetrahedron is the matrix that performs the Discrete Haar Wavelet Transform [28, 31]:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 1 \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & 1 \\ -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & 1 \\ -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \bar{x} \end{bmatrix} \iff \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \bar{x} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \tag{7}$$

However, this transformation incurs a cost in terms of log-likelihood:

$$\Delta \log p_{(x_1,x_2,x_3,x_4)\to(y_1,y_2,y_3,\bar{x})} = \log \left|\det(\mathbf{M}^{-1})\right| = \log(1/2) \tag{8}$$

and is therefore not volume preserving. Other simple scaling of Equation 7 has been used in the past, for example multiplying the last row of Equation 7 by 2, yielding an orthogonal transformation, such as in WaveletFlow [4]. However, this transformation neither preserves the volume i.e. the log determinant is not 0, nor the maximum i.e. the range of $\mathbf{x}_s$ changes.

We wish to find a transformation $\mathbf{M}$ where: one of the results is the average of the inputs, $\bar{x}$; it is unit determinant; the columns are orthogonal; and it preserves the range of $\bar{x}$. Fortunately such a matrix exists – although we have not seen it discussed in prior literature. It can be seen as a variant of the Discrete Haar Wavelet Transformation matrix that is unimodular, i.e. has a determinant of 1 (and is therefore volume preserving), while also preserving the range of the images for the input and its average (shown in Figure 2):

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \frac{1}{a} \begin{bmatrix} c & c & c & a \\ c & -c & -c & a \\ -c & c & -c & a \\ -c & -c & c & a \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \bar{x} \end{bmatrix} \iff \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \bar{x} \end{bmatrix} = \begin{bmatrix} c^{-1} & c^{-1} & -c^{-1} & -c^{-1} \\ c^{-1} & -c^{-1} & c^{-1} & -c^{-1} \\ c^{-1} & -c^{-1} & -c^{-1} & c^{-1} \\ a^{-1} & a^{-1} & a^{-1} & a^{-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \tag{9}$$

where $c = 2^{2/3}$, $a = 4$. Hence, there is no cost to the log-likelihood due to the transformation:

$$\Delta \log p_{(x_1,x_2,x_3,x_4)\to(y_1,y_2,y_3,\bar{x})} = \log \left|\det(\mathbf{M}^{-1})\right| = \log(1) = 0 \tag{10}$$

This can be scaled up to larger spatial regions by performing the same calculation for each 2x2 patch. Let M be the function that uses matrix $\mathbf{M}$ from above and combines every pixel in $\mathbf{x}_{s+1}$ with the three corresponding pixels in $\mathbf{y}_s$ to make the 2x2 patch at that location in $\mathbf{x}_s$ using Equation 9:

$$\mathbf{x}_s = M(\mathbf{y}_s, \mathbf{x}_{s+1}) \iff \mathbf{y}_s, \mathbf{x}_{s+1} = M^{-1}(\mathbf{x}_s) \tag{11}$$

Equation 1 can be used to compute the change in log likelihood from this transformation $\mathbf{x}_s \rightarrow (\mathbf{y}_s, \mathbf{x}_{s+1})$:

$$\begin{aligned}
&\log p(\mathbf{x}_s) \\
&= \Delta \log p_{\mathbf{x}_s \rightarrow (\mathbf{y}_s, \mathbf{x}_{s+1})} + \log p(\mathbf{y}_s, \mathbf{x}_{s+1}) \\
&= \log \left|\det(M^{-1})\right| + \log p(\mathbf{y}_s, \mathbf{x}_{s+1})
\end{aligned} \tag{12}$$

where $\log \left|\det(M^{-1})\right|$ can be determined for the Wavelet transform in Equation 7 using Equation 8 as $\log \left|\det(M^{-1})\right| = \text{dims}(\mathbf{x}_{s+1}) \log(1/2)$ where "dims" is the number of pixels times the number of channels (typically 3) in the image, and for our unimodular transform in Equation 9 using Equation 10 as :

$$\log \left|\det(M^{-1})\right| = 0 \tag{13}$$

3.3 Multi-Resolution Continuous Normalizing Flows

Using the multi-resolution image representation in Equation 6, we characterize the conditional distribution over the additional degrees of freedom ($\mathbf{y}_s$) required to generate a higher resolution image ($\mathbf{x}_s$) that is consistent with the average ($\mathbf{x}_{s+1}$) over the equivalent pixel space. At each resolution s, we use a CNF to reversibly map between $\mathbf{y}_s$ (or $\mathbf{x}_S$ when $s{=}S$) and a sample $\mathbf{z}_s$ from a known noise distribution. For generation, $\mathbf{y}_s$ only adds information missing in $\mathbf{x}_{s+1}$, but conditional on it.

This framework ensures that one coarse image could generate several potential fine images, but these fine images have the same coarse image as their average. This fact is preserved across resolutions. Note that the 3 additional pixels in $\mathbf{y}_s$ per pixel in $\mathbf{x}_{s+1}$ are generated conditioned on the entire coarser image $\mathbf{x}_{s+1}$, thus maintaining consistency using the full context.

In principle, any generative model could be used to map between the multi-resolution image and noise. Normalizing flows are good candidates for this as they are probabilistic generative models that perform exact likelihood estimates, and can be run in reverse to generate novel data from the model's distribution. This allows model comparison and measurement of generalization to unseen data. We choose to use the CNF variant of normalizing flows at each resolution. CNFs have recently been shown to be effective in modeling image distributions using a fraction of the number of parameters typically used in normalizing flows (and non flow-based approaches), and their underlying framework of Neural ODEs have been shown to be more robust than convolutional layers [32].

Training: We train an MRCNF by maximizing the average log-likelihood of the images in the training dataset under the model. The log probability of each image $\log p(\mathbf{x})$ can be estimated recursively using the sequence of variables in Equation 6, and the corresponding simplification of the log-probability using Equation 12 as (here, $\mathbf{x}_1 = \mathbf{x}$):

$$\begin{aligned}
&\log p(\mathbf{x}) = \Delta \log p_{\mathbf{x}_1 \rightarrow (\mathbf{y}_1, \mathbf{x}_2)} + \log p(\mathbf{y}_1, \mathbf{x}_2) \\
&= \Delta \log p_{\mathbf{x}_1 \rightarrow (\mathbf{y}_1, \mathbf{x}_2)} + \log p(\mathbf{y}_1 \mid \mathbf{x}_2) + \log p(\mathbf{x}_2) \\
&= \Delta \log p_{\mathbf{x}_1 \rightarrow (\mathbf{y}_1, \mathbf{x}_2)} + \log p(\mathbf{y}_1 \mid \mathbf{x}_2)
\end{aligned}$$

$$+ \Delta \log p_{\mathbf{x}_2 \to (\mathbf{y}_2, \mathbf{x}_3)} + \log p(\mathbf{y}_2 \mid \mathbf{x}_3) + \log p(\mathbf{x}_3)$$

$$\vdots$$

$$= \sum_{s=1}^{S-1} \left(\Delta \log p_{\mathbf{x}_s \to (\mathbf{y}_s, \mathbf{x}_{s+1})} + \log p(\mathbf{y}_s \mid \mathbf{x}_{s+1}) \right) + \log p(\mathbf{x}_S) \tag{14}$$

where $\Delta \log p_{\mathbf{x}_s \to (\mathbf{y}_s, \mathbf{x}_{s+1})}$ is given by Equation 13, while $\log p(\mathbf{y}_s \mid \mathbf{x}_{s+1})$ and $\log p(\mathbf{x}_S)$ (at the coarsest resolution S) are given by Equation 5:

$$\mathbf{z}_s = g_s(\mathbf{y}_s \mid \mathbf{x}_{s+1}); \quad \log p(\mathbf{y}_s \mid \mathbf{x}_{s+1}) = \Delta \log p_{(\mathbf{y}_s \to \mathbf{z}_s) \mid \mathbf{x}_{s+1}} + \log p(\mathbf{z}_s) \tag{15}$$

$$\mathbf{z}_S = g_S(\mathbf{x}_S); \quad \log p(\mathbf{x}_S) = \Delta \log p_{\mathbf{x}_S \to \mathbf{z}_S} + \log p(\mathbf{z}_S) \tag{16}$$

The coarsest resolution S can be chosen such that the last CNF operates on the image distribution at a small enough resolution that is easy to model unconditionally. All other CNFs are conditioned on the immediate coarser image. The conditioning itself is achieved by concatenating the input image of the CNF with the coarser image. This model could be seen as a stack of CNFs connected in an autoregressive fashion.

Typically, likelihood-based generative models are compared using the metric of bits-per-dimension (BPD), i.e. the negative log likelihood per pixel in the image. Hence, we train our MRCNF to minimize the average BPD of the images in the training dataset, computed using Equation 17:

$$\text{BPD}(\mathbf{x}) = -\log p(\mathbf{x}) / \text{dims}(\mathbf{x}) \tag{17}$$

We use FFJORD [8] as the baseline model for our CNFs. In addition, we use to two regularization terms introduced by RNODE [9] to speed up the training of FFJORD models by stabilizing the learnt dynamics: the kinetic energy $\mathcal{K}(\theta)$ and the Jacobian norm $\mathcal{B}(\theta)$ of the flow $f(\mathbf{v}(t), t, \theta)$ described in subsection 2.2:

$$\mathcal{K}(\theta) = \int_{t_0}^{t_1} \|f(\mathbf{v}(t), t, \theta)\|_2^2 \, \mathrm{d}t \quad ; \tag{18}$$

$$\mathcal{B}(\theta) = \int_{t_0}^{t_1} \|\epsilon^\top \nabla_z f(\mathbf{v}(t), t, \theta)\|_2^2 \, \mathrm{d}t, \quad \epsilon \sim \mathcal{N}(0, I) \tag{19}$$

Parallel training: Note that although the final log likelihood $\log p(\mathbf{x})$ involves sequentially summing over values returned by all S CNFs, the log likelihood term of each CNF is independent of the others. Conditioning is done using ground truth images. Hence, each CNF can be trained independently, in parallel.

Generation: Given an S-resolution model, we first sample $\mathbf{z}_s, s = 1, \ldots, S$ from the latent noise distributions. The CNF g_s at resolution s transforms the noise sample $\mathbf{z}_s$ to high-level information $\mathbf{y}_s$ conditioned on the immediate coarse image $\mathbf{x}_{s+1}$ (except g_S which is unconditioned). $\mathbf{y}_s$ and $\mathbf{x}_{s+1}$ are then combined to form $\mathbf{x}_s$ using M from Equation 9. This process is repeated progressively from coarser to finer resolutions, until the finest resolution image $\mathbf{x}_1$ is computed (see Figure 1). It is to be noted that the

generated image at one resolution is used to condition the CNF at the finer resolution.

$$\begin{cases} \mathbf{x}_S = g_S^{-1}(\mathbf{z}_S) & s = S \\ \mathbf{y}_s = g_s^{-1}(\mathbf{z}_s \mid \mathbf{x}_{s+1}); \quad \mathbf{x}_s = M(\mathbf{y}_s, \mathbf{x}_{s+1}) & s = S\text{-}1 \rightarrow 1 \end{cases} \tag{20}$$

3.4 Multi-Resolution Noise

We further decompose the noise image as well into its respective coarser components. This means ultimately we use only one noise image at the finest level, and it is decomposed into multiple resolutions using Equation 9. $\mathbf{x}_{s+1}$ is mapped to noise of $1/4$ variance, $\mathbf{y}_s$ is mapped to noise of c-factored variance (see Figure 1). Although this is optional, it preserves interpretation between the single- and multi-resolution models.

4 Related work

Multi-resolution approaches already serve as a key component of state-of-the-art GAN [33–35] and VAE [16, 36] based deep generative models. The idea is to take advantage of the fact that much of the information in an image is contained in a coarsened version, which allows us to deal with simpler problems (coarser images) in a progressive fashion. This helps make models more efficient and effective. Deconvolutional CNNs [37, 38] use upsampling layers to generate images more effectively. Modern state-of-the-art generative models have also injected noise at different levels to improve sample quality [13, 15, 16]. Several works [36, 39–41] have also shown how the inductive bias of the multi-resolution structure helps alleviate some of the problems of image quality in likelihood-based models.

Several prior works on normalizing flows [2–4, 42–48] build on RealNVP [1]. Although they achieve great results in terms of BPD and image quality, they nonetheless report results from significantly higher number of parameters (some with 100x!), and several times GPU hours of training.

STEER [21] introduced temporal regularization to CNFs by making the final time of integration stochastic. However, we found that this increased training time without significant BPD improvement.

$$\text{STEER [21]: } \begin{cases} \mathbf{v}(t_1) = \mathbf{v}(t_0) + \int_{t_0}^{T} f(\mathbf{v}(t), t)\ \mathrm{d}t; \\ T \sim \text{Uniform}(t_1 - b, t_1 + b);\ b < t_1 - t_0 \end{cases} \tag{21}$$

"Multiple scales" in prior normalizing flows: Normalizing flows [1, 2, 8] try to be "multi-scale" by transforming the input in a smart way (squeezing operation) such that the width of the features progressively reduces in the direction of image to noise, while maintaining the total dimensions. This happens while operating at a *single resolution*. In contrast, our model stacks normalizing flows at *multiple resolutions* in an autoregressive fashion by conditioning on the images at coarser resolutions.

Other classes of generative models that map from a complex distribution to a known noise distribution are Denoising diffusion probabilistic models (DDPM) [49–51] which use a predefined noising process, and score-based generative models [52–55]

which estimate the gradient of the log density with respect to the input (i.e. the *score*) of corrupted data with progressively lesser intensities of noise. In contrast, CNFs learn a reversible noising/denoising process using a Neural ODE.

4.1 Wavelet Flow [4]

WaveletFlow is a recent innovation on the normalizing flow, wherein the image is decomposed into a lower-resolution average image, and 3 other informative components using the Discrete Wavelet Transformation. The 3 components at each resolution are mapped to noise using a normalizing flow conditioned on the average image at that resolution. WaveletFlow builds on the Glow [2] architecture. It uses an orthogonal transformation, which does not preserve range, and adds a constant term to the log likelihood at each resolution. Best results are obtained when WaveletFlow models with a high parameter count are trained for a long period of time. We fix these issues using our MRCNF.

Comparison to WaveletFlow: We emphasize that there are important and crucial differences between our MRCNF and WaveletFlow. We generalize the notion of a multi-resolution image representation (subsection 3.2), and show that Wavelets are one case of this general formulation. WaveletFlow builds on the Glow [2] architecture, while ours builds on CNFs [8, 9]. We also make use of the notion of multi-resolution decomposition of the noise, which is optional, but is not taken into account by WaveletFlow. WaveletFlow uses an orthogonal transformation which does not preserve range ; our MRCNF uses Equation 9 which is volume-preserving and range-preserving. Finally, WaveletFlow applies special sampling techniques to obtain better samples from its model. We have so far not used such techniques for generation, but we believe they can potentially help our models as well. By making these important changes, we fix many of the previously discussed issues with WaveletFlow. For a more detailed ablation study, please check subsection 5.3.

5 Experimental results

We train MRCNF models on the CIFAR10 [61] dataset at finest resolution of 32x32, and the ImageNet [62] dataset at 32x32, 64x64, 128x128. We build on top of the code provided in [9][1]. In all cases, we train using *only one* NVIDIA RTX 20280 Ti GPU with 11GB.

In Table 1, we compare our results with prior work in terms of (lower is better in all cases) the BPD of the images of the test datasets under the trained models, the number of parameters used by the model, and the number of GPU hours taken to train. The most relevant models for comparison are the 1-resolution FFJORD [8] models, and their regularized version RNODE [9], since our model directly converts their architecture into multi-resolution. Other relevant comparisons are previous flow-based methods [1–4, 44], however their core architecture (RealNVP [1]) is quite different from FFJORD.

BPD: At lower resolution spaces, we achieve comparable BPDs in lesser time with far fewer parameters than previous normalizing flows (and non flow-based approaches). However, the power of the multi-resolution formulation is more evident at higher

[1] https://github.com/cfinlay/ffjord-rnode

Table 1: Bits-per-dimension (BPD) (lower is better) of images in the corresponding evaluation sets for CIFAR10, ImageNet (32x32), and ImageNet (64x64). We also report the number of PARAMETERS (P) in the models (in millions), and the TIME taken to train (in GPU hours). All our models were trained on only one GPU. Lower is better in all cases.

	CIFAR10			IMAGENET32			IMAGENET64		
(↓)	BPD	P	TIME	BPD	P	TIME	BPD	P	TIME
Non Flow-based Prior Work									
Gated PixelCNN [56]	3.03	-	-	3.83	-	60	3.57	-	60
SPN [41]	-	-	-	3.85	150.0M	-	3.53	150.0M	-
Sparse Transformer [57]	2.80	59.0M	-	-	-	-	3.44	152.0M	7days
NVAE [16]	2.91	-	55	3.92	-	70	-	-	-
DistAug [58]	2.56	152.0M	-	-	-	-	3.42	152.0M	-
Flow-based Prior Work									
RealNVP [1]	3.49	-	-	4.28	46.0M	-	3.98	96.0M	-
Glow [2]	3.35	44.0M	-	4.09	66.1M	-	3.81	111.1M	-
MaCow [45]	3.16	43.5M	-	-	-	-	3.69	122.5M	-
Flow++ [3]	3.08	31.4M	-	3.86	169.0M	-	3.69	73.5M	-
DenseFlow [59]	2.98	-	250	3.63	-	310	3.35	-	224
1-Resolution Continuous Normalizing Flow									
FFJORD [8]	3.40	0.9M	≥5days	3.96‡	2.0M‡	>5days‡	x	x	x
RNODE [9]	3.38	1.4M	31.8	3.49§	1.6M§	40.4§	3.83*	2.0M	256.4*
FFJORD + STEER [21]	3.40	1.4M	86.3	3.84	2.0M	>5days	-	-	-
RNODE + STEER [21]	3.397	1.4M	22.2	3.49§	1.6M§	30.1§	-	-	-
(OURS) Multi-Resolution Continuous Normalizing Flow (MRCNF)									
2-resolution MRCNF	3.65	1.3M	19.8	3.77	1.3M	18.2	3.44	2.0M	42.3
2-resolution MRCNF	3.54	3.3M	36.5	3.78	6.7M	18.0	x	6.7M	x
3-resolution MRCNF	3.79	1.5M	17.4	3.97	1.5M	13.8	3.55	2.0M	35.4
3-resolution MRCNF	3.60	5.1M	38.3	3.93	10.2M	41.2	x	7.6M	x

- Unreported values.
†As reported in [60].
‡As reported in [21].
§Re-implemented by us.
'x': Fails to train.
*RNODE [9] used 4 GPUs to train on ImageNet64.

resolutions: we achieve better BPD for ImageNet64 with significantly fewer parameters and lesser time using only one GPU. A more complete table can be found in the appendix.

Train time: All our experiments used only one GPU, and took significantly less time to train than 1-resolution CNFs, and all prior works including flow-based and non-flow-based models. For example on CIFAR-10, Glow [2] used 8 GPUs for 7 days, MintNet [44] used 2 GPUs for $\approx$ 5 days, 1-resolution FFJORD [8] used 6 GPUs for $\approx$ 5 days. All our models used 1 GPU for $\leq$ 1 day.

To make a fair comparison with previous methods, we report the total time taken to train the CNFs of all resolutions one after another on a single GPU. We also maintained the batch size of the finest resolution the same as that in the previous CNF works, but

used bigger batch sizes to train coarser resolutions. However, since all the CNFs can be trained in parallel, the actual training time in practice could be much lower.

5.1 Super-resolution

Our formulation also allows for super-resolution of images (Figure 3) free of cost since our framework is autoregressive in resolution. At any stage, one can condition on a ground truth low-resolution image and generate the corresponding high-resolution image.

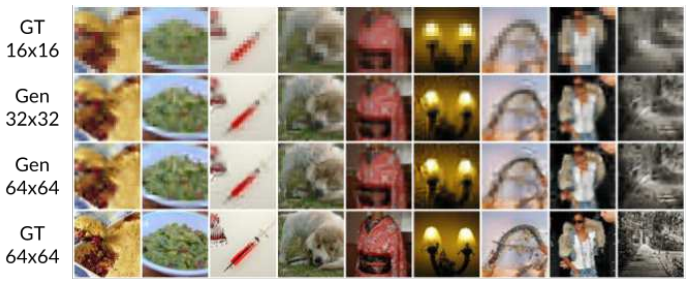


Fig. 3: ImageNet: Example of super-resolving to 64x64 from ground truth 16x16. Row 1: ground truth 16x16, Row 2: generated 32x32, Row 3: generated 64x64 Row 4: ground truth 64x64.

5.2 Progressive training

We trained an MRCNF model on ImageNet128 by training only the finest resolution (128x128) conditioned on the immediate coarser (64x64) images, and attached it to a 3-resolution model trained on ImageNet64. The resultant 4-resolution ImageNet128 model gives a BPD of 3.31 (Table 2) with just 2.74M parameters in ≈60 GPU hours.

Table 2: Metrics for unconditional ImageNet128 generation. PARAM is number of parameters, TIME is in hours. '-' indicates unreported values.

IMAGENET128 (↓)	BPD	PARAM	TIME
Parallel Multiscale [40]	3.55	-	-
SPN [41]	3.08	250.00M	-
(OURS) 4-resolution MRCNF	3.31	2.74M	58.59

5.3 Ablation study

Our MRCNF method differs from WaveletFlow in three respects:

1. we use CNFs, while WaveletFlow uses the discrete vairant of normalizing flows,

2. we use Equation 9 instead of Equation 7 as used by WaveletFlow,
3. we use multi-resolution noise.

We check the individual effects of these changes in an ablation study in Table 3, and conclude that:

1. Simply replacing the normalizing flows in WaveletFlow with CNFs does not produce the best results. It does improve the BPD and training time compared to WaveletFlow.
2. Using our unimodular transformation in Equation 9 instead of the original Wavelet Transformation of Equation 7 not only improves the BPD, it also consistently decreases training time.
3. As expected, the use of multi-resolution noise does not have a critical impact on either BPD or training time. We use it anyway so as to retain interpretation with 1-resolution models.

Table 3: Ablation study across using Wavelet in Equation 7, and multi-resolution noise formulation in subsection 3.4. P is number of parameters, TIME is in hours. Lower is better in all cases. '-' indicates unreported values. 'x' : Fails to train.

	CIFAR10			IMAGENET64		
(↓)	BPD	P	TIME	BPD	P	TIME
WaveletFlow [4]	-	-	-	3.78	98.0M	822.00
1-resolution CNF (RNODE) [9]	3.38	1.4M	31.84	3.83	2.0M	256.40
2-resolution						
eq. (7) WaveletFlow with CNF w/o multi-res noise	3.68	1.3M	27.25	x	2.0M	x
eq. (7) WaveletFlow with CNF w/ multi-res noise	3.69	1.3M	25.88	x	2.0M	x
eq. (9) MRCNF w/o multi-res noise	3.66	1.3M	19.79	3.48	2.0M	42.33
eq. (9) MRCNF w/ multi-res noise **(Ours)**	3.65	1.3M	19.69	3.44	2.0M	42.30
3-resolution						
eq. (7) WaveletFlow with CNF w/o multi-res noise	3.82	1.5M	22.99	3.62	2.0M	43.37
eq. (7) WaveletFlow with CNF w/ multi-res noise	3.82	1.5M	25.28	3.62	2.0M	44.21
eq. (9) MRCNF w/o multi-res noise	3.79	1.5M	17.25	3.57	2.0M	35.42
eq. (9) MRCNF w/ multi-res noise **(Ours)**	3.79	1.5M	17.44	3.55	2.0M	35.39

Thus, our MRCNF model is not a trivial replacement of normalizing flows with CNFs in WaveletFlow. We generalize the notion of multi-resolution image representation, in which the Discrete Wavelet Transform is one of many possibilities. We then derived a unimodular transformation that adds no change in likelihood.

5.4 Adversarial Loss

Several works [63–66] have found it useful to add an adversarial loss to pre-existing losses to generate images that better resemble the true data distribution. Similar to [64], we conducted experiments with an additional adversarial loss at each resolution. However in our experiments so far, we could achieve neither better BPDs nor better Fréchet Inception Distance (FID)s [67]. As noted in [68], since likelihood-based models tend to

cover all the modes by minimizing KL-divergence while GAN-based methods tend to mode collapse by minimizing JS-divergence, it is possible that the two approaches are incompatible, and so combining them is not trivial.

6 Examining Out-of-Distribution behaviour

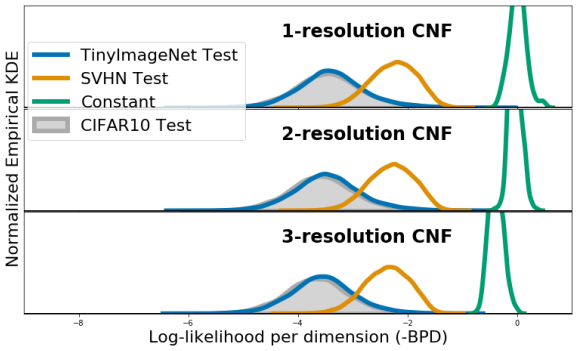


Fig. 4: Histogram of log likelihood per dimension i.e. −BPD (estimated using normalized empirical Kernel Density Estimation) of OoD datasets (TinyImageNet, SVHN, Constant) under (MR)CNF models trained on CIFAR10. As with other likelihood-based generative models such as Glow & PixelCNN, OoD datasets have higher likelihood under (MR)CNFs.

The derivation of likelihood-based models suggests that the density of an image under the model is an effective measure of its likelihood of being in-distribution. However, recent works [68–71] have pointed out that it is possible that images drawn from other distributions have higher model likelihood. Examples have been shown where normalizing flow models (such as Glow) trained on CIFAR10 images assign higher likelihood to SVHN [72] images. This could have serious implications on their practical applicability. Some also note that likelihood-based models do not generate images with good sample quality as they avoid assigning small probability to out-of-distribution (OoD) data points, hence model likelihood (-BPD) is not effective for detecting OoD data in such models.

We conduct the same experiments with (MR)CNFs, and find that similar conclusions can be drawn. Figure 4 plots the histogram of log likelihood per dimension (-BPD) of OoD images (SVHN, TinyImageNet) under MRCNF models trained on CIFAR10. It can be observed that the likelihood of the OoD SVHN is higher than CIFAR10 for MRCNF, similar to the findings for Glow, PixelCNN, VAE in earlier works [69–71, 73, 74].

One possible explanation put forward by [71] is that "typical" images are less "likely" than constant images, which is a consequence of the distribution of a Gaussian in high dimensions. Indeed, as our Figure 4 shows, constant images have the highest likelihood under MRCNFs, while randomly generated (uniformly distributed) pixels have the least likelihood (not shown in figure due to space constraints).

[71, 73] suggest using "typicality" as a better measure of OoD. However, [70] observe that the complexity of an image plays a significant role in the training of likelihood-based generative models. They propose a new metric S as an out-of-distribution detector:

$$S(\mathbf{x}) = \text{BPD}(\mathbf{x}) - L(\mathbf{x}) \tag{22}$$

where $L(\mathbf{x})$ is the complexity of an image $\mathbf{x}$ measured as the length of the best compressed version of $\mathbf{x}$ (we use FLIF [75] following [70]) normalized by the number of dimensions.

We perform a similar analysis as [70] to test how S compares with -bpd for OoD detection. For different MRCNF models trained on CIFAR10, we compute the area under the receiver operating characteristic curve (auROC) using -bpd and S as standard evaluation for the OoD detection task [70, 76]. Table 4 shows that S does perform better than -bpd in the case of (MR)CNFs, similar to the findings in [70] for Glow and PixelCNN++. SVHN seems easier to detect as OoD for Glow than MRCNFs. However, OoD detection performance is about the same for TinyImageNet. We also observe that MRCNFs are better at OoD than CNFs.

Other OoD methods [76–81] are not suitable, as identified in [70].

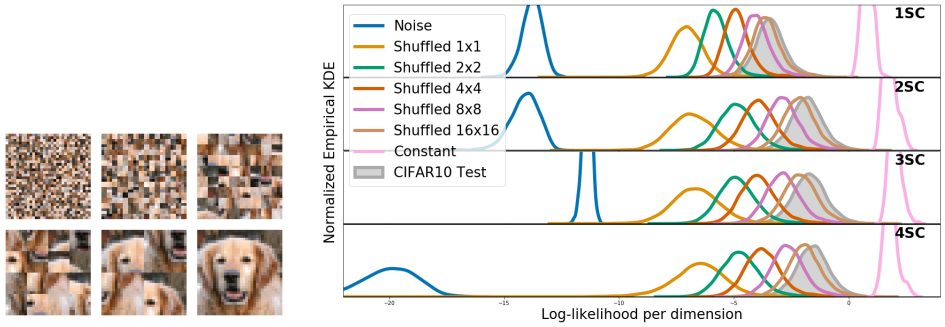
Table 4: auROC for OoD detection using -bpd and S[70], for models trained on CIFAR10.

CIFAR10	SVHN		TIN	
(trained)	-BPD	S	-BPD	S
Glow	0.08	0.95	0.66	0.72
1-res CNF	0.07	0.16	0.48	0.60
2-res MRCNF	0.06	0.25	0.46	0.66
3-res MRCNF	0.05	0.25	0.46	0.66

6.1 Shuffled in-distribution images

Prior work [74] concludes that normalizing flows do not represent images based on their semantic contents, but rather directly encode their visual appearance. We verify this for continuous normalizing flows by estimating the density of in-distribution test images, but with patches of pixels randomly shuffled. Figure 5 (a) shows an example of images of shuffled patches of varying size, Figure 5 (b) shows the graph of the their log-likelihoods.

That shuffling pixel patches would render the image semantically meaningless is reflected in the FID between CIFAR10-Train and these sets of shuffled images — 1x1: 340.42, 2x2: 299.99, 4x4: 235.22, 8x8: 101.36, 16x16: 33.06, 32x32 (i.e. CIFAR10-Test): 3.15. However, we see that images with large pixel patches shuffled are quite close in likelihood to the unshuffled images (Figure 5 (b)), suggesting that since their visual content has not changed much they are almost as likely as unshuffled images under MRCNFs.



(a) (b)

Fig. 5: (a) Example of shuffling different-sized patches of a 32x32 image: (left to right, top to bottom) 1x1, 2x2, 4x4, 8x8, 16x16, 32x32 (unshuffled) (b) Histogram of log likelihood per dimension (normalized empirical Kernel Density Estimate) for MRCNF models at different resolutions, trained on CIFAR10.

7 Conclusion

We have presented a Multi-Resolution approach to Continuous Normalizing Flows (MRCNF). MRCNF models achieve comparable or better performance in significantly less training time, training on a single GPU, with a fraction of the number of parameters of other competitive models. Although the likelihood values for 32x32 resolution datasets such as CIFAR10 and ImageNet32 do not improve over the baseline, ImageNet64 and above see a marked improvement. The performance is better for higher resolutions, as seen in the case of ImageNet128. We also conducted an ablation study to note the effects of each change we introduced in the formulation.

In addition, we show that (Multi-Resolution) Continuous Normalizing Flows have similar out-of-distribution properties as other Normalizing Flows.

Acknowledgments. Chris Finlay contributed to this paper while a postdoc at McGill University; he is now affiliated with Deep Render. His postdoc was funded in part by a Healthy Brains Healthy Lives Fellowship. Adam Oberman was supported by the Air Force Office of Scientific Research under award number FA9550-18-1-0167 and by IVADO. Christopher Pal is funded in part by CIFAR. We thank CIFAR for their support through the CIFAR AI Chairs program. We also thank Samsung for partially supporting Vikram Voleti for this work. We thank Adam Ibrahim, Etienne Denis, Gauthier Gidel, Ioannis Mitliagkas, and Roger Girgis for their valuable feedback.

Declarations

Compliance with ethical standards:. Not applicable

Funding. Chris Finlay contributed to this paper while a postdoc at McGill University, funded in part by a Healthy Brains Healthy Lives Fellowship. Adam Oberman was

supported by the Air Force Office of Scientific Research under award number FA9550-18-1-0167 and by IVADO. Christopher Pal is funded in part by CIFAR. We thank CIFAR for their support through the CIFAR AI Chairs program.

Author Contribution:. This project began when Vikram Voleti contacted Chris Finlay, then a PhD candidate at McGill University with Prof. Adam Oberman. Chris Finlay had earlier worked on a publication that improved the dynamics of Neural ODEs for image generation, and Vikram had worked on a project that used Neural ODEs for video generation. Vikram Voleti and Chris Finlay brainstormed over ideas for improving image generation using the continuous normalizing flows framework of Neural ODEs. Adam Oberman and Christopher Pal provided advice and guidance throughout the project and wrote parts of the paper. With help from Adam Oberman and Christopher Pal, Vikram derived the mathematical framework. With help from Chris Finlay, Vikram designed the experiments, wrote the code, ran experiments, proposed and executed on out-of-distribution analysis, and wrote the paper.

Appendix A Full Table 1

Table A1 presents the full version of Table 1 including other results relevant to the conclusion but not mentioned in the main paper for brevity.

Appendix B Qualitative samples

Here we present qualitative examples of our method for the datasets of MNIST and CIFAR10.

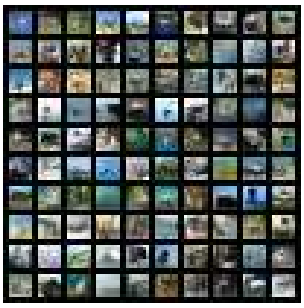
(a) Generated samples at 16x16

(b) Corresponding generated samples at 32x32

Fig. B1: Generated samples from MNIST.

Table A1: Unconditional image generation metrics (lower is better in all cases): parameters in the model, bits-per-dimension, time (in hours). All our models were trained on only *one* NVIDIA V100 GPU. ‡As reported in [21]. *used 4 GPUs. 'x': Fails to train.

	CIFAR10			IMAGENET32			IMAGENET64		
	BPD	P	TIME	BPD	P	TIME	BPD	P	TIME
Non Flow-based Prior Work									
PixelRNN [39]	3.00			3.86			3.63		
Gated PixelCNN [56]	3.03			3.83		60	3.57		60
Parallel Multiscale [40]				3.95			3.70		
Image Transformer [82]	2.90			3.77					
PixelSNAIL [83]	2.85			3.80					
SPN [41]				3.85	150.0M		3.53	150.0M	
Sparse Transformer [57]	2.80	59.0M					3.44	152.0M	7days
Axial Transformer [84]				3.76			3.44		
PixelFlow++ [85]	2.92								
NVAE [16]	2.91		55	3.92		70			
Dist-Aug Sparse Tx [58]	2.56	152.0M					3.42	152.0M	
Flow-based Prior Work									
IAF [86]				3.11					
RealNVP [1]	3.49			4.28	46.0M		3.98	96.0M	
Glow [2]	3.35	44.0M		4.09	66.1M		3.81	111.1M	
i-ResNets [87]									
Emerging [42]	3.34	44.7M		4.09	67.1M		3.81	67.1M	
IDF [43]	3.34			4.18			3.90		
S-CONF [88]	3.34								
MintNet [44]	3.32	17.9M	≥5days	4.06	17.4M				
Residual Flow [60]	3.28			4.01			3.76		
MaCow [45]	3.16	43.5M					3.69	122.5M	
Neural Spline Flows [46]	3.38	11.8M					3.82	15.6M	
Flow++ [3]	3.08	31.4M		3.86	169.0M		3.69	73.5M	
ANF [89]	3.05			3.92			3.66		
MEF [90]	3.32	37.7M		4.05	37.7M		3.73	46.6M	
VFlow [47]	2.98			3.83					
Woodbury NF [91]	3.47			4.20			3.87		
NanoFlow [48]	3.25								
ConvExp [92]	3.218								
Wavelet Flow [4]				4.08	64.0M		3.78	96.0M	822
TayNODE [93]	1.039								
1-resolution Continuous Normalizing Flow									
FFJORD [8]	3.40	0.9M	≥5days	‡3.96	‡2.0M	‡>5days	x		x
RNODE [9]	3.38	1.4M	31.84	‡2.36 §3.49	2.0M §1.6M	‡30.1 §40.39	*3.83	2.0M	*256.4
FFJORD + STEER [21]	3.40	1.4M	86.34	3.84	2.0M	>5days			
RNODE + STEER [21]	3.397	1.4M	22.24	2.35 §3.49	2.0M §1.6M	24.90 §30.07			
(OURS) Multi-Resolution Continuous Normalizing Flow (MRCNF)									
2-resolution MRCNF	3.65	1.3M	19.79	3.77	1.3M	18.18	3.44	2.0M	42.30
2-resolution MRCNF	3.54	3.3M	36.47	3.78	6.7M	17.98	x	6.7M	x
3-resolution MRCNF	3.79	1.5M	17.44	3.97	1.5M	13.78	3.55	2.0M	35.39
3-resolution MRCNF	3.60	5.1M	38.27	3.93	10.2M	41.20	x	7.6M	x



(a) Generated samples 8x8

(b) Generated samples 16x16

(c) Generated samples 32x32

Fig. B2: Generated samples from CIFAR10.

Appendix C Simple example of density estimation

For example, if we use Euler method as our ODE solver, for density estimation Equation 2 reduces to:

$$\mathbf{v}(t_1) = \mathbf{v}(t_0) + (t_1 - t_0) f_s(\mathbf{v}(t_0), t_0 \mid \mathbf{c}) \tag{C1}$$

where f_s is a neural network, t_0 represents the "time" at which the state is image $\mathbf{x}$, and t_1 is when the state is noise $\mathbf{z}$. We start at scale S with an image sample $\mathbf{x}_S$, and assume t_0 and t_1 are 0 and 1 respectively:

$$\begin{cases} \mathbf{z}_S = \mathbf{x}_S + f_S(\mathbf{x}_S,\ t_0 \mid \mathbf{x}_{S-1}) \\ \mathbf{z}_{S-1} = \mathbf{x}_{S-1} + f_{S-1}(\mathbf{x}_{S-1},\ t_0 \mid \mathbf{x}_{S-2}) \\ \vdots \\ \mathbf{z}_1 = \mathbf{x}_1 + f_1(\mathbf{x}_1,\ t_0 \mid \mathbf{x}_0) \\ \mathbf{z}_0 = \mathbf{x}_0 + f_0(\mathbf{x}_0,\ t_0) \end{cases} \tag{C2}$$

Appendix D Simple example of generation

For example, if we use Euler method as our ODE solver, for generation Equation 2 reduces to:

$$\mathbf{v}(t_0) = \mathbf{v}(t_1) + (t_0 - t_1) f_s(\mathbf{v}(t_1), t_1 \mid \mathbf{c}) \tag{D3}$$

i.e. the state is integrated backwards from t_1 (i.e. $\mathbf{z}_s$) to t_0 (i.e. $\mathbf{x}_s$). We start at scale 0 with a noise sample $\mathbf{z}_0$, and assume t_0 and t_1 are 0 and 1 respectively:

$$\begin{cases} \mathbf{x}_0 = \mathbf{z}_0 - f_0(\mathbf{z}_0,\ t_1) \\ \mathbf{x}_1 = \mathbf{z}_1 - f_1(\mathbf{z}_1,\ t_1 \mid \mathbf{x}_0) \\ \vdots \\ \mathbf{x}_{S-1} = \mathbf{z}_{S-1} - f_{S-1}(\mathbf{z}_{S-1},\ t_1 \mid \mathbf{x}_{S-2}) \\ \mathbf{x}_S = \mathbf{z}_S - f_S(\mathbf{z}_S,\ t_1 \mid \mathbf{x}_{S-1}) \end{cases} \tag{D4}$$

Appendix E Models

We used the same neural network architecture as in RNODE [9]. The CNF at each resolution consists of a stack of *bl* blocks of a 4-layer deep convolutional network comprised of 3x3 kernels and softplus activation functions, with 64 hidden dimensions, and time t concatenated to the spatial input. In addition, except at the coarsest resolution, the immediate coarser image is also concatenated with the state. The integration time of each piece is [0, 1]. The number of blocks *bl* and the corresponding total number of parameters are given in Table E2.

Table E2: Number of parameters for different models with different total number of resolutions (res), and the number of channels (ch) and number of blocks (bl) per resolution.

MRCNF

resolutions	ch	bl	Param
1	64	2	0.16M
	64	4	0.32M
	64	14	1.10M
2	64	8	1.33M
	64	20	3.34M
	64	40	6.68M
3	64	6	1.53M
	64	8	2.04M
	64	20	5.10M

Appendix F Gradient norm

In order to avoid exploding gradients, We clipped the norm of the gradients [94] by a maximum value of 100.0. In case of using adversarial loss, we first clip the

gradients provided by the adversarial loss by 50.0, sum up the gradients provided by the log-likelihood loss, and then clip the summed gradients by 100.0.

Appendix G 8-bit to uniform

The change-of-variables formula gives the change in probability due to the transformation of $\mathbf{u}$ to $\mathbf{v}$:

$$\log p(\mathbf{u}) = \log p(\mathbf{v}) + \log \left| \det \frac{\mathrm{d}\mathbf{v}}{\mathrm{d}\mathbf{u}} \right|$$

Specifically, the change of variables from an 8-bit image to an image with pixel values in range [0, 1] is:

$$\begin{aligned}
\mathbf{b}_S^{(p)} &= \frac{\mathbf{a}_S^{(p)}}{256} \\
&\implies \log p(\mathbf{a}_S) = \log p(\mathbf{b}_S) + \log \left| \det \frac{\mathrm{d}\mathbf{b}}{\mathrm{d}\mathbf{a}} \right| \\
&\implies \log p(\mathbf{a}_S) = \log p(\mathbf{b}_S) + \log \left(\frac{1}{256} \right)^{D_S} \\
&\implies \log p(\mathbf{a}_S) = \log p(\mathbf{b}_S) - D_S \log 256 \\
\implies \mathrm{bpd}(\mathbf{a}_S) &= \frac{-\log p(\mathbf{a}_S)}{D_S \log 2} \\
&= \frac{-(\log p(\mathbf{b}_S) - D_S \log 256)}{D_S \log 2} \\
&= \frac{-\log p(\mathbf{b}_S)}{D_S \log 2} + \frac{\log 256}{\log 2} \\
&= \mathrm{bpd}(\mathbf{x}) + 8
\end{aligned}$$

where $\mathrm{bpd}(\mathbf{x})$ is given from Equation 17.

Appendix H FID v/s Temperature

Table H3 lists the FID values of generated images from MRCNF models trained on CIFAR10, with different temperature settings on the Gaussian.

Conflict of Interest (COI) statement. Conflict of Interest: The authors declare that they have no conflict of interest.

data availability statement (DAS). All data generated or analysed during this study are included in their respective published articles, as mentioned in the main draft: CIFAR10 [61], ImageNet [62]

	Temperature					
	1.0	**0.9**	**0.8**	**0.7**	**0.6**	**0.5**
1-resolution CNF	138.82	147.62	175.93	284.75	405.34	466.16
2-resolution MRCNF	89.55	106.21	171.53	261.64	370.38	435.17
3-resolution MRCNF	88.51	104.39	152.82	232.53	301.89	329.12
4-resolution MRCNF	92.19	104.35	135.58	186.71	250.39	313.39

Table H3: FID v/s temperature for MRCNF models trained on CIFAR10.

References

[1] Dinh, L., Sohl-Dickstein, J., Bengio, S.: Density estimation using real nvp. In: International Conference on Learned Representations (2017)

[2] Kingma, D.P., Dhariwal, P.: Glow: Generative flow with invertible 1x1 convolutions. In: Advances in Neural Information Processing Systems, pp. 10215–10224 (2018)

[3] Ho, J., Chen, X., Srinivas, A., Duan, Y., Abbeel, P.: Flow++: Improving flow-based generative models with variational dequantization and architecture design. In: International Conference on Machine Learning (2019)

[4] Yu, J., Derpanis, K., Brubaker, M.: Wavelet flow: Fast training of high resolution normalizing flows. In: Advances in Neural Information Processing Systems (2020)

[5] Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y.: Deep learning **1** (2016)

[6] Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)

[7] Chen, R.T.Q., Rubanova, Y., Bettencourt, J., Duvenaud, D.: Neural ordinary differential equations. Advances in Neural Information Processing Systems (2018)

[8] Grathwohl, W., Chen, R.T.Q., Bettencourt, J., Sutskever, I., Duvenaud, D.: Ffjord: Free-form continuous dynamics for scalable reversible generative models. International Conference on Learning Representations (2019)

[9] Finlay, C., Jacobsen, J.-H., Nurbekyan, L., Oberman, A.: How to train your neural ode: the world of jacobian and kinetic regularization. International Conference on Machine Learning (2020)

[10] Lin, Z., Khetan, A., Fanti, G., Oh, S.: Pacgan: The power of two samples in generative adversarial networks. In: Advances in Neural Information Processing Systems, pp. 1498–1507 (2018)

[11] Arjovsky, M., Bottou, L.: Towards principled methods for training generative adversarial networks. arXiv preprint arXiv:1701.04862 (2017)

[12] Berard, H., Gidel, G., Almahairi, A., Vincent, P., Lacoste-Julien, S.: A closer look at the optimization landscapes of generative adversarial networks. In: International Conference on Machine Learning (2020)

[13] Brock, A., Donahue, J., Simonyan, K.: Large scale GAN training for high fidelity natural image synthesis. In: International Conference on Learning Representations (2019)

[14] Shaham, T.R., Dekel, T., Michaeli, T.: Singan: Learning a generative model from a single natural image. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 4570–4580 (2019)

[15] Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8110–8119 (2020)

[16] Vahdat, A., Kautz, J.: Nvae: A deep hierarchical variational autoencoder. In: Advances in Neural Information Processing Systems (2020)

[17] Tabak, E.G., Turner, C.V.: A family of nonparametric density estimation algorithms. Communications on Pure and Applied Mathematics **66**(2), 145–164 (2013)

[18] Jimenez Rezende, D., Mohamed, S.: Variational inference with normalizing flows. In: International Conference on Machine Learning, pp. 1530–1538 (2015)

[19] Papamakarios, G., Nalisnick, E., Rezende, D.J., Mohamed, S., Lakshminarayanan, B.: Normalizing flows for probabilistic modeling and inference. arXiv preprint arXiv:1912.02762 (2019)

[20] Kobyzev, I., Prince, S., Brubaker, M.: Normalizing flows: An introduction and review of current methods. IEEE Transactions on Pattern Analysis and Machine Intelligence (2020)

[21] Ghosh, A., Behl, H.S., Dupont, E., Torr, P.H., Namboodiri, V.: Steer: Simple temporal regularization for neural odes. In: Advances in Neural Information Processing Systems (2020)

[22] Onken, D., Fung, S.W., Li, X., Ruthotto, L.: Ot-flow: Fast and accurate continuous normalizing flows via optimal transport. AAAI Conference on Artificial Intelligence (2021)

[23] Huang, H.-H., Yeh, M.-Y.: Accelerating continuous normalizing flow with trajectory polynomial regularization. AAAI Conference on Artificial Intelligence (2021)

[24] Burt, P.J.: Fast filter transform for image processing. Computer graphics and image processing **16**(1), 20–51 (1981)

[25] Marr, D.: Vision: A computational investigation into the human representation and processing of visual information (2010)

[26] Witkin, A.P.: Scale-space filtering, 329–332 (1987)

[27] Burt, P., Adelson, E.: The laplacian pyramid as a compact image code. IEEE Transactions on communications **31**(4), 532–540 (1983)

[28] Mallat, S.G.: A theory for multiresolution signal decomposition: the wavelet representation. IEEE transactions on pattern analysis and machine intelligence **11**(7), 674–693 (1989)

[29] Lindeberg, T.: Scale-space for discrete signals. IEEE transactions on pattern analysis and machine intelligence **12**(3), 234–254 (1990)

[30] Adelson, E.H., Anderson, C.H., Bergen, J.R., Burt, P.J., Ogden, J.M.: Pyramid methods in image processing. RCA engineer **29**(6), 33–41 (1984)

[31] Mallat, S.G., Peyré, G.: A Wavelet Tour of Signal Processing: the Sparse Way, (2009)

[32] Yan, H., Du, J., Tan, V.Y.F., Feng, J.: On robustness of neural ordinary differential equations. International Conference on Learning Representations (2020)

[33] Denton, E.L., Chintala, S., Fergus, R., *et al.*: Deep generative image models using a laplacian pyramid of adversarial networks. In: Advances in Neural Information Processing Systems, pp. 1486–1494 (2015)

[34] Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation. In: International Conference on Learned Representations (2018)

[35] Karnewar, A., Wang, O.: Msg-gan: Multi-scale gradients for generative adversarial networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7799–7808 (2020)

[36] Razavi, A., van den Oord, A., Vinyals, O.: Generating diverse high-fidelity images with vq-vae-2. In: Advances in Neural Information Processing Systems, pp. 14866–14876 (2019)

[37] Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3431–3440 (2015)

[38] Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)

[39] Oord, A.v.d., Kalchbrenner, N., Kavukcuoglu, K.: Pixel recurrent neural networks. International Conference on Machine Learning (2016)

[40] Reed, S., Oord, A.v.d., Kalchbrenner, N., Colmenarejo, S.G., Wang, Z., Belov, D., De Freitas, N.: Parallel multiscale autoregressive density estimation. In: International Conference on Machine Learning (2017)

[41] Menick, J., Kalchbrenner, N.: Generating high fidelity images with subscale pixel networks and multidimensional upscaling. In: International Conference on Learning Representations (2019)

[42] Hoogeboom, E., Berg, R.v.d., Welling, M.: Emerging convolutions for generative normalizing flows. In: International Conference on Machine Learning (2019)

[43] Hoogeboom, E., Peters, J., van den Berg, R., Welling, M.: Integer discrete flows and lossless compression. In: Advances in Neural Information Processing Systems, vol. 32, pp. 12134–12144 (2019). https://proceedings.neurips.cc/paper/2019/file/9e9a30b74c49d07d8150c8c83b1ccf07-Paper.pdf

[44] Song, Y., Meng, C., Ermon, S.: Mintnet: Building invertible neural networks with masked convolutions. In: Advances in Neural Information Processing Systems, pp. 11004–11014 (2019)

[45] Ma, X., Kong, X., Zhang, S., Hovy, E.: Macow: Masked convolutional generative flow. In: Advances in Neural Information Processing Systems, pp. 5893–5902 (2019)

[46] Durkan, C., Bekasov, A., Murray, I., Papamakarios, G.: Neural spline flows. In: Advances in Neural Information Processing Systems, vol. 32, pp. 7511–7522 (2019). https://proceedings.neurips.cc/paper/2019/file/7ac71d433f282034e088473244df8c02-Paper.pdf

[47] Chen, J., Lu, C., Chenli, B., Zhu, J., Tian, T.: Vflow: More expressive generative flows with variational data augmentation. In: International Conference on Machine Learning (2020)

[48] Lee, S.-g., Kim, S., Yoon, S.: Nanoflow: Scalable normalizing flows with sublinear parameter complexity. In: Advances in Neural Information Processing Systems (2020)

[49] Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., Ganguli, S.: Deep unsupervised learning using nonequilibrium thermodynamics. In: International Conference on Machine Learning, pp. 2256–2265 (2015). PMLR

[50] Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Advances in Neural Information Processing Systems (2020)

[51] Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. International Conference on Learning Representations (2020)

[52] Song, Y., Ermon, S.: Generative modeling by estimating gradients of the data distribution. Advances in Neural Information Processing Systems (2019)

[53] Song, Y., Ermon, S.: Improved techniques for training score-based generative models. Advances in Neural Information Processing Systems (2020)

[54] Jolicoeur-Martineau, A., Piché-Taillefer, R., Combes, R.T.d., Mitliagkas, I.: Adversarial score matching and improved sampling for image generation. International Conference on Learning Representations (2021)

[55] Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., Poole, B.: Score-based generative modeling through stochastic differential equations. International Conference on Learning Representations (2021)

[56] 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 (2016)

[57] Child, R., Gray, S., Radford, A., Sutskever, I.: Generating long sequences with sparse transformers. arXiv preprint arXiv:1904.10509 (2019)

[58] Jun, H., Child, R., Chen, M., Schulman, J., Ramesh, A., Radford, A., Sutskever, I.: Distribution augmentation for generative modeling. In: International Conference on Machine Learning, pp. 10563–10576 (2020)

[59] Grcić, M., Grubišić, I., Šegvić, S.: Densely connected normalizing flows. arXiv preprint (2021)

[60] Chen, R.T., Behrmann, J., Duvenaud, D.K., Jacobsen, J.-H.: Residual flows for invertible generative modeling. In: Advances in Neural Information Processing Systems, pp. 9916–9926 (2019)

[61] Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. Technical Report, University of Toronto (2009)

[62] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255 (2009). IEEE

[63] Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., Frey, B.: Adversarial autoencoders. arXiv preprint arXiv:1511.05644 (2015)

[64] Grover, A., Dhar, M., Ermon, S.: Flow-gan: Combining maximum likelihood and adversarial learning in generative models. In: AAAI Conference on Artificial Intelligence (2018)

[65] Lee, A.X., Zhang, R., Ebert, F., Abbeel, P., Finn, C., Levine, S.: Stochastic adversarial video prediction. ArXiv **abs/1804.01523** (2018)

[66] Beckham, C., Honari, S., Verma, V., Lamb, A.M., Ghadiri, F., Hjelm, R.D., Bengio, Y., Pal, C.: On adversarial mixup resynthesis. In: Advances in Neural Information Processing Systems, pp. 4346–4357 (2019)

[67] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: Advances in Neural Information Processing Systems, pp. 6626–6637 (2017)

[68] Theis, L., Oord, A.v.d., Bethge, M.: A note on the evaluation of generative models. In: International Conference on Learning Representations (2016)

[69] Nalisnick, E., Matsukawa, A., Teh, Y.W., Gorur, D., Lakshminarayanan, B.: Do deep generative models know what they don't know? In: International Conference on Learning Representations (2019)

[70] Serrà, J., Álvarez, D., Gómez, V., Slizovskaia, O., Núñez, J.F., Luque, J.: Input complexity and out-of-distribution detection with likelihood-based generative models. In: International Conference on Learning Representations (2020)

[71] Nalisnick, E., Matsukawa, A., Teh, Y.W., Lakshminarayanan, B.: Detecting out-of-distribution inputs to deep generative models using a test for typicality. arXiv preprint arXiv:1906.02994 **5** (2019)

[72] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. NIPS Workshop on Deep Learning and Unsupervised Feature Learning (2011)

[73] Choi, H., Jang, E., Alemi, A.A.: Waic, but why? generative ensembles for robust anomaly detection. arXiv preprint arXiv:1810.01392 (2018)

[74] Kirichenko, P., Izmailov, P., Wilson, A.G.: Why normalizing flows fail to detect out-of-distribution data. In: Advances in Neural Information Processing Systems, vol. 33 (2020)

[75] Sneyers, J., Wuille, P.: Flif: Free lossless image format based on maniac compression. In: 2016 IEEE International Conference on Image Processing (ICIP), pp. 66–70 (2016). IEEE

[76] Hendrycks, D., Mazeika, M., Dietterich, T.: Deep anomaly detection with outlier exposure. In: International Conference on Learning Representations (2019)

[77] Hendrycks, D., Gimpel, K.: A baseline for detecting misclassified and out-of-distribution examples in neural networks. In: International Conference on Learning Representations (2017)

[78] Liang, S., Li, Y., Srikant, R.: Enhancing the reliability of out-of-distribution image detection in neural networks. In: International Conference on Learning Representations (2018)

[79] Lee, K., Lee, K., Lee, H., Shin, J.: A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In: Advances in Neural Information Processing Systems, pp. 7167–7177 (2018)

[80] Sabeti, E., Høst-Madsen, A.: Data discovery and anomaly detection using atypicality for real-valued data. Entropy **21**(3), 219 (2019)

[81] Høst-Madsen, A., Sabeti, E., Walton, C.: Data discovery and anomaly detection using atypicality: Theory. IEEE Transactions on Information Theory **65**(9), 5302–5322 (2019)

[82] Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, Ł., Shazeer, N., Ku, A., Tran, D.: Image transformer. In: International Conference on Machine Learning (2018)

[83] Chen, X., Mishra, N., Rohaninejad, M., Abbeel, P.: Pixelsnail: An improved autoregressive generative model. In: International Conference on Machine Learning, pp. 864–872 (2018). PMLR

[84] Ho, J., Kalchbrenner, N., Weissenborn, D., Salimans, T.: Axial attention in multidimensional transformers. arXiv preprint arXiv:1912.12180 (2019)

[85] Nielsen, D., Winther, O.: Closing the dequantization gap: Pixelcnn as a single-layer flow. In: Advances in Neural Information Processing Systems (2020)

[86] Kingma, D.P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., Welling, M.: Improving variational inference with inverse autoregressive flow (2016). cite arxiv:1606.04934

[87] Behrmann, J., Grathwohl, W., Chen, R.T., Duvenaud, D., Jacobsen, J.-H.: Invertible residual networks. In: International Conference on Machine Learning, pp. 573–582 (2019)

[88] Karami, M., Schuurmans, D., Sohl-Dickstein, J., Dinh, L., Duckworth, D.: Invertible convolutional flow. In: Advances in Neural Information Processing Systems, vol. 32, pp. 5635–5645 (2019). https://proceedings.neurips.cc/paper/2019/file/b1f62fa99de9f27a048344d55c5ef7a6-Paper.pdf

[89] Huang, C.-W., Dinh, L., Courville, A.: Augmented normalizing flows: Bridging the gap between generative flows and latent variable models. arXiv preprint arXiv:2002.07101 (2020)

[90] Xiao, C., Liu, L.: Generative flows with matrix exponential. In: International Conference on Machine Learning (2020)

[91] Lu, Y., Huang, B.: Woodbury transformations for deep generative flows. In: Advances in Neural Information Processing Systems (2020)

[92] Hoogeboom, E., , Satorras, V.G., Tomczak, J., Welling, M.: The convolution exponential and generalized sylvester flows. In: Advances in Neural Information Processing Systems (2020)

[93] Kelly, J., Bettencourt, J., Johnson, M.J., Duvenaud, D.: Learning differential equations that are easy to solve. In: Advances in Neural Information Processing Systems (2020)

[94] Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. In: International Conference on Machine Learning, pp. 1310–1318 (2013)