# Normalizing Flow: An Introduction

Hongli Zhao

5:00 pm, June 16, 2021, Zoom (Internal Group Seminar)

**Abstract**

In this short note, we present normalizing flow as an expressive and powerful generative model under which both sampling and density evaluations are efficiently able to be implemented. Possible applications of normalizing flow can be found in solving high-dimensional partial differential equations (PDE).

## 1 Notations, Definitions and Theorems

Below we present some common notations and results that we will use in the following sections. We assume that the audience has an application-focused understanding of probability theory (but not the measure-theoretic counterpart) on a similar level of UC Berkeley course EECS 126. With machine learning applications, we assume the audience has attained the level of understanding on par with Berkeley course CS 189.

In the probabilistic setting, we denote $x \sim p_X(x)$ as "random variable $x$ is distributed according to probability distribution with density $p_X(x)$". Let $(\Omega, \mathcal{F}, \mathbb{P})$ denote a probability space. For the probability density function $p_X$, in certain cases we encounter the fact that it may be parametrized (such as in the setting of a Gaussian normal $\mathcal{N}(\mu, \sigma)$. We denote $\theta$ as our class of parameters, and $p_\theta(x)$ as the density.

A *random variable* $X$ is a mapping from $\Omega \to \mathbb{R}^d$. We will explicitly distinguish cases where measure theory needs to be considered. Otherwise, $x$ can be assumed as in $\mathbb{R}^d$, and let:

$$\mathbb{E}[x] = \int_\Omega x \cdot p_X(x)dx$$

denote the expectation or expected value of $x \sim p_X(x)$. Furthermore, in numerical methods, oftentimes we must seek approximations to true values, we use the hat notation to denote the approximate value. For example, an approximation to the expected value is often done by sampling a probability distribution and taking the average over sample points $\{x_i\}_{i=1}^N$:

$$\widehat{\mathbb{E}}[x] = \frac{1}{N}\sum_{i=1}^N x_i$$

Another commonly used quantity is the variance, defined as the expected squared deviation from the mean:

$$\mathbb{V}ar\big[x\big] = \mathbb{E}\big[(x - \mathbb{E}\big[x\big])^2\big]$$

In certain cases we may find it notationally convenient to simplify the writing of the above quantities, we will denote in certain cases the expectation $\hat{x}$ or $\langle x \rangle$, and variance $\sigma_x^2$. Finally, when the probabilistic models become increasingly complex, it is more important and appropriate to clarify what probability density is in question, with the additional subscript notations:

$$\mathbb{E}\big[x\big]_{x\sim p_X}, \sigma^2_{x\sim p_X}$$

• A random variable $x$ can be either discrete or continuous. In the continuous setting ($d = 1$):

$$\mathbb{P}(x \in [a,b]) = \int_{[a,b]} p_X(x)dx$$

• In continuous space where theories are derived, integration is $\int$. The equivalent version of it in discrete space where theories are verified using data, we use $\sum$.

• (Normalizing) A valid probability density is nonnegative and has unit mass. In order to learn probability densities from data, it is imperial that we need to enforce:

$$\int_{\mathbb{R}^d} p(x)dx = 1$$

or:

$$\sum_{i=1}^N p(x_i) = 1$$

• (Chain rule of probability) High-dimensional settings often make many operations computationally intractable. We are able to decompose a multivariate density into a sequence of single variable ones:

$$p(x_1, x_2, \cdots, x_d) = \prod_{j=1}^{d} p(x_j | x_1, x_2, \cdots, x_{j-1})$$

where:

$$p(x_j | x_{<j}) = \frac{p(x_1, x_2, \cdots, x_j)}{p(x_1, x_2, \cdots, x_{j-1})}$$

where all missing variables $x_{j+1}, \cdots, x_d$ are "integrated out". Namely $p(x_{\leq j})$ is in fact a marginal density:

$$p(x_{\leq j}) = \int_{\mathbb{R}\times\cdots\times\mathbb{R}} p(x_1, x_2, \cdots, x_d) dx_{j+1} \cdots dx_d$$

• (Probability change of variable formula, $d = 1$) Let $x \sim p_X(x)$ and $z = f(x)$ or $x = f^{-1}(z)$ where $f$ is a mapping $x \mapsto z$ (invertible, differentiable). Then the density of $z$ can be derived as:

$$p_Z(z) = p_X(x) \cdot \left| \frac{dx}{dz} \right|$$

the derivation is given by considering the CDF:

$$F_X(x) = \int_{[-\infty, x]} p_X(x) dx$$

and using $x = f^{-1}(z)$ (to transform the density), the inverse function theorem (to correct for unit mass), and $\frac{d}{dx} F_X(x) = p_X(x)$ (to recover the PDF). It is possible to define change of variables for general measurable functions, which we will skip for the purpose of this discussion.

• (Probability change of variable formula, general $d$) In multivariate $x \in \mathbb{R}^d$, the change of variable formula can be written by replacing the definition of derivative and absolute value.

$$f : \mathbb{R}^d \to \mathbb{R}^d$$

$$\frac{dz}{dx} = J_f(z)$$

$$|\cdot| = \det(\cdot)$$
$$\det\big(A^{-1}\big) = \det(A)^{-1}$$

for nonsingular $A \in \mathbb{R}^{d\times d}$. Here $\left|\frac{dx}{dz}\right|$ can be considered as volume correction.

• (closure under composition) Let $\{f_i\}_{i=1}^{M}$ be a sequence of invertible and differentiable functions, one can show that:

$$f = f_M \circ f_{M-1} \circ \cdots \circ f_1$$

is also invertible and differentiable, with inverse:

$$f^{-1} = f_1^{-1} \circ \cdots \circ f_M^{-1}$$

and chain rule of differentiation:

$$\frac{df}{dz} = \prod_{i=1}^{M} \frac{df_i}{df_{i-1}}$$

where $f_0(z) = z$.

Below is a result that demonstrates the benefit of a normalizing flow, that evaluating the expected value of a transformed quantity does not require us to know the (potentially complex) distribution.

• (Law of Unconscious Statistician) let $z \sim p_Z(z)$ which is known and easy to evaluate. Consider the mapping $x = f(z)$, with $p_X$ being difficult to evaluate or unknown, then $\mathbb{E}\big[x\big]$ is still computable by:

$$\mathbb{E}\big[x\big] = \mathbb{E}\big[f(z)\big] = \int_{\mathbb{R}^d} f(z)\cdot p_Z(z)dz$$

• (Inverse CDF Sampling) Let $x = (x_1, x_2, \cdots, x_d) \in \mathbb{R}^d$, by product rule:

$$p_X(x_1, x_2, \cdots, x_d) = \prod_{i=1}^{d} p(x_i|x_1, x_2, \cdots, x_{i-1})$$

The CDF of each conditional distribution is given by:

$$F_X(x_i|x_{<i}) = \int_{-\infty}^{x_i} p(x_i|x_{<i})dx_i$$

let $f_i = F_X(\cdot|x_{<i})$ and consider the invertible, differentiable transformation:

$$z_i = f_i(x_i)$$

this forms a mapping from $\mathbb{R}^d \to [0,1]^d$. This mapping is also invertible as:

$$x_i = f_i^{-1}(z_i)$$

since the CDF is monotonic. Finally, regarding the distribution of $z$, we have by the change of variables formula that:

$$p_Z(z) = p_X(x)\left|\det\left(\frac{dz}{dx}\right)\right|^{-1}$$

however, the Jacobian is lower triangular because the conditional distribution $x_i$ has no dependence on $x_{>i}$. Specifically on the diagonals:

$$\frac{dz_i}{dx_i} = \frac{d}{dx_i}f_i(x_i)$$

which is by definition the conditional CDF, which yields:

$$= p(x_i|x_{<i})$$

such that the determinant is:

$$\left|\det\left(\frac{dz}{dx}\right)\right| = \prod_{i=1}^{d} p(x_i|x<i) = p(x_1, x_2, \cdots, x_d)$$

by product rule. This means:

$$p_Z(z) = p_X(x)\cdot\frac{1}{p_X(x)} = 1$$

and implies that $z$ is $U([0,1])$.

One can consider the above derivation as the reason why normalizing flow is expressive enough to capture any distribution $x \sim p_X(x)$ may follow. Furthermore, in other literatures, the *generative direction* is named *conditional distribution sampling method* where a random vector $x = (x_1, x_2, \cdots, x_d)$ is iteratively generated via inverse CDF at each step. The two directions refer to the same underlying process.

The following is a result from matrix calculus that we can make use in a proof of continuous-time flows:

- (adjugate matrix) The adjugate matrix of an $n \times n$ matrix is:

$$\text{adj}(A) = C^T = \det(A)\cdot A^{-1}$$

where $C$ is the cofactor matrix.

- (Jacobi's formula) Let $A(t)$ be a differentiable map $\mathbb{R} \to \mathbb{R}^{n\times n}$, then:

$$\frac{d}{dt}A(t) = \text{tr}\left(\text{adj}(A(t))\frac{dA(t)}{dt}\right)$$

## 2 Normalizing Flow as a Generative Model

In applications of machine learning, one major goal is to discover underlying patterns of data. In generative models, we achieve this by directly approximating the probability distribution from which samples are drawn, and provide an efficient way to generate or predict new samples. In short, the main goals we would like to achieve with a good **generative probabilistic model** [7] [5] is as follows:

- provide a satisfactory fit and generalization of training data with unknown distribution
- computationally convenient sampling (prediction of new data) and density evaluation from learned model, especially in the high-dimensional case [2].
- qualitatively explainable latent space / low-dimensional representation

In the following sections, we briefly discuss why normalizing flow can achieve these criterion, and common directions for building such tools.

## 3 Forms of Normalizing Flow and Computational Complexity

In the following formulations we discuss the methods in the *generative direction*, which also gives us an algorithm of sampling from a complex distribution once we have trained a flow:

(1) sample latent dataset $z$ is according to desired $p_Z(z)$.

(2) obtain $x = f(z)$ as samples from $p_X(x)$.

Equivalently one can also describe the process in the *normalizing direction*, which also describes how normalizing flow can be used for density evaluation:

(1) dataset $x$ is acquired.

(2) evaluate $p_X(x)$ by evaluating $p_Z(f^{-1}(x))$ and volume correction.

### 3.1 Computational Considerations

In the following method presentations, there is a balance between the ease of:

• density evaluation: cost of computing $\left|\det\left(\frac{dx}{dz}\right)\right|$. The naive expansion of determinant of a dense matrix is $O(n!)$. One also considers LU factorization or SVD decomposition, which costs $O(n^3)$ at least. Research efforts have been given to consider special $\frac{dx}{dz}$ matrix forms (such as triangular, diagonal, or block diagonal) to reduce time complexity.

• sampling: cost of computing $f^{-1}$. The specific cost depends on the choice of $f$. For instance, when $f = Az + b$ is an affine transformation, and $A$ is dense, computing $f^{-1}$ amounts to the cost of finding $A^{-1}$, which by Gaussian elimination is at least $O(n^3)$. In certain cases (such as using neural networks), the inversion is intractable.

• preservation of dimensions: as dimensions between invertible maps must be the same, cost of evaluating $f(z)$ itself scales with $d$.

## 3.2 Expressiveness of Normalizing Flow

We demonstrate the result in $d = 1$. Multivariate PDF can be formed similarly by *conditional distribution sampling*. Inverse CDF sampling is a special case of normalizing flow. Namely, regardless of the complexity of $p_X(x)$, the target distribution, we can always find the cumulative distribution function as a suitable flow. CDFs are monotonically increasing and differentiable, and therefore would satisfy the desired properties.

Precisely, the inverse CDF sampling can be described as:

(1) generate $u \sim U([0, 1])$.

(2) let $f = F_X^{-1}$ where $F_X$ is the CDF of $p_X$.

(3) $x = f(u)$ produces the desired distribution $x \sim p_X(x)$.

By inverse CDF sampling, we have demonstrated that there always exists an invertible map $f$ such that $f(z)$ can be flexibly expressive. In other words, any complex distribution can be normalized as a uniform distribution; this also implies that the base distribution does not need to be $U([0, 1])$, as compositions of flows form other flows, as demonstrated in section 1. To transform an arbitrary distribution $p_Z$ to another arbitrary distribution $p_X$, one can always consider:

$$p_X \xrightarrow{f_1} U([0,1]) \xrightarrow{f_2} p_Z$$

## 3.3 Affine Normalizing Flow: Building Block

$$\theta = (A, b) \in \mathbb{R}^{d\times d} \times \mathbb{R}^d$$

where we require $A$ to be invertible.

$$x = f(z) = Az + b$$
$$\frac{df}{dz} = A^T$$

The linear flow has several limitations both in expressiveness and computations. Suppose $z \sim \mathcal{N}(0, I)$, then:

$$p_X(x) = \frac{1}{|\det(A)|} \cdot p_Z(f^{-1}(x)) = \frac{1}{|\det(A)|} \cdot p_Z(A^{-1}(x - b))$$

We have the multivariate Gaussian normal distribution:

$$p_Z(z) = (2\pi)^{-d/2} \exp\left( -\frac{1}{2} z^T z \right)$$

Then:

$$p_X(x) = \frac{1}{|\det(A)|} \cdot (2\pi)^{-d/2} \exp\left( -\frac{1}{2} (A^{-1}(x-b))^T (A^{-1}(x-b)) \right)$$
$$= \frac{1}{(2\pi)^{d/2} |\det(A)|^{1/2} |\det(A^T)|^{1/2}} \exp\left( -\frac{1}{2} (x-b)^T (AA^T)^{-1} (x-b) \right)$$

which means $x \sim \mathcal{N}(b, AA^T)$.

Furthermore, compositions of linear flows result in linear flows. Let $f_1(z) = Az + b$, $f_2(z) = Cz + d$:

$$f_2(f_1(z)) = C(Az + b) + d = CAz + (Cb + d)$$

Finally, training $O(d^3 + d)$ parameters directly, evaluation of density and sampling are inefficient in high dimensions, as both computing $A^{-1}$ and $|\det(A)|$ requires $O(d^3)$ flops. Further restrictions on $A$ can be made in order to simplify the computations (such as diagonal or triangular $A$).

## 3.4 Coupling Layers

A coupling layer can benefit us in high-dimensions in terms of computational cost of evaluating the determinant while only using affine transformations. This is done by moving work to the region of the Jacobian that is "free of charge", by taking advantage of the observation that the determinant of a triangular matrix is the product of its diagonals.

A coupling layer takes the general form (partitioning $z$) into two parts:

$$z = \begin{bmatrix} z_{\leq m} \\ z_{>m} \end{bmatrix}$$

and define:

$$x = \begin{bmatrix} x_{\leq m} \\ x_{>m} \end{bmatrix} = \begin{bmatrix} z_{\leq m} \\ f(z_{>m}; h(z_{\leq m})) \end{bmatrix}$$

where $f$ is invertible and $h$ has no restrictions. It is the restriction-free $h(\cdot)$ that brings nonlinearity at a low cost of training and evaluation. We can explicitly compute the form of the Jacobian:

$$\frac{dx}{dz} = \begin{pmatrix} I_m & 0 \\ \frac{dx_{>m}}{dz_{\leq m}} & D \end{pmatrix}$$

where $D$ is a matrix with entries $\frac{dx_{>m}}{d_{z>m}}$.

This is an interesting choice as we are essentially allowed to have any nonlinearity without it causing computational cost to skyrocket in high dimensions, as the determinant is:

$$\left|\det\left(\frac{dx}{dz}\right)\right| = \prod_{i>m} \frac{d}{dz_i} f(z_{>m}; h(z_{\leq m}))$$

and $h(z_{\leq m})$ has no dependence on $z_{>m}$ and therefore does not need to be differentiated. Furthermore, the volume correction only requires the determinant, rather than the Jacobian matrix itself, therefore we would not have to be concerned with the lower triangular part, even in the case where $h$ is non-differentiable.

### 3.4.1 Nonlinearity "for Free"

Due to this result, for example, the NICE[8] paper was able to generate impressive results using only additive coupling layer and learned $h$ as a neural network, without adding computational costs.

Since the lower triangular region can help us perform operations "for free", there are research efforts focused on populating that region as much as possible. The HINT[6] paper offers a recursive construction. It has been shown that multiple layers of coupling flow is an *universal approximator*, however one needs to enforce sufficient "mixing", namely permuting the state variables so that the information from each can be learned. A permutation of variables can be viewed as a volume-preserving invertible map as well.

## 3.5 Autoregressive Flow

Autoregressive flow is one of the commonly used flow structures that have conveniently computable determinant. This is done by letting the flow map of $x_i$ depend on certain parameters generated via all previous latent variables $z_{<i}$ but not "future ones" $z_{\geq i}$.

It takes the general form:

$$x_i = f(z_i, h_i)$$

$$h_i = h_i(z_{<i})$$

here $f$ needs to be strictly monotonic, and $h_i$ serves as parametrizations. Similar to the coupling layers, the Jacobian matrix also takes a lower triangular form, since $x_i = f(z_i, h_i(z_1, z_2, \cdots, z_{i-1}))$ has dependence on all the previous variables, the upper triangular derivatives are all 0:

$$\frac{dx}{dz} = \begin{pmatrix} \frac{\partial x_1}{\partial z_1} & & 0 \\ & \ddots & \\ A & & \frac{\partial x_d}{\partial z_d} \end{pmatrix}$$

where $A$ is the rest of the derivatives, however, we do not need to explicitly know its form (and hence saving us computations) since the determinant is still the diagonal product:

$$\left|\det\left(\frac{dx}{dz}\right)\right| = \prod_{i=1}^{d} \frac{\partial x_i}{\partial z_i}$$

Autoregressive flows can be shown to be universal approximators, and equivalently, any autoregressive model is an autoregressive flow of one layer. Due to the lower triangular property, similar to coupling layers, $h(\cdot)$ can also be quite flexible (it does not need to be invertible).

The training process of an autoregressive model can be understood as the following:

(1) Decompose the joint probability by product rule.

$$p(x_1, x_2, \cdots, x_d) = \prod_{i=1}^{d} p(x_i|x_1, x_2, \cdots, x_{i-1}) = \prod_{i=1}^{d} p(x_i|x_{<i})$$

(2) Model $p(x_i|x_{<i})$ as a parametrized single variable distribution (such as a normal, or mixture Gaussian):

$$p(x_i|x_{<i}) = p(x_i; h_i(x_{<i}))$$

By the conditional distribution sampling result, let:

$$x_i = f(z_i, h_i) := F_X^{-1}(z_i; h_i)$$

where:

$$F_X(x_i; h_i) = \int_{-\infty}^{x_i} p(x_i; h_i) dx_i$$

we see that this is an autoregressive flow with $z \sim U([0, 1])$ as base distribution.

One main drawback of autoregressive flow is that computation of the inverse is sequential and cannot be parallelized:

$$z_i = f^{-1}(\cdot; h_i)(x_i)$$

in order to invert $z_i$, one must wait until all $x_{<i}$ have been inverted in order to have access to $z_{<i}$ information required in the parametrization of $h_i$.

A modification, called Inverse Autoregressive Flow moves the issue "to the other side" (does not solve the issue [4]) such that inversion is parallelizable, but not the generative direction. IAF redefines the conditioning variables:

$$x_i = f(z_i; h_i)$$

$$h_i = h_i(x_{<i})$$

the information or correlation among these variables learned are still the same, thus we expect the same expressive power as AF.

### 3.5.1 special case: elementwise flow

To make parallelization always possible, one can also consider flowing each variable independently of all others, namely:

$$x_i = f(z_i)$$

which is the same as autoregressive flow but with no conditioning on any variables. This limits expressive power as it removes mixing in between variables, but it is highly parallelizable with very cheap determinant:

$$\left|\det\left(\frac{dx}{dz}\right)\right| = \prod_{i=1}^{d} \frac{dx_i}{dz_i}$$

## 3.6 Continuous-Time Flow

So far the flows that we have discussed are characterized by stages of discrete applications of finite transformations $f_i$:

$$x = f_K \circ f_{K-1} \circ \cdots f_2 \circ f_1(z)$$

we can naturally consider the indices $\{1, 2, \cdots, K\}$ to be discrete time snapshots of a continuous flow $f_t$ in the space of diffeomorphisms, where the group algebra is $f_{t_1} \circ f_{t_2} = f_{t_1+t_2}$ to reflect the fact that compositions of two flows that are infinitesmally close in time is still a valid flow. Equivalently one can consider the discrete compositions to be evolutions of the base distribution $z \sim p_Z(z)$:

$$\begin{aligned} z_1 &= id(z_0) \\ z_2 &= f_1(z_1) \\ z_3 &= f_2(z_2) \\ &\vdots \\ z_{K+1} &= f_K(z_K) \\ x &= z_{K+1} \end{aligned}$$

which is equivalent to discretizing a continuous underlying ODE:

$$\frac{d}{dt}z(t) = f(t, z(t); \theta(t))$$

Given base distribution $z \sim p_Z(z)$, we can obtain a complex distribution $x \sim p_X(x)$ by using the fundamental theorem of calculus:

$$x = z_T = z_0 + \int_0^T f(t, z(t); \theta(t))dt = z + \int_0^T f(t, z(t); \theta(t))dt$$

or:

$$z = x - \int_0^T f(t, z(t); \theta(t))dt$$

is our flow in normalizing direction.

A continuous-time flow has certain advantages over discrete flows:

• ease of inverse flow.

• requires less parameters to train than discrete counterparts.

• flexible tolerance as the granularity can be made arbitrarily fine.

• optimization only requires ODE solvers, instead of complex graph back-propogation.

Below we present two proofs to demonstrate why ODE flows require fewer parameters in training.

**Theorem 3.1** *(Change of variables formula, continuous case)*

Let $z_t \in \mathbb{R}^d$ be a random variable distributed according to $p_Z(z_t)$. And:

$$\frac{dz_t}{dt} = f(t, z_t)$$

describe the dynamics of $z_t$. If $f : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$ is uniformly Lipschitz continuous in $z_t$ and continuous in $t$, then the log probability follows an ODE:

$$\frac{\partial \log p_Z(z_t)}{\partial t} = -\operatorname{tr}\left(\frac{\partial f}{\partial z_t}\right)$$

Proof:

We use the definition of the partial derivative in $t$:

$$\frac{\partial \log p(z(t))}{\partial t} = \lim_{\epsilon \to 0^+} \frac{\log p(z(t+\epsilon)) - \log p(z(t))}{\epsilon}$$

One can consider the $\epsilon$ time evolution to be a mapping $f_\epsilon : \mathbb{R}^d \to \mathbb{R}^d$ as claimed in the beginning of the section (in the space of diffeomorphisms). We write:

$$z(t+\epsilon) = f_\epsilon(z(t)) := z(t) + \epsilon \frac{dz(t)}{dt} + O(\epsilon^2) + \cdots$$

Certainly:

$$\lim_{\epsilon\to 0^+} z(t+\epsilon) = z(t)$$

which means:

$$\lim_{\epsilon\to 0^+} f_\epsilon = id$$

in the space of diffeomorphisms.

Then by the change of variables formula:

$$p(z(t+\epsilon)) = p(f_\epsilon(z(t))) = p(z(t)) \cdot \left|\det\left(\frac{df_\epsilon(z(t))}{dz(t)}\right)\right|^{-1}$$

such that:

$$\log p(z(t+\epsilon)) = \log p(z(t)) - \log\left|\det\left(\frac{df_\epsilon}{dz(t)}\right)\right|$$

Substitute this result back:

$$\frac{\partial \log p(z(t))}{\partial t} = \lim_{\epsilon\to 0^+} \frac{\log p(z(t)) - \log\left|\det\left(\frac{\partial f_\epsilon}{\partial z}\right)\right| - \log p(z(t))}{\epsilon}$$

$$= \lim_{\epsilon\to 0^+} -\frac{\log\left|\det\left(\frac{\partial f_\epsilon}{\partial z}\right)\right|}{\epsilon}$$

here both numerator and denominator goes to 0 as $\epsilon \to 0$. Specifically:

$$\lim_{\epsilon\to 0^+} \log\left|\det\left(\frac{\partial f_\epsilon}{\partial z}\right)\right| = \log|\det I| = 0$$

therefore we can use L'Hopital's rule:

$$= \lim_{\epsilon\to 0^+} -\left|\det\left(\frac{\partial f_\epsilon}{\partial z}\right)\right| \cdot \frac{\partial}{\partial \epsilon}\left|\det\left(\frac{\partial}{\partial z} f_\epsilon(z)\right)\right|$$

we have shown above:

$$\lim_{\epsilon\to 0^+} \left|\det\left(\frac{\partial f_\epsilon}{\partial z}\right)\right| = 1$$

then we are left with:

$$= -\lim_{\epsilon\to 0^+} \frac{\partial}{\partial \epsilon}\left|\det\left(\frac{\partial}{\partial z} f_\epsilon(z(t))\right)\right|$$

one of the conditions that the authors from NeurODE implicitly assumed is that it the continuous flow is orientation-preserving, namely the determinant is positive, otherwise one will run into the issue of absolute value when differentiating with respect to $\epsilon$:

$$\frac{\partial}{\partial\epsilon}|\det(\cdot)| = \frac{\partial}{\partial\epsilon}\det(\cdot)$$

using this result along with the Jacobi's formula:

$$= -\lim \operatorname{tr}\left(\operatorname{adj}(\frac{\partial f_\epsilon}{\partial z})\cdot\frac{\partial}{\partial\epsilon}\frac{\partial}{\partial z}f_\epsilon(z)\right)$$

the trace operator is a continuous operator, thus we can exchange limit:

$$= -\operatorname{tr}\left(\lim_{\epsilon\to 0^+}\operatorname{adj}(\frac{\partial f_\epsilon}{\partial z})\cdot\frac{\partial}{\partial\epsilon}\frac{\partial}{\partial z}f_\epsilon(z)\right)$$

using again the fact that $\lim_{\epsilon\to 0^+}\frac{\partial f_\epsilon}{\partial z} = I$, we obtain finally:

$$= -\operatorname{tr}\left(\lim_{\epsilon\to 0^+}\frac{\partial}{\partial\epsilon}\frac{\partial}{\partial z}f_\epsilon(z)\right)$$

and use the Taylor expansion definition of $f_\epsilon$, along with continuity in $z$ and in $t$ of $f$, we obtain:

$$= -\operatorname{tr}\left(\lim_{\epsilon\to 0^+}\frac{\partial}{\partial\epsilon}\frac{\partial}{\partial z}(z+\epsilon\frac{\partial z}{\partial t}+O(\epsilon^2)+\cdots)\right)$$

$$= -\operatorname{tr}\left(\lim_{\epsilon\to 0^+}\frac{\partial}{\partial z}\frac{\partial}{\partial\epsilon}(z+\epsilon\frac{\partial z}{\partial t}+O(\epsilon^2)+\cdots)\right)$$

$$= -\operatorname{tr}\left(\lim_{\epsilon\to 0^+}\frac{\partial}{\partial z}(\frac{\partial z}{\partial t}+O(\epsilon)+\cdots)\right)$$

we use the original ODE to replace $\partial z/\partial t = f(t, z(t))$. Then we have:

$$= -\operatorname{tr}\left(\frac{\partial f(t,z(t))}{\partial z}\right)$$

without the need to compute the determinant.

Concerning optimization of a cost function $L(\cdot)$, one needs to backpropogate the gradients at each layers in order to optimize the parameters. In the discrete case for one hidden layer:

$$z \to x := f(z) \to L(f(z))$$

we need the gradients:

$$(\frac{\partial L}{\partial z})^T = (\frac{\partial L}{\partial x})^T \cdot \frac{\partial x}{\partial z}$$

the transposes are to cover the fact that $\frac{\partial x}{\partial z}$ is a Jacobian matrix.

In the continuous case one can do the same. Naively, a continuous flow can be considered as a network with infinitely many layers, hence the last theorem describing the continuous evolution of the gradient.

**Theorem 3.2** *(continuous version of backpropogation) Let the continuous dynamics of $z_t = z(t)$ be defined as before. Define the continuous version of the gradient of the cost function with respect to the state $z(t)$:*

$$a(t) = \frac{\partial L}{\partial z(t)}$$

*then it satisfies the ODE:*

$$\frac{d}{dt}a(t) = -a(t)^T \frac{\partial f(t, z(t))}{\partial z(t)}$$

Proof: The key to the proof is to somehow convert this continuous form into a discrete form on an infinitesimal scale (an infinitesimally close discrete layer), and we can use the discrete form of the backpropogation from before.

Evolve the $z(t)$ dynamics by $\epsilon$ in time:

$$z(t+\epsilon) = z(t) + \int_t^{t+\epsilon} f(t, z(t))dt \approx z(t) + \epsilon f(t, z(t)) + O(\epsilon^2)$$

again similar as before, we can consider this in the space of diffeomorphisms:

$$f_\epsilon(z) := z(t) + \epsilon f(t, z(t)) + O(\epsilon^2)$$

Now we essentially have a hidden layer, $z(t+\epsilon)$, which yields:

$$\frac{dL}{dz(t)} = \frac{dL}{dz(t+\epsilon)} \cdot \frac{dz(t+\epsilon)}{dz(t)}$$

Now use the definition of derivative:

$$\frac{d}{dt}a(t) = \lim_{\epsilon\to 0^+}\frac{a(t+\epsilon)-a(t)}{\epsilon}$$

$$= \lim_{\epsilon\to 0^+}\frac{\frac{\partial L}{\partial z(t+\epsilon)} - \frac{\partial L}{\partial z(t)}}{\epsilon} = \lim_{\epsilon\to 0^+}\frac{1}{\epsilon}\left(\frac{\partial L}{\partial z(t+\epsilon)} - \frac{\partial L}{\partial z(t+\epsilon)}\cdot\frac{\partial z(t+\epsilon)}{\partial z(t)}\right)$$

$$= \lim_{\epsilon\to 0^+}\epsilon^{-1}\left(\frac{\partial L}{\partial z(t+\epsilon)}\left(I - \frac{\partial}{\partial z(t)}(z(t)+\epsilon f(t,z(t)) + O(\epsilon^2))\right)\right)$$

by Taylor expansion.

$$= \lim_{\epsilon\to 0^+}\epsilon^{-1}\left(\frac{\partial L}{\partial z(t+\epsilon)}\left(I - I - \epsilon\frac{\partial f(t,z(t))}{\partial z(t)} + O(\epsilon^2)\right)\right)$$

moving $\epsilon^{-1}$ inside the parenthesis:

$$= -\lim_{\epsilon\to 0^+}\left(\frac{\partial L}{\partial z(t+\epsilon)}\frac{\partial f(t,z(t))}{\partial z(t)} + O(\epsilon)\right) = -\frac{\partial L}{\partial z(t)}\cdot\frac{\partial f(t,z(t))}{\partial z(t)}$$

then use the fact that $z(t+\epsilon)\to z(t)$ as $\epsilon\to 0$, we finally obtain the last equality as desired.

The fact that the above two problems can be solved by an ODE solver implies that density evaluation and sampling can be both made symmetrically efficient [1] [3].

## References

[1] Tian Qi Chen et al. "Neural Ordinary Differential Equations". In: *CoRR* abs/1806.07366 (2018). arXiv: 1806.07366. URL: http://arxiv.org/abs/1806.07366.

[2] Christina Gao, Joshua Isaacson, and Claudius Krause. "i- flow: High-dimensional integration and sampling with normalizing flows". In: *Machine Learning: Science and Technology* 1.4 (Nov. 2020), p. 045023. ISSN: 2632-2153. DOI: 10.1088/2632-2153/abab62. URL: http://dx.doi.org/10.1088/2632-2153/abab62.

[3] Will Grathwohl et al. *FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models*. 2018. arXiv: 1810.01367 [cs.LG].

[4] Jonathan Ho et al. *Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design*. 2019. arXiv: `1902.00275 [cs.LG]`.

[5] Ivan Kobyzev, Simon Prince, and Marcus Brubaker. "Normalizing Flows: An Introduction and Review of Current Methods". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), pp. 1–1. ISSN: 1939-3539. DOI: `10.1109/tpami.2020.2992934`. URL: `http://dx.doi.org/10.1109/TPAMI.2020.2992934`.

[6] Jakob Kruse et al. *HINT: Hierarchical Invertible Neural Transport for Density Estimation and Bayesian Inference*. 2021. arXiv: `1905.10687 [stat.ML]`.

[7] George Papamakarios et al. *Normalizing Flows for Probabilistic Modeling and Inference*. 2021. arXiv: `1912.02762 [stat.ML]`.

[8] Danilo Jimenez Rezende and Shakir Mohamed. *Variational Inference with Normalizing Flows*. 2016. arXiv: `1505.05770 [stat.ML]`.