Generative Convolutional Flow for Density Estimation

Mahdi Karami
Department of Computer Science
University of Alberta
karami1@ualberta.ca

Laurent Dinh, Daniel Duckworth, Jascha Sohl-Dickstein, Dale Schuurmans
Google Brain

Abstract

Normalizing flows can be used to construct high quality generative probabilistic models. However, training and generating samples from the flow requires evaluating the determinant of the input-output Jacobian, and inverting the input-output function, respectively. In order to make these computations feasible, existing normalizing flow models have highly constrained architectures, in most cases producing a Jacobian which is a diagonal, triangular, or low-rank matrix, enabling fast Jacobian calculation. In this work, we introduce a set of novel and powerful normalizing flows based on the circular convolution transform. We show that the Jacobian of this transform is a circulant matrix that admits efficient determinant computation and inverse mapping (deconvolution) in $\mathcal{O}(N \log N)$ time. Moreover, element-wise multiplications, that are widely used in normalizing flow architectures, can be combined with these convolution transforms to increase the flexibility of the flow. Since convolutional layers are important for many applications, especially in image and audio processing, we expect the proposed bijective mapping to enable richer and more powerful normalizing flows for these domains.

1 Introduction

Flow-based generative networks have shown tremendous promise for modeling complex observations in high dimensional datasets. In flow-based models, a complex probability density is constructed by transforming a simple base density, such as a standard normal distribution, via a chain of smooth and invertible mappings (bijections), yielding a *normalizing flow*. Such normalizing flows are employed in various contexts, including approximating a complex posterior distribution in variational inference [Rezende and Mohamed, 2015], or for density estimation with generative models [Dinh et al., 2016].

Unfortunately, using a complex transformation to define a normalized density requires the computation of a Jacobian determinant, which is generally impractical for arbitrary neural network transformations. To overcome this difficulty, previous work carefully designs architectures to impose a simple structure in the Jacobian matrix, enabling fast computation of the Jacobian determinant. For example, one approach has been to consider transformations that have a Jacobian corresponding to low rank perturbations of a diagonal matrix, enabling the use of Sylvester's determinant lemma [Rezende and Mohamed, 2015, van den Berg et al., 2018]. Other works, such as [Dinh et al., 2014, 2016, Kingma et al., Papamakarios et al., 2017], use a constrained transformation where the Jacobian has a triangular structure. This latter scheme has proved particularly successful as this constraint is easy to enforce without large sacrifices in expressiveness or computational efficiency. More recently, Kingma and Dhariwal [2018] proposed the use of 1×1 convolutions, which owes its tractability to a block diagonal Jacobian.

In this work, we propose an alternative non-linear convolution layer we call an *adaptive non-linear convolution filter*. Here, the convolution kernel for a layer adapts to the input of the layer. Broadly speaking, splitting the input of a layer $\boldsymbol{x}$ into two disjoint subsets $\{\boldsymbol{x}_1,\ \boldsymbol{x}_2\}$, the convolution updates a subset of input as $\boldsymbol{w}(\boldsymbol{x}_1) * \boldsymbol{x}_2$, while the convolution kernel $\boldsymbol{w}(\boldsymbol{x}_2)$ is a function of a disjoint subset of input which can be modeled with deep neural networks. We present convolution operations that are invertible and whose Jacobians can be computed efficiently, making them amenable for the normalizing flow. As apposed to the causal convolution employed in [van den Oord et al., 2016] to generate audio waveform, or in [Zheng et al., 2017] to approximate the posterior in variational autoencoder, the convolution operations presented here are not constrained to depend only on the preceding input variables.

1.1 Normalizing flows

Given an invertible and differentiable mapping $g : \mathbb{R}^n \to \mathbb{R}^n$ of a random variable $\mathbf{z} \sim p(\mathbf{z})$, with inverse transform $f = g^{-1}$, the probability density function of the transform $\mathbf{x} = g(\mathbf{z})$ can be recovered by the *change of variable rule* as

$$\begin{aligned} p(\mathbf{x}) &= p(\mathbf{z}) \left|\det \boldsymbol{J}_g\right|^{-1} \\ &= p(f(\mathbf{x})) \left|\det \boldsymbol{J}_f\right| \end{aligned} \tag{1}$$

where $J_g = \frac{\partial g}{\partial \mathbf{z}^\top}$ and $\boldsymbol{J}_f = \frac{\partial f}{\partial \mathbf{x}^\top}$ are the Jacobian matrices of functions g and f, respectively. Furthermore, one can build a complex mapping g by composing a chain of simple bijective maps, $g = g_1 \circ g_2 \circ ... \circ g_K$, that preserves the invertibility property with the inverse transformation being $f = f_K \circ f_{K-1} \circ ... \circ f_1$; subsequently, applying the chain rule to the Jacobian of the composition, and using the fact that $\det \boldsymbol{AB} = \det \boldsymbol{A} \det \boldsymbol{B}$, the log-likelihood can be written as

$$\log p(\mathbf{x}) = \log p(\mathbf{z}) + \sum_{k=1}^{K} \log \left|\det \boldsymbol{J}_{f_k}\right| \tag{2}$$

This mapping, which enables specifying complex densities by flow of a simple density $p(z)$ through a sequence of bijections, is called a *normalizing flow* [Rezende and Mohamed, 2015]. The basic density is chosen from well known distribution families that can be easily evaluated, such as a standard normal distribution. In comparison to variational inference methods that maximize a variational lower bound on the likelihood by learning an approximate posterior distribution on the latent variable, equation (2) provides a mechanism for exact likelihood maximization and density estimation.

Evaluating the Jacobian-determinant is the main computational bottleneck in equation (2) since, in general, it can scale cubicly in the size of input. It is natural to seek general classes of structured transformations that can mitigate this cost while retaining useful modeling flexibility. In the following a class of such transformations will be introduced in detail.

2 Toeplitz structure and Circular Convolution

Although previous work has considered Jacobians with a block-diagonal or triangular form, these are not the only useful possibilities. In fact, there are many other forms of transformation whose Jacobian has sufficient structure to allow computationally efficient determinant calculation. One such structure is *Toeplitz* property, *i.e.* the diagonal elements of a square matrix are identical, then calculation of the determinant can be significantly simplified. Let $\boldsymbol{J}$ be

$$\boldsymbol{J} = \begin{bmatrix} a_0 & a_{-1} & \cdots & a_{-s} & & & \mathbf{0} \\ a_1 & a_0 & a_{-1} & \ddots & a_{-s} & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \\ a_r & & & & & & a_{-s} \\ \vdots & a_r & & & & & \\ & & \ddots & & a_1 & a_0 & a_{-1} \\ \mathbf{0} & & & a_r & \cdots & a_1 & a_0 \end{bmatrix}$$

of size $N * N$ with bandwidth size $K := r + s$. In general, the determinant of a Toeplitz matrix can be evaluated in $\mathcal{O}(N^2)$ time [Monahan, 2011], and specially for matrices of the form $\mathbf{J}$ with small bandwidth size, this can even be more simplified to algorithms of $\mathcal{O}(K^2 \log N + K^3)$ time [Cinkir, 2011]. Moreover, there are also efficient algorithms for inverting Toeplitz matrices [Martinsson et al., 2005]. So the transformations with Jacobian matrices of Toeplitz form are suitable candidates for normalizing flows. The Toeplitz matrices are of particular interest in *convolutional neural networks (CNNs)* as the discrete convolution operation can be expressed in the form of a matrix product of a Toeplitz matrix by the input [Gray et al., 2006].

Herein, we consider particular transformations whose Jacobian is a *circulant matrix*, a special and well established form of Toeplitz structure. In a circulant matrix the rows (columns) are cyclic permutations of the first row (column) *i.e.* $J_{l,m} = J_{1,(l-m) \bmod N}$. This structure allows certain algebraic operations, such as determinant calculation, inversion and eigenvalue decomposition, to be readily performed in $\mathcal{O}(N \log N)$ time by exploiting the fact that a square circulant matrix can be diagonalized by a discrete Fourier transform [Gray et al., 2006]. Let's define the circular convolution as $\mathbf{y} := \boldsymbol{w} \circledast \mathbf{x}$ where $y(i) := \sum_{n=0}^{N-1} x(n) w(i-n)_{\bmod N}$, the key property we exploit in developing an efficient normalizing layer is that the Jacobian of this convolution forms a circulant matrix, hence its determinant and inverse transforms (deconvolution) can be computed efficiently. In the following some of the main properties of this operation are summarized.

Theorem 1 *Let $\mathbf{y} = \boldsymbol{w} \circledast \mathbf{x}$ be a circular convolution on the input vector $\mathbf{x}$, then:*

a) *The circular convolution operation can be expressed as a matrix multiplication equation $\mathbf{y} = \boldsymbol{C}_w \mathbf{x}$ where $\boldsymbol{C}_w$ is a circulant square matrix with first row being $\boldsymbol{w}$.*

b) *The Jacobian of this transform is $\boldsymbol{J}_y = \boldsymbol{C}_w$.*

c) *The eigenvalues of the circulant Jacobian matrix $\boldsymbol{C}_w$ are equal to the discrete Fourier transform (DFT) of $\boldsymbol{w}$, thus*

$$\log |\det \boldsymbol{J}_y| = \sum_{n=0}^{N-1} \log |\boldsymbol{w}_f(n)|$$

where $\boldsymbol{w}_f(n) := \mathcal{F}_N\{\boldsymbol{w}\}_n$.

d) *The circular convolution can be expressed by element-wise multiplication in the frequency domain, $\boldsymbol{y}_f(k) = \boldsymbol{w}_f(k)\, \boldsymbol{x}_f(k)$, which is also known as the convolution-multiplication property. If $\boldsymbol{w}_f(n) \neq 0\ \forall n$, this linear transform is invertible with inverse $\boldsymbol{x}_f(n) = \boldsymbol{w}_f^{-1}(n)\, \boldsymbol{y}_f(n)$. Moreover, its inverse transform (deconvolution) is also a circular convolution operation with kernel $\boldsymbol{w}^{inv} := \mathcal{F}_N^{-1}\{\boldsymbol{w}_f^{-1}\}$.*

e) *Since the fast Fourier transform (FFT) can be computed in time $\mathcal{O}(N \log N)$ for DFT, the circular convolution operation, its inverse transform, and computation of determinant-Jacobian can all be efficiently performed in the frequency domain.*

3 Input dependent convolution layer

The special form of convolution introduced so far appears to be particularly well suited to capturing structure in images and audio signals, thus we seek to design more expressive normalizing flows using this form of convolution bijection as a building block. For modeling flexibility, we propose a convolution layer that uses a filter kernel that is a function of the input of the layer. Inspired by the idea of the coupling layer in Dinh et al. [2016], we can build a modular bijection by splitting the input $\mathbf{x} \in \mathbb{R}^d$ into two parts $\{\mathbf{x}_1 \in \mathbb{R}^{d_1}, \mathbf{x}_2 \in \mathbb{R}^{d_2} : d_1 + d_2 = d\}$ termed as the *base input* and *update input*, respectively, and only update the update input $\mathbf{x}_2$ by a convolution operation where the kernel of this transform is a function of base input $\mathbf{x}_1$. More specifically, we use

$$\begin{aligned} \mathbf{y}_1 &= \mathbf{x}_1 \\ \mathbf{y}_2 &= \boldsymbol{w}(\mathbf{x}_1) \circledast \mathbf{x}_2 + \boldsymbol{b}(\mathbf{x}_1) \end{aligned} \tag{3}$$

The kernel $\boldsymbol{w}(\mathbf{x}_1)$ and the translation $\boldsymbol{b}(\mathbf{x}_1)$ can be any nonlinear functions that are not required to be invertible, hence they can be modeled by deep convolution networks with an arbitrary number of hidden units, offering flexibility and rich representation capacity while preserving an efficient learning algorithm.

3.1 Determinant-Jacobian calculation and the inverse transform

Due to modular structure of the flow in (3), its Jacobian can be expressed in terms of the Jacobian of its convolution operation, denoted as $\boldsymbol{J}_*$. More precisely, its Jacobian is

$$\boldsymbol{J}_y = \frac{\partial \mathbf{y}}{\partial \mathbf{x}^\top} = \begin{bmatrix} \boldsymbol{I}_{d_1} & \mathbf{0} \\ \frac{\partial \mathbf{y}_2}{\partial \mathbf{x}_1^\top} & \boldsymbol{J}_* \end{bmatrix} \tag{4}$$

where $\boldsymbol{J}_* = \frac{\partial (\boldsymbol{w} * \mathbf{x}_2)}{\partial \mathbf{x}_2^\top}$. Noticeably, the Jacobian is a block triangular matrix, so determinant can be readily calculated by the product of determinant of the square blocks on its diagonal, therefore

$$\log |\det \boldsymbol{J}_y| = \log |\det \boldsymbol{J}_*| \tag{5}$$

Hence, the Jacobian of the flow in (3) can be expressed in terms of the Jacobian of its convolution operation, that, according to Theorem 1, can be expressed in terms of the Fourier transform of the convolution kernel and can be computed efficiently in $\mathcal{O}(d_2 \log d_2)$ by the fast Fourier transform algorithm.

It is worth noting that the term (5) plays the role of a log barrier in the final loss function that prevents the eigenvalues of the Jacobian (the filter coefficients in the frequency domain) from falling to zero. This in turn enforces non-singularity of the convolution kernel resulting in an invertible transform. The inverse of the transform (3) is

$$\begin{aligned} \mathbf{x}_1 &= \mathbf{y}_1 \\ \mathbf{x}_2 &= \boldsymbol{w}(\mathbf{y}_1)^{inv} * (\mathbf{y}_2 - \boldsymbol{b}(\mathbf{y}_1)). \end{aligned} \tag{6}$$

The inverse kernel $\boldsymbol{w}(\mathbf{y}_1)^{inv}$ can indeed be derived through the procedure explained in Theorem 1 using the same network as for the forward kernel $\boldsymbol{w}(\mathbf{y}_1)$, where only the matrix output of the network, and not the network itself, needs to be inverted. The inverse convolution, also know as deconvolution, can also be performed in the frequency domain as observed in Theorem 1.

3.2 Combined convolution multiplication layer

The convolution in (5) slides a filter spatially and applies the same weighted summation on all locations of the signal[1]. To achieve a more flexible and richer filtering scheme, we can combine an element-wise multiplication and convolution so that the filtering scheme varies over the location and we achieve a more flexible and richer transformation. Also, the product of a diagonal matrix with a circulant matrix was proposed in [Cheng et al., 2015] as a structured approximation for dense (fully connected) linear layers while [Moczulski et al., 2015] showed that any $N \times N$ linear operator can be approximated to arbitrary precision by composing order N such products. Thus, the flows can be modified to

$$\begin{aligned} \mathbf{y}_1 &= \mathbf{x}_1 \\ \mathbf{y}_2 &= \boldsymbol{s}(\mathbf{x}_1) \odot \big(\boldsymbol{w}(\mathbf{x}_1) * \mathbf{x}_2 + \boldsymbol{b}(\mathbf{x}_1)\big) + \boldsymbol{t}(\mathbf{x}_1). \end{aligned} \tag{7}$$

Where again the Jacobian is a block triangular matrix and hence the determinant calculation will be significantly simplified resulting in

$$\log |\det \boldsymbol{J}_y| = \log |\det \boldsymbol{J}_*| + \log |\det \boldsymbol{J}_\odot| \tag{8}$$

where $\boldsymbol{J}_\odot$ is the Jacobian of element-wise product that reduces to $\boldsymbol{J}_\odot = \sum_i \log |\boldsymbol{s}_i|$. Note again that the term (8) in the optimization loss induces an invertible mapping with inverse transform

$$\begin{aligned} \mathbf{x}_1 &= \mathbf{y}_1 \\ \mathbf{x}_2 &= \boldsymbol{w}(\mathbf{y}_1)^{inv} * \big(\boldsymbol{s}(\mathbf{y}_1)^{inv} \odot (\mathbf{y}_2 - \boldsymbol{t}(\mathbf{y}_1)) - \boldsymbol{b}(\mathbf{y}_1)\big). \end{aligned} \tag{9}$$

where $\boldsymbol{s}_{i,j}^{inv} := 1/\boldsymbol{s}_{i,j}$.

[1]For multichannel inputs such as images one convolution filter is applied per channel.

Relation to planar normalizing flow Rezende and Mohamed [2015], introduced a simple and tractable planar normalizing flow that has the form

$$\mathbf{y} = \boldsymbol{u} \odot h(\boldsymbol{v}^\top \mathbf{x} + b) + \mathbf{x}. \tag{10}$$

where $h(.)$ is a differentiable and invertible nonlinearity and $\boldsymbol{u} \in \mathbb{R}^d, \boldsymbol{v} \in \mathbb{R}^d, b \in \mathbb{R}$ are the parameters of the model. This transform is known as planar flow. The combined multiplication-convolution transform (7) can be viewed as the generalization of the planar flow where the parameters of the model are nonlinear function of input and the inner vector product is replaced with invertible convolution operation. Also, in the transforms proposed in this work, the nonlinearity is embedded into the deep neural networks modeling the parameter functions $\{\boldsymbol{s}(.), \boldsymbol{t}(.), \boldsymbol{w}(.), \boldsymbol{b}(.)\}$, and we didn't add a nonlinearity after the convolution. Subsequently, the output of the transforms are nonlinear function of the input hence they are termed as *non-linear convolution filter*.

Remark: Multi-dimensional spectral transformations can be expressed in separable forms so that the these operations can be performed by successively applying 1-dimensional transforms along each dimension [Gonzalez and Woods, 1992]. The separability property induces that the results mentioned so far can be generalized to multi-dimensional settings. In this work, we are particularly interested in 2-D operations that are used in image processing; more details on the 2-D circular convolution, its corresponding block-circulant matrix and the diagonalization method can be found in [Gonzalez and Woods, 1992, Ch. 5]. It can be readily verified that results presented in theorem 1 can be extended to 2-D case using 2-D circular convolution and the 2-D DFT[2].

4 Model architecture

An arbitrarily complex density approximation can be performed by composing a chain of the convolution coupling layers, introduced in this work. As explained prior to section 1.1, the determinant of Jacobian and the inverse of this composition can be obtained readily.

Due to special structure of the convolutional coupling layer, only one part of the input, update input, is transformed by each layer and the remaining part, base input, is left unaltered. To overcome this, we permute the assignment of pixels and/or channels to the base input and update input in every other layer so that both parts of the input are updated; note that one layer exhibits a lower block-wise triangular Jacobian and the other has an upper block-wise triangular Jacobian[3].

This architecture can also be modified in order to provide latent representations at multiple layers and scales, by factoring out a subset of latent variables every time the scale of the image is changed, as in Dinh et al. [2016]. This both enables a multi-scale representation, and reduces computational cost.

5 Experiments

We evaluated the proposed generative model on MNIST [Y. LeCun, 1998] and CIFAR-10 [Krizhevsky, 2009]. To transform the inputs from a bounded to an unbounded domain, the logit mapping of the form $y = \text{logit}(\alpha + (1-\alpha)\frac{x}{256})$ is applied with $\alpha = 0.05$, 10^{-6} to images from CIFAR-10 and MNIST, respectively. The model consists of 4 stacked coupling layers each composed of 8 combined multiplication-convolution flows (MCONV) (7). ResNet neural networks [He et al., 2016] with 8 residual blocks, each with 64 feature maps, are employed to model the non-linear functions specifying the parameters of the flow. The Adam optimizer [Kingma and Ba, 2014] was used to train the model, with a learning rate of .0005 and mini-batch sizes of 128 in the training phase.

Three generative models: Real NVP [Dinh et al., 2016], masked autoregressive flow (MAF) [Papamakarios et al., 2017] and inverse autoregressive flow (IAF-VAE) [Kingma et al.], are used as benchmark models. The results of MAF are obtained from the original paper which implemented a general-purpose density estimation model and does not take into account the 2D structure of image data.

[2] Due to separability property, the 2-D DFT of matrices of size $M \times N$ can be computed in $\mathcal{O}(MN(\log M + \log N))$ time.

[3] This is analogous to LU factorization, where one can compose a lower triangular matrix with and upper triangular one to obtain an arbitrary square matrix, with proper row permutation [Horn et al., 1990].

	MNIST	CIFAR-10
Real NVP	1.09	3.49
MAF (10)	1.91	4.31
IAF-VAE	-	<3.28
MCONV	1.07	3.44

Table 1: Bits per dimension achieved by different density estimators on the MNIST and CIFAR-10 test data (training results shown in parentheses).

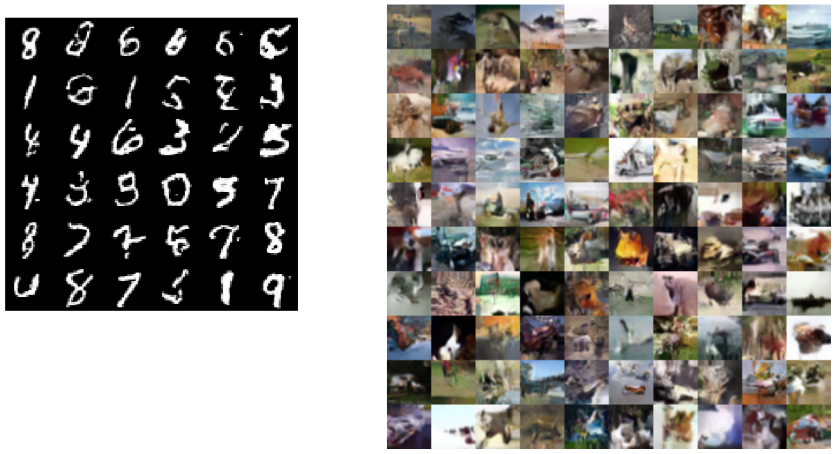

Figure 1: (left) Samples generated from an MCONV flow model trained on the MNIST dataset. (right) Samples generated from an MCONV flow model trained on the CIFAR-10 dataset.

The results in Table 1, given in bits per dimension, show that the proposed normalizing flow model is able to improve on the results of previous methods on these datasets.

References

Yu Cheng, Felix X Yu, Rogerio S Feris, Sanjiv Kumar, Alok Choudhary, and Shi-Fu Chang. An exploration of parameter redundancy in deep networks with circulant projections. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2857–2865, 2015.

Z. Cinkir. A fast elementary algorithm for computing the determinant of Toeplitz matrices. *ArXiv e-prints*, January 2011.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Rafael C Gonzalez and Richard E Woods. Digital image processing, 1992.

Robert M Gray et al. Toeplitz and circulant matrices: A review. *Foundations and Trends® in Communications and Information Theory*, 2(3):155–239, 2006.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Roger A Horn, Roger A Horn, and Charles R Johnson. *Matrix analysis*. Cambridge university press, 1990.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. A fast algorithm for the inversion of general toeplitz matrices. *Computers & Mathematics with Applications*, 50(5-6):741–752, 2005.

Marcin Moczulski, Misha Denil, Jeremy Appleyard, and Nando de Freitas. Acdc: A structured efficient linear layer, 2015.

John F Monahan. *Numerical methods of statistics*. Cambridge University Press, 2011.

George Papamakarios, Iain Murray, and Theo Pavlakou. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 1530–1538, 2015.

Rianne van den Berg, Leonard Hasenclever, Jakub M Tomczak, and Max Welling. Sylvester normalizing flows for variational inference. *arXiv preprint arXiv:1803.05649*, 2018.

Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016.

C. Cortes Y. LeCun. The mnist database of handwritten digit. 1998.

Guoqing Zheng, Yiming Yang, and Jaime Carbonell. Convolutional normalizing flows. *arXiv preprint arXiv:1711.02255*, 2017.