FFJORD: Free-Form Continuous Dynamics For Scalable Reversible Generative Models

Joeri Hermans, University of Liège

INFO8004: Advanced Machine Learning

5th of April, 2019

ABSTRACT

A promising class of generative models maps points from a simple distribution to a complex distribution through an invertible neural network. Likelihood-based training of these models requires restricting their architectures to allow cheap computation of Jacobian determinants. Alternatively, the Jacobian trace can be used if the transformation is specified by an ordinary differential equation. In this paper, we use Hutchinson's trace estimator to give a scalable unbiased estimate of the log-density. The result is a continuous-time invertible generative model with unbiased density estimation and one-pass sampling, while allowing unrestricted neural network architectures. We demonstrate our approach on high-dimensional density estimation, image generation, and variational inference, achieving the state-of-the-art among exact likelihood methods with efficient sampling.

What is a generative model?

> A probabilistic model as a simulator of the data and is typically applied as an approach for unsupervised learning of data.

The purpose is to generate high-dimensional data

$$\mathbf{x} \sim p(\mathbf{x}; \theta),$$

from the unknown but true data distribution $p(\mathbf{x})$. If we can represent a dataset of samples with a distribution, we can:

1. Generate new data by sampling from the learned distribution $p(\mathbf{x}; \theta)$.
2. Evaluate the likelihood of the observed data.
3. Finding the conditional relationship between variables.

We are pretty good at sampling $\mathbf{x} \sim p(\mathbf{x}; \theta)$.

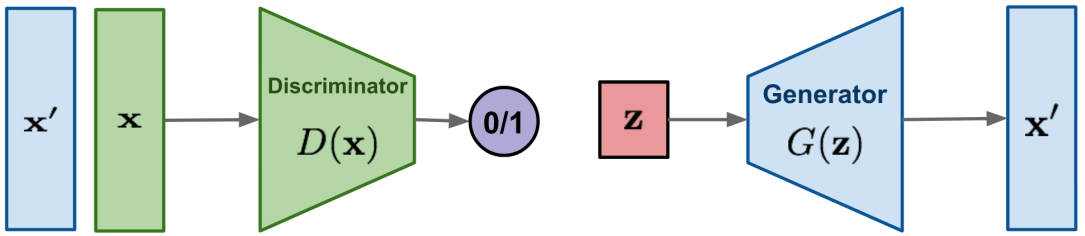
---

https://www.thispersondoesnotexist.com/
https://arxiv.org/pdf/1812.04948.pdf

However, we are not very good at estimating how likely the data is.

Generative Adversarial Networks

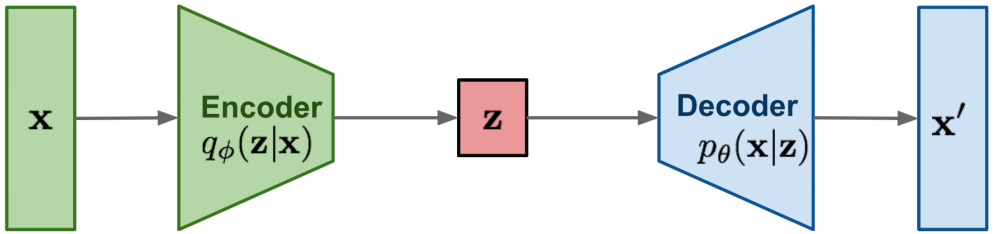
GAN: minimax the classification error loss.



- The generator $\mathcal{G}_\theta(\cdot) : \mathcal{Z} \to \mathcal{X}$ transforms samples from a base distribution $p(\mathbf{z})$ to data space.
- No closed-form likelihood $\to$ discriminator provides a learning signal.

Variational Autoencoders

VAE: maximize ELBO.



The likelihood $p(\mathbf{x}|\mathbf{z})$ is parameterized by a decoder network, which are conditioned on latent variables $\mathbf{z}$.

The posterior $p(\mathbf{z}|\mathbf{x})$ is parameterized by an encoder network.

Goal: what is the posterior $p(\mathbf{z}|\mathbf{x})$ given observations $\mathbf{x}$? Bayes' rule gives us:

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}$$

This depends on the evidence $p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$, which can not be evaluated practically. Therefore, we need to rely on approximations.

Variational inference

> Approximate $p(\mathbf{z}|\mathbf{x})$ with a family of tractable distributions $q_\phi(\mathbf{z}|\mathbf{x})$.
> This turns posterior inference into an optimization problem!

How well does $q_\phi(\mathbf{z}|\mathbf{x})$ approximate the true posterior $p(\mathbf{z}|\mathbf{x})$?

$$KL(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) = \mathbb{E}_q\left[\log q(\mathbf{z}|\mathbf{x})\right] - \mathbb{E}_q\left[\log p(\mathbf{x},\ \mathbf{z})\right] + \log p(\mathbf{x})$$

Thus, the optimal q_ϕ^* is the one which minimizes the KL-divergence, but this depends on $p(\mathbf{x})$...

Notice that the marginal model does not depend on ϕ (independent of the q_ϕ), and can therefore be threated as a constant:

$$-\mathbb{E}_q\left[\log q(\mathbf{z}|\mathbf{x})\right] + \mathbb{E}_q\left[\log p(\mathbf{x},\ \mathbf{z})\right] = \log p(\mathbf{x}) - KL(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x}))$$

$$-\mathbb{E}_q\left[\log q(\mathbf{z}|\mathbf{x})\right] + \mathbb{E}_q\left[\log p(\mathbf{x},\ \mathbf{z})\right] \leq \log p(\mathbf{x})$$

Other approaches include auto-regressive models:

$$p(\mathbf{x}) = \prod_{d}^{D} p(x_d | \mathbf{x}_1, ..., \mathbf{x}_d)$$

Figure 1. Image completions sampled from a PixelRNN.

Active research-area!

MADE: Masked Autoencoder for Distribution Estimation https://arxiv.org/pdf/1502.03509.pdf

Can we find better representation?

Ability to model rich, multi-modal data distributions.

Retain the comfort of a Normal distribution: sampling, density evaluation?

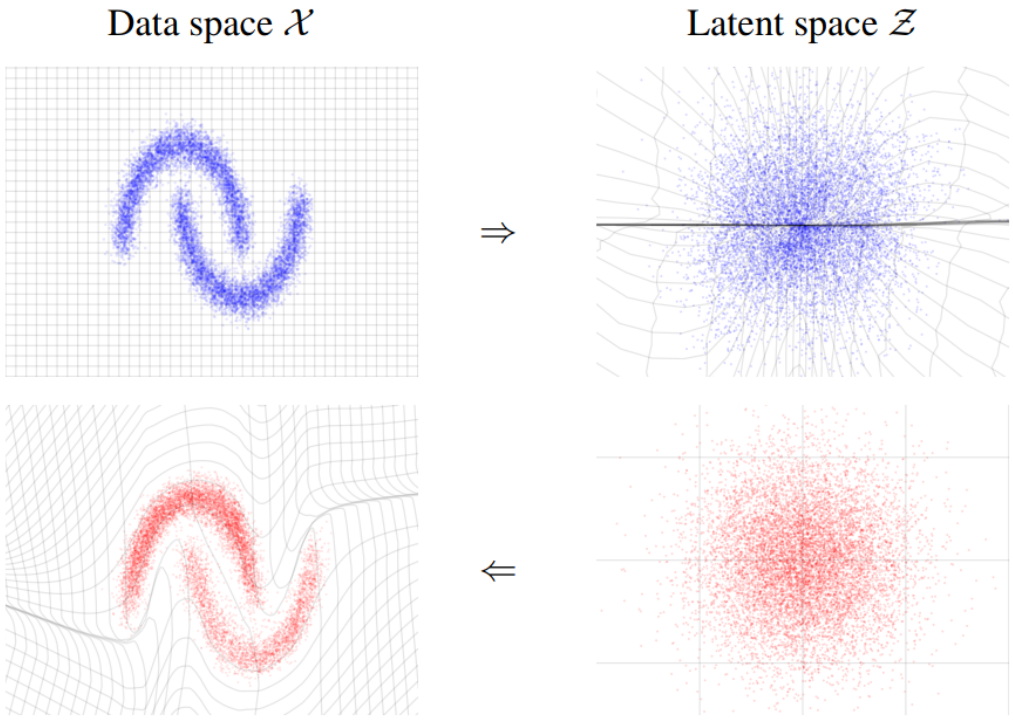
Invertible generative models

Inference

$$x \sim \hat{p}_X$$
$$z = f(x)$$

Generation

$$z \sim p_Z$$
$$x = f^{-1}(z)$$



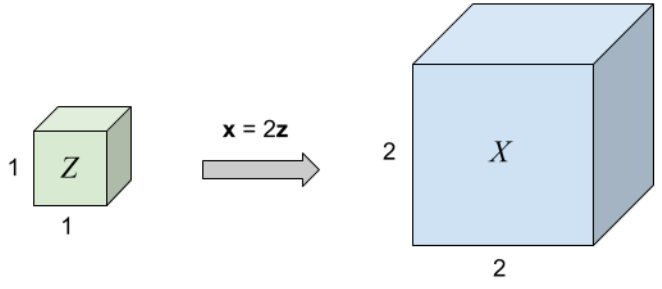
NICE: Non-Linear Independent Components Estimation https://arxiv.org/pdf/1410.8516.pdf
Density estimation using Real NVP https://arxiv.org/pdf/1605.08803.pdf

Change of Variable Theorem (I)

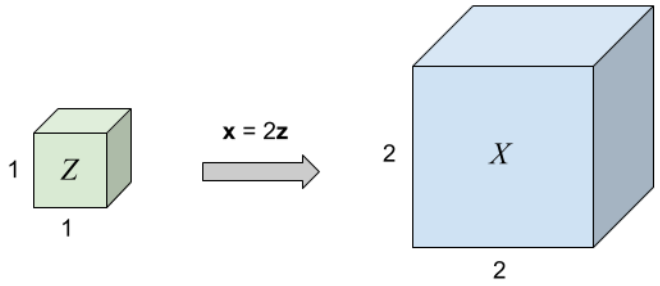
> Given a random variable $\mathbf{z}$ and its known density $p(\mathbf{z})$, we would like to construct a new random variable using a mapping $\mathbf{x} = f(\mathbf{z})$.

How does the density $p(\mathbf{z})$ relate to $p(\mathbf{x})$?

Assume $p(\mathbf{z})$ is a uniformly distributed unit cube, and scale this by a factor of 2 to obtain a new density $p(\mathbf{x})$.



Change of Variable Theorem (II)



The volume for the larger cube is easy to compute and invertible: $V_{\mathbf{x}} = 2^3$.
The total probability mass must be conserved so:

$$p(\mathbf{x}) = p(\mathbf{z})\frac{V_z}{V_x} = p(\mathbf{z})\frac{1}{8},$$

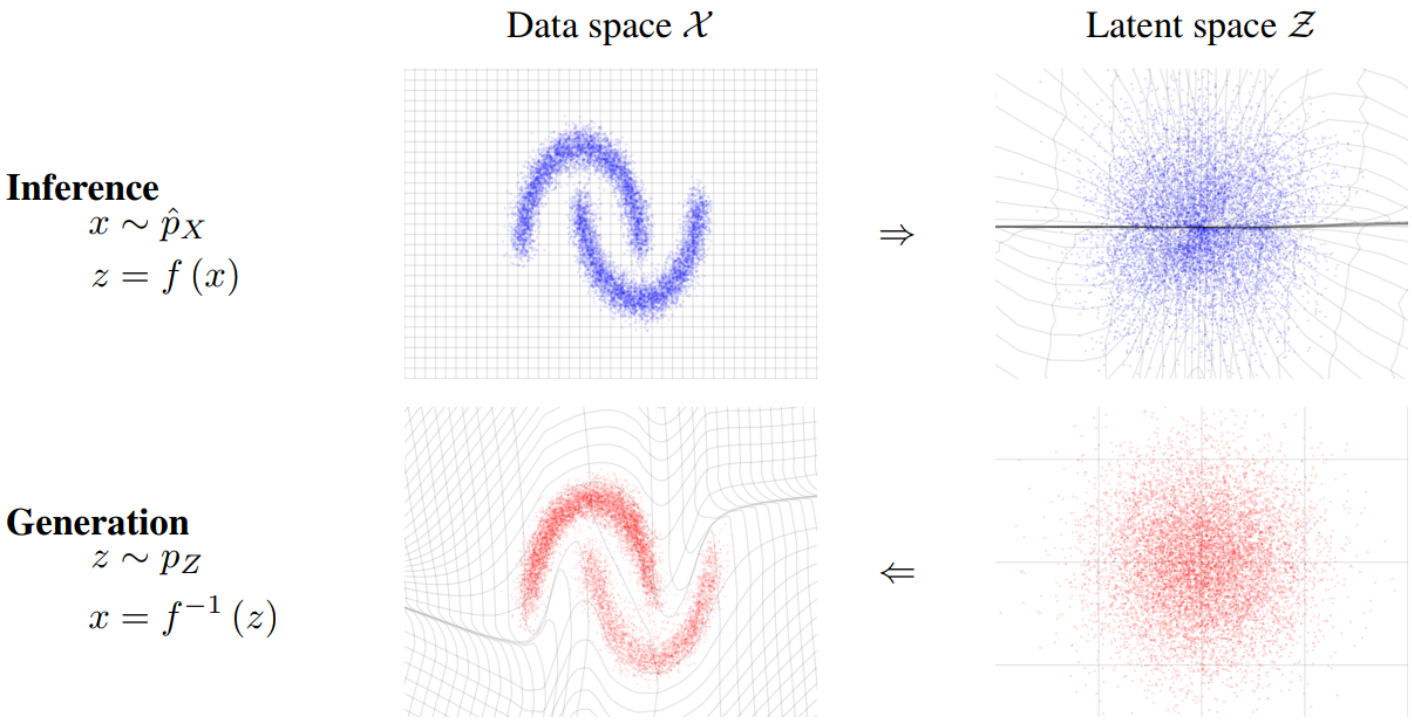
which is the determinant of the linear transformation $\left|\det \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}\right|^{-1}$.

What if the transformation is non-linear?

$$\log p(\mathbf{x}) = \log p(\mathbf{z}) - \log \det \left| \frac{\partial f(\mathbf{z})}{\partial \mathbf{z}} \right|^{-1}$$

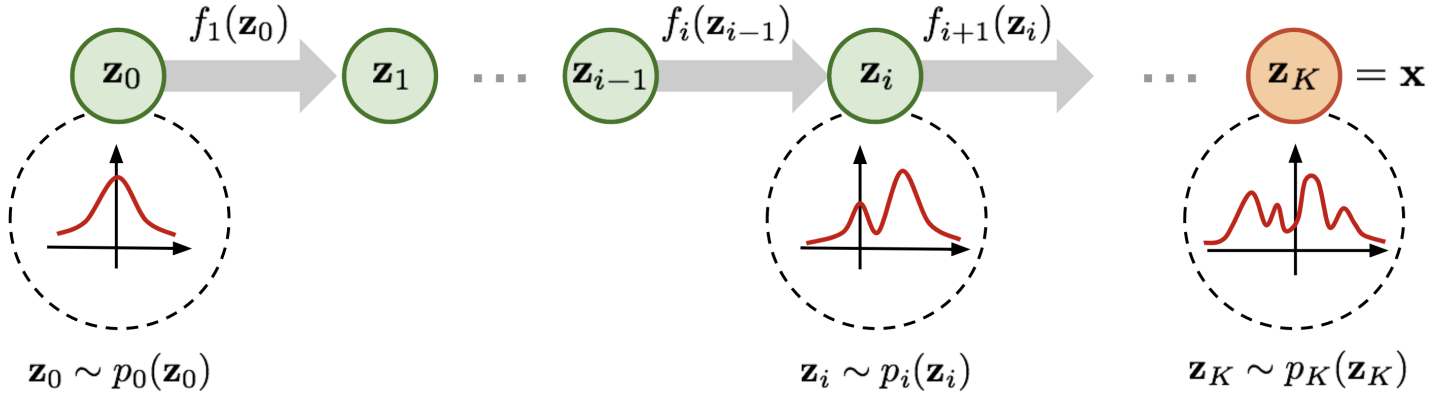
Your transformation still needs to be invertible mapping!

How should we define $f(\mathbf{z})$ and its inverse?



Normalizing Flows

> Describes the transformation of a base probability density $p(\mathbf{z})$ into a target density $p(\mathbf{x})$ through a sequence of invertible mappings $f_i(\cdot)$.

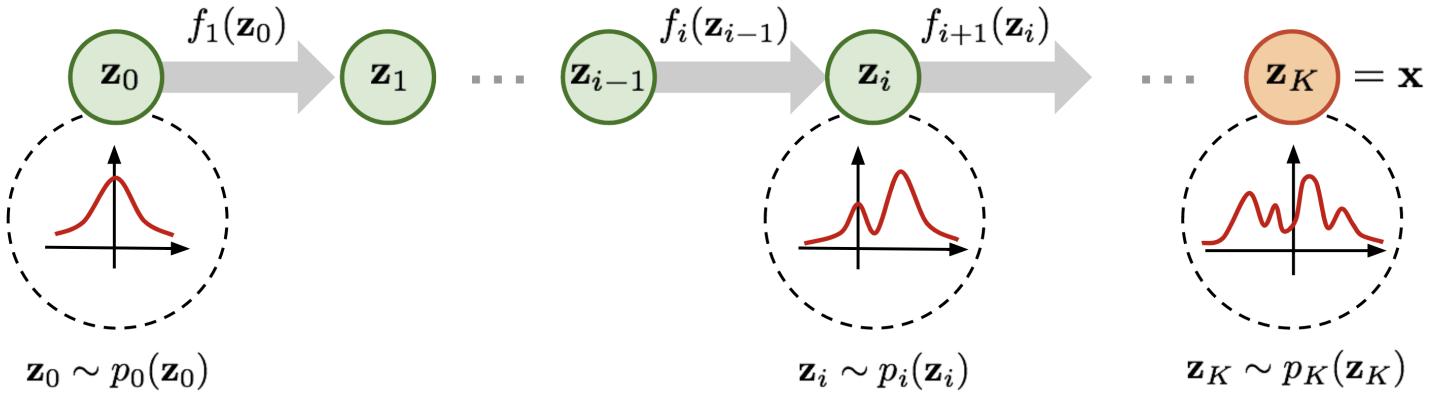


Distribution flows through a sequence of invertible mappings!

$$\mathbf{x} = f(\mathbf{z})$$

with the log-density of $\mathbf{x}$ given by

$$\log p(\mathbf{x}) = \log p(\mathbf{z}_0) - \sum_{i}^{K} \log \det \left| \frac{\partial f_i(\mathbf{z}_i)}{\partial \mathbf{z}_i} \right|^{-1}.$$



As a result:

- The initial density $p(\mathbf{z}_0)$ flows through the sequence of mappings.
- The full chain of successive operations is the normalizing flow.
- At the end of this sequence, we obtain a valid probability distribution.
- The mapping $f(\mathbf{z})$ should be easily invertible.
- The Jacobian determinant should be easy to compute.

With Normalizing Flows, the $\log p(\mathbf{x})$ becomes tractable.

$$\log p(\mathbf{x}) = \log p(\mathbf{z}) - \log \det \left| \frac{\partial f(\mathbf{z})}{\partial \mathbf{z}} \right|^{-1}$$

Simply apply $f^{-1}(\mathbf{x})$ to obtain $\mathbf{z}$ and evaluate $\mathbf{z}$ under the prior $p(\mathbf{z})$.

$$\log p(\mathbf{x}) = \log p(f^{-1}(\mathbf{x})) - \log \det \left| \frac{\partial f^{-1}(\mathbf{x})}{\partial \mathbf{x}} \right|$$

However: constructing f with sufficient expressivity is difficult.

Issues

Computational bottleneck: computing the determinant is $\mathcal{O}(D^3)$.

Expressiveness: restrict the functional form of f to use determinant identities.

Neural Ordinary Differential Equations

Several models such as recurrent neural networks or normalizing flows compose a sequence transformations of transformations to hidden states

$$h_{t+1} = h_t + f(h_t, \theta_t).$$

View these as a discritization of a continuous transformation.

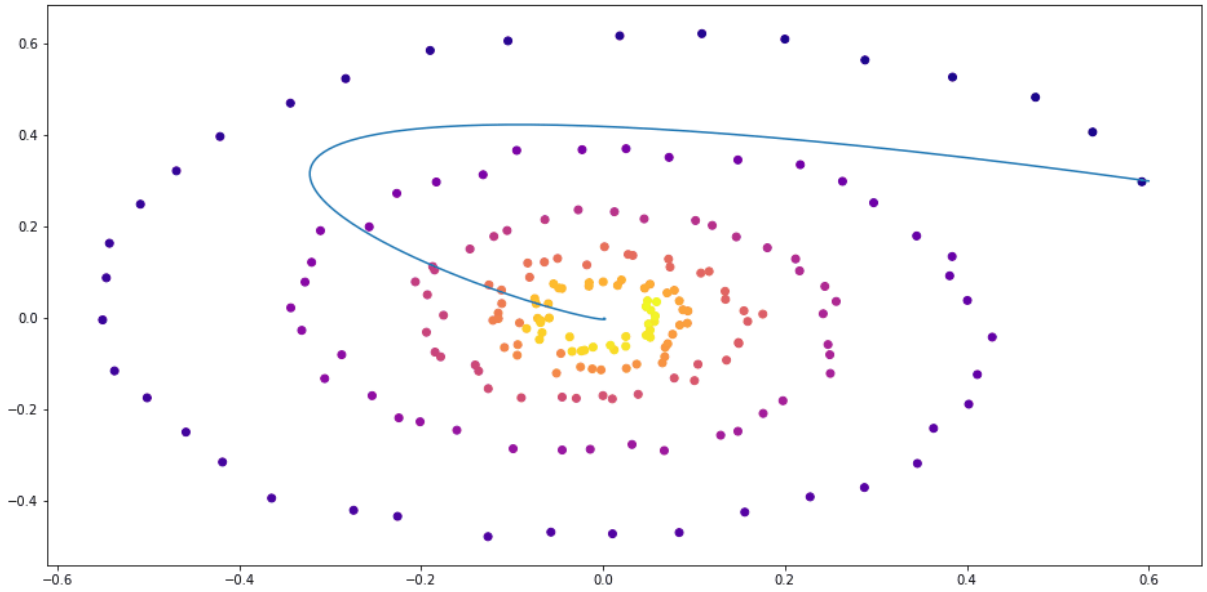
In the limit we obtain the dynamics of the differential equation

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

specified by a neural network parameterized by θ, and use an ODE solver to compute the "output".

---

https://papers.nips.cc/paper/7892-neural-ordinary-differential-equations.pdf
https://arxiv.org/pdf/1709.01179.pdf


0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6

Idea: it is easier to model a small change to an almost-correct answer than to output the whole improved answer at once.

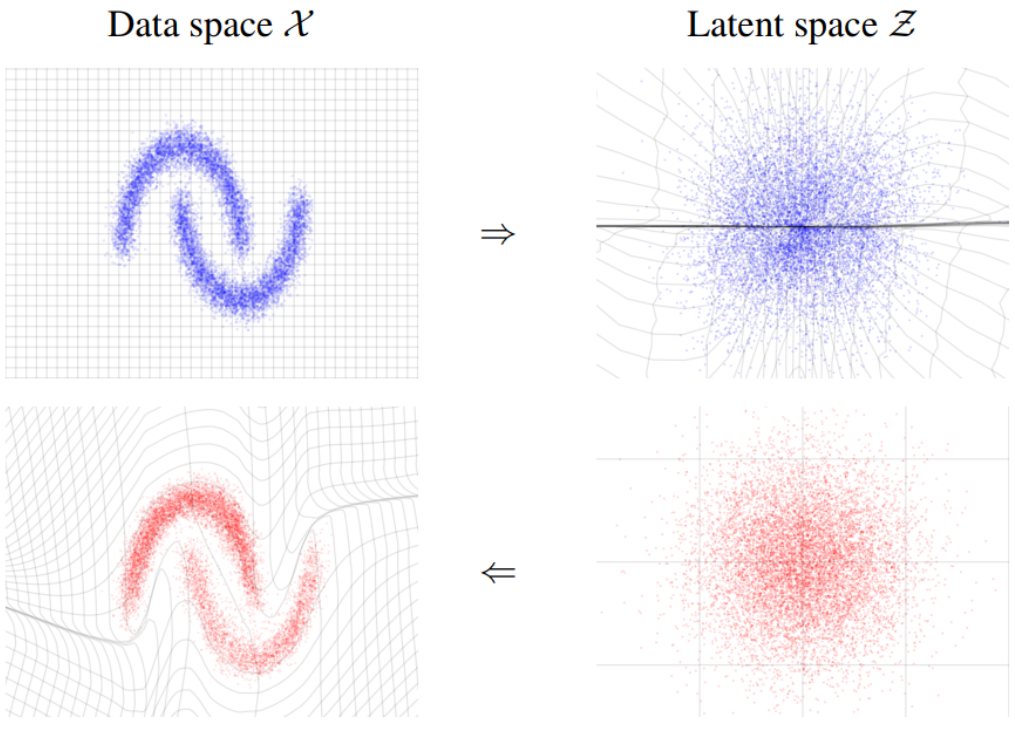
Sounds familiar?

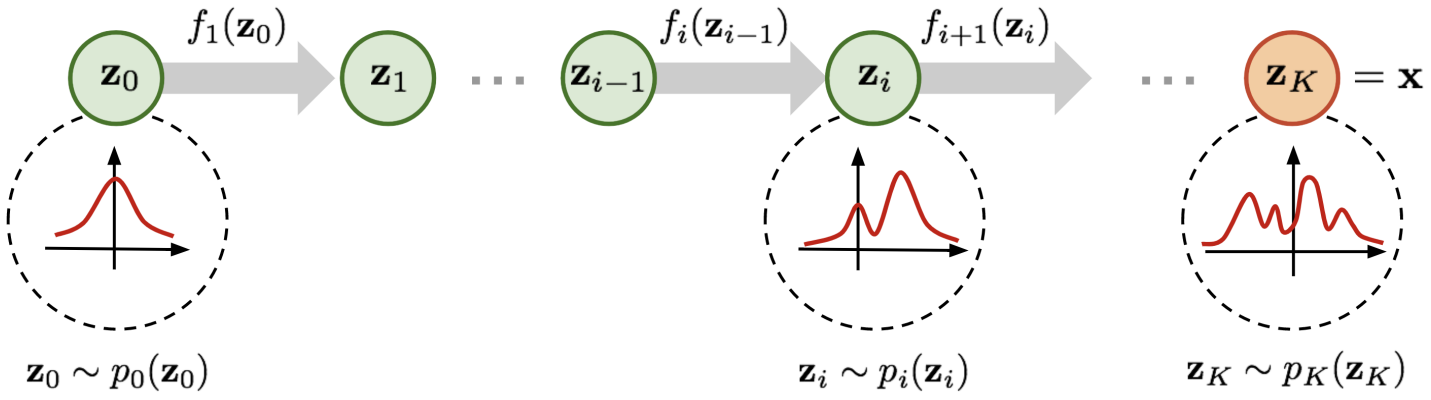
Inference

$$x \sim \hat{p}_X$$
$$z = f(x)$$

Generation

$$z \sim p_Z$$
$$x = f^{-1}(z)$$




$f_1(\mathbf{z}_0)$
$f_i(\mathbf{z}_{i-1})$
$f_{i+1}(\mathbf{z}_i)$
$\mathbf{z}_0$
$\mathbf{z}_1$
$\mathbf{z}_{i-1}$
$\mathbf{z}_i$
$\mathbf{z}_K$
$= \mathbf{x}$
$\mathbf{z}_0 \sim p_0(\mathbf{z}_0)$
$\mathbf{z}_i \sim p_i(\mathbf{z}_i)$
$\mathbf{z}_K \sim p_K(\mathbf{z}_K)$

Take this idea to the extreme

What if we define a (residual) network as a continuously evolving system?

Instead of updating hidden units layer by layer, define their derrivative with respect to depth instead.

Comments by senior author: https://news.ycombinator.com/item?id=18676986

Continuous Normalizing Flows

Recall that the previous methods use the change of variables theorem to compute exact changes in probability if samples are transformed through a bijective function.

Instantaneous Change of Variables:

- Let $\mathbf{z}(t)$ be a finite continuous random variable with density $p(\mathbf{z}(t))$.
- Let $\frac{d\mathbf{z}}{dt} = f(\mathbf{z}(t), t)$ be a differential equation describing the transformation of $\mathbf{z}(t)$.

The change in log probability also follows a differential equation:

$$\frac{\partial \log p(\mathbf{z}(t))}{\partial t} = -\mathrm{tr}\left(\frac{df}{d\mathbf{z}(t)}\right)$$

The change in log-density can be computed by integrating over time:

$$\log p(\mathbf{z}(t_1)) = \log p(\mathbf{z}(t_0)) - \int \operatorname{tr}\left(\frac{\partial f}{\partial \mathbf{z}(t)}\right) dt.$$

Scalable density evaluation

$$\frac{\partial \log p(\mathbf{z}(t))}{\partial t} = -\mathrm{tr}\left(\frac{df}{d\mathbf{z}(t)}\right)$$

We only require a trace instead of the log-determinant ($\mathcal{O}(D^2)$)

The differential equation f does not have to be bijective

Dependence on t is added as a parameterization through a hypernetwork.

At the cost of including an ODE solver.

Hypernetworks https://arxiv.org/abs/1609.09106

Use an unbiased estimator of trace (Hutchinson's trace estimator):

$$tr(A) = \mathbb{E}_{p(\epsilon)}[\epsilon^T A \epsilon]$$

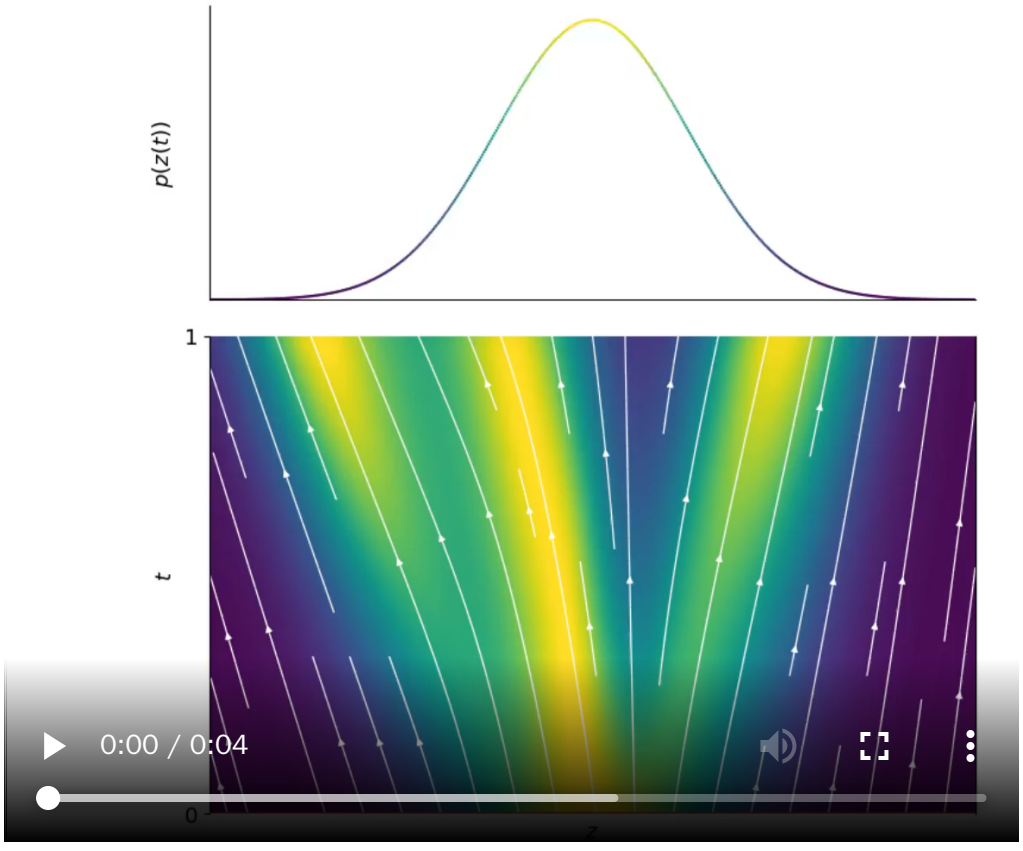
such that $\mathbb{E}[\epsilon] = 0$ and $\mathbf{Cov}[\epsilon] = I$.

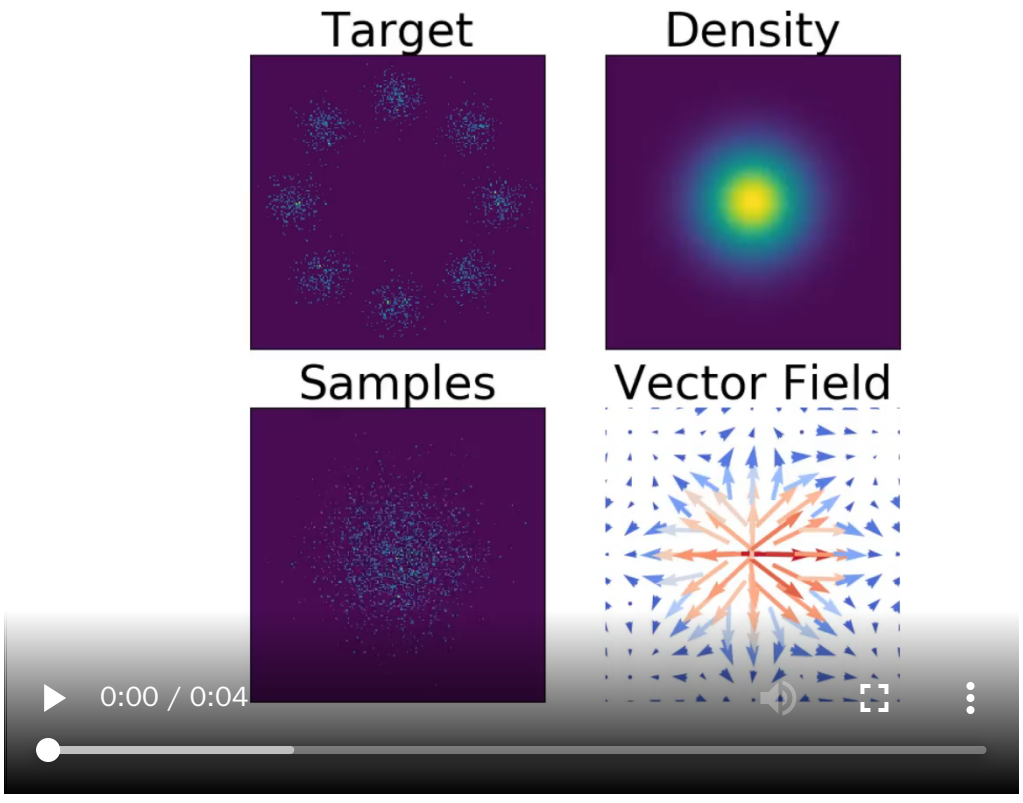
$$\begin{aligned}
\log p(\mathbf{z}(t_1)) &= \log p(\mathbf{z}(t_0)) - \int_{t_0}^{t_1} \mathrm{Tr}\left(\frac{\partial f}{\partial \mathbf{z}(t)}\right) dt \\
&= \log p(\mathbf{z}(t_0)) - \int_{t_0}^{t_1} \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\boldsymbol{\epsilon}^T \frac{\partial f}{\partial \mathbf{z}(t)} \boldsymbol{\epsilon}\right] dt \\
&= \log p(\mathbf{z}(t_0)) - \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\int_{t_0}^{t_1} \boldsymbol{\epsilon}^T \frac{\partial f}{\partial \mathbf{z}(t)} \boldsymbol{\epsilon} dt\right]
\end{aligned}$$

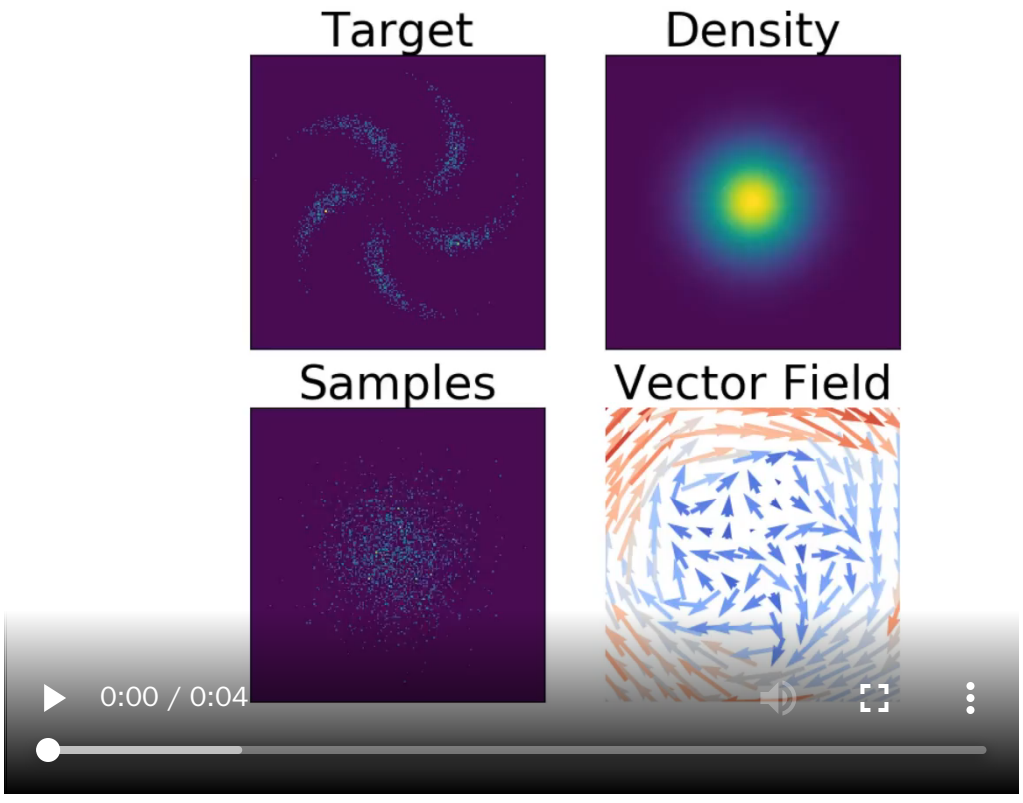
FFJORD = CNF + unbiased linear-time log-density estimation

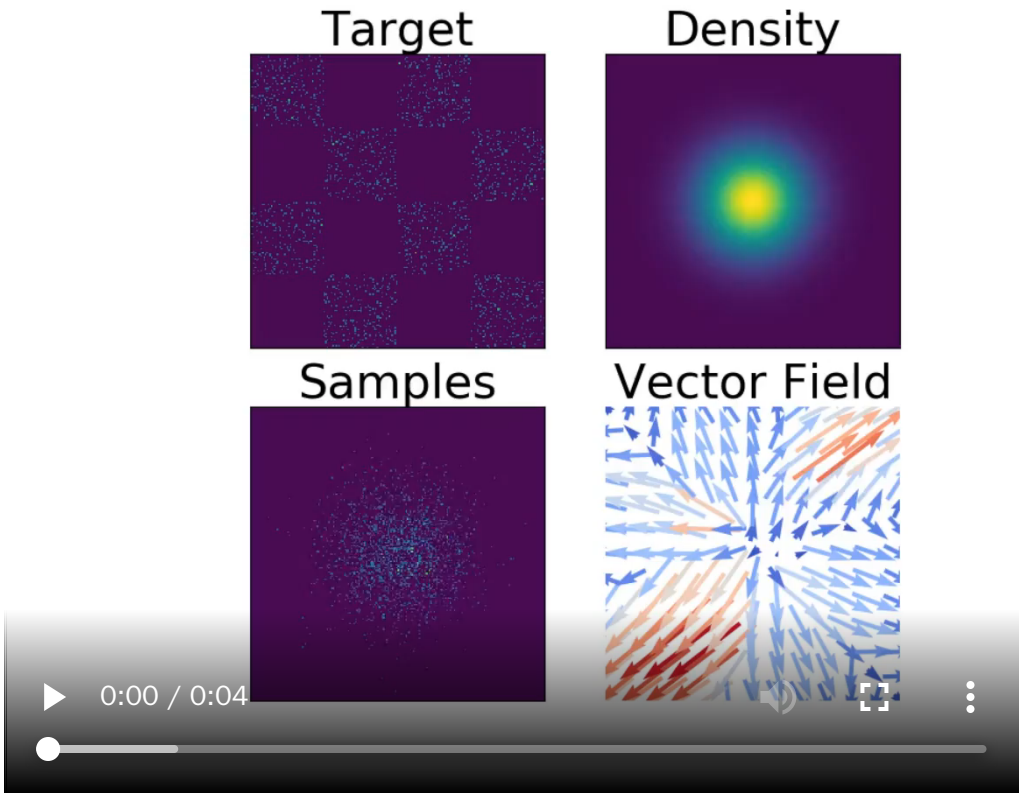
	Method	Train on data	One-pass Sampling	Exact/Unbiased Log-likelihood	Free-form Jacobian
	Variational Autoencoders	✓	✓	✗	✓
	Generative Adversarial Nets	✓	✓	✗	✓
	Likelihood-based Autoregressive	✓	✗	✓	✗
Change of Variables	Normalizing Flows	✗	✓	✓	✗
	Reverse-NF, MAF, TAN	✓	✗	✓	✗
	NICE, Real NVP, Glow, Planar CNF	✓	✓	✓	✗
	FFJORD	✓	✓	✓	✓

Table 1: A comparison of recent generative modeling approaches.


p(z(t))
1
t
0
z
0:00 / 0:04


Target
Density
Samples
Vector Field
0:00 / 0:04


Target
Density
Samples
Vector Field
0:00 / 0:04


Target
Density
Samples
Vector Field
0:00 / 0:04

Conclusion

- Neural Ordinary Differential Equations provide a nice fresh perspective.
- Some details regarding computational complexity are very vague.
- While a nice mathemathical formulation, the ODE approach quite demanding.
  - NODE's solve the issue with the expresiveness and the log-determinant.
  - Introduce an ODE solver.
- FFJORD mainly uses the unbiased estimator during training. Is it really required to have a free-form Jacobian?

fin