Multi-Resolution Continuous Normalizing Flows

Anonymous Author(s)
Affiliation
Address
email

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 1 GPU.

1 Introduction

Reversible generative models derived through the use of the change of variables technique [15, 37, 23, 80] are growing in interest as alternatives to generative models based on Generative Adversarial Networks (GANs) [19] and Variational Autoencoders (VAEs) [36]. 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.

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 [4, 64, 35, 74]. While shaping noise is fundamental to normalizing flows, only recently have normalizing flows exploited the multi-resolution properties of images, using wavelets [80]. In this work, we consider a non-trivial multi-resolution approach to normalizing flows, which we find performs better than the corresponding wavelet approach.

A promising variation of the change-of-variable approach is based on the use of a continuous time variant of normalizing flows [8, 20], 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. 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 [44], and are notoriously difficult to train [2] compared to feed forward networks because their adversarial loss seeks a saddle point instead of a local minimum [3]. CNFs are trained by mapping images to noise and their reversible architecture allows images to be

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

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 transforming the input into the latent space 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.

In this work, we focus on making the training of continuous normalizing flows feasible for higher resolution images, and help reduce computation time. Our hypothesis is that a multi-resolution representation will work better at higher resolutions. We thus introduce a novel multi-resolution technique for continuous normalizing flows, by modelling the conditional distribution of high-level information at each resolution in an autoregressive fashion. We show that this makes the models perform better at higher resolutions. A high-level view of our approach is shown in Figure 1. Our main contributions are:

1. We introduce **Multi-Resolution Continuous Normalizing Flows (MRCF)**, through which we achieve state-of-the-art Bits-per-dimension (BPD) (negative log likelihood per pixel) on ImageNet64 using fewer model parameters relative to comparable methods.
2. We propose a multi-resolution transformation that does not add cost in terms of likelihood.
3. We explore the out-of-distribution (OoD) properties of continuous normalizing flows, and find that they are quite similar to those of other likelihood-based generative models.

2 Background

2.1 Normalizing Flows

Normalizing flows [72, 30, 15, 57, 39] 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 $\mathcal{N}$.

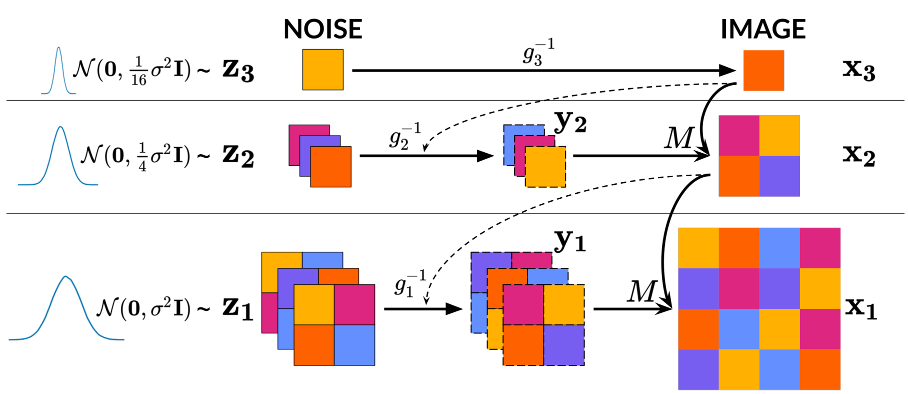


Figure 1: The architecture of our Multi-Resolution Continuous Normalizing Flow (MRCNF) method (best viewed in color). Continuous normalizing flows (CNFs) g_s are used to generate images $\mathbf{x}_s$ from noise $\mathbf{z}_s$ at each resolution, with those at finer resolutions conditioned (dashed lines) 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 eq. (8) to form $\mathbf{x}_s$. The multiscale maps are defined by eq. (16).

2.2 Continuous Normalizing Flows

Continuous Normalizing Flows (CNF) [8, 20, 17] are a variant of normalizing flows that operate in the continuous domain, using the framework of Neural ODEs [8]. 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). If the differential function is parameterized by a neural network, this model is called a Neural ODE. Suppose CNF g transforms its state $\mathbf{v}(t)$ using a Neural ODE [8] with neural network f defining the differential. Here, $\mathbf{v}(t_0) = \mathbf{x}$ is, say, an image, and at the final time step $\mathbf{v}(t_1) = \mathbf{z}$ is a sample from a known noise distribution.

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

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, Chen et al. [8], Grathwohl et al. [20] have proposed a more efficient method in the context of CNFs, called the instantaneous variant of the change-of-variables formula:

$$\frac{\partial \log p(\mathbf{v}(t))}{\partial t} = -\mathrm{Tr}\left(\frac{\partial f}{\partial \mathbf{v}(t)}\right) \implies \Delta \log p_{\mathbf{v}(t_0)\to\mathbf{v}(t_1)} = -\int_{t_0}^{t_1} \mathrm{Tr}\left(\frac{\partial f}{\partial \mathbf{v}(t)}\right) \mathrm{d}t \tag{3}$$

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 eq. (2) and eq. (3) 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 [20, 17, 18, 55, 29] have trained CNFs as reversible generative models of images, by maximizing the likelihood of the images under the model:

$$\begin{cases} \mathbf{z} = g(\mathbf{x}) \\ \log p(\mathbf{x}) = \Delta \log p_{\mathbf{x}\to\mathbf{z}} + \log p(\mathbf{z}) \end{cases} \tag{4}$$

where $\mathbf{x}$ is an image, $\mathbf{z}$ and $\Delta \log p_{\mathbf{x}\to\mathbf{z}}$ are computed by the CNF using eq. (2) and eq. (3), and $\log p(\mathbf{z})$ is the likelihood of the computed $\mathbf{z}$ under a known noise distribution, typically the standard Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$. CNF g is trained by maximizing $\mathbb{E}_{\mathbf{x}} \log p(\mathbf{x})$. Novel images are generated by sampling $\mathbf{z}$ from the known noise distribution, and running it through 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. Moreover, we used 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 [6, 50, 77, 5, 48, 45]. This implies that much of the content of an image at a resolution is a composition of low-level information captured at coarser resolutions, and some high-level information not present in the coarser images. We take advantage of this property 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, or a Gaussian Pyramid if the upsampling-downsampling operations include a Gaussian filter [6, 5, 1, 77, 45]. In this work, we obtain a coarser image simply by averaging pixels in every 2×2 patch, thereby halving the width and height.

However, this representation is redundant since much of the information in $\mathbf{x}_1$ is contained in $\mathbf{x}_{s>1}$. Instead, we 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 $\mathbf{x}_S$:

$$\mathbf{x} \to (\mathbf{y}_1, \mathbf{x}_2) \to (\mathbf{y}_1, \mathbf{y}_2, \mathbf{x}_3) \to \cdots \to (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{S-1}, \mathbf{x}_S) \tag{5}$$

Our overall method is to map these S terms to S noise samples using S CNFs.

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

The multi-resolution representation in eq. (5) needs to be 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. Further, it is preferable that this transformation incurs minimal additional computational cost, and does not add too much change in terms of log-likelihood. We choose to perform a linear transformation taking into account the following properties: 1) volume preserving i.e. determinant is 1, 2) angle preserving, and 3) range preserving (respecting the maximum principle, studied for some time, under the notion of *the maximum principle* [76]).

Consider the simplest case of 2 resolutions where $\mathbf{x}_1$ is a 2×2 image with pixel values x_1, x_2, x_3, x_4, and $\mathbf{x}_2$ is a 1×1 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 when they are combined with $\mathbf{x}_2$, $\mathbf{x}_1$ is obtained.

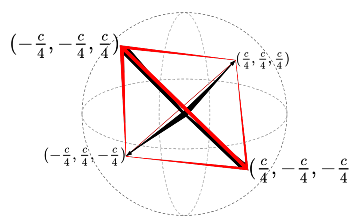


Figure 2: Tetrahedron in 3D space with 4 corners

This could be viewed as a problem of finding a matrix $\boldsymbol{M}$ such that: $[x_1, x_2, x_3, x_4]^\top = \boldsymbol{M}[y_1, y_2, y_3, \bar{x}]^\top$. We fix the last column of $\boldsymbol{M}$ 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 (ideally) non-trivially equally spaced. 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.

Out of the many possibilities for this tetrahedron, we could choose the matrix that performs the Discrete Haar Wavelet Transform [48, 49]:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 1 \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & 1 \\ -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & 1 \\ -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \bar{x} \end{bmatrix} \iff \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \bar{x} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \tag{6}$$

However, this has $\log\left|\det(\boldsymbol{M}^{-1})\right| = \log(1/2)$ (6), and is therefore not volume preserving. Other simple scaling of (6) has been used in the past, for example multiplying the last row of (6) by 2, yielding an orthogonal transformation. However, this transformation does not preserve the maximum (i.e. the range changes).

We wish to find a transformation $\boldsymbol{M}$ where: one of the results is the average, $\bar{x}$, of the inputs; the columns are orthogonal; it is unit determinant, and which also preserves the range of $\bar{x}$. Fortunately such a matrix exists – although we have not seen it discussed in the 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 (therefore volume preserving), while also preserving the range of the images for the input and its average:

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

where $c = 2^{2/3}$, and $\log\left|\det(\boldsymbol{M}^{-1})\right| = \log(1) = 0$. This can be scaled up to larger spatial regions by performing the same calculation for each 2×2 patch. Let M be the function that uses matrix $\boldsymbol{M}$ from above and combines every pixel in $\mathbf{x}_{s+1}$ with 3 corresponding pixels in $\mathbf{y}_s$ to make the 2×2 patch at that location in $\mathbf{x}_s$ using eq. (7):

$$\mathbf{x}_s = M(\mathbf{y}_s, \mathbf{x}_{s+1}) \iff \mathbf{y}_s, \mathbf{x}_{s+1} = M^{-1}(\mathbf{x}_s) \tag{8}$$

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

$$\log p(\mathbf{x}_s) = \Delta \log p_{\mathbf{x}_s \to (\mathbf{y}_s, \mathbf{x}_{s+1})} + \log p(\mathbf{y}_s, \mathbf{x}_{s+1}) \tag{9}$$

$$\Delta \log p_{\mathbf{x}_s \to (\mathbf{y}_s, \mathbf{x}_{s+1})} = \log\left|\det(M^{-1})\right| \tag{10}$$

where $\log\left|\det(M^{-1})\right| = \text{dims}(\mathbf{x}_{s+1}) \log(1/2)$ in the case of eq. (6), where "dims" is the number of pixels times the number of channels (typically 3) in the image, while it is 0 for eq. (7).

3.3 Multi-Resolution Continuous Normalizing Flows

Using the multi-resolution image representation in eq. (5), 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. At generation, $\mathbf{y}_s$ only adds 3 degrees of freedom to $\mathbf{x}_{s+1}$, which contain 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 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 [79].

Training: We train an MRCNF by maximizing the average log-likelihood of the images in the training dataset under the model, i.e. $\max \mathbb{E}_{\mathbf{x}} \log p(\mathbf{x})$. The log probability of each image $\log p(\mathbf{x})$ can be estimated recursively from eq. (9) as:

$$\begin{aligned}\log p(\mathbf{x}) &= \Delta \log p_{\mathbf{x}_1 \to (\mathbf{y}_1, \mathbf{x}_2)} + \log p(\mathbf{y}_1, \mathbf{x}_2) \\ &= \Delta \log p_{\mathbf{x}_1 \to (\mathbf{y}_1, \mathbf{x}_2)} + \log p(\mathbf{y}_1 \mid \mathbf{x}_2) + \log p(\mathbf{x}_2) \\ &= \sum_{s=1}^{S-1} \left(\Delta \log p_{\mathbf{x}_s \to (\mathbf{y}_s, \mathbf{x}_{s+1})} + \log p(\mathbf{y}_s \mid \mathbf{x}_{s+1}) \right) + \log p(\mathbf{x}_S) \end{aligned} \tag{11}$$

where $\Delta \log p_{\mathbf{x}_s \to (\mathbf{y}_s, \mathbf{x}_{s+1})}$ is given by eq. (10), $\log p(\mathbf{x}_S)$ is computed by CNF g_S using eq. (4):

$$\begin{cases} \mathbf{z}_S = g_S(\mathbf{x}_S) \\ \log p(\mathbf{x}_S) = \Delta \log p_{\mathbf{x}_S \to \mathbf{z}_S} + \log p(\mathbf{z}_S) \end{cases} \tag{12}$$

and $\log p(\mathbf{y}_s \mid \mathbf{x}_{s+1})$ is also computed by CNFs g_s similarly, conditioning on the coarser image:

$$\begin{cases} \mathbf{z}_s = g_s(\mathbf{y}_s \mid \mathbf{x}_{s+1}) \\ \log p(\mathbf{y}_s \mid \mathbf{x}_{s+1}) = \Delta \log p_{(\mathbf{y}_s \to \mathbf{z}_s) \mid \mathbf{x}_{s+1}} + \log p(\mathbf{z}_s) \end{cases} \tag{13}$$

S can be chosen such that the last CNF operates on the image distribution at a small resolution that is easy enough to model unconditionally. The rest of the CNFs are all conditioned on the immediate 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:

$$\text{BPD}(\mathbf{x}) = \frac{-\log p(\mathbf{x})}{\text{dims}(\mathbf{x})} \tag{14}$$

Hence, we train our MRCNF to minimize the average BPD of the images in the training dataset, computed using eq. (14). 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. Hence, each CNF can be trained independently, in parallel.

We use FFJORD [20] as the baseline model for our CNFs. In addition, we use to two regularization terms introduced by RNODE [17] to speed up the training of FFJORD models by stabilizing the learnt dynamics: the kinetic energy of the flow $\mathcal{K}(\theta)$, and the Jacobian norm $\mathcal{B}(\theta)$:

$$\mathcal{K}(\theta) = \int_{t_0}^{t_1} \|f(\mathbf{v}(t), t, \theta)\|_2^2 \, \mathrm{d}t; \qquad \mathcal{B}(\theta) = \int_{t_0}^{t_1} \|\epsilon^\top \nabla_z f(\mathbf{v}(t), t, \theta)\|_2^2 \, \mathrm{d}t, \quad \epsilon \sim \mathcal{N}(0, I) \tag{15}$$

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 $\mathbf{x}_s$ as described in section 3.2 (see fig. 1). This process is repeated progressively from coarser to finer resolutions, until the finest resolution image $\mathbf{x}_1$ is computed:

$$\begin{aligned} &\mathbf{x}_S = g_S^{-1}(\mathbf{z}_S) && s = S \\ &\begin{cases} \mathbf{y}_s = g_s^{-1}(\mathbf{z}_s \mid \mathbf{x}_{s+1}) \\ \mathbf{x}_s = M(\mathbf{y}_s, \mathbf{x}_{s+1}) \end{cases} && s = S\text{-}1 \to 1 \end{aligned} \tag{16}$$

Multi-Resolution Noise: We further decompose the noise image as well into its respective coarser and high-level components. Using eq. (7), assuming the finest level noise is a Gaussian, this leads to the average image $\bar{x}$ mapping to noise of a quarter variance, while $\mathbf{y}$ mapped to noise of variance factored by c.

4 Related work

Multi-resolution approaches already serve as a key component of state-of-the-art GAN [14, 34, 33] and VAE [60, 74] based deep generative models. Deconvolutional CNNs [46, 59] use upsampling layers to generate images more effectively. Modern state-of-the-art generative models have also injected noise at different levels of the hierarchy to improve sample quality [4, 35, 74]. Several works [56, 61, 51, 60] 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 [37, 26, 27, 70, 47, 16, 7, 23, 42, 80] build on RealNVP [15]. Although they achieve great results in terms of BPD and image quality, they nonetheless report results from significantly higher parameter (some with 100x!), and several times GPU hours of training.

Our MRCNF model is similar to the recently published WaveletFlow [80], which uses the Wavelet representation and maps to noise using a normalizing flow. We emphasize that we generalize the notion of a multi-resolution image representation, and show that Wavelets are one case of this general formulation. WaveletFlow builds on the Glow [37] architecture, while ours builds on CNFs [20, 17]. 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 claims to have orthonormal transformation, but its code has the orthogonal transformation which does not preserve range, our eq. (7) is different from both. 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.

STEER [18] 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.

Other classes of generative models that map from a complex distribution to a known noise distribution are Denoising diffusion probabilistic models (DDPM) [66, 25, 67] which use a predefined noising process, and score-based generative models [68, 69, 31, 71] 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.

"Multiple scales" in prior normalizing flows: Normalizing flows [15, 37, 20] 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.

5 Experimental results

We train MRCNF and Multi-Resolution Continuous Normalizing Flow - Wavelet (MRCNF-Wavelet) models on the CIFAR10 [40] dataset at finest resolution of 32x32, and the ImageNet [13] dataset at

	CIFAR10			ImageNet32			ImageNet64		
	BPD	Param	Time	BPD	Param	Time	BPD	Param	Time
Non Flow-based Prior Work									
PixelRNN [56]	3.00			3.86			3.63		
Gated PixelCNN [75]	3.03			3.83		60	3.57		60
Parallel Multiscale [61]				3.95			3.70		
Image Transformer [58]	2.90			3.77					
PixelSNAIL [10]	2.85			3.80					
SPN [51]				3.85	150M		3.53	150M	
Sparse Transformer [11]	2.80	59M					3.44	152M	7days
Axial Transformer [24]				3.76			3.44		
PixelFlow++ [54]	2.92								
NVAE [74]	2.91		55	3.92		70			
DistAug [32]	2.56	152M					3.42	152M	
Flow-based Prior Work									
RealNVP [15]	3.49			4.28	46.0M		3.98	96.0M	
Glow [37]	3.35	44.0M		4.09	66.1M		3.81	111.1M	
MintNet [70]	3.32	17.9M	$\geq$5days	4.06	17.4M				
Residual Flow [9]	3.28			4.01			3.76		
MaCow [47]	3.16	43.5M					3.69	122.5M	
Neural Spline Flows [16]	3.38	11.8M					3.82	15.6M	
Flow++ [23]	3.08	31.4M		3.86	169M		3.69	73.5M	
MEF [78]	3.32	37.7M		4.05	37.7M		3.73	46.6M	
Wavelet Flow [80]				4.08	64M		3.78	96M	822
1-Resolution Continuous Normalizing Flow									
FFJORD [20]	3.40	0.88M	$\geq$5days	3.96‡	2.00M‡	>5days‡	x		x
RNODE [17]	3.38	1.36M	31.84	2.36‡ 3.49§	2.00M 1.58M§	30.1‡ 40.39§	3.83*	2.00M	64.1*
FFJORD + STEER [18]	3.40	1.36M	86.34	3.84	2.00M	>5days			
RNODE + STEER [18]	3.397	1.36M	22.24	2.35 3.49§	2.00M 1.58M§	24.9 30.07§			
(Ours) Multi-Resolution Continuous Normalizing Flow (MRCNF)									
2-resolution MRCNF	3.65 $\pm$0.62	1.33M	19.79	3.77$\pm$0.74	1.33M	18.18	-	-	-
2-resolution MRCNF	3.54$\pm$0.64	3.34M	36.47	3.78$\pm$0.71	6.68M	17.98	-	-	-
3-resolution MRCNF	3.79$\pm$0.60	1.53M	17.44	3.97$\pm$0.70	1.53M	13.78	3.61$\pm$0.71	2.04M	28.64
3-resolution MRCNF	3.60$\pm$0.63	5.10M	38.27	-	-	-	-	-	-

Table 1: Bits-per-dimension (lower is better) of images in the corresponding evaluation sets for CIFAR10, ImageNet at 32×32, and ImageNet at 64×64, reported as the mean and standard deviation across the dataset. We also report the number of parameters in the models, and the time taken to train (in GPU hours). Most previous models use multiple GPUs for training, all our models were trained on only *one* GPU: NVIDIA RTX 2080 Ti with 11GB.
‡As reported in [18]. §Re-implemented by us. 'x': Fails to train. Blank spaces indicate unreported values. *RNODE [17] used 4 GPUs to train on ImageNet64.

32x32, 64x64, 128x128. We build on top of the code provided in [17][1]. In all cases, we train using *only one* NVIDIA V100 GPU with 16GB.

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 time taken to train. The most relevant models for comparison are the 1-resolution FFJORD [20] models, and their regularized version RNODE [17], since our model directly converts their architecture into multi-resolution. Other relevant comparisons are previous flow-based methods [15, 37, 70, 23, 80], however their core architecture (RealNVP [15]) is quite different from FFJORD.

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 resolutions: we achieve state-of-the-art BPD for ImageNet64 with significantly fewer parameters and lower time using only one GPU.

Although each CNF can be trained independently (potentially on separate GPUs), we train our models on a single GPU from coarser to finer resolutions, and report the total number of GPU hours taken.

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

We find that the multi-resolution framework eases the burden of modelling the image space in entirety, instead choosing to model the incremental changes from one resolution to the next, which results in faster training. Other non flow-based methods typically report their training time as several days using multiple GPUs. In contrast, all our experiments used only 1 GPU, and took $\approx$1 day to train (depending on the number of parameters). We observed that the BPD was almost saturated after ≈ 20 hours, however the BPD could go lower given more time.

Progressive training: Since each resolution can be trained independently, we train an MRCNF model on ImageNet128 by training only the finest resolution (128×128) conditioned on the immediate coarser (64×64) images, and attach that to a 3-resolution model trained on 64×64. The resultant 4-resolution ImageNet128 model gives a BPD of **3.31** (Table 2) with just 2.74M parameters about 60 GPU hours of total training time.

IMAGENET128	BPD	PARAM	TIME
Parallel Multiscale [61]		3.55	
SPN [51]	3.08	250M	
(OURS) 4-resolution MRCNF	3.3 $\pm$0.69	2.74M	58.59

Table 2: Metrics for unconditional ImageNet128 generation.

Our formulation also allows for super-resolution of images (Figure 3). This comes free of cost since our framework is autoregressive in resolution space. We shall include more qualitative results in the appendix.

6 Examining Out-of-Distribution behaviour

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 [73, 52, 63, 53] 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 images. This could have serious implications on the practical applicability of these models. 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 using model likelihood (-BPD) for detecting OoD data is not effective.

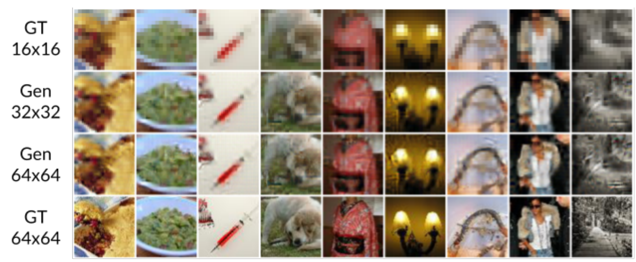


Figure 3: ImageNet: Example of super-resolving from ground truth 16×16 to 64×64. Top ground truth, middle generated, bottom ground truth.

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 [52, 12, 63, 53, 38].

One possible explanation put forward by Nalisnick et al. [53] 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).

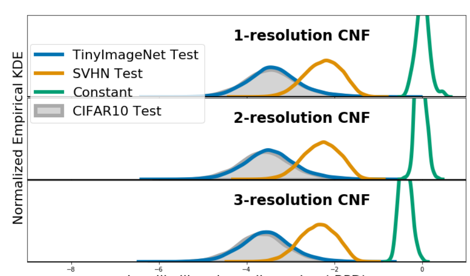


Figure 4: Histogram of log likelihood per dimension (-BPD) of out-of-distribution datasets (TinyImageNet Test, SVHN Test, and Constant images) under MRCNF models trained on CIFAR10. OoD datasets such as SVHN have higher likelihood, a property shared with other likelihood-based generative models such as Glow, PixelCNN.

Choi et al. [12], Nalisnick et al. [53] suggest using "typicality" as a better measure of OoD. However, Serrà et al. [63] 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:

$$S(\mathbf{x}) = \text{bpd}(\mathbf{x}) - L(\mathbf{x}) \tag{17}$$

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 [65] following Serrà et al. [63]) normalized by the number of dimensions.

We perform a similar analysis as Serrà et al. [63] 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 [22, 63]. Table 3 shows that S does perform better than -bpd in the case of (MR)CNFs, similar to the findings in Serrà et al. [63] for Glow and PixelCNN++. Other OoD methods [21, 43, 41, 62, 28, 22] are not as suitable in our case, as identified in Serrà et al. [63].

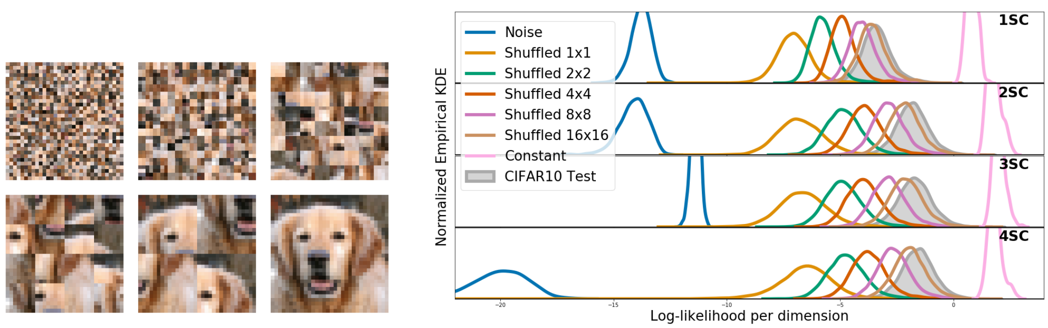
	CIFAR 10			
	SVHN		TIN	
	-bpd	S	-bpd	S
1-res	0.07	0.16	0.48	0.60
2-res	0.06	0.25	0.46	0.66
3-res	0.05	0.25	0.46	0.66

Table 3: auROC for OoD detection using -bpd and S[63]

6.1 Shuffled in-distribution images

Kirichenko et al. [38] conclude 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 Fréchet Inception Distance (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 are quite close in likelihood to the unshuffled images, suggesting that since their visual content has not changed much they are almost as likely as unshuffled images under MRCNFs.



(a) (b)

Figure 5: (a) Example of shuffling different-sized patches of a 32×32 image: (left to right, top to bottom) 1×1, 2×2, 4×4, 8×8, 16×16, 32×32 (unshuffled) (b) Bits-per-dim vs Epoch at each resolution for different MRCNF models trained on CIFAR10.

7 Conclusion

We have presented a Multi-Resolution approach to Continuous Normalizing Flows, which provides an efficient framework for exact likelihood calculations on several datasets of images by training on a single GPU in lesser time with a fraction of the number of parameters of other competitive models. Although the likelihood values for 32×32 resolution datasets such as CIFAR10 and ImageNet32 do not improve over the baseline, ImageNet64 and above see a marked improvement. In addition, we show that Continuous Normalizing Flows have similar out-of-distribution properties as other Normalizing Flows. In terms of broader social impacts of this work, generative models of images can be used to generated so-called fake images and this issue has been discussed at length in other work.

References

[1] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden, “Pyramid methods in image processing,” *RCA engineer*, vol. 29, no. 6, pp. 33–41, 1984.

[2] M. Arjovsky and L. Bottou, “Towards principled methods for training generative adversarial networks,” *arXiv preprint arXiv:1701.04862*, 2017.

[3] H. Berard, G. Gidel, A. Almahairi, P. Vincent, and S. Lacoste-Julien, “A closer look at the optimization landscapes of generative adversarial networks,” in *International Conference on Machine Learning*, 2020.

[4] A. Brock, J. Donahue, and K. Simonyan, “Large scale GAN training for high fidelity natural image synthesis,” in *International Conference on Learning Representations*, 2019.

[5] P. Burt and E. Adelson, “The laplacian pyramid as a compact image code,” *IEEE Transactions on communications*, vol. 31, no. 4, pp. 532–540, 1983.

[6] P. J. Burt, “Fast filter transform for image processing,” *Computer graphics and image processing*, vol. 16, no. 1, pp. 20–51, 1981.

[7] J. Chen, C. Lu, B. Chenli, J. Zhu, and T. Tian, “Vflow: More expressive generative flows with variational data augmentation,” in *International Conference on Machine Learning*, 2020.

[8] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, “Neural ordinary differential equations,” *Advances in Neural Information Processing Systems*, 2018.

[9] R. T. Chen, J. Behrmann, D. K. Duvenaud, and J.-H. Jacobsen, “Residual flows for invertible generative modeling,” in *Advances in Neural Information Processing Systems*, 2019, pp. 9916–9926.

[10] X. Chen, N. Mishra, M. Rohaninejad, and P. Abbeel, “Pixelsnail: An improved autoregressive generative model,” in *International Conference on Machine Learning*. PMLR, 2018, pp. 864–872.

[11] R. Child, S. Gray, A. Radford, and I. Sutskever, “Generating long sequences with sparse transformers,” *arXiv preprint arXiv:1904.10509*, 2019.

[12] H. Choi, E. Jang, and A. A. Alemi, “Waic, but why? generative ensembles for robust anomaly detection,” *arXiv preprint arXiv:1810.01392*, 2018.

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*. IEEE, 2009, pp. 248–255.

[14] E. L. Denton, S. Chintala, R. Fergus *et al.*, “Deep generative image models using a laplacian pyramid of adversarial networks,” in *Advances in neural information processing systems*, 2015, pp. 1486–1494.

[15] L. Dinh, J. Sohl-Dickstein, and S. Bengio, “Density estimation using real nvp,” in *International Conference on Learned Representations*, 2017.

[16] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios, “Neural spline flows,” in *Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 7511–7522. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/7ac71d433f282034e088473244df8c02-Paper.pdf

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

[18] A. Ghosh, H. S. Behl, E. Dupont, P. H. Torr, and V. Namboodiri, “Steer: Simple temporal regularization for neural odes,” in *Advances in Neural Information Processing Systems*, 2020.

[19] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT Press, 2016, vol. 1.

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

[21] D. Hendrycks and K. Gimpel, “A baseline for detecting misclassified and out-of-distribution examples in neural networks,” in *International Conference on Learning Representations*, 2017.

[22] D. Hendrycks, M. Mazeika, and T. Dietterich, “Deep anomaly detection with outlier exposure,” in *International Conference on Learning Representations*, 2019.

[23] J. Ho, X. Chen, A. Srinivas, Y. Duan, and P. Abbeel, “Flow++: Improving flow-based generative models with variational dequantization and architecture design,” in *International Conference on Machine Learning*, 2019.

[24] J. Ho, N. Kalchbrenner, D. Weissenborn, and T. Salimans, “Axial attention in multidimensional transformers,” *arXiv preprint arXiv:1912.12180*, 2019.

[25] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” *Advances in Neural Information Processing Systems*, 2020.

[26] E. Hoogeboom, R. v. d. Berg, and M. Welling, “Emerging convolutions for generative normalizing flows,” in *International Conference on Machine Learning*, 2019.

[27] E. Hoogeboom, J. Peters, R. van den Berg, and M. Welling, “Integer discrete flows and lossless compression,” in *Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 12 134–12 144. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/9e9a30b74c49d07d8150c8c83b1ccf07-Paper.pdf

[28] A. Høst-Madsen, E. Sabeti, and C. Walton, “Data discovery and anomaly detection using atypicality: Theory,” *IEEE Transactions on Information Theory*, vol. 65, no. 9, pp. 5302–5322, 2019.

[29] H.-H. Huang and M.-Y. Yeh, “Accelerating continuous normalizing flow with trajectory polynomial regularization,” *AAAI Conference on Artificial Intelligence*, 2021.

[30] D. Jimenez Rezende and S. Mohamed, “Variational inference with normalizing flows,” in *International Conference on Machine Learning*, 2015, pp. 1530—-1538.

[31] A. Jolicoeur-Martineau, R. Piché-Taillefer, R. T. d. Combes, and I. Mitliagkas, “Adversarial score matching and improved sampling for image generation,” *International Conference on Learning Representations*, 2021. [Online]. Available: arXivpreprintarXiv:2009.05475

[32] H. Jun, R. Child, M. Chen, J. Schulman, A. Ramesh, A. Radford, and I. Sutskever, “Distribution augmentation for generative modeling,” in *International Conference on Machine Learning*, 2020, pp. 10 563–10 576.

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

[34] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive growing of gans for improved quality, stability, and variation,” in *International Conference on Learned Representations*, 2018.

[35] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, “Analyzing and improving the image quality of stylegan,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8110–8119.

[36] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.

[37] D. P. Kingma and P. Dhariwal, “Glow: Generative flow with invertible 1x1 convolutions,” in *Advances in neural information processing systems*, 2018, pp. 10 215–10 224.

[38] P. Kirichenko, P. Izmailov, and A. G. Wilson, “Why normalizing flows fail to detect out-of-distribution data,” in *Advances in neural information processing systems*, vol. 33, 2020.

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

[40] A. Krizhevsky, G. Hinton *et al.*, “Learning multiple layers of features from tiny images,” *Technical Report, University of Toronto*, 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html

[41] K. Lee, K. Lee, H. Lee, and J. Shin, “A simple unified framework for detecting out-of-distribution samples and adversarial attacks,” in *Advances in Neural Information Processing Systems*, 2018, pp. 7167–7177.

[42] S.-g. Lee, S. Kim, and S. Yoon, “Nanoflow: Scalable normalizing flows with sublinear parameter complexity,” in *Advances in Neural Information Processing Systems*, 2020.

[43] S. Liang, Y. Li, and R. Srikant, “Enhancing the reliability of out-of-distribution image detection in neural networks,” in *International Conference on Learning Representations*, 2018.

[44] Z. Lin, A. Khetan, G. Fanti, and S. Oh, “Pacgan: The power of two samples in generative adversarial networks,” in *Advances in neural information processing systems*, 2018, pp. 1498–1507.

[45] T. Lindeberg, “Scale-space for discrete signals,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 12, no. 3, pp. 234–254, 1990.

[46] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[47] X. Ma, X. Kong, S. Zhang, and E. Hovy, “Macow: Masked convolutional generative flow,” in *Advances in Neural Information Processing Systems*, 2019, pp. 5893–5902.

[48] S. G. Mallat, “A theory for multiresolution signal decomposition: the wavelet representation,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 11, no. 7, pp. 674–693, 1989.

[49] S. G. Mallat and G. Peyré, *A wavelet tour of signal processing: the sparse way*. Elsevier, 2009.

[50] D. Marr, *Vision: A computational investigation into the human representation and processing of visual information*. MIT press, 2010.

[51] J. Menick and N. Kalchbrenner, “Generating high fidelity images with subscale pixel networks and multidimensional upscaling,” in *International Conference on Learning Representations*, 2019.

[52] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan, "Do deep generative models know what they don't know?" in *International Conference on Learning Representations*, 2019.

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

[54] D. Nielsen and O. Winther, "Closing the dequantization gap: Pixelcnn as a single-layer flow," in *Advances in Neural Information Processing Systems*, 2020.

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

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

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

[58] N. Parmar, A. Vaswani, J. Uszkoreit, Ł. Kaiser, N. Shazeer, A. Ku, and D. Tran, "Image transformer," in *International Conference on Machine Learning*, 2018.

[59] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[60] A. Razavi, A. van den Oord, and O. Vinyals, "Generating diverse high-fidelity images with vq-vae-2," in *Advances in Neural Information Processing Systems*, 2019, pp. 14 866–14 876.

[61] S. Reed, A. v. d. Oord, N. Kalchbrenner, S. G. Colmenarejo, Z. Wang, D. Belov, and N. De Freitas, "Parallel multiscale autoregressive density estimation," in *International Conference on Machine Learning*, 2017.

[62] E. Sabeti and A. Høst-Madsen, "Data discovery and anomaly detection using atypicality for real-valued data," *Entropy*, vol. 21, no. 3, p. 219, 2019.

[63] J. Serrà, D. Álvarez, V. Gómez, O. Slizovskaia, J. F. Núñez, and J. Luque, "Input complexity and out-of-distribution detection with likelihood-based generative models," in *International Conference on Learning Representations*, 2020.

[64] T. R. Shaham, T. Dekel, and T. Michaeli, "Singan: Learning a generative model from a single natural image," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 4570–4580.

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

[66] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International Conference on Machine Learning*. PMLR, 2015, pp. 2256–2265.

[67] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *International Conference on Learning Representations*, 2020. [Online]. Available: arXivpreprintarXiv:2010.02502

[68] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," *Advances in Neural Information Processing Systems*, 2019.

[69] ——, "Improved techniques for training score-based generative models," *Advances in Neural Information Processing Systems*, 2020.

[70] Y. Song, C. Meng, and S. Ermon, "Mintnet: Building invertible neural networks with masked convolutions," in *Advances in Neural Information Processing Systems*, 2019, pp. 11 004–11 014.

[71] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," *International Conference on Learning Representations*, 2021. [Online]. Available: arXivpreprintarXiv:2011.13456

[72] E. G. Tabak and C. V. Turner, "A family of nonparametric density estimation algorithms," *Communications on Pure and Applied Mathematics*, vol. 66, no. 2, pp. 145–164, 2013.

[73] L. Theis, A. v. d. Oord, and M. Bethge, "A note on the evaluation of generative models," in *International Conference on Learning Representations*, 2016.

[74] A. Vahdat and J. Kautz, "Nvae: A deep hierarchical variational autoencoder," in *Advances in Neural Information Processing Systems*, 2020.

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

[76] R. S. Varga, "On a discrete maximum principle," *SIAM Journal on Numerical Analysis*, vol. 3, no. 2, pp. 355–359, 1966.

[77] A. P. Witkin, "Scale-space filtering," in *Readings in Computer Vision*. Elsevier, 1987, pp. 329–332.

[78] C. Xiao and L. Liu, “Generative flows with matrix exponential,” in *International Conference on Machine Learning*, 2020.

[79] H. Yan, J. Du, V. Y. F. Tan, and J. Feng, “On robustness of neural ordinary differential equations,” *International Conference on Learning Representations*, 2020.

[80] J. Yu, K. Derpanis, and M. Brubaker, “Wavelet flow: Fast training of high resolution normalizing flows,” in *Advances in Neural Information Processing Systems*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We present a new technique as is, and present an entires section that describes some of the disadvantages.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We mention the importance and impact of the out-of-distribution properties of our method.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [No] Our theoretical results follow directly from first principles in the derivations and the analysis that we present.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Anonymized code will be included in supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] The code contains all details, and we refer to the public code repository that we build upon.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We report error bars for our main results in Tables 1 and 2.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We give lots of details about the computational requirements of our method, since that is one of the advantages. The reader would need to add up the numbers from all our experiments if they really wanted a total, but the point is our method is much less computationally intense compared to popular alternatives.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No] We use standard datasets here, i.e. CIFAR and ImageNet.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] We use a standard set of public Deep Learning benchmark datasets.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]