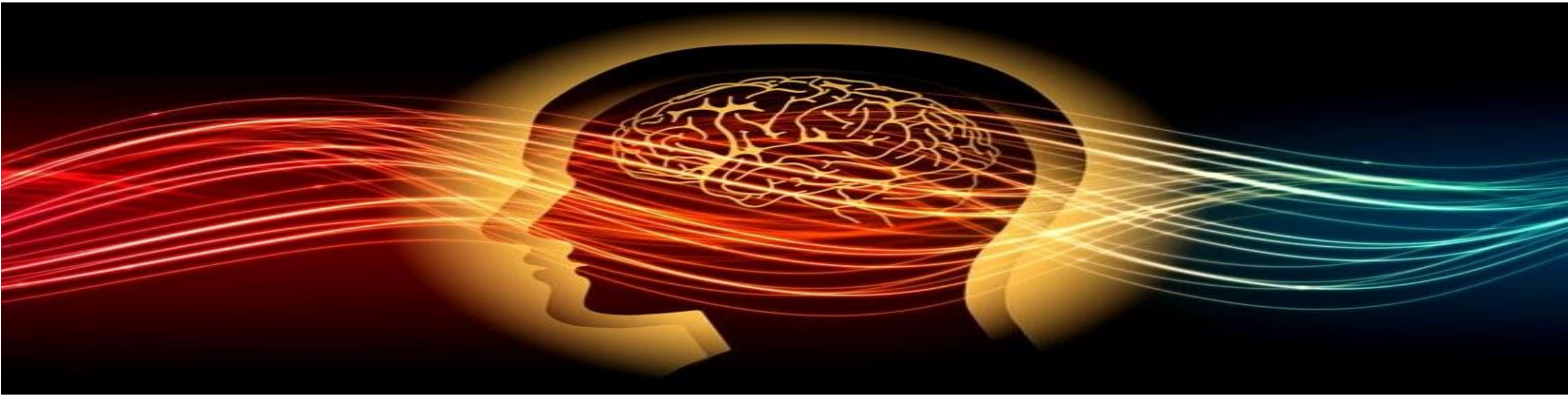
Normalizing Flow



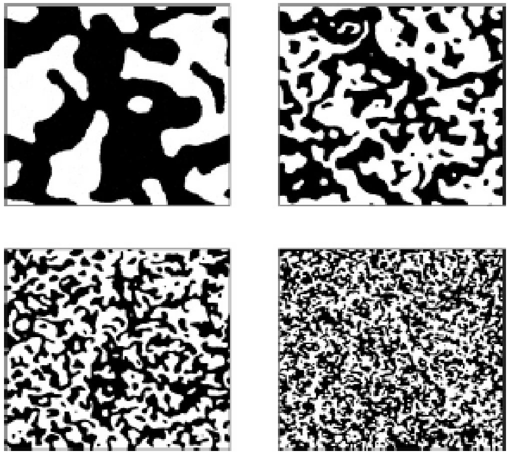
Hao Wu (吴昊)
hwu@tongji.edu.cn

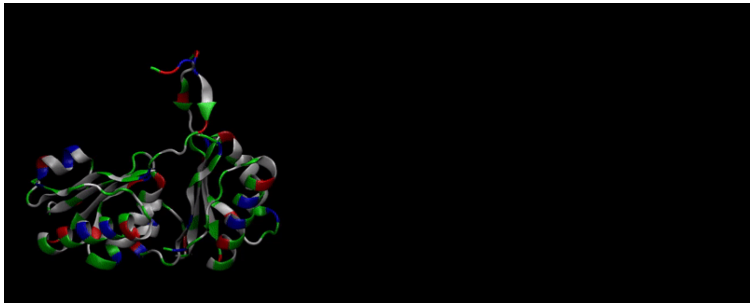
13-Oct-21

(Some contents are borrowed from Laurent Dinh's slides)

Background

Sampling from Boltzmann distributions





Input: Reduced potential energy $u(\mathbf{x})$ in coordinates $\mathbf{x} \in \mathbb{R}^n$.

Aim: Sample equilibrium (Boltzmann) distribution

$$\mu(\mathbf{x}) \propto e^{-u(\mathbf{x})}$$

Limitations of Monte Carlo / MCMC sampling

Problem 1: For large n,

$$\frac{\mathrm{Vol(low\!-\!energy\ configurations)}}{\mathrm{Vol(possible\ configurations)}} \ll 1$$

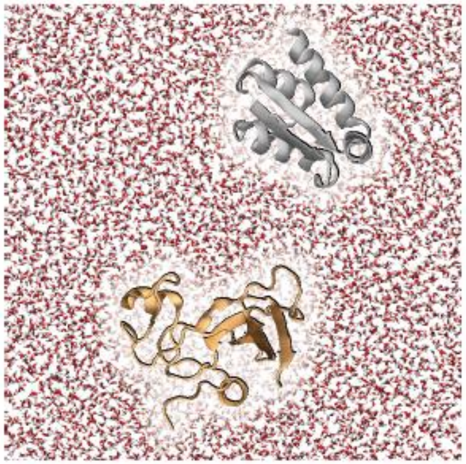
Problem 2: Multiple potential wells yields

$$\frac{\text{mixing time}}{\text{simulation time}} \geq \mathcal{O}(1)$$

- Example: Protein folding/unfolding needs $10^9 - 10^{15}$ MD simulation steps.

Direct MCMC/MD is hopeless for many-body systems.

Standard methods are INSANELY expensive



very long time

→

a lot of energy

2 ms MD = 20,000 GPU days
500 gigajoule

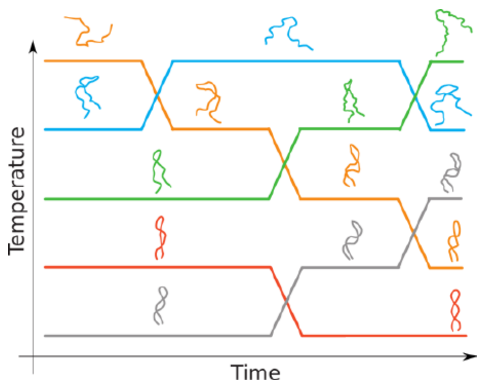
1 non-transferable model

Burn a Saturn V rocket and deliver
50 ton payload to lunar orbit

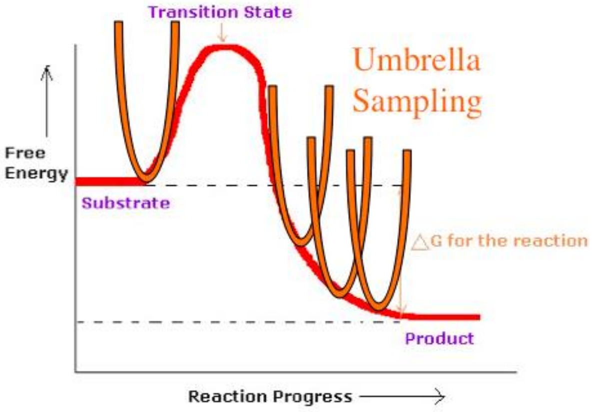
1500 gigajoule

Enhanced sampling

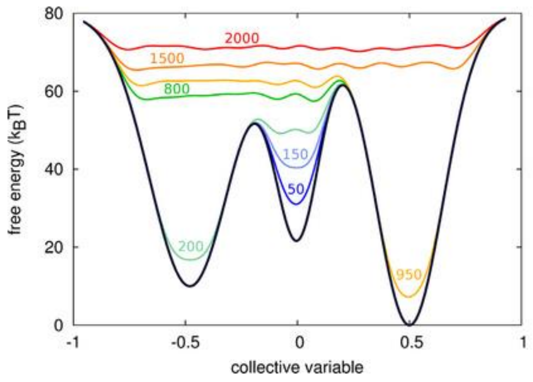
Draw samples according to biased but more “efficient” potentials



Source: Cragnolini, JPCM, 2014



Source: https://sites.google.com/site/wjfriend1/



Source: Pietrucci, Rev. Phys., 2017

Parallel tempering

Umbrella sampling

Metadynamics

Computational effort remains enormous.

Idea: Sampling in latent space

Sample tractable latent distribution:

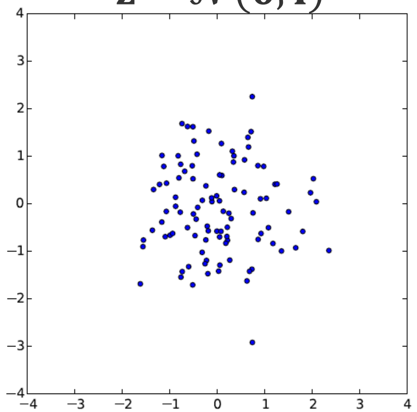
$$\mathbf{z} \sim p_Z(\mathbf{z})$$

Learn a nonlinear transformation from the latent space to the configuration space:

$$\mathbf{x} = F_{ZX}(\mathbf{z};\theta) \sim p_X(\mathbf{x}) = \mu(\mathbf{x})$$

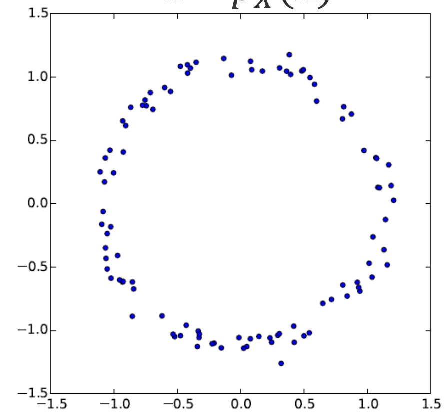
Example:

$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$



$$F_{ZX}(\mathbf{z}) = \frac{\mathbf{z}}{10} + \frac{\mathbf{z}}{\|\mathbf{z}\|}$$

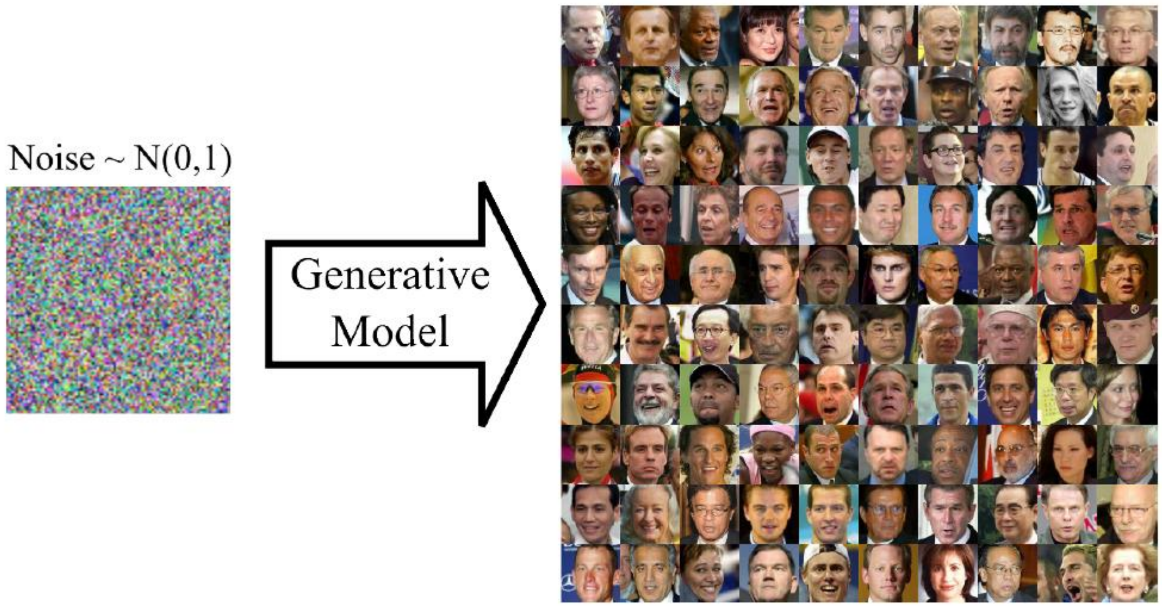
$\mathbf{x} \sim p_X(\mathbf{x})$



Computational cost can be extremely small after learning.

Idea: Sampling in latent space

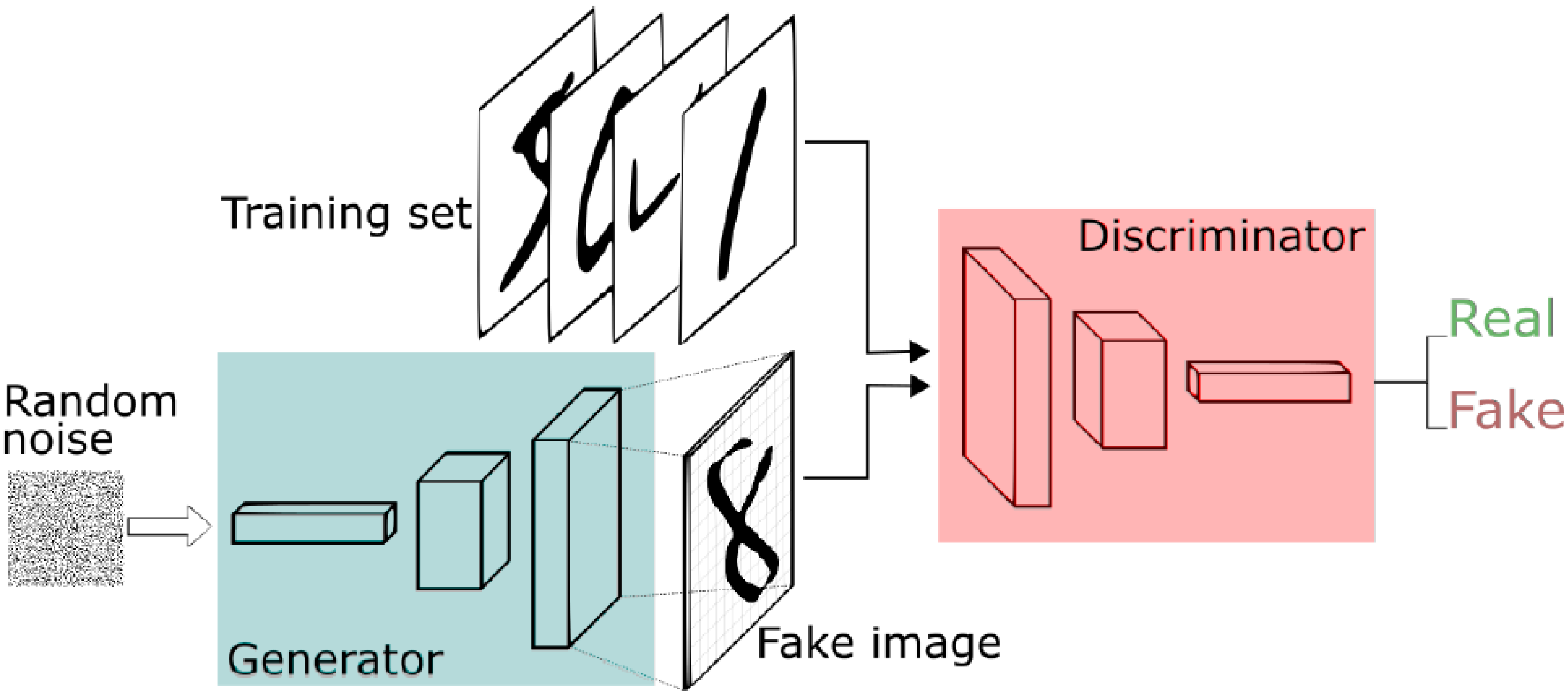
Similar ideas have been widely applied in machine learning community.



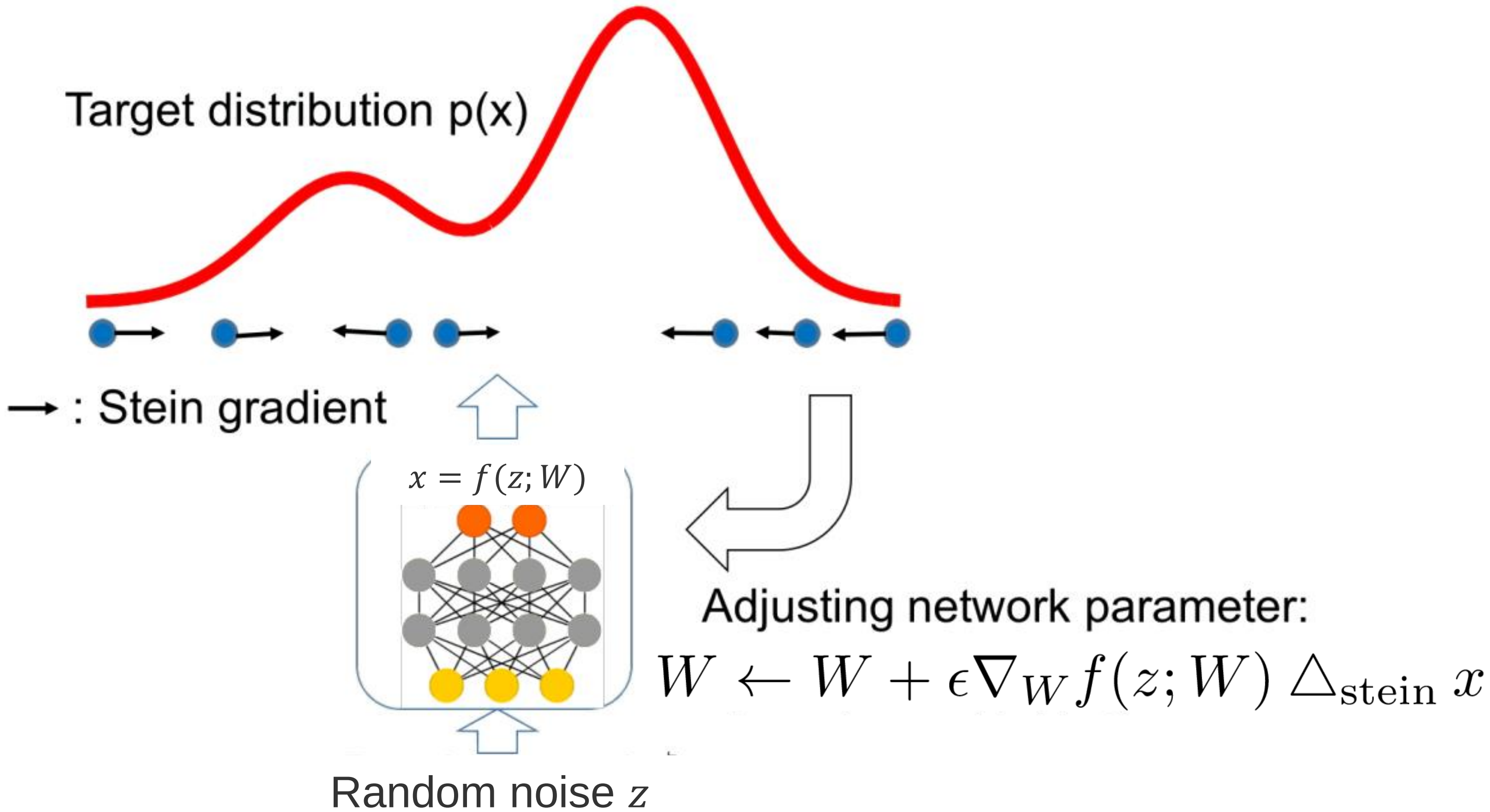
Well-known architectures:

- Generative adversarial net (Goodfellow et al., NeurIPS, 2014)
- Variational autoencoder (Kingma et al., ICLR, 2014)

Generative adversarial network (training by data)



Amortized SVGD (training by energy)

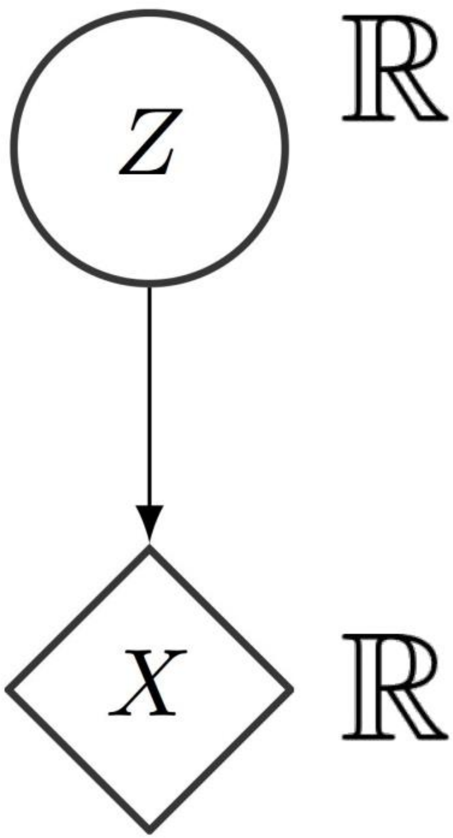


Disadvantage: The estimation bias cannot be reduced by more samples.

(Wang, arXiv:1611.01722; Liu, arXiv:1612.00081)

What if we have an invertible mapping

Scalar case



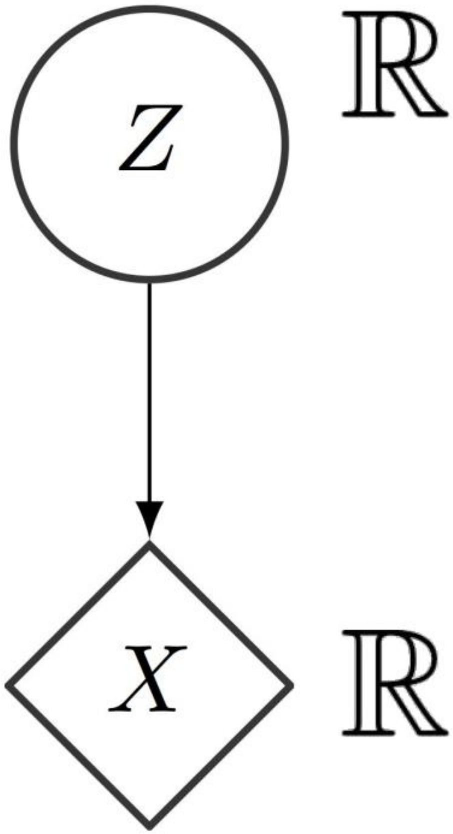
$$p_X(x)\,dx = p_Z(z)\,dz$$

Density Volume

Mass

What if we have an invertible mapping

Scalar case



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

A trivial example

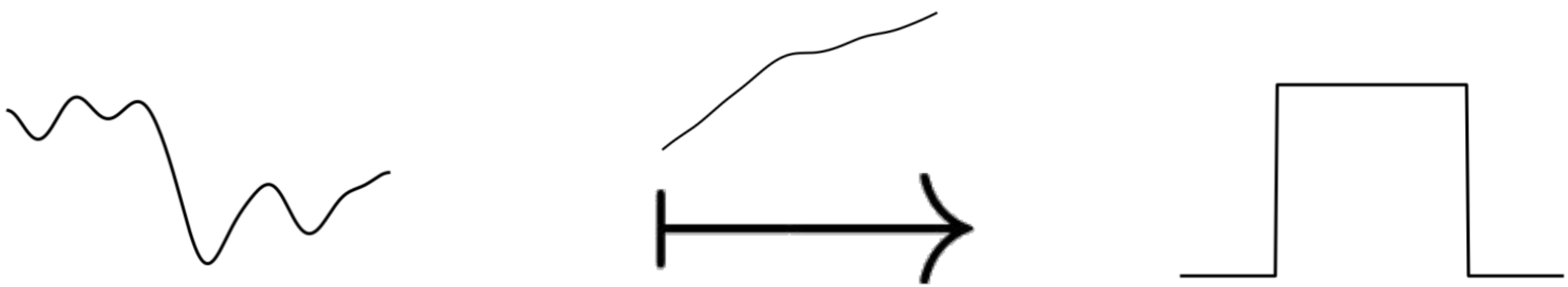
$$\mathcal{N}(x|\mu, \sigma) = \mathcal{N}(z|0, 1)\sigma^{-1}$$

$$z = \frac{x - \mu}{\sigma}$$

A non-trivial example

Inverse transform sampling $\quad x \mapsto z = CDF(x)$

$$p_X(x) = \mathcal{U}(z; [0, 1]) \frac{\partial CDF}{\partial x}(x)$$

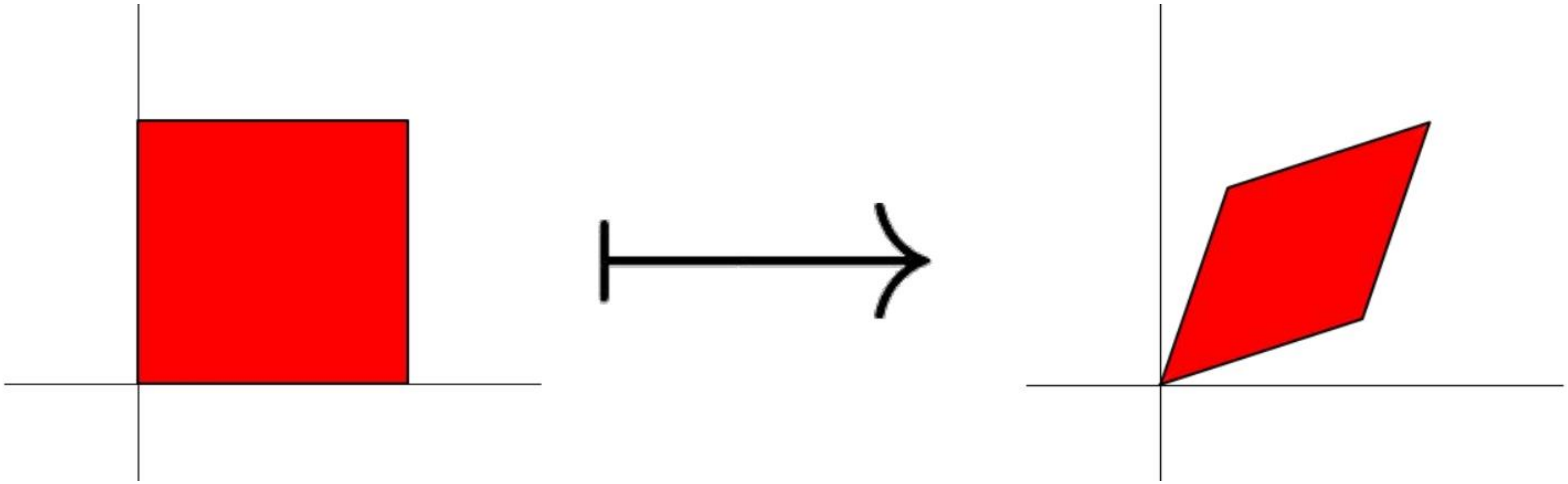


Multi-dimensional case $\mathbb{R}^d$

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

Multi-dimensional case $\mathbb{R}^d$

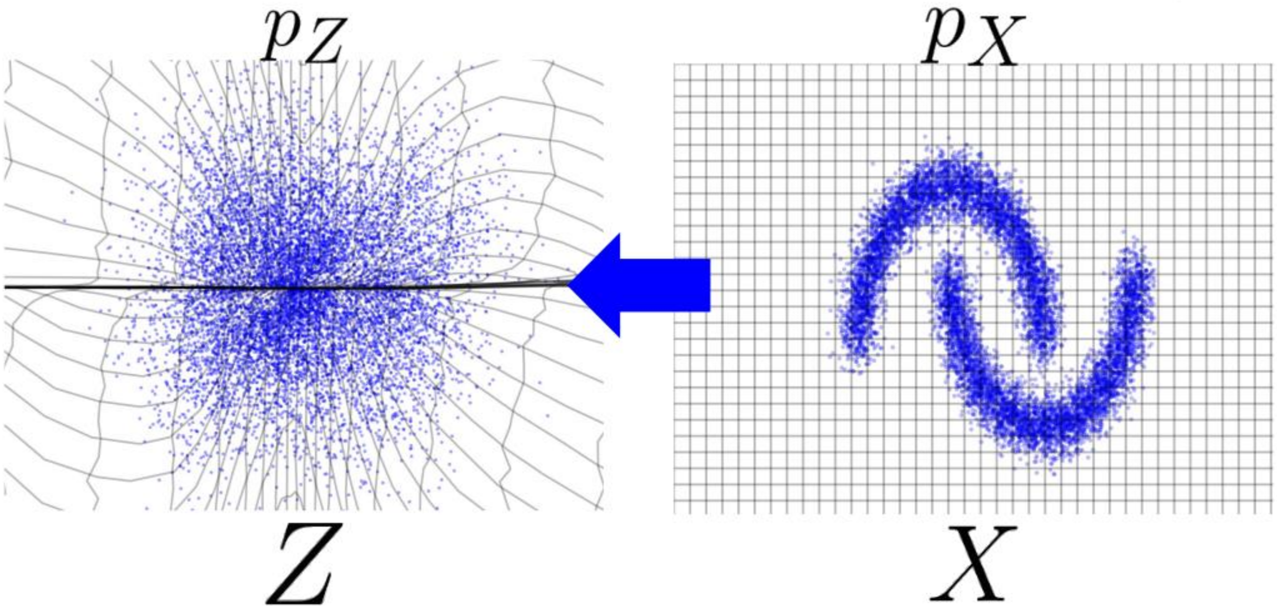
$$p_X(x) = p_Z(z) \left| \det \left(\frac{\partial z}{\partial x} \right) \right|$$



Applications

- Density estimation $f_\theta = g_\theta^{-1}$

$$\log\left(p_X^{(\theta)}(x)\right) = \log\left(p_Z\left(f_\theta(x)\right)\right) + \log\left(\left|\frac{\partial f_\theta}{\partial x}\right|(x)\right)$$

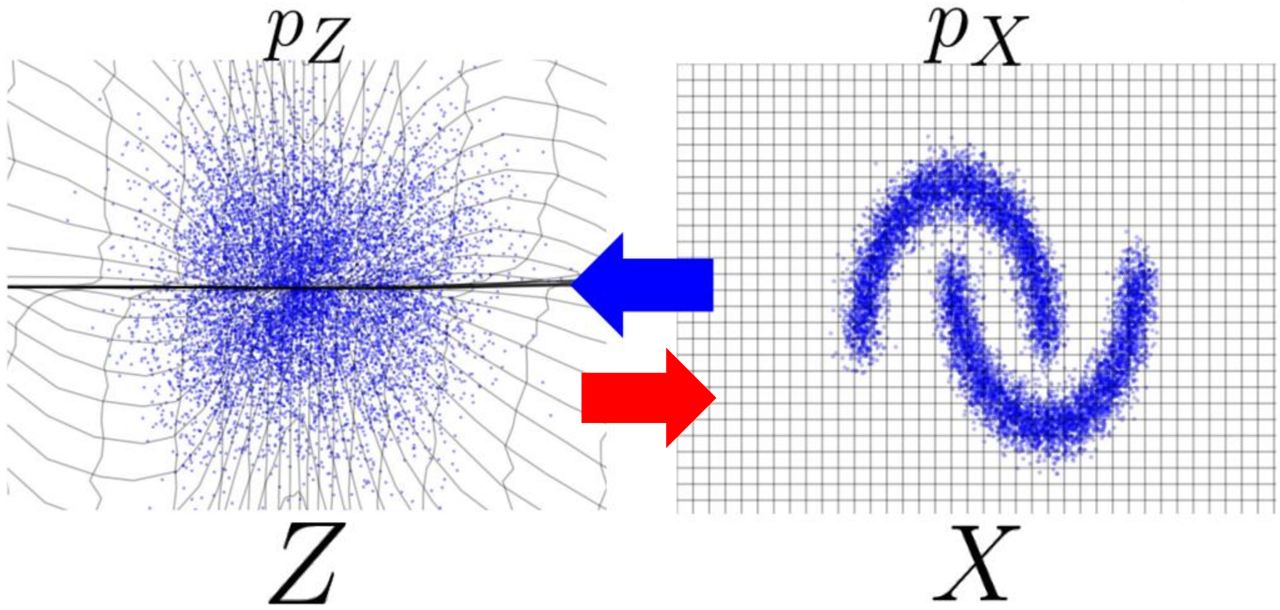


Applications

- Draw samples

$$f_\theta = g_\theta^{-1}$$

$$\log\left(p_X^{(\theta)}(x)\right) = \log\left(p_Z\left(f_\theta(x)\right)\right) + \log\left(\left|\frac{\partial f_\theta}{\partial x}\right|(x)\right)$$



Advantage: The bias can be removed by importance sampling / MCMC.

Normalizing flow

Normalizing flow

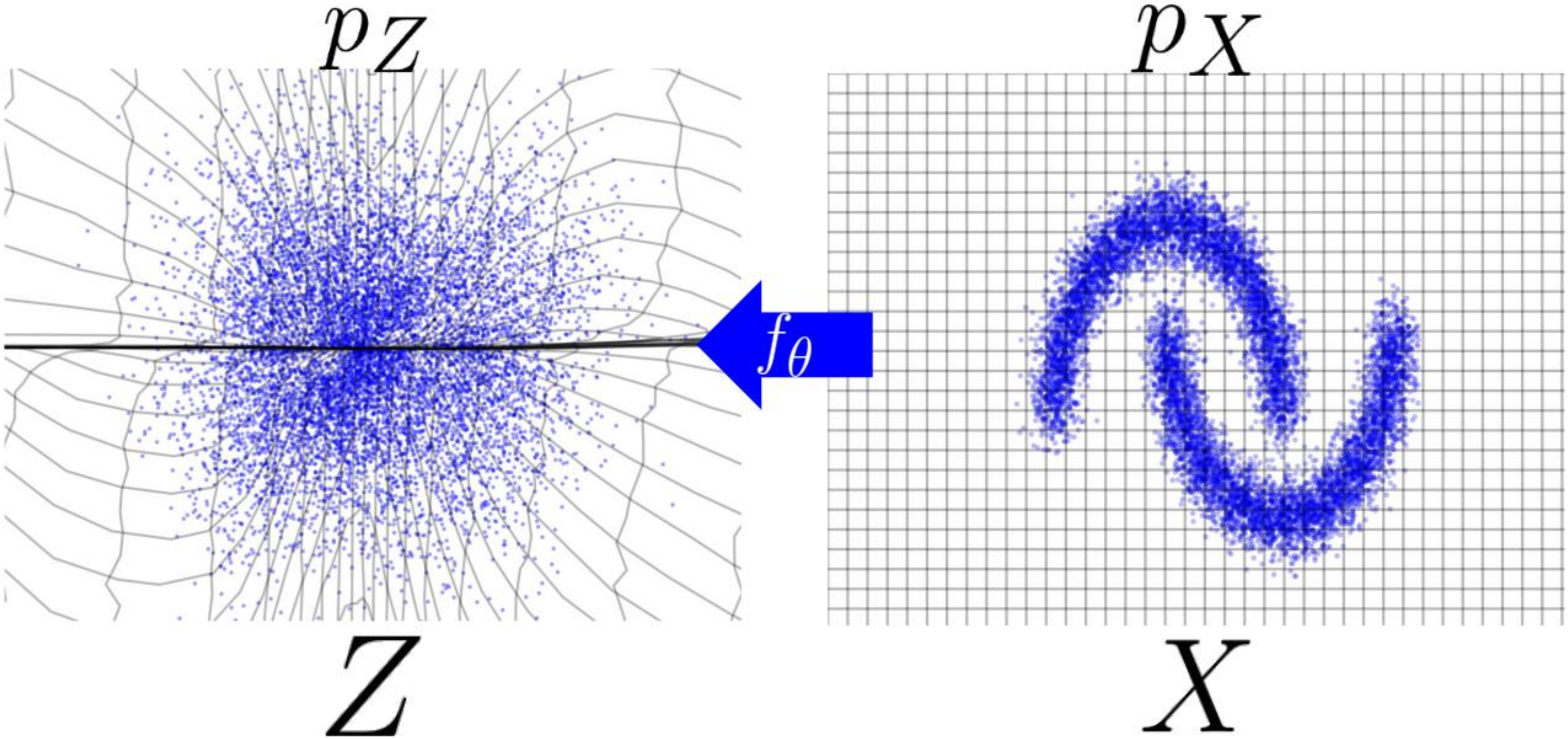
Motiviation: Probability manipulation by invertible nerual networks

(Kingma & Dhariwal, 2018)

Challenges

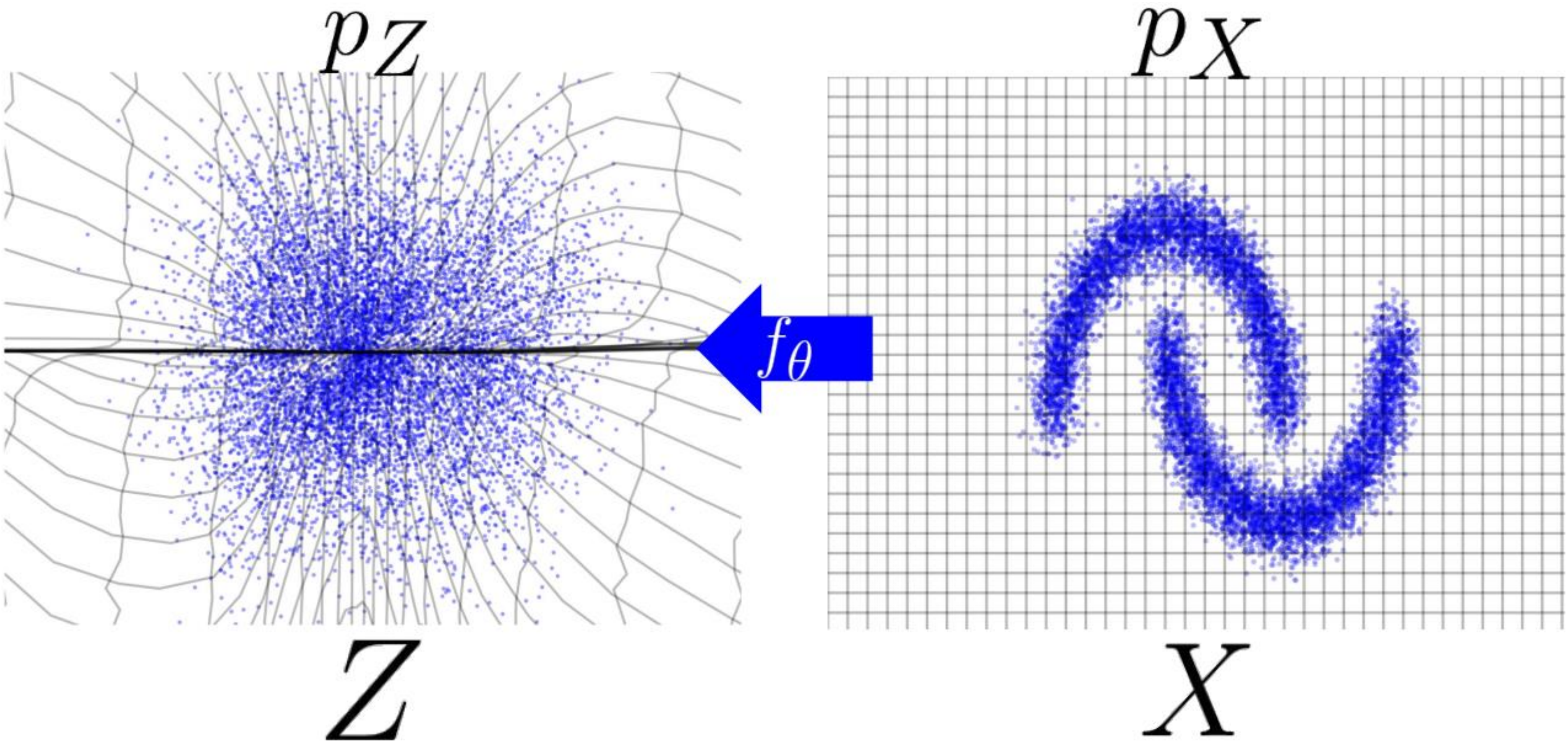
- Jacobian determinant
- Inversion

Study case: density estimation



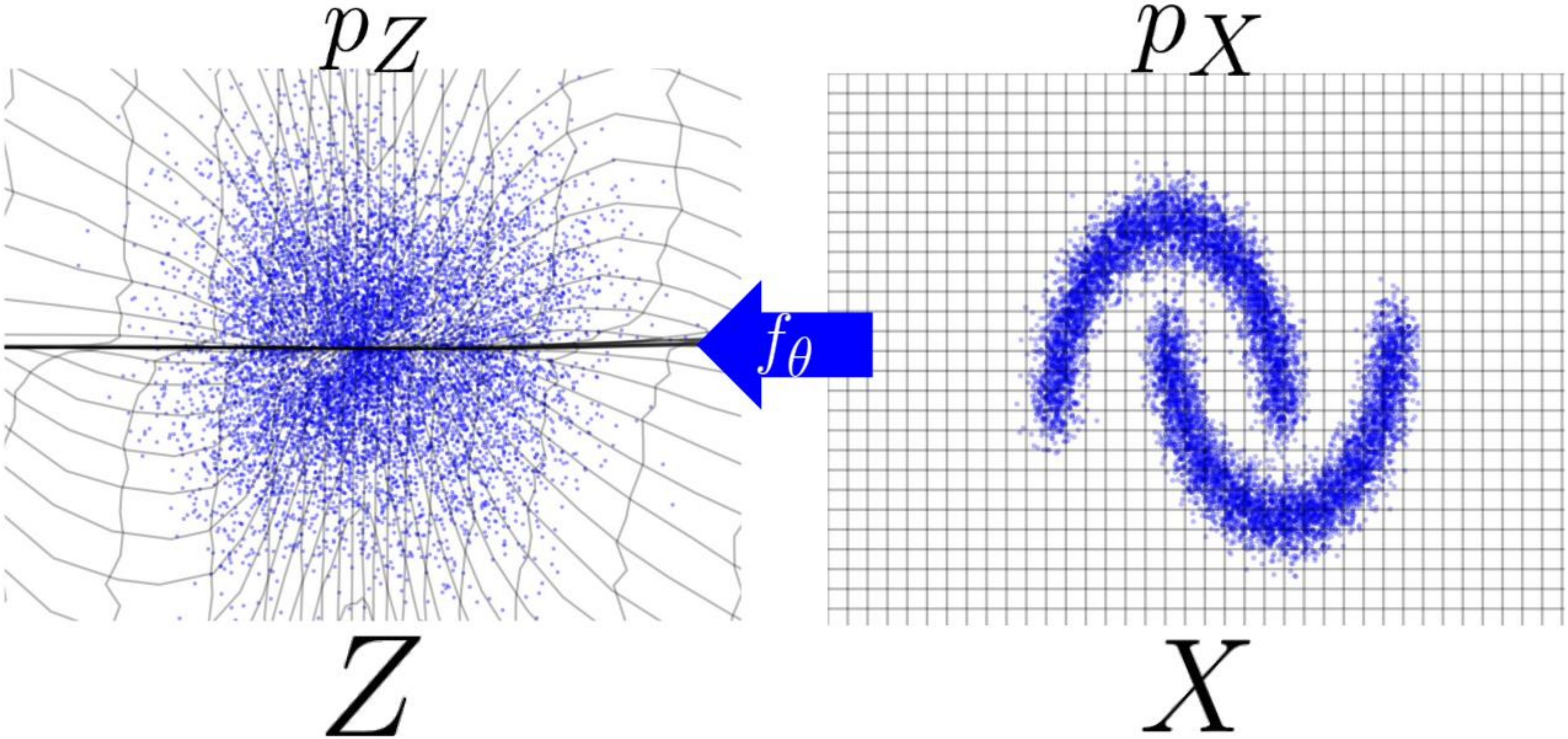
$f_\theta(x)$

Study case: density estimation



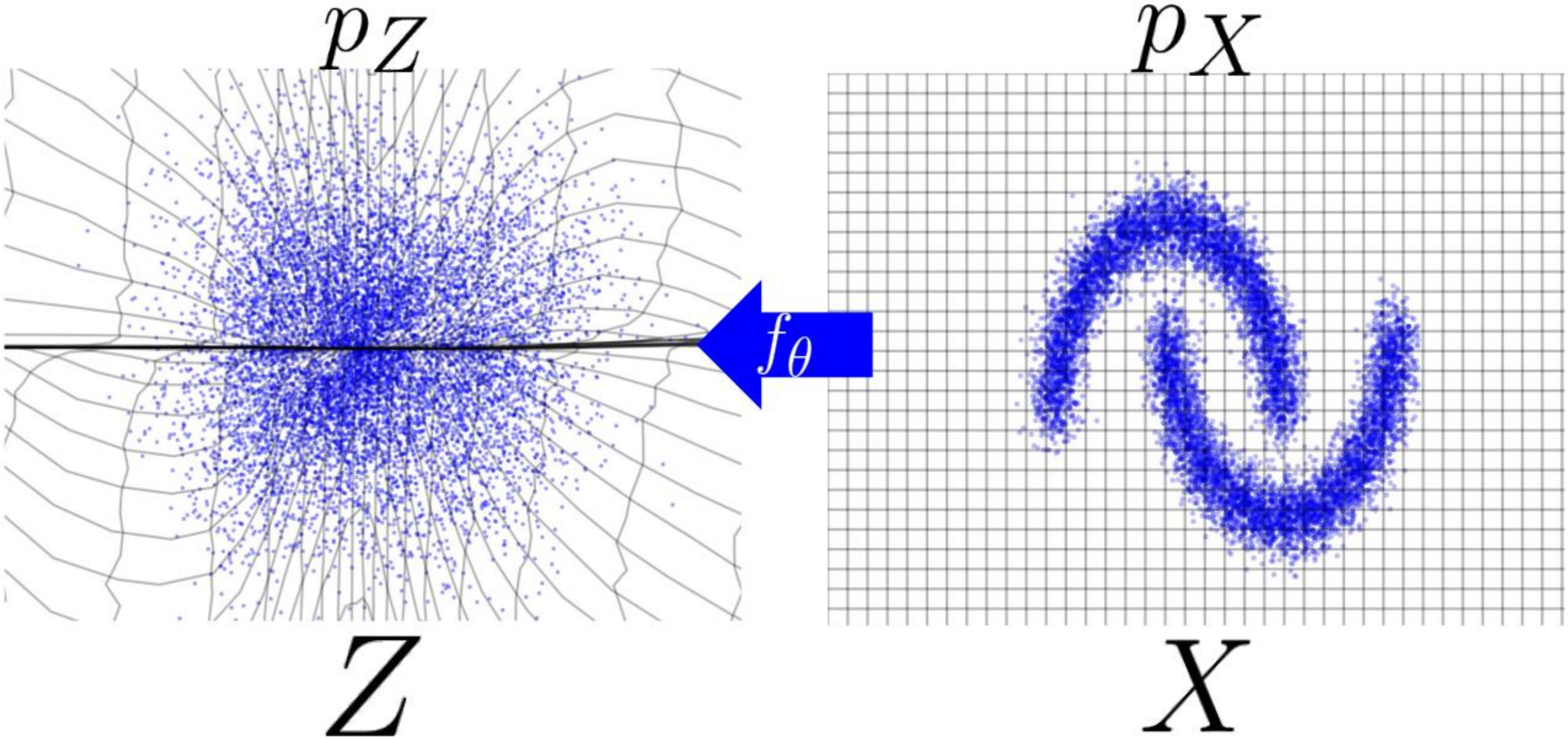
$p_Z\left(f_\theta(x)\right)$

Study case: density estimation



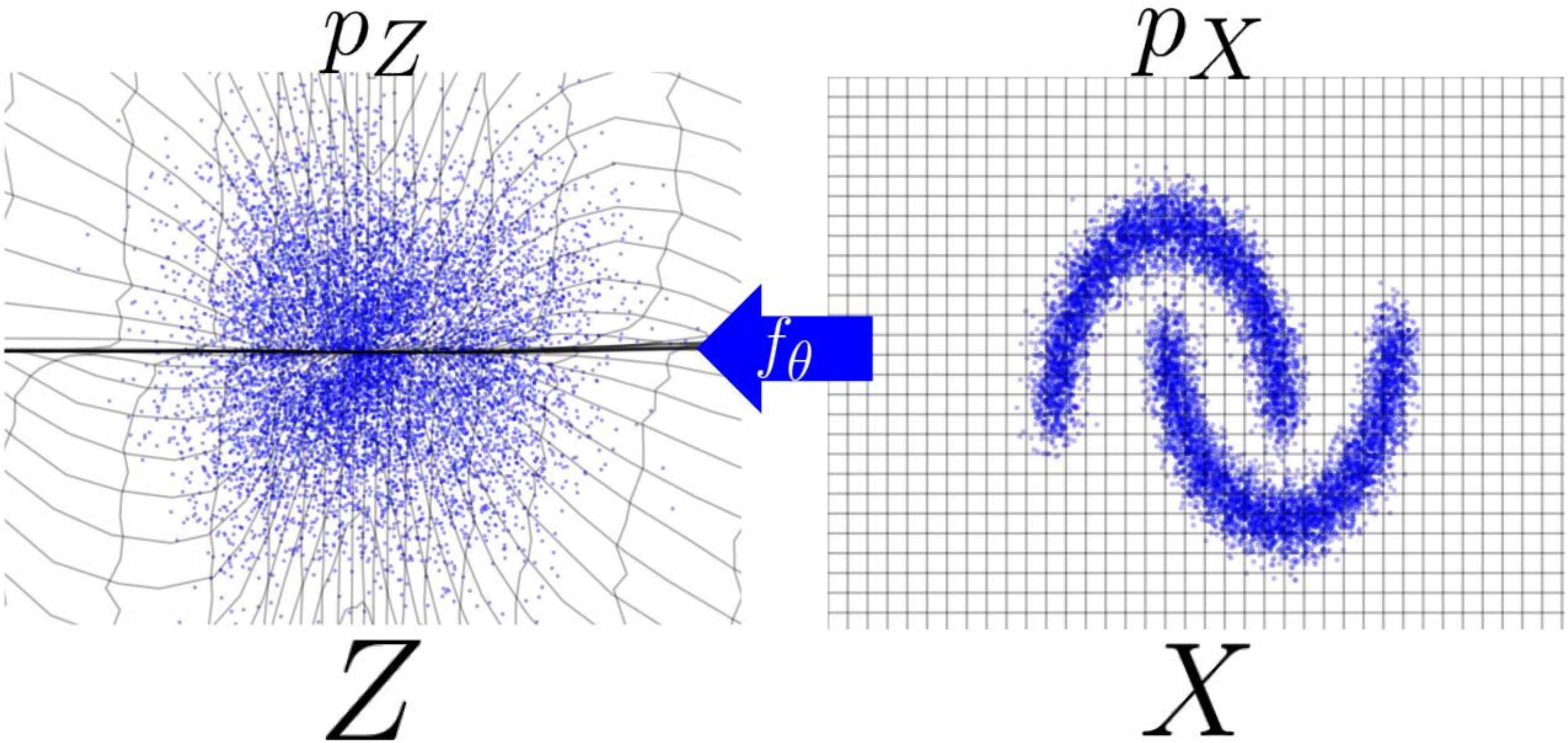
$$\log\left(p_X^{(\theta)}(x)\right) = \log\left(p_Z\left(f_\theta(x)\right)\right) + \log\left(\left|\frac{\partial f_\theta}{\partial x}\right|(x)\right)$$

Study case: density estimation



$$\log\left(p_X^{(\theta)}(x)\right) = \log\left(p_Z\left(f_\theta(x)\right)\right) + \log\left(\left|\frac{\partial f_\theta}{\partial x}\right|(x)\right)$$

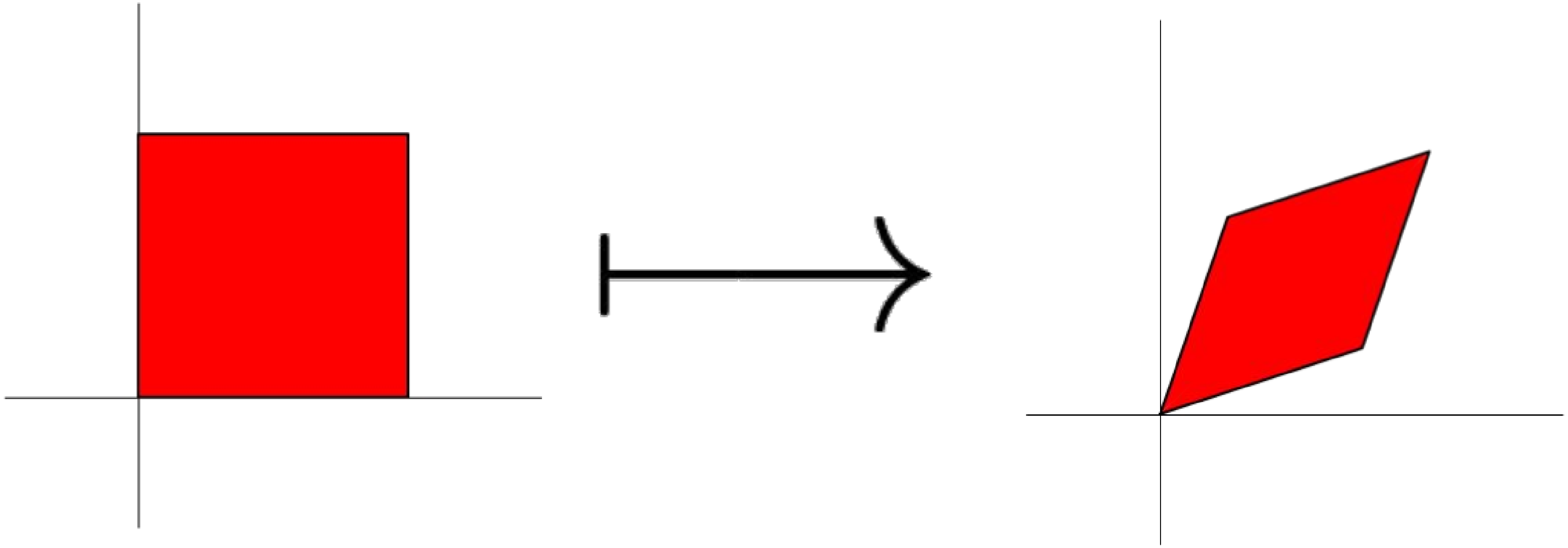
Study case: density estimation



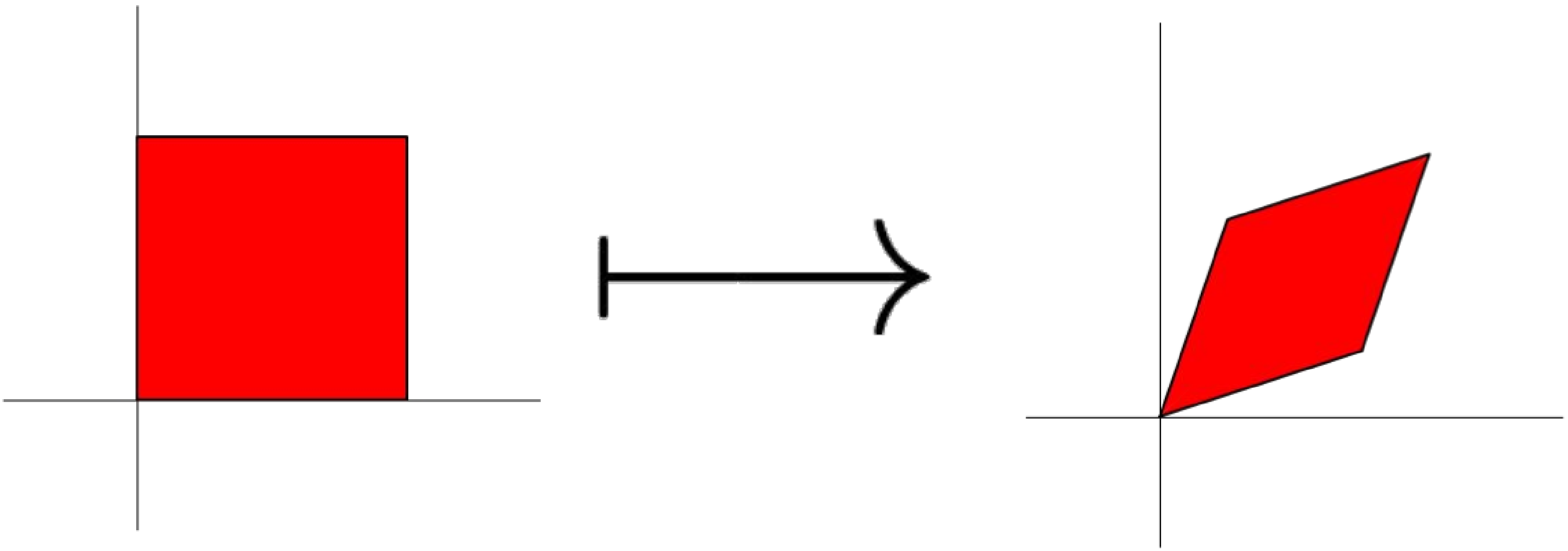
$$\log\left(p_X^{(\theta)}(x)\right) = \log\left(p_Z\left(f_\theta(x)\right)\right) + \log\left(\left|\frac{\partial f_\theta}{\partial x}\right|(x)\right)$$

Determinant

$$\frac{\partial f_\theta}{\partial x} \in \mathbb{M}(d, d)$$

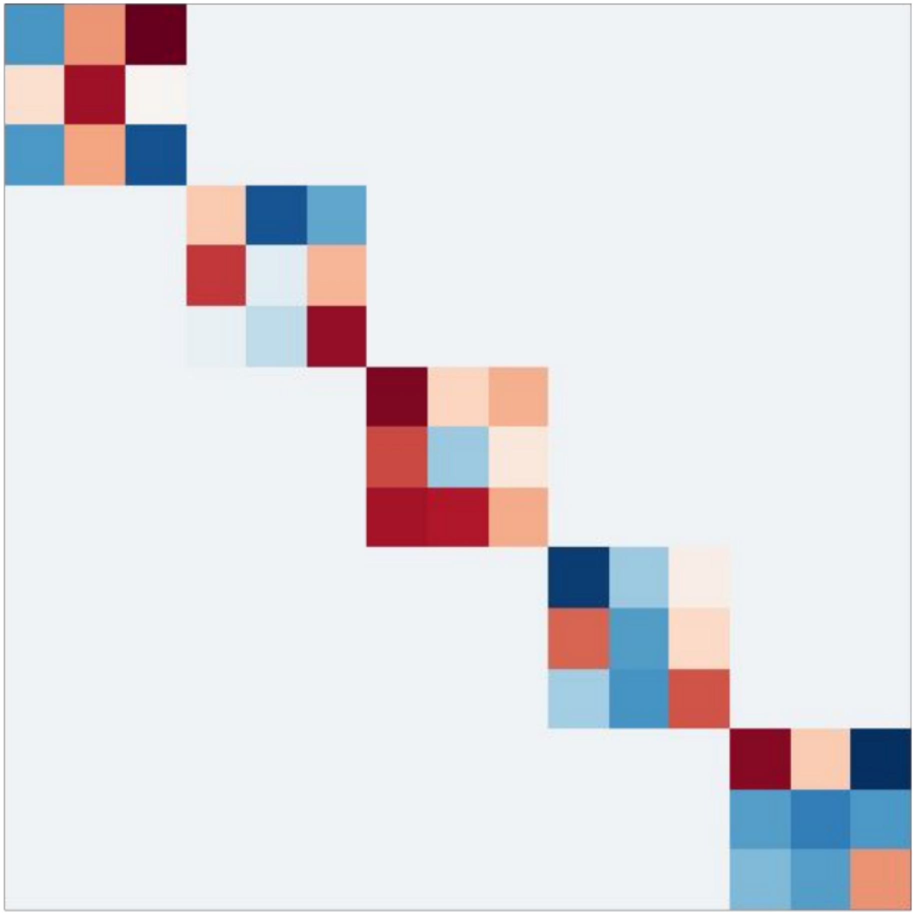


Determinant

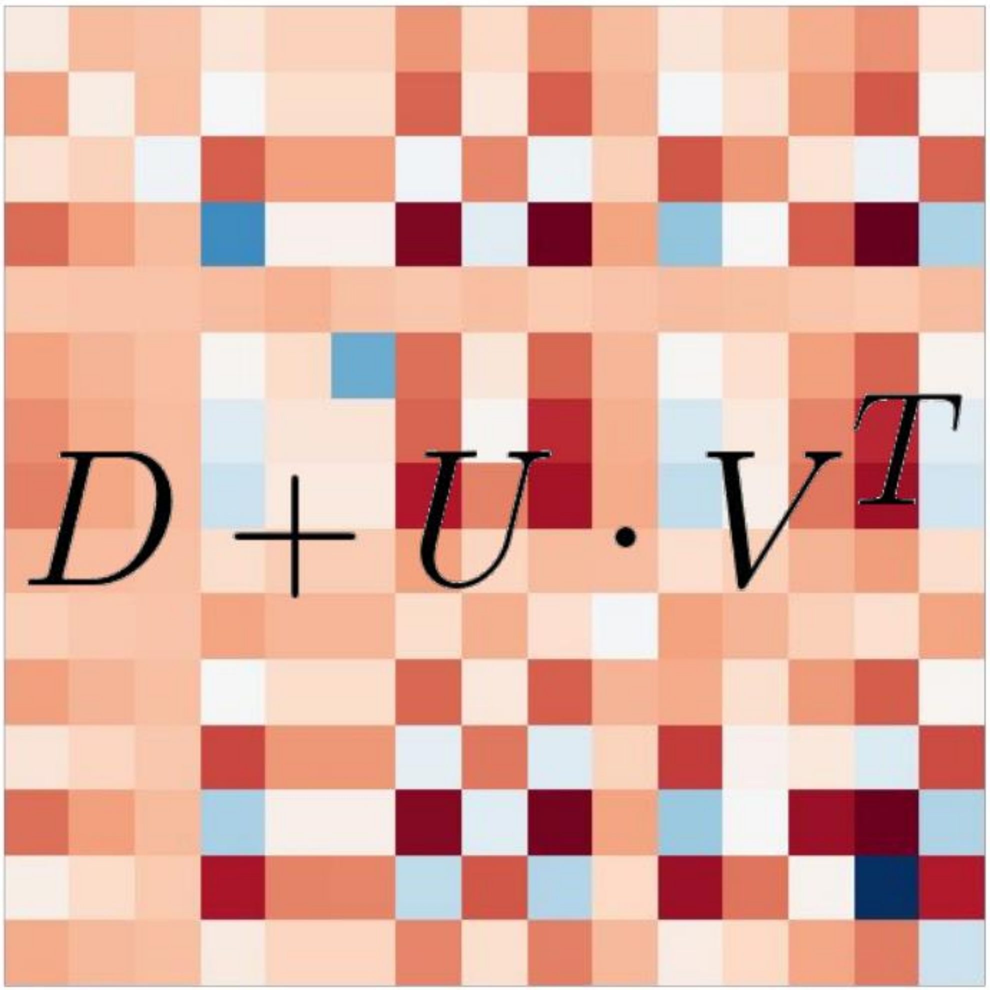


- Computational time is from $O(d^{2.376})$ to $O(d!)$.
- High variance unbiased estimator exists (Hutchinson estimator).

More tractable determinants



More tractable determinants



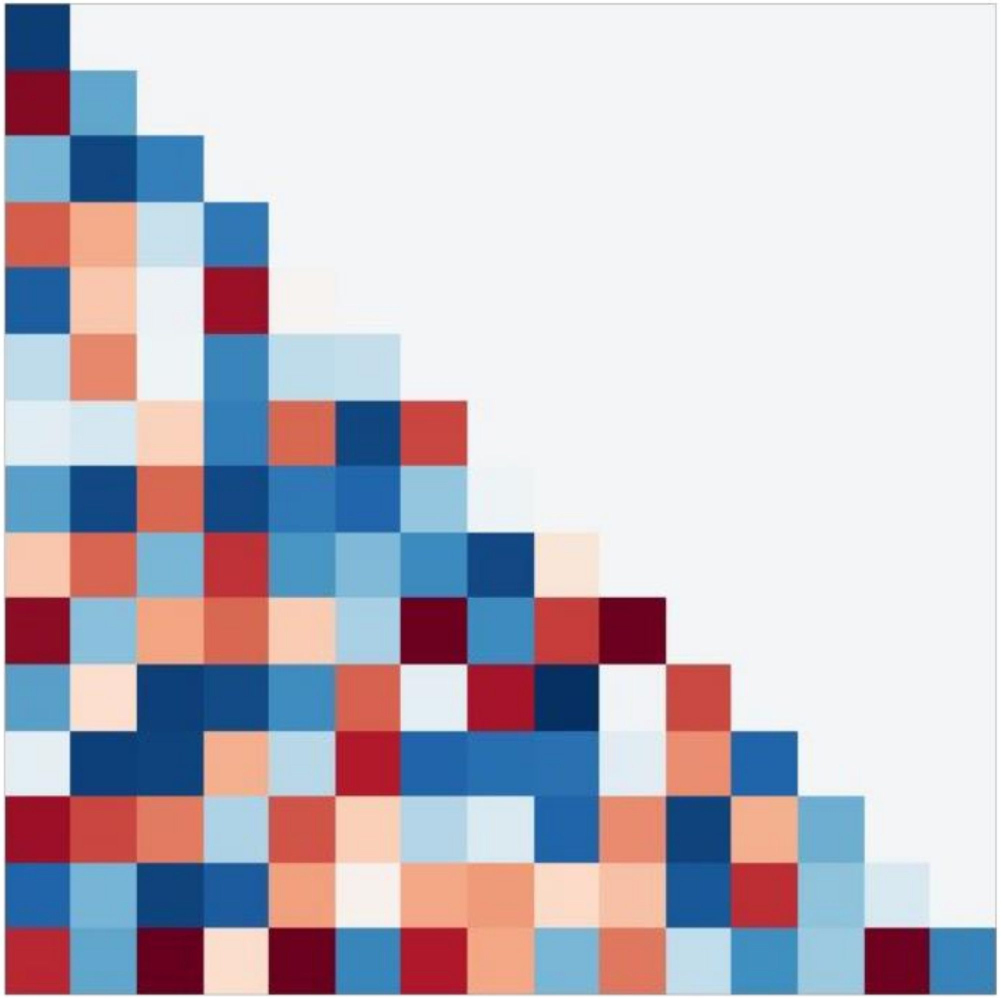
$$\det(D + UV^\top)$$

$$= \det(D) \cdot \det(I + D^{-1}UV^\top)$$

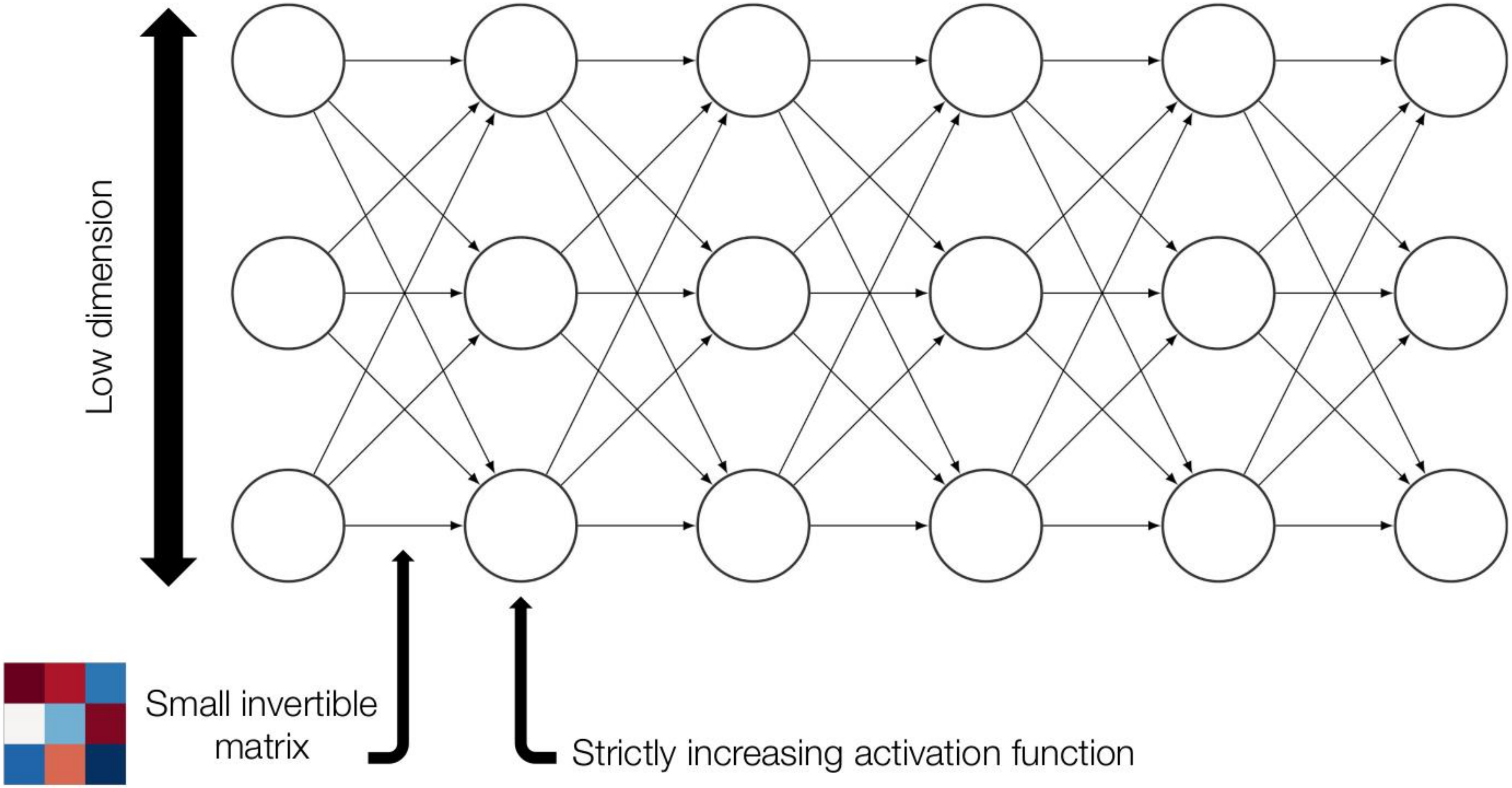
$$= \det(D) \cdot \det(I + V^\top D^{-1}U)$$

(Sylvester’s determinant identity)

More tractable determinants

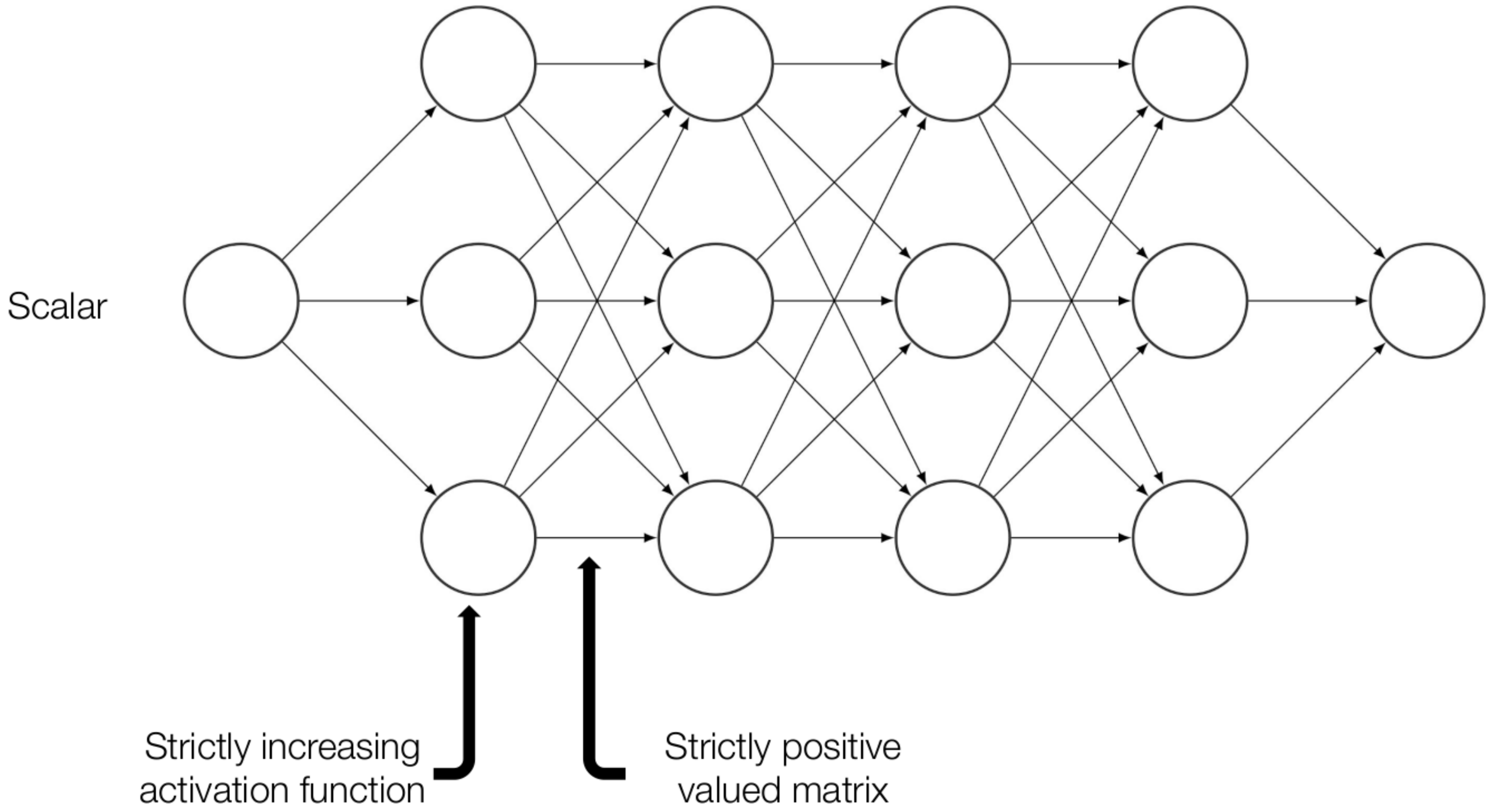


Deep learning with tractable Jacobian determinant



(Baird et al., 2005)

Neural scalar flow

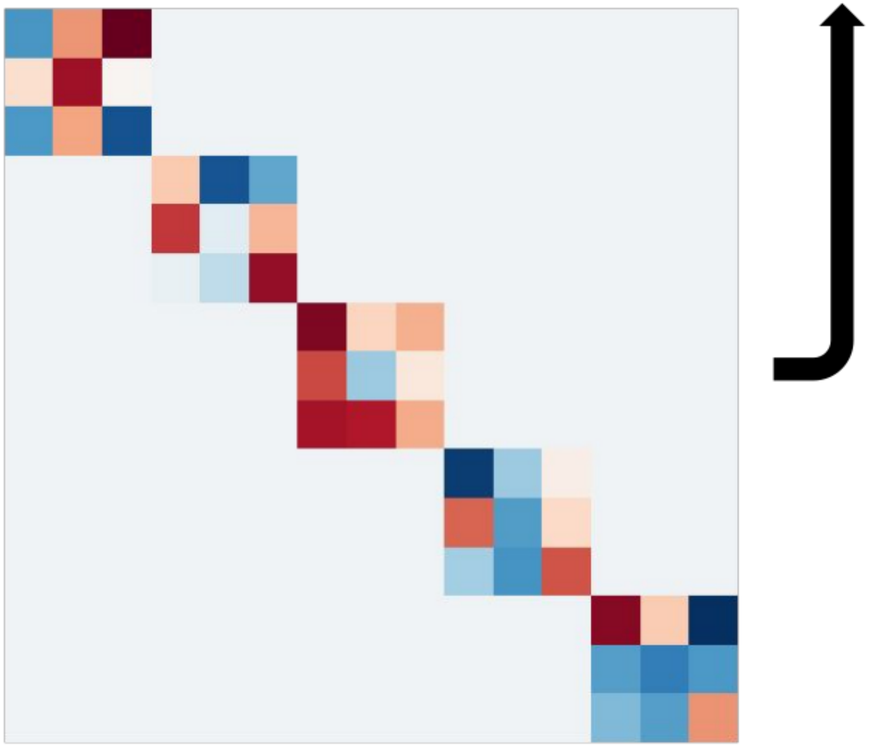


(Huang et al., 2018; De Cao, 2019)

Fourier convolution

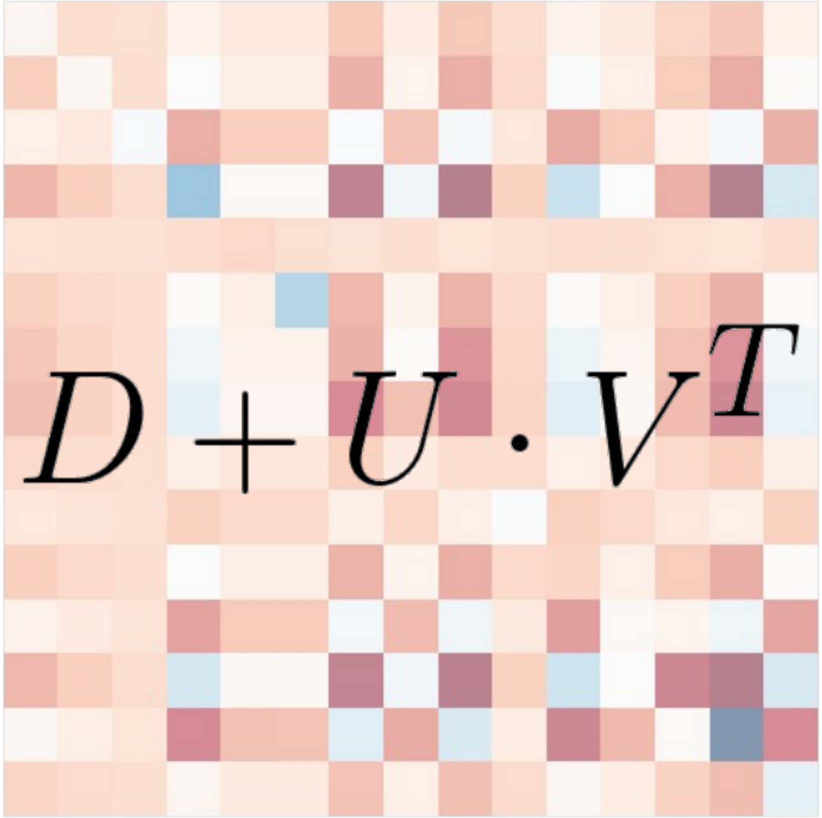
(Periodic) convolution theorem

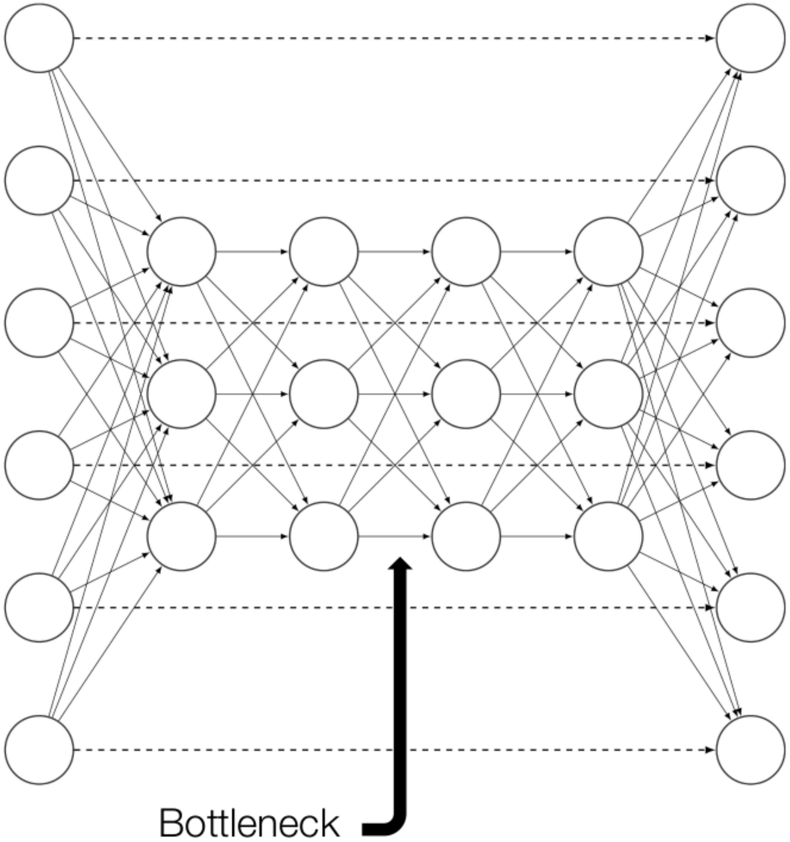
$$\mathcal{F}(x * w) = \mathcal{F}(x) \cdot \mathcal{F}(w)$$



(Hoogeboom et al., 2019; Karami et al., 2019)

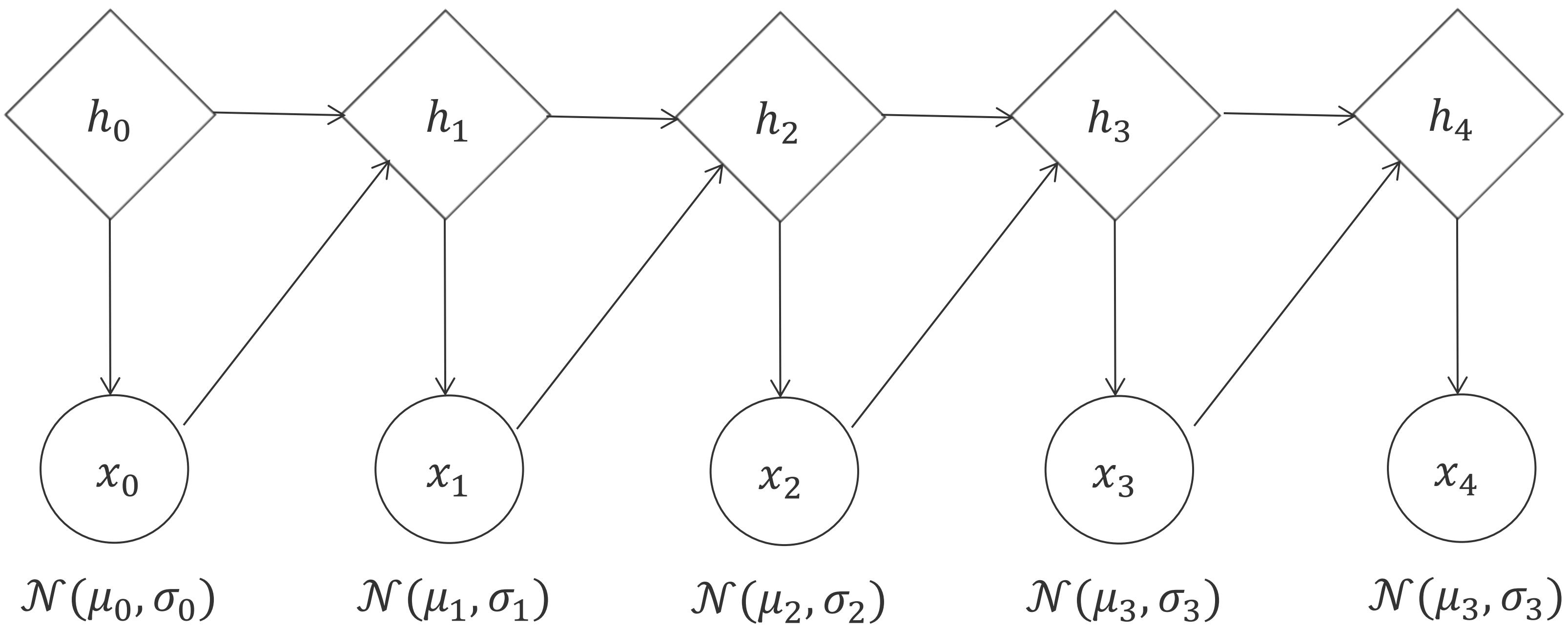
Sylvester normalizing flows



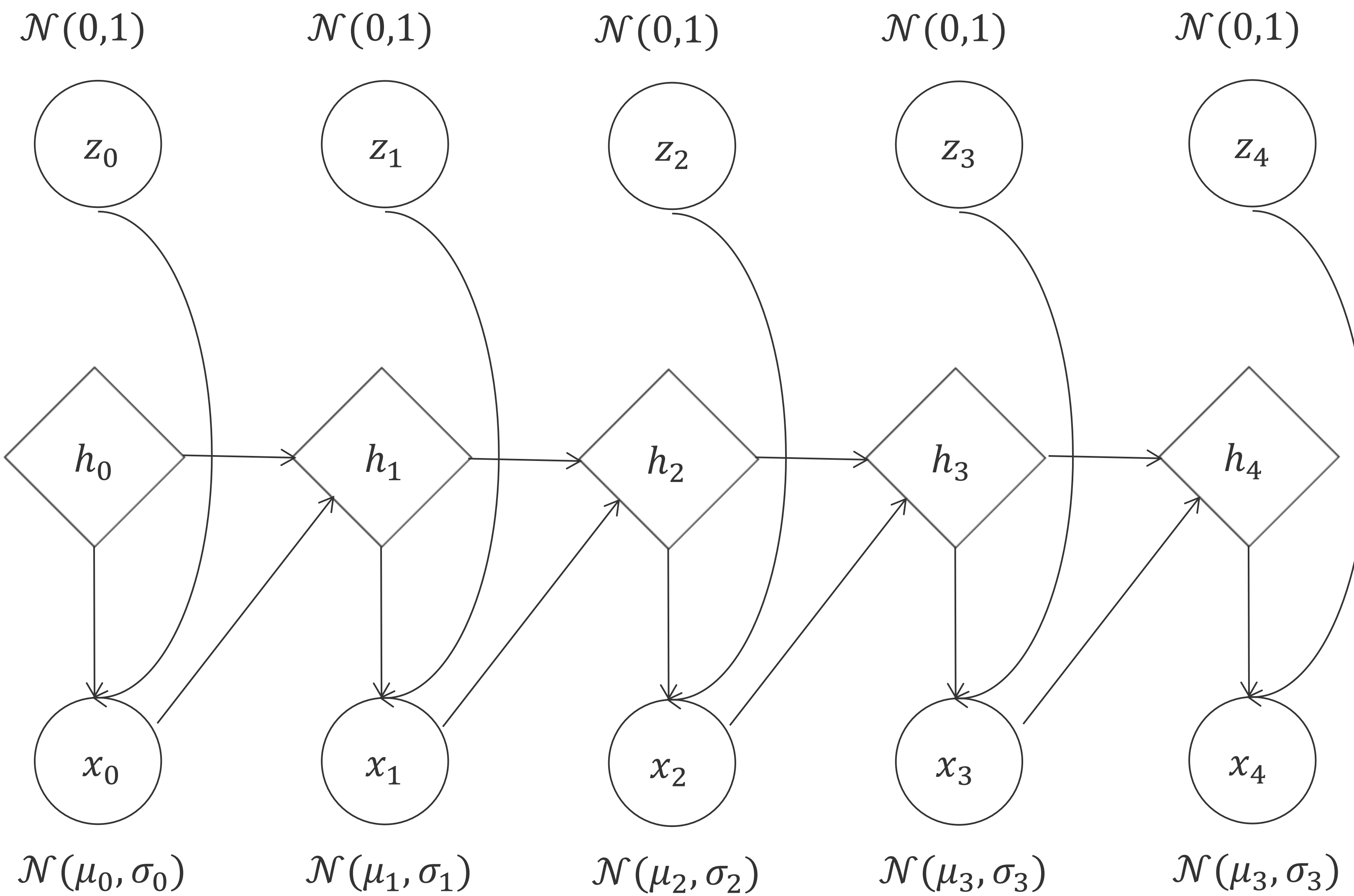


(van den Berg, Hansclever et al., 2018)

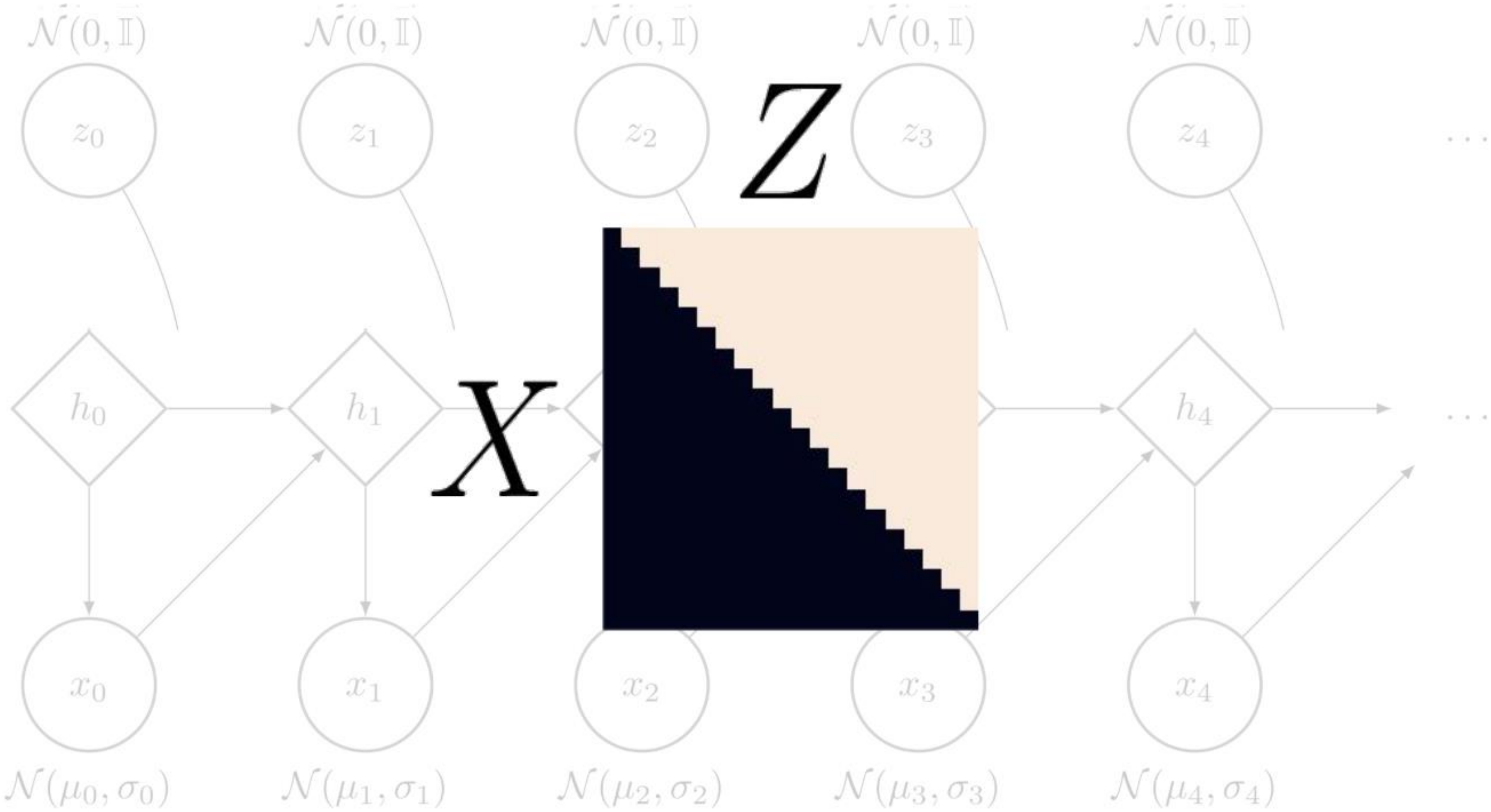
Autoregressive models



Autoregressive models

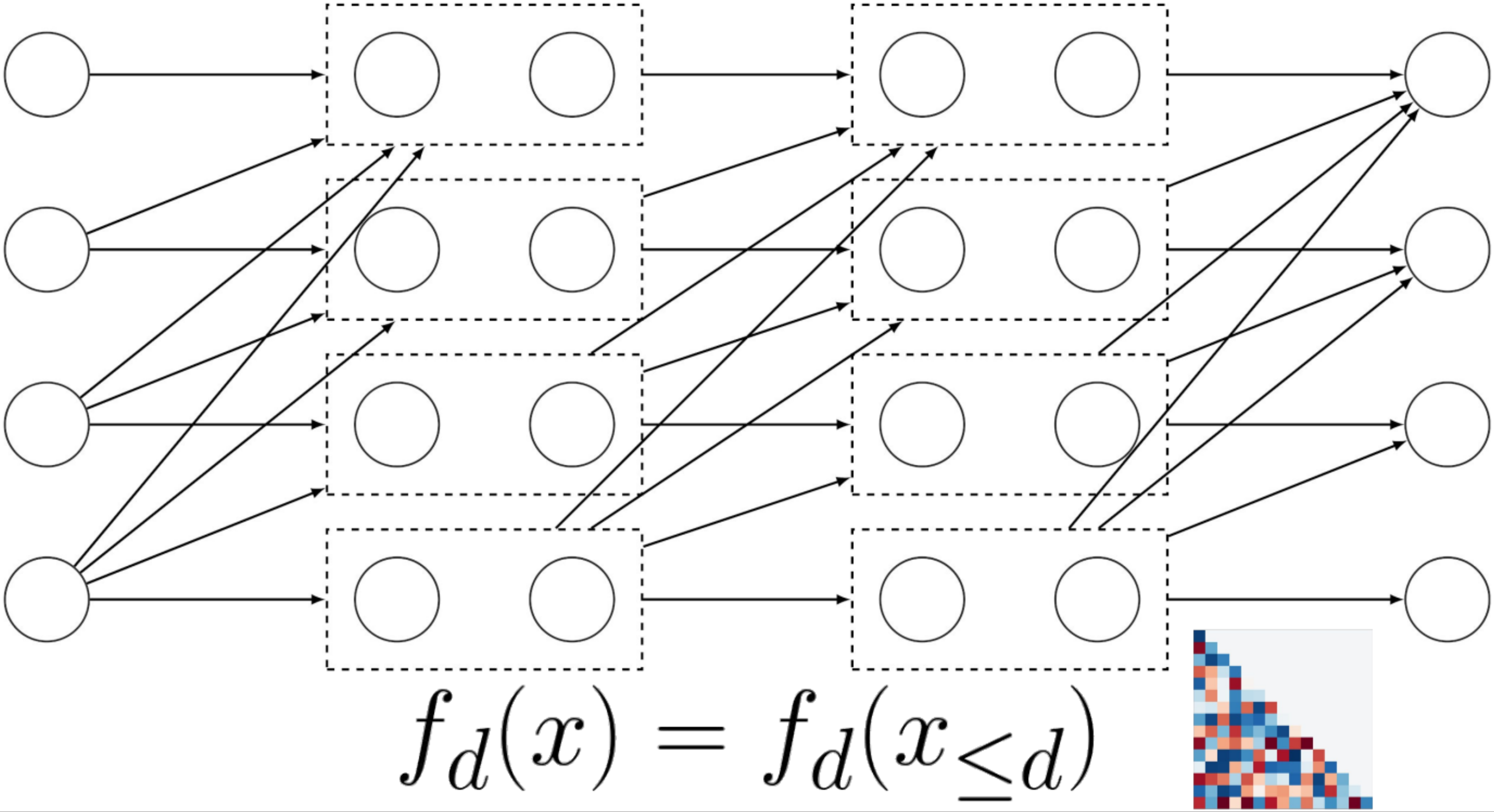


Autoregressive models



(Deco & Brauer, 1995; Hyvarinen & Pajunen, 1998; Moselhy & Marzouk, 2012)

Neural autoregressive models

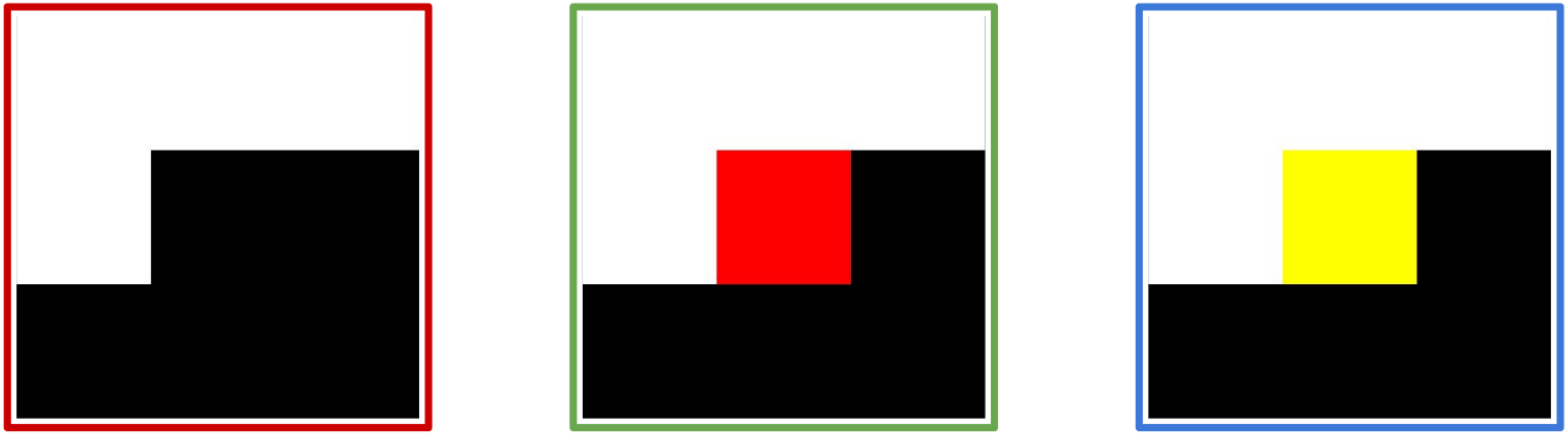


$$f_d(x) = f_d(x_{\leq d})$$

(Bengio, 1999; Larochelle & Murray, 2011; van den Oord et al., 2015; Uria et al., 2016)

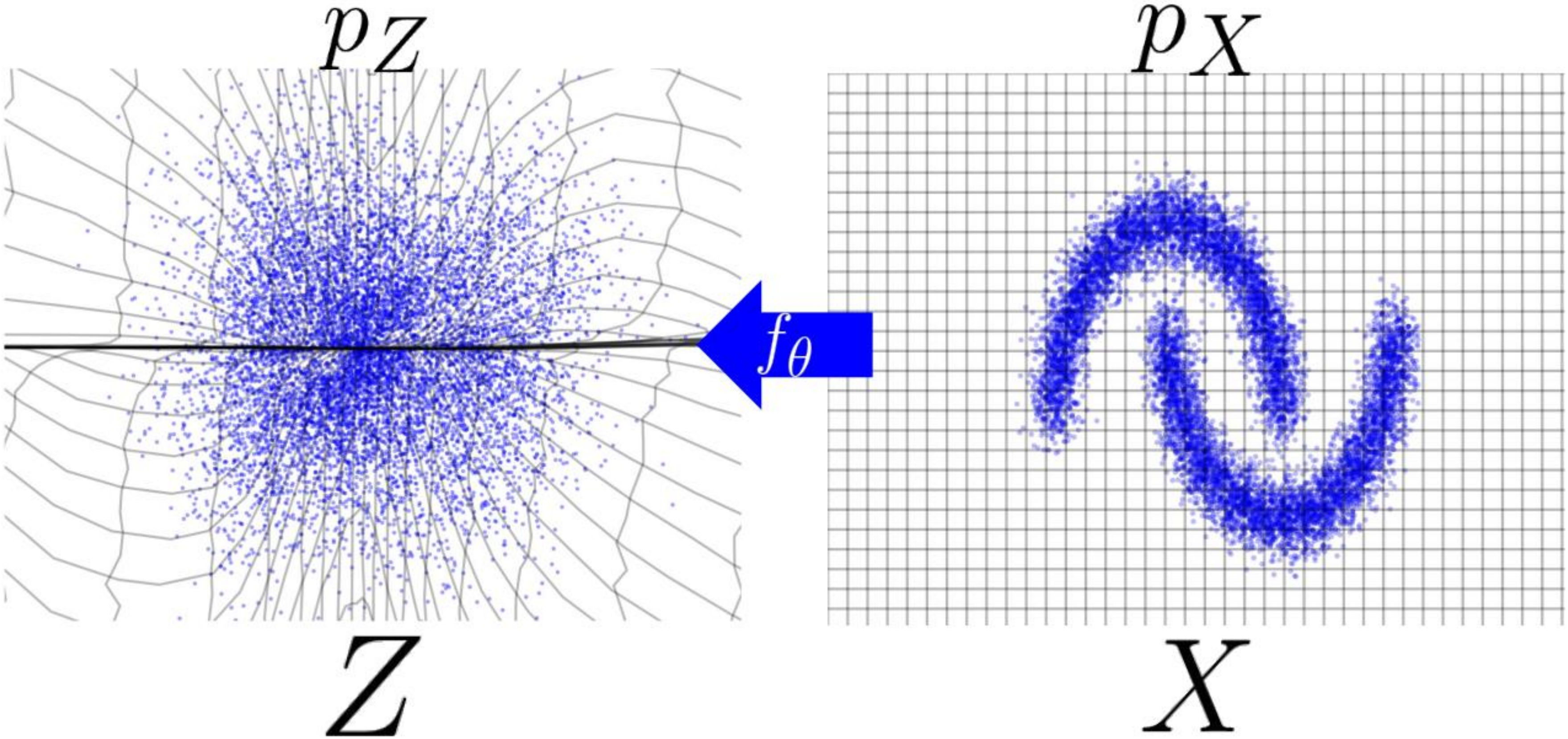
Convolutional autoregressive models

Masked convolutions



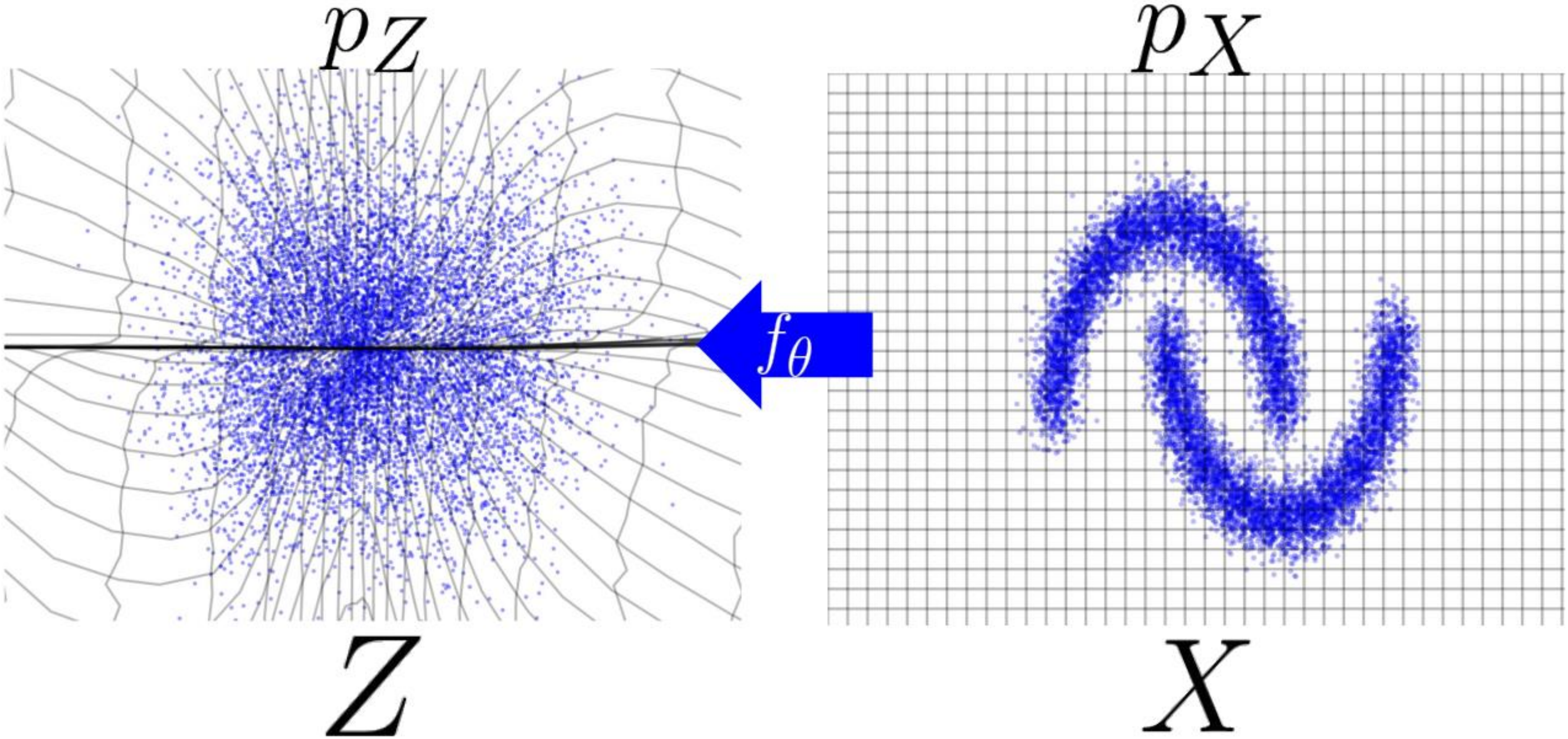
(van den Oord et al., 2016)

Study case: density estimation



$$\log\left(p_X^{(\theta)}(x)\right) = \log\left(p_Z\left(f_\theta(x)\right)\right) + \log\left(\left|\frac{\partial f_\theta}{\partial x}\right|(x)\right)$$

Study case: density estimation

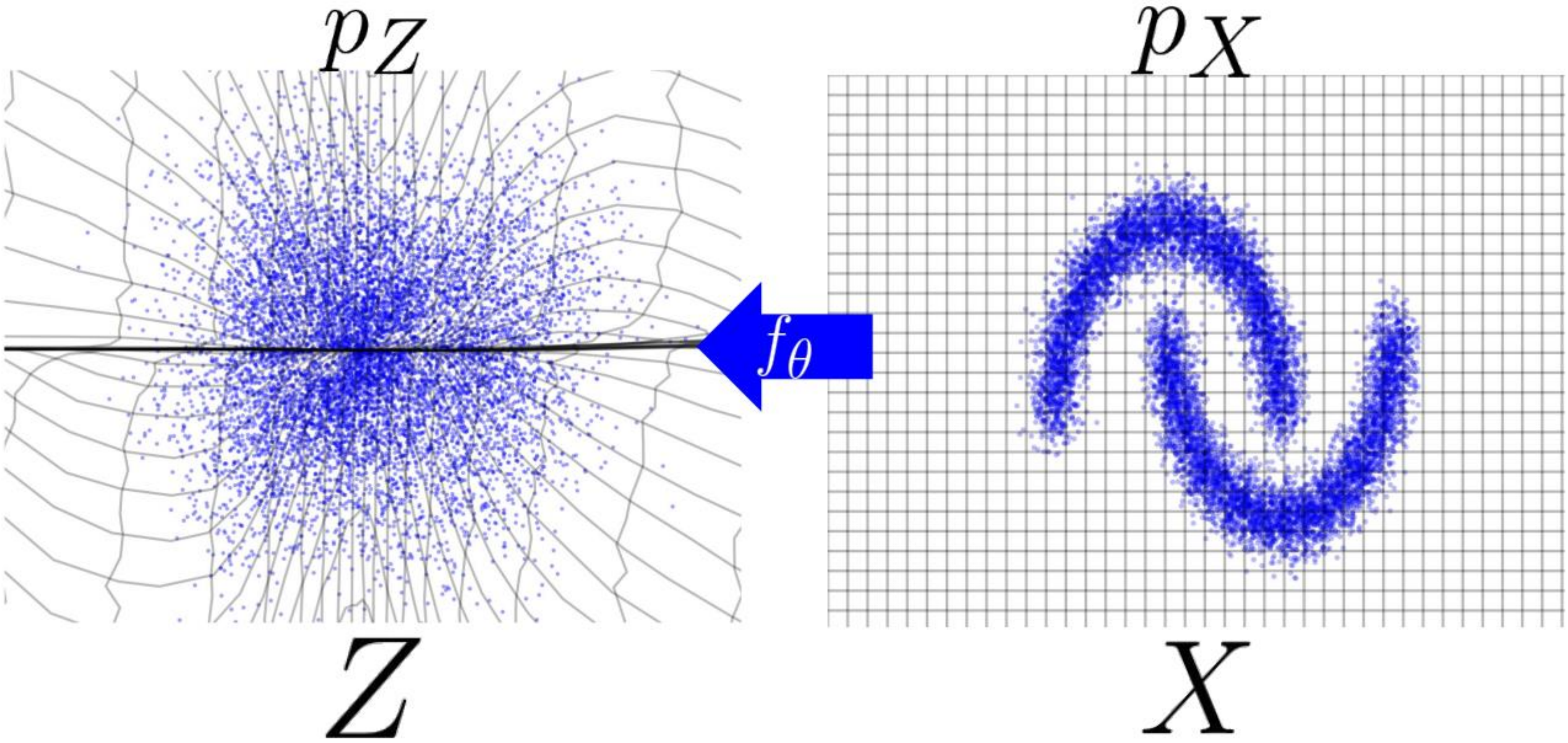


$$\log\left(p_X^{(\theta)}(x)\right) = \log\left(p_Z\left(f_\theta(x)\right)\right) + \log\left(\left|\frac{\partial f_\theta}{\partial x}\right|(x)\right)$$

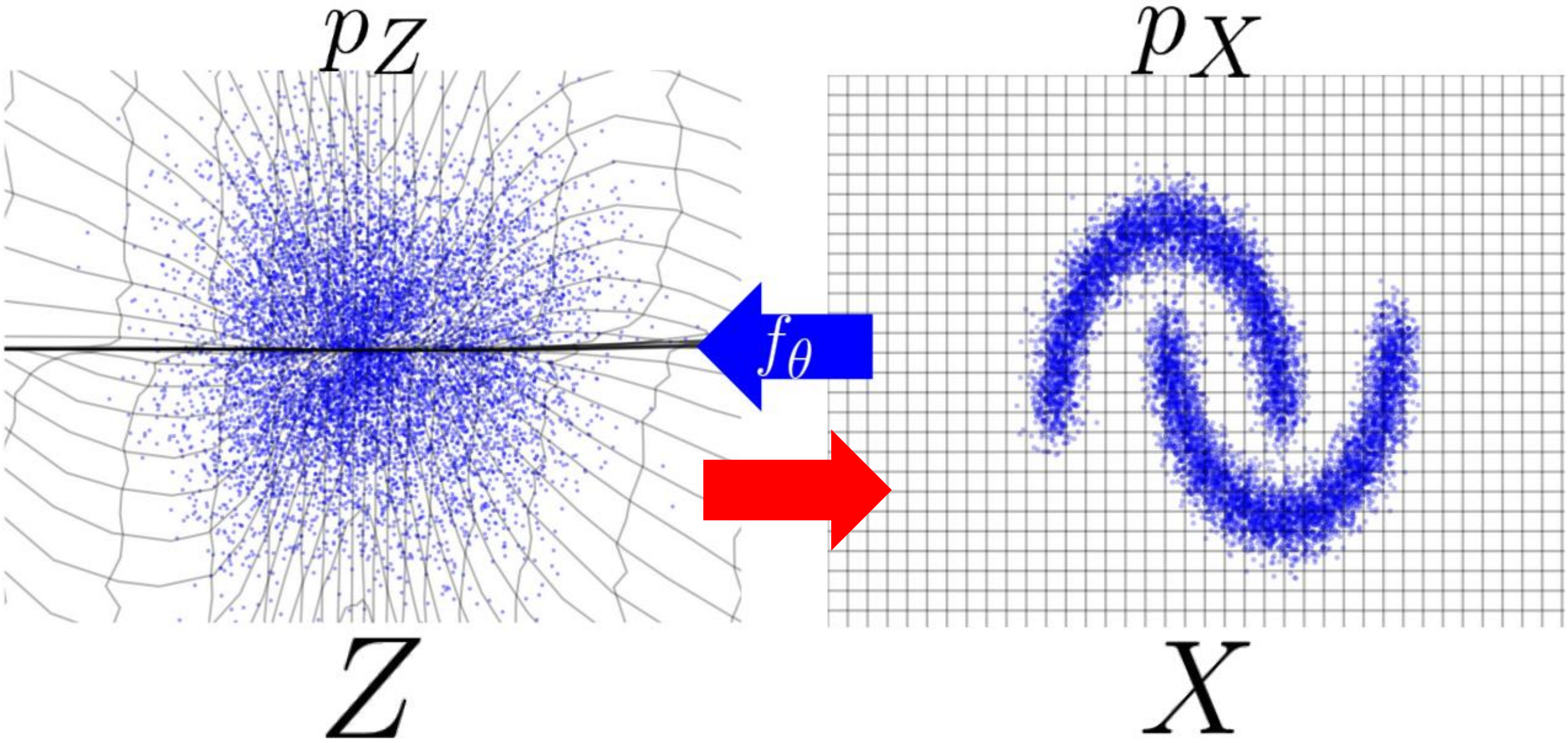
Inverting a neural network

$$f_\theta^{-1} = \ ?$$

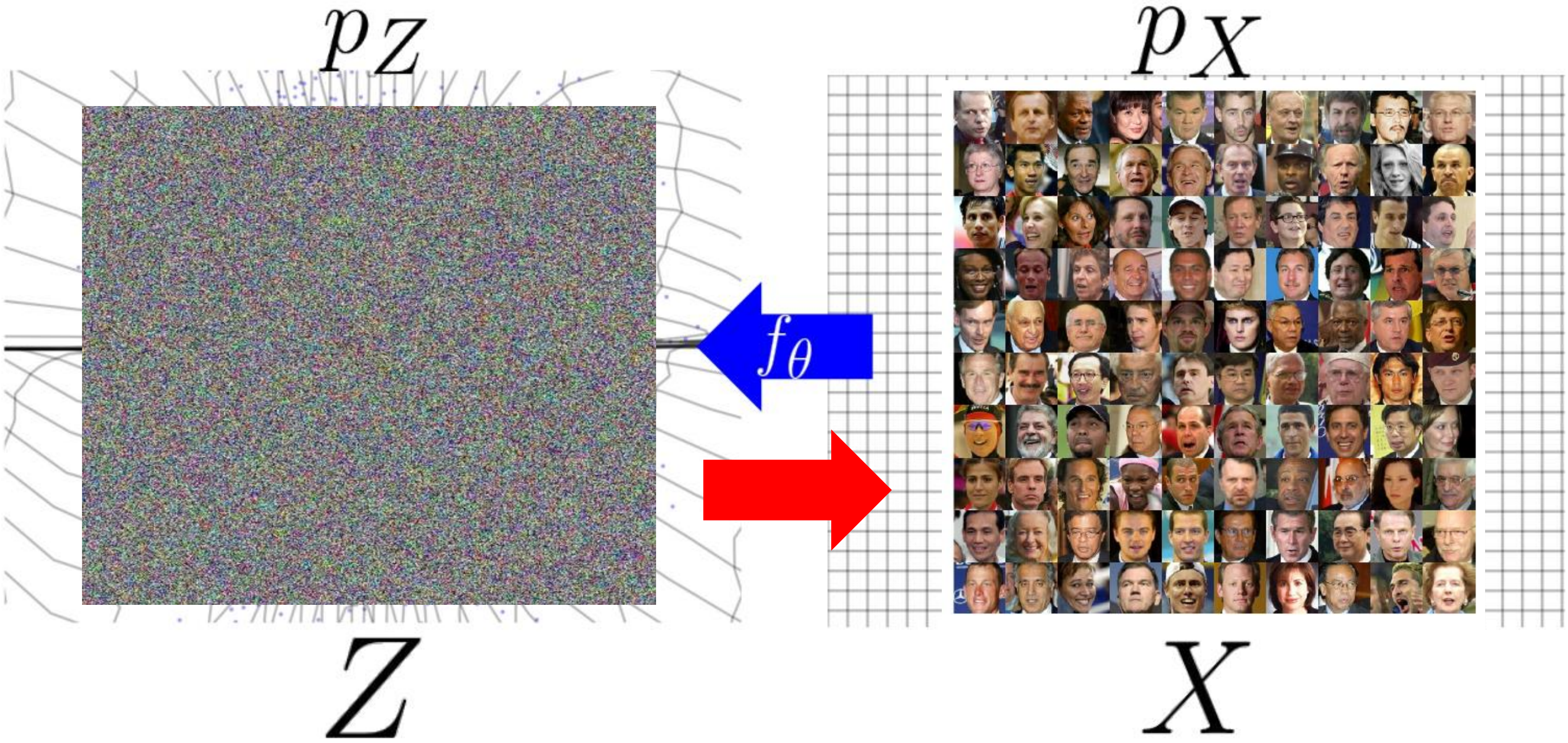
Generation through process reversion



Generation through process reversion



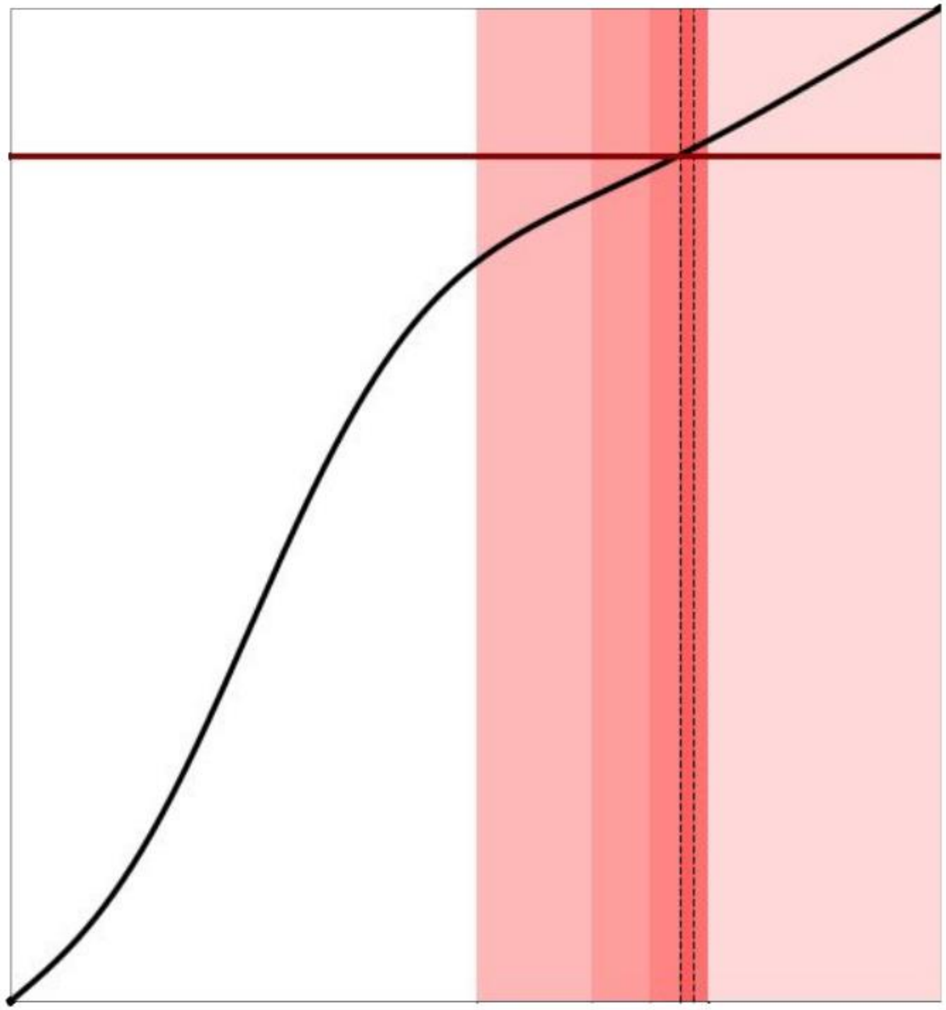
Generation through process reversion



Iterative inversion

- Bisection / binary search
- Root finding algorithm (Newton Raphson)
- Fixed point iteration

Bisection



(Ho, Chen et al., 2019)

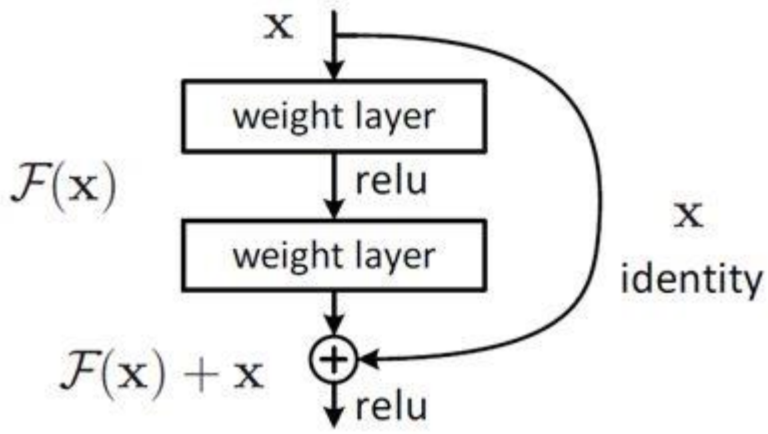
Root finding algorithm

$$x^{(t+1)} = x^{(t)} - \alpha \left(\frac{\partial f}{\partial x} \right)^{-1} \left(f(x^{(t)}) - y \right)$$

Newton-Raphson: **Local convergence**

(Song et al., 2019)

Residual flow



A block of residual learning

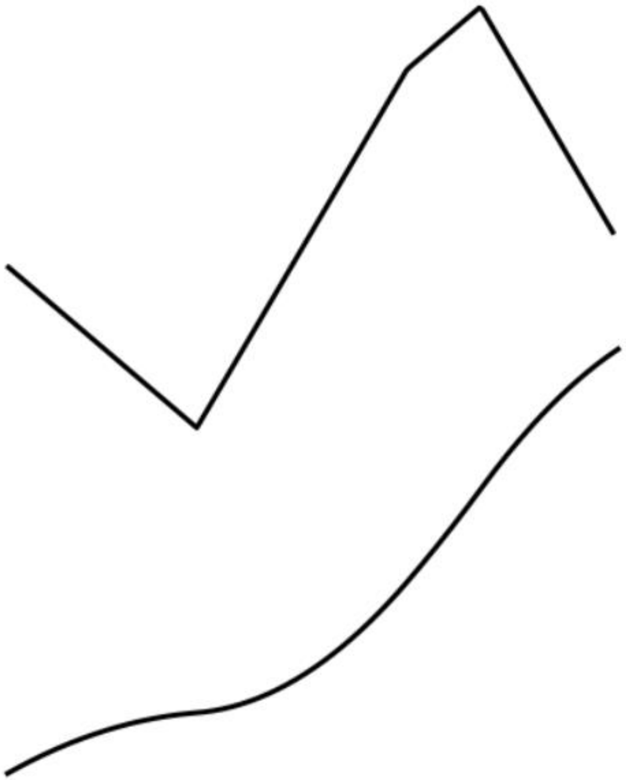
$$x \mapsto x + f(x) = y$$
$$\|f(x^{(1)}) - f(x^{(2)})\| \leq c\|x^{(1)} - x^{(2)}\|$$
$$x^{(t+1)} = y - f(x^{(t)})$$

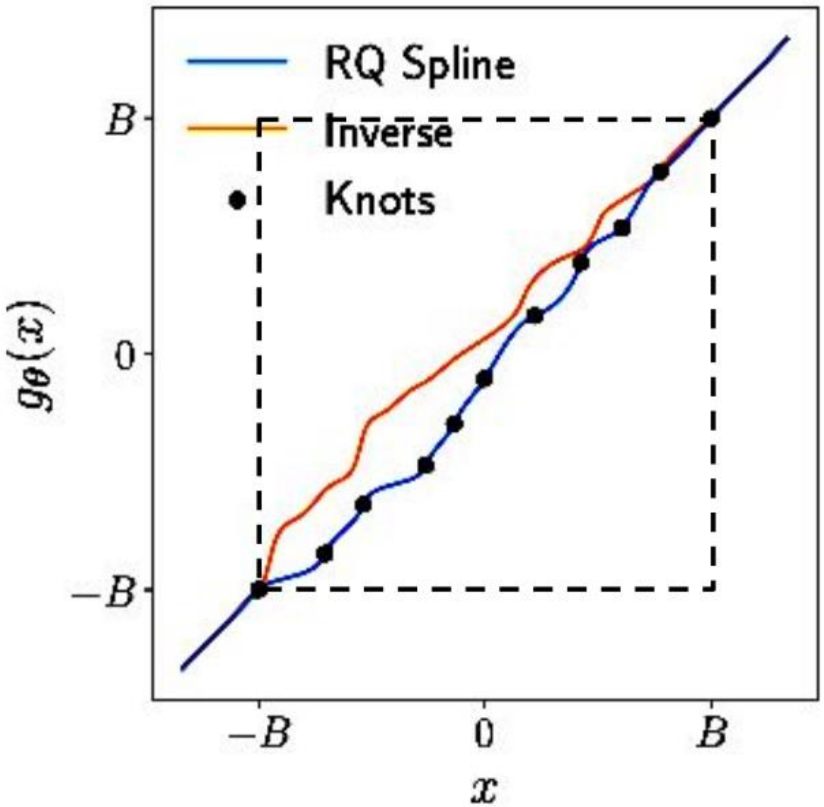
Fix-point iteration: **Global convergence**

(Behrmann et al., 2019)

Closed form inverse: scalar case

Invertible piecewise functions





(Müller et al., 2019; Durkan, Bekasov et al., 2019)

Autoregressive case

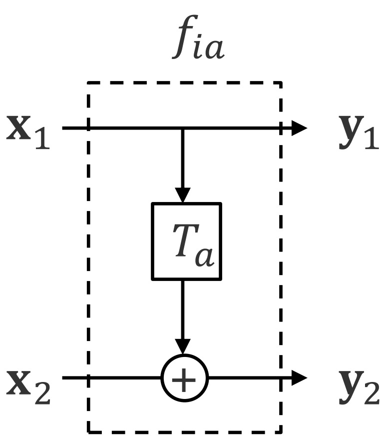
Forward substitution

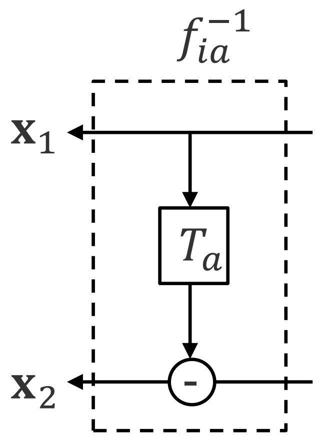
$$z_d = f_d(x_d; x_{<d})$$

$$x_d = f_d^{-1}(z_d; x_{<d})$$

Non parallel

Coupling layer





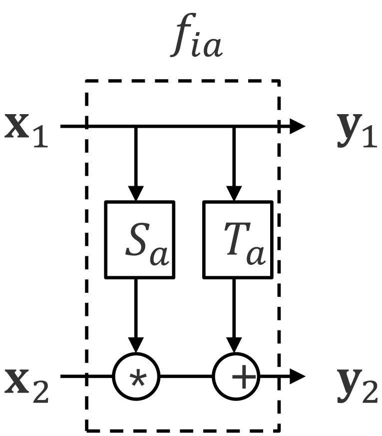
$$\mathbf{y}_1 = \mathbf{x}_1$$
$$\mathbf{y}_2 = \mathbf{x}_2 + T_a(\mathbf{x}_1)$$

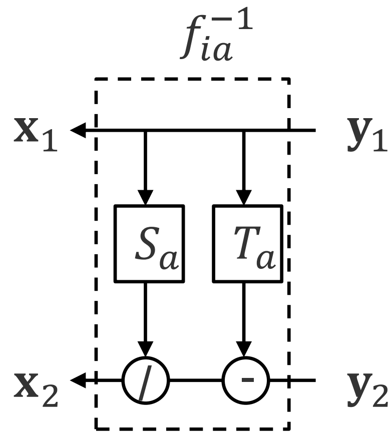
$$\mathbf{x}_1 = \mathbf{y}_1$$
$$\mathbf{x}_2 = \mathbf{y}_2 - T_a(\mathbf{y}_1)$$

T_a: Deep netural networks

(Dinh et al., 2015)

Coupling layer





$$\mathbf{y}_1 = \mathbf{x}_1$$
$$\mathbf{y}_2 = S_a(\mathbf{x}_1) * \mathbf{x}_2 + T_a(\mathbf{x}_1)$$

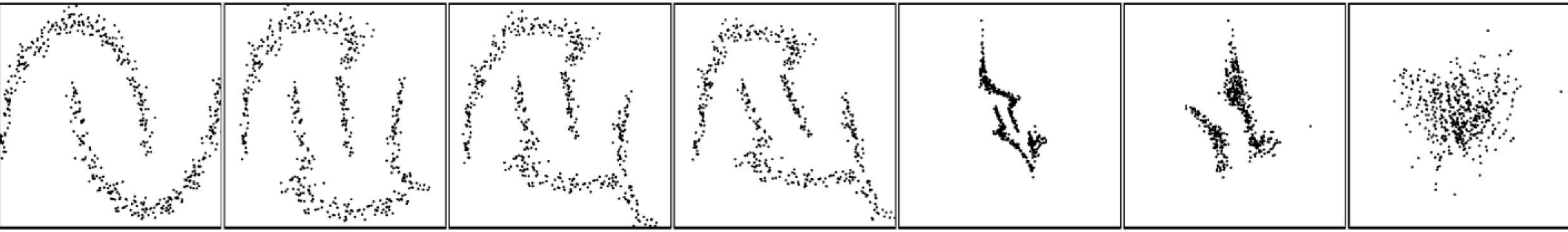
$$\mathbf{x}_1 = \mathbf{y}_1$$
$$\mathbf{x}_2 = (\mathbf{y}_2 - T_a(\mathbf{y}_1))/S_a(\mathbf{y}_1)$$

$T_a, \log S_a$: Deep netural networks

(Dinh et al., 2017)

Composing flows

$$f_3 \circ f_2 \circ f_1$$



Composing flows

- Inversion and sampling

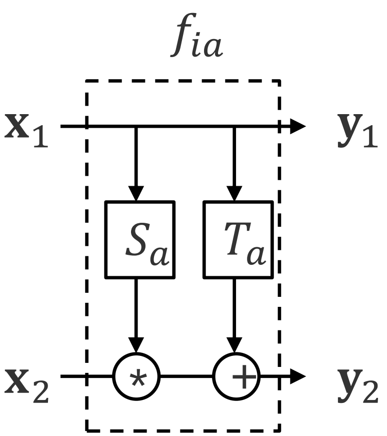
$$(f_2 \circ f_1)^{-1} = f_1^{-1} \circ f_2^{-1}$$

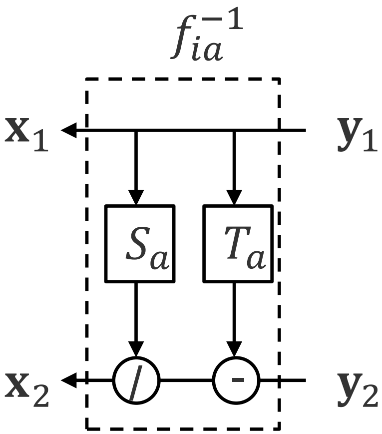
- Determinant and inference

$$\nabla(f_2 \circ f_1)(x) = \nabla f_2\big(f_1(x)\big)\nabla f_1(x)$$

$$\det(A \cdot B) = \det(A) \cdot \det(B)$$

Combining coupling layers: RealNVP





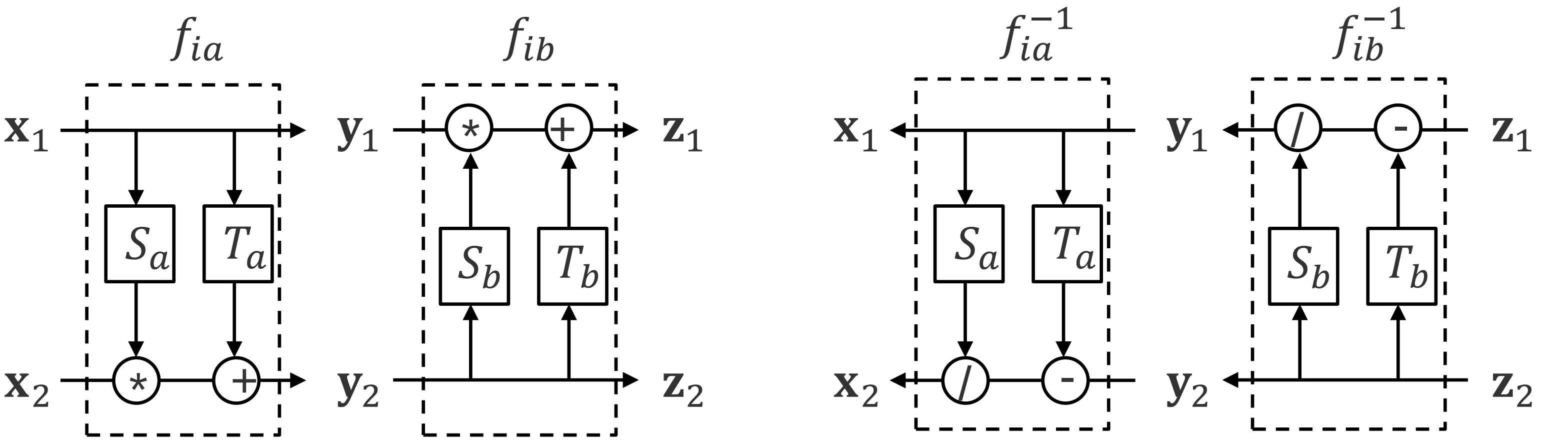
$$\mathbf{y}_1 = \mathbf{x}_1$$
$$\mathbf{y}_2 = S_a(\mathbf{x}_1) * \mathbf{x}_2 + T_a(\mathbf{x}_1)$$

$$\mathbf{x}_1 = \mathbf{y}_1$$
$$\mathbf{x}_2 = (\mathbf{y}_2 - T_a(\mathbf{y}_1))/S_a(\mathbf{y}_1)$$

$T_a, \log S_a$: Deep netural networks

(Dinh et al., 2017)

Combining coupling layers: RealNVP



$$\mathbf{y}_1 = \mathbf{x}_1$$
$$\mathbf{y}_2 = S_a(\mathbf{x}_1) * \mathbf{x}_2 + T_a(\mathbf{x}_1)$$

$$\mathbf{z}_1 = S_b(\mathbf{y}_2) * \mathbf{y}_1 + T_b(\mathbf{y}_2)$$
$$\mathbf{z}_2 = \mathbf{y}_2$$

$$\mathbf{x}_1 = \mathbf{y}_1$$
$$\mathbf{x}_2 = (\mathbf{y}_2 - T_a(\mathbf{y}_1))/S_a(\mathbf{y}_1)$$

$$\mathbf{y}_1 = (\mathbf{z}_1 - T_b(\mathbf{z}_2))/S_b(\mathbf{z}_2)$$
$$\mathbf{y}_2 = \mathbf{z}_2$$

$T_a, T_b, \log S_a, \log S_b$: Deep netural networks

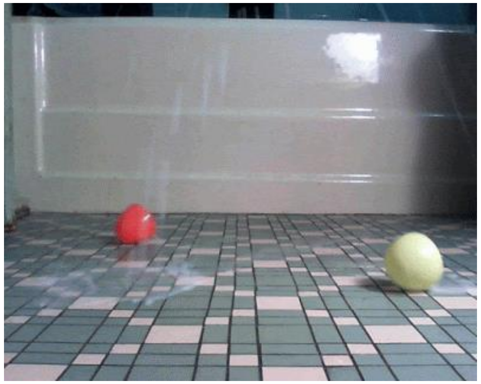
(Dinh et al., 2017)

Some recent progress

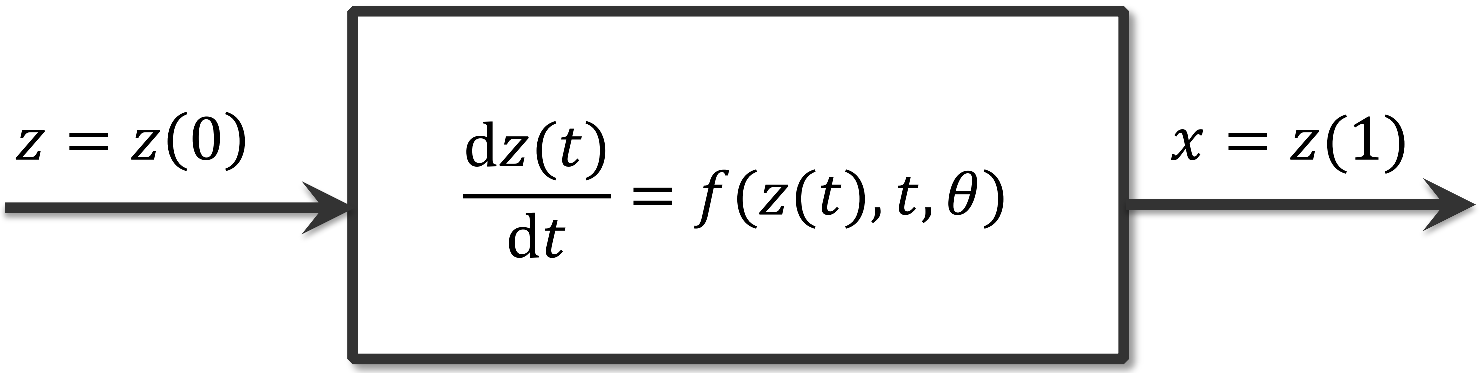
- Continuous time flow
- Discrete value flows

Time reversibility in physics

In classical mechanics, the time-reversibility is common

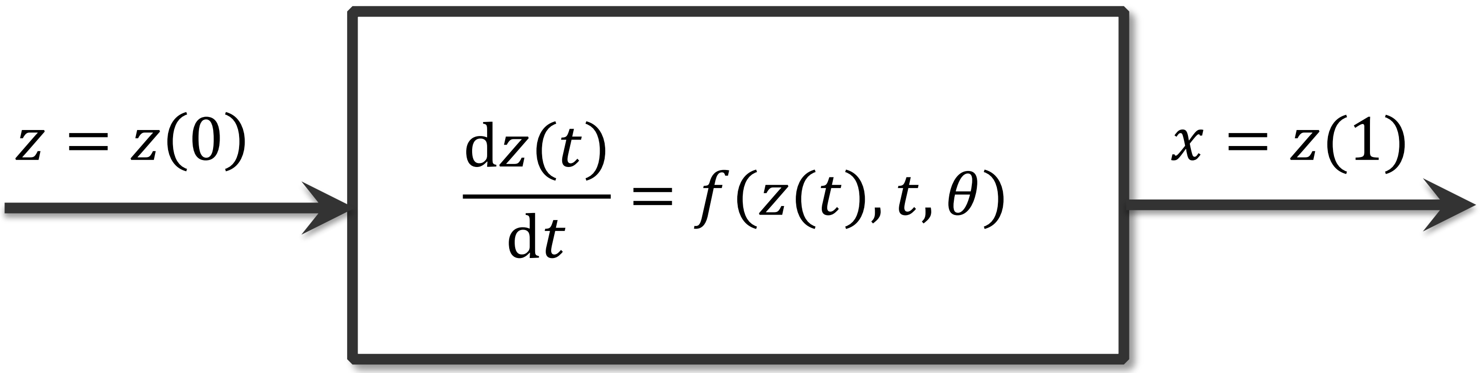


Continuous time flow



$z \longrightarrow x$ is invertible if f is uniformly Lipschitz continuous in z and continuous in t.

Continuous time flow

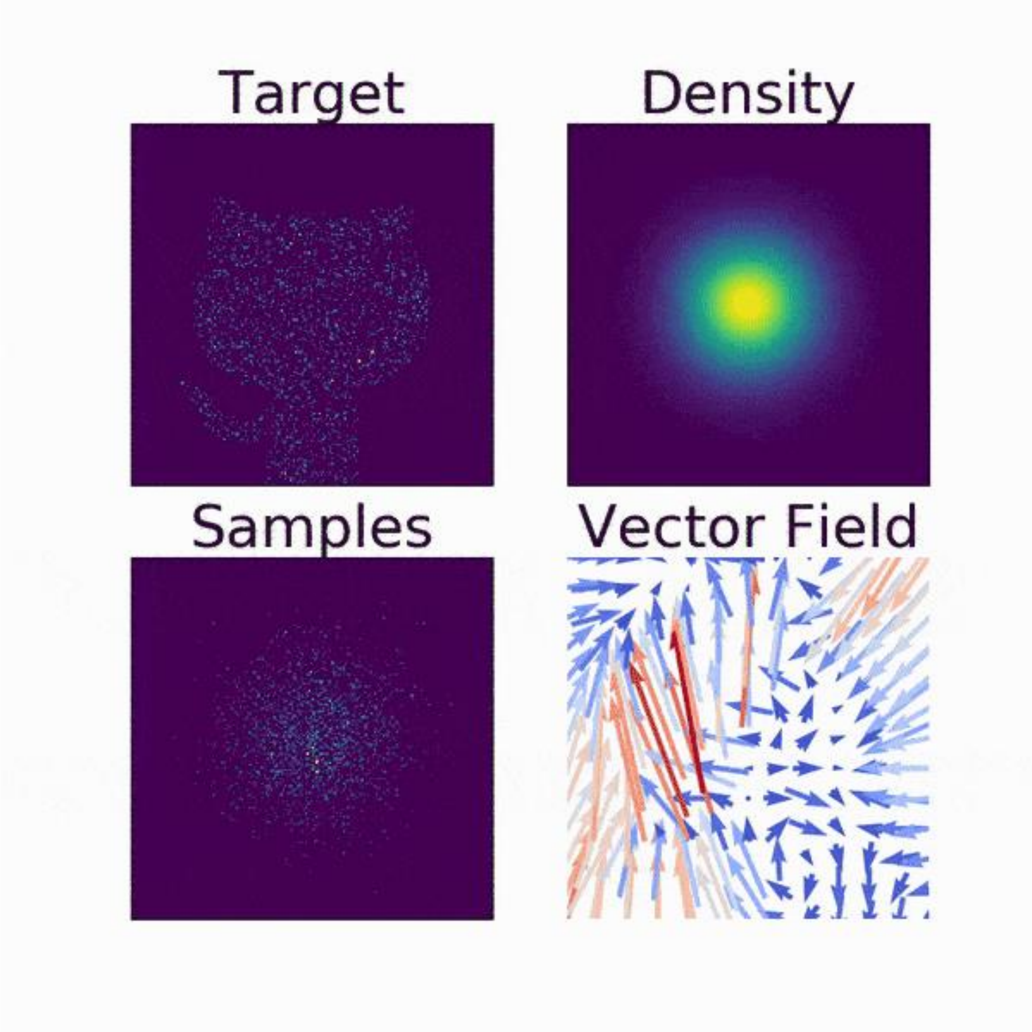


$$x = z(0) + \int_0^1 f(z(t), t, \theta)\,\mathrm{d}t$$

$$\log p(x) = \log p(z(0)) - \int_0^1 tr\left(\frac{\partial f}{\partial z(t)}\right)\mathrm{d}t$$

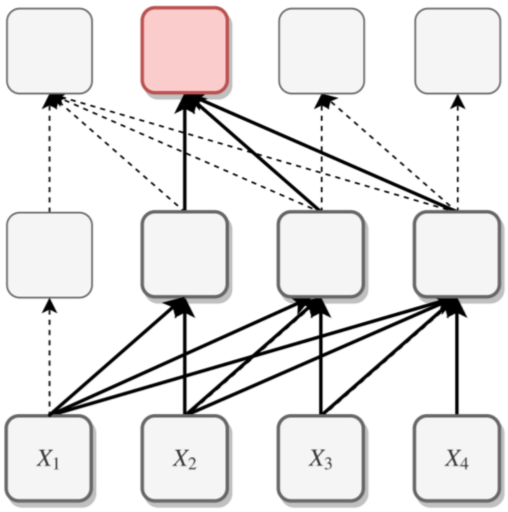
Chen, et al., 2018. Grathwohl, Chen, et al., 2019.

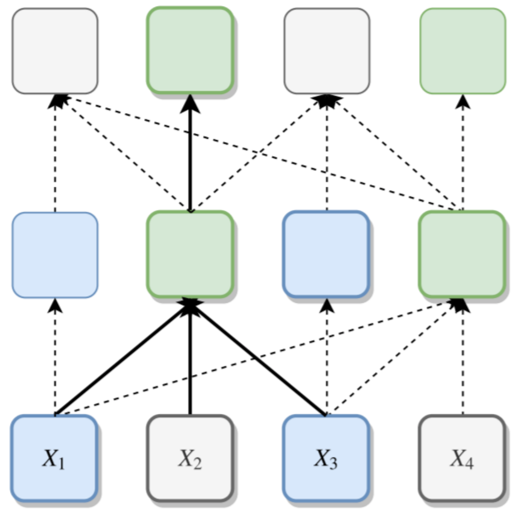
Continuous time flow



https://github.com/rtqichen/ffjord

Discrete values flow





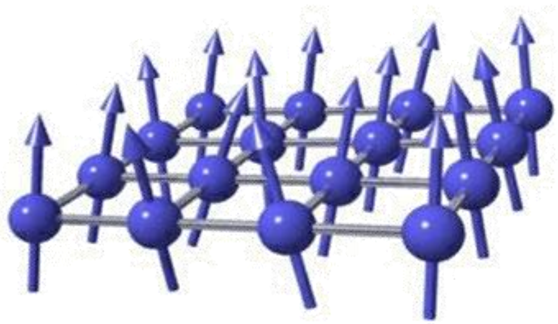
$$\mathbf{y}_d = \boldsymbol{\mu}_d \oplus \mathbf{x}_d,$$

$$\mathbf{y}_d = (\boldsymbol{\mu}_d + \boldsymbol{\sigma}_d \cdot \mathbf{x}_d) \bmod K.$$

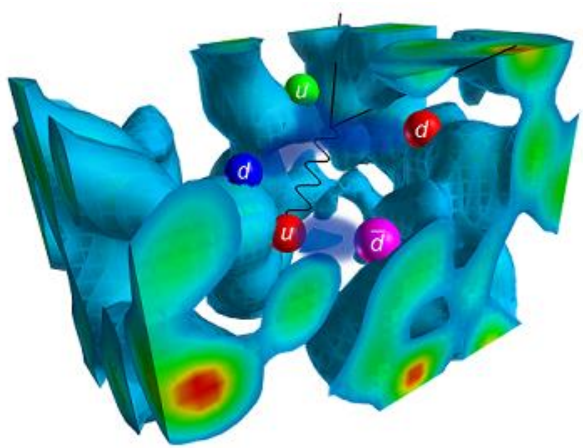
$$p(\mathbf{y} = y) = p(\mathbf{x} = f^{-1}(y))$$

NFs for energy landscape exploration

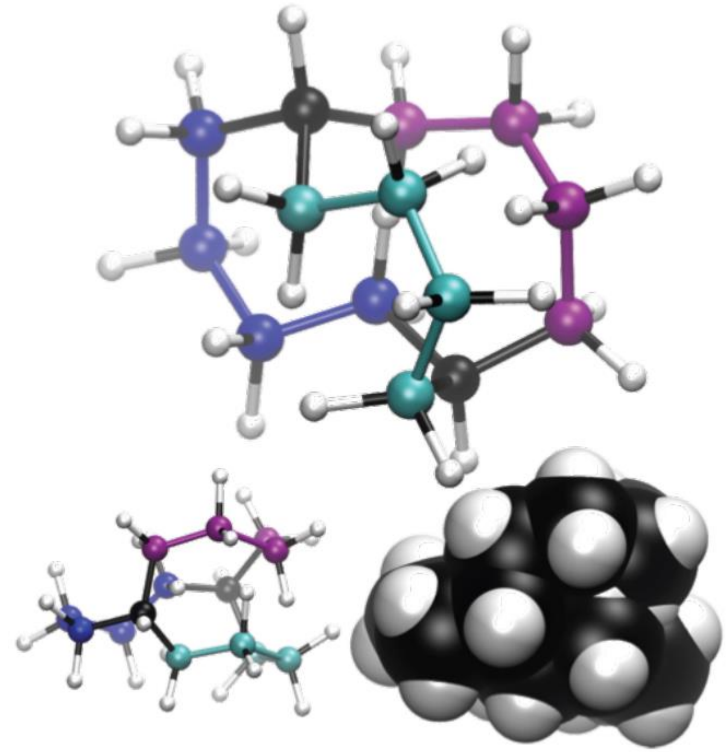
NFs for energy landscape exploration



Spin systems: Li and Wang, PRL, 2018

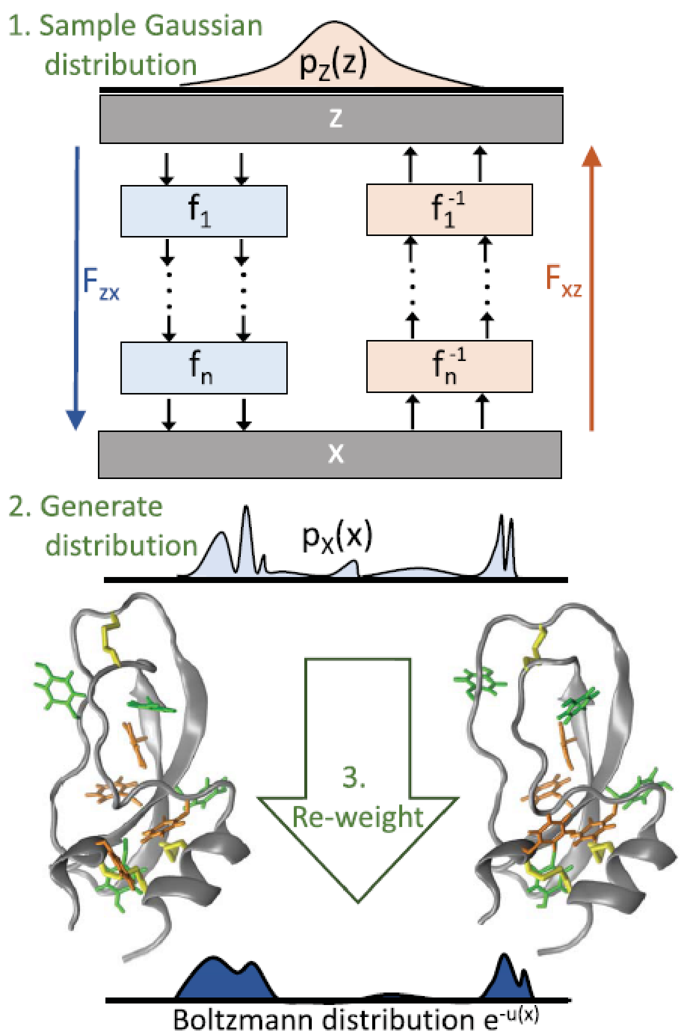


Lattice QCD: Kanwar et al., PRL, 2020



Molecular systems: Noé, Olsson, Köhler and Wu, Science, 2019

Why are NFs interesting?



- Normalizing flows (NFs) can be trained based on both energy and data:

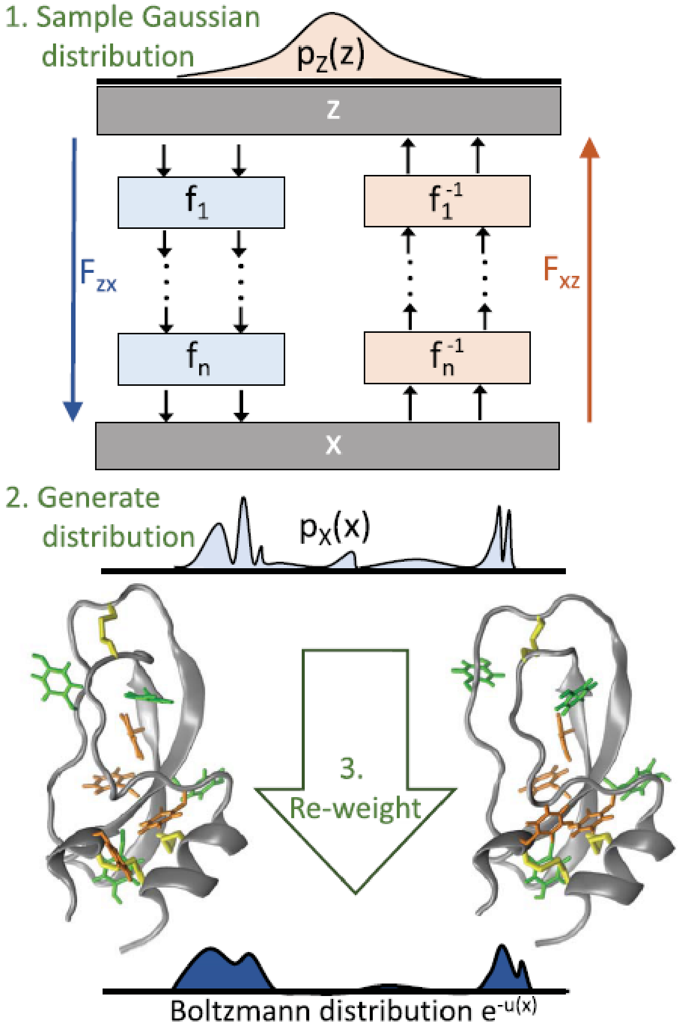
Energy based learning:

$$\min J_{KL} = \mathbb{E}_{p_Z}[\log q_X(F_{ZX}(\mathbf{z})) + u(F_{ZX}(\mathbf{z}))]$$

Data (likelihood) based learning:

$$\min J_{ML} = \mathbb{E}_{\text{data}}[-\log p_X(x)]$$

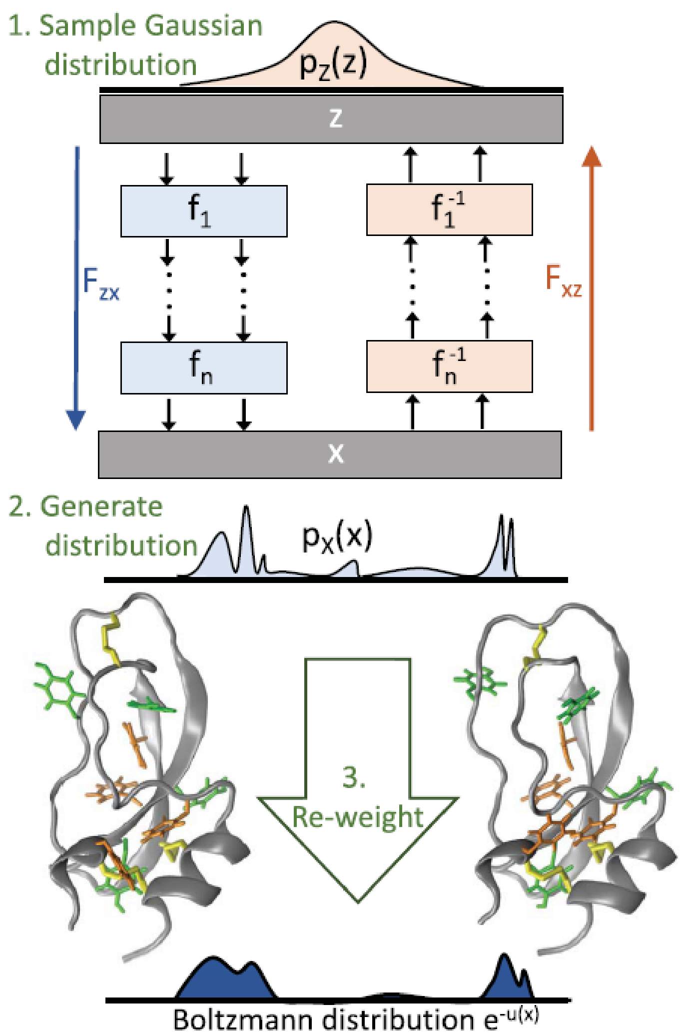
Why are NFs interesting?



- Asymptotically unbiased estimation can be obtained based on the exact density:

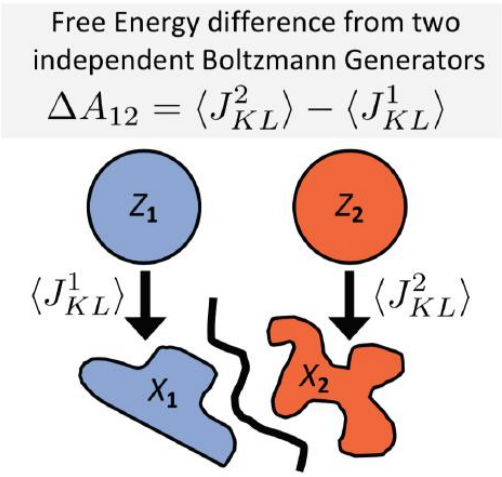
$$\mathbb{E}_{\mu}[O(x)] = \mathbb{E}_{p_X}\left[\frac{\mu(x)}{p_X(x)} O(x)\right]$$

Why are NFs interesting?



- The free energy difference can be directly calculated:

$$\mathrm{KL}(q_X||\mu)$$
$$= \mathbb{E}_{\mathbf{z}\sim q_Z}[\log q_X(F_{ZX}(\mathbf{z})) + u\,(F_{ZX}(\mathbf{z}))] + \mathrm{const}$$
$$= J_{KL} + \text{free energy}$$



Boltzmann Generators: NF + MCMC

1. Sample batch $\{\mathbf{x}_1, \ldots, \mathbf{x}_B\}$ from X.

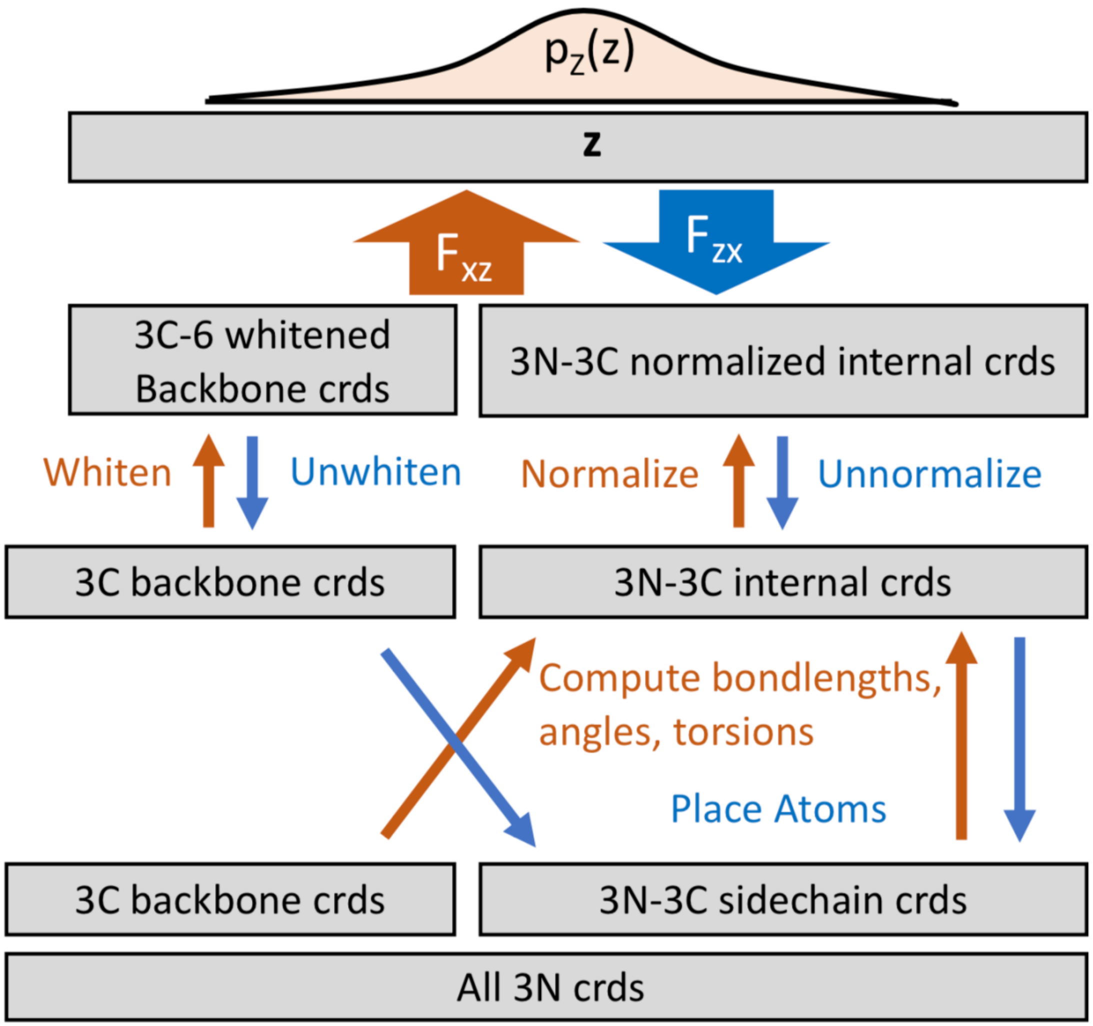
2. Update normalizing flow parameters θ by training on batch.

3. For each $\mathbf{x}$ in batch, project it to the latent space with $\mathbf{z} = F_{XZ}(\mathbf{x})$

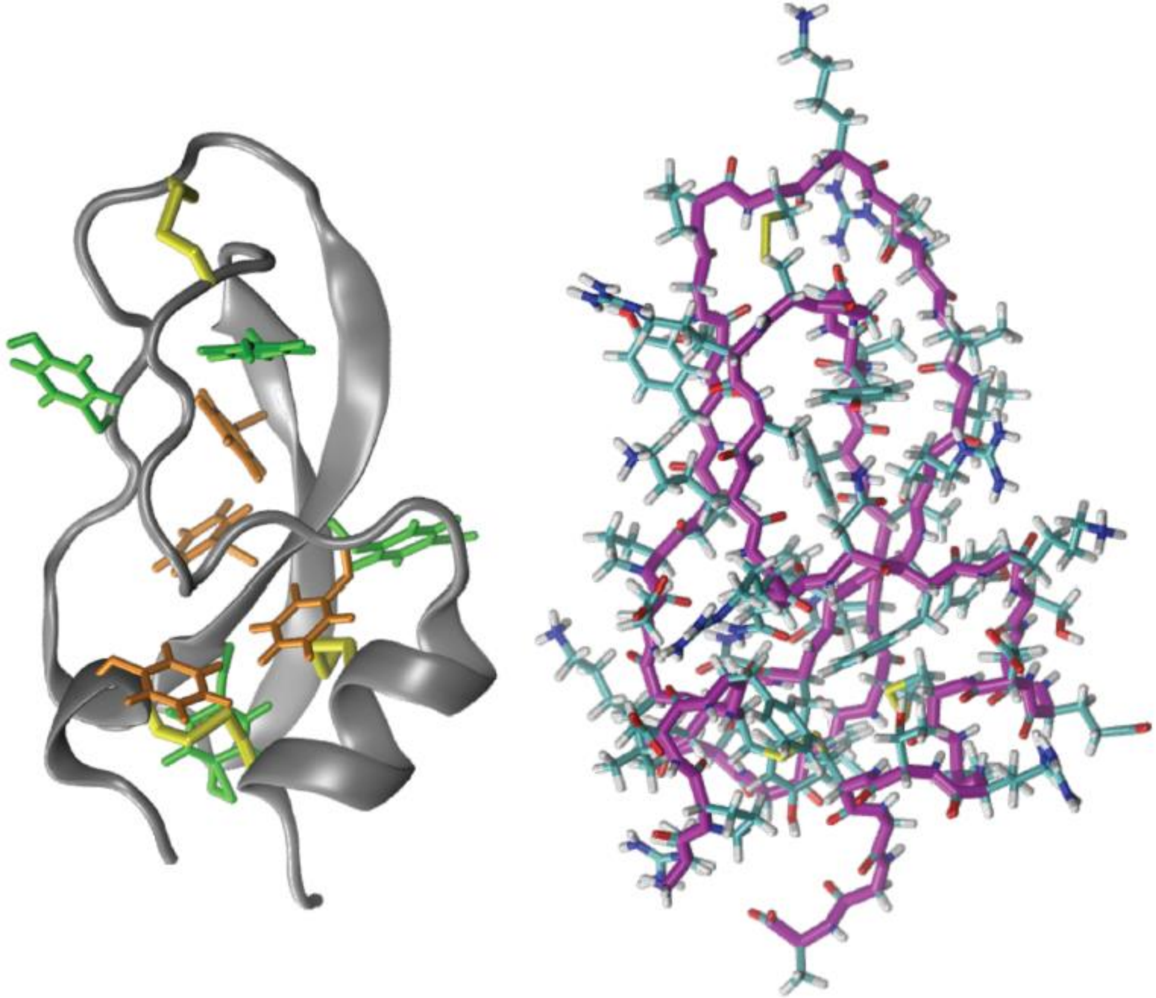
4. For each $\mathbf{z}$, perform MCMC with target distribution $\mu_Z(\mathbf{z}) = \left|\frac{\partial F_{XZ}(\mathbf{x})}{\partial \mathbf{x}}\right|^{-1} \mu_X(\mathbf{x})$, and get a new sample $\mathbf{z}'$.

5. Replace $\mathbf{x}$ by $\mathbf{x}' = F_{ZX}(\mathbf{x})$.

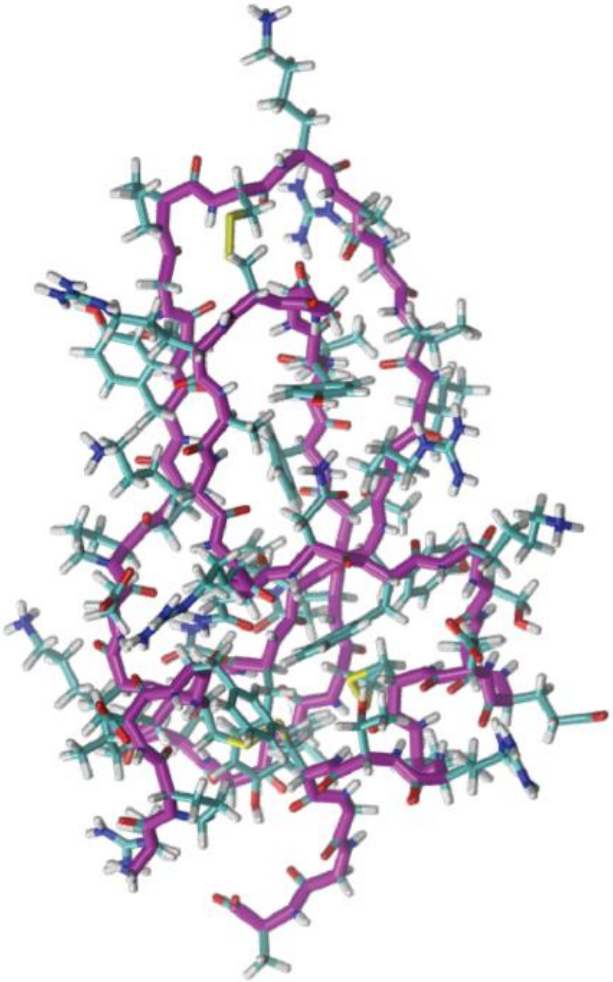
Towards proteins

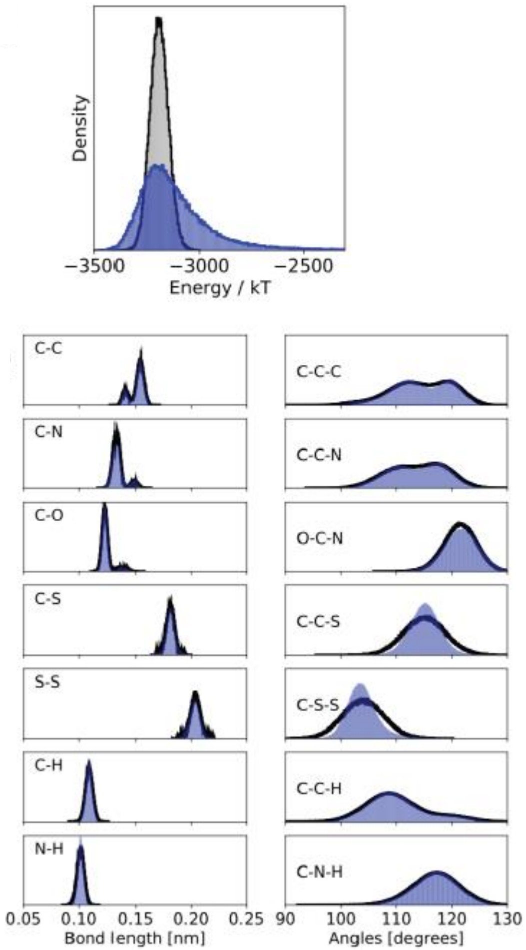


Towards proteins

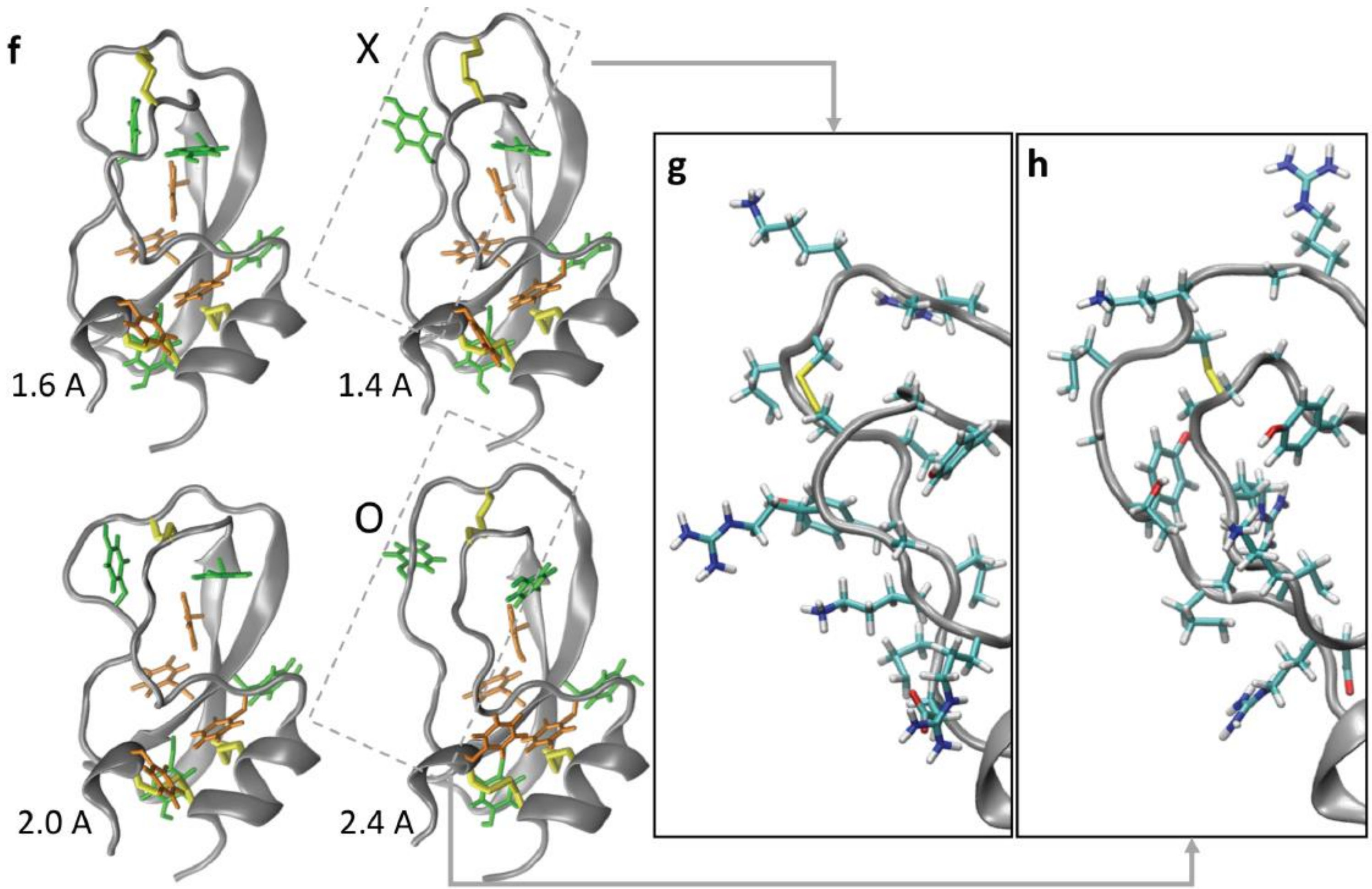


Towards proteins

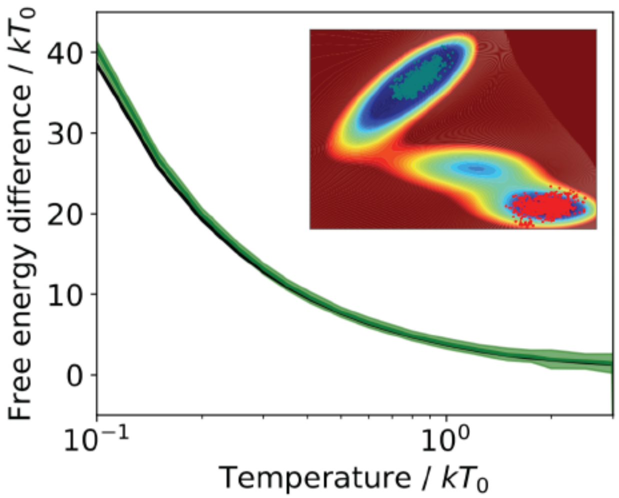


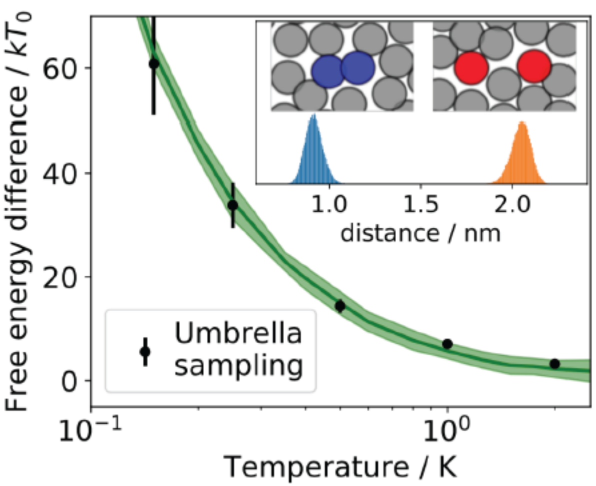


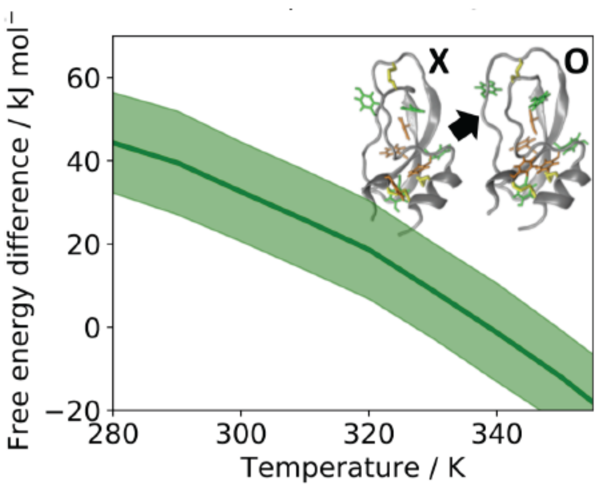
Towards proteins



Free energy differences



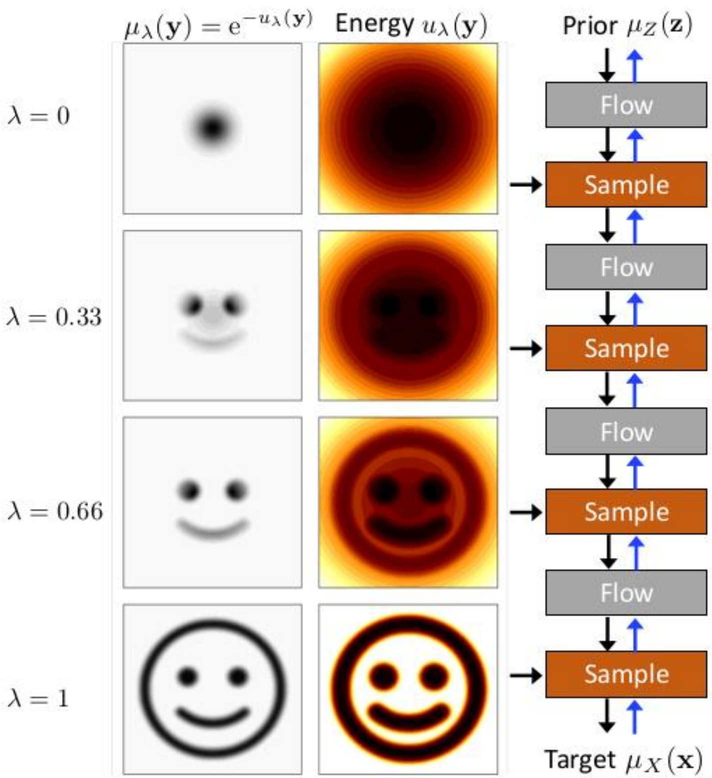


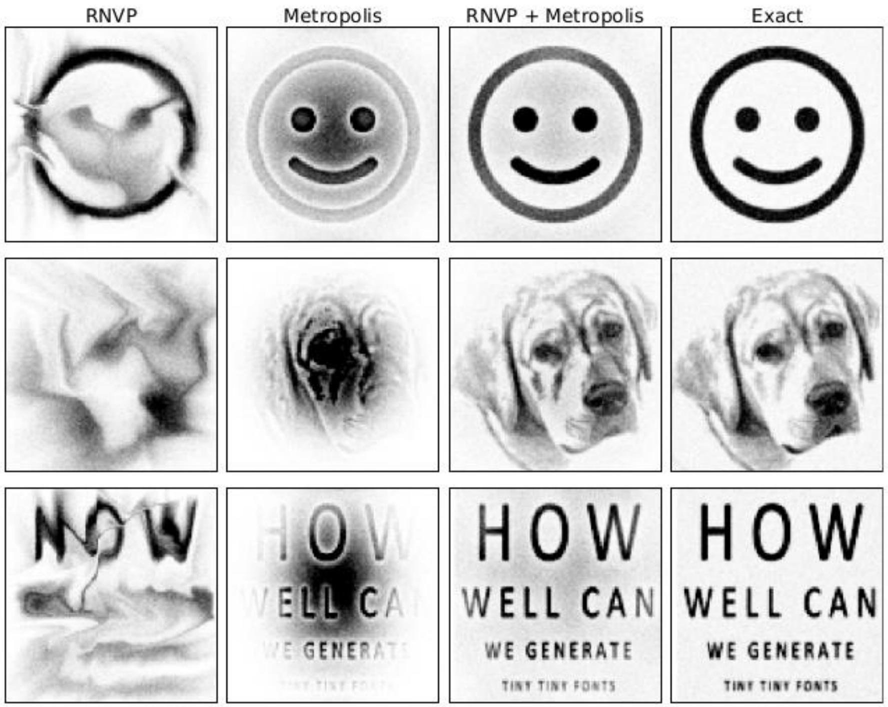


Extensions

Stochastic normalizing flows

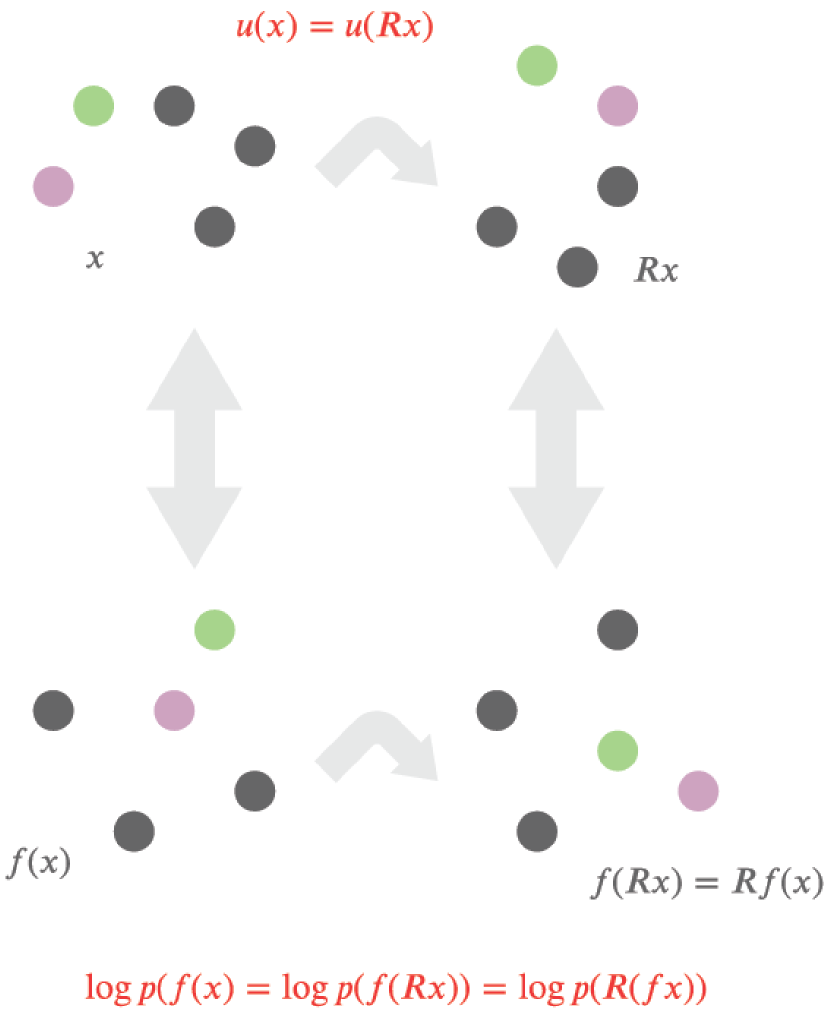
Combining normalizing flows and MCMC samplers

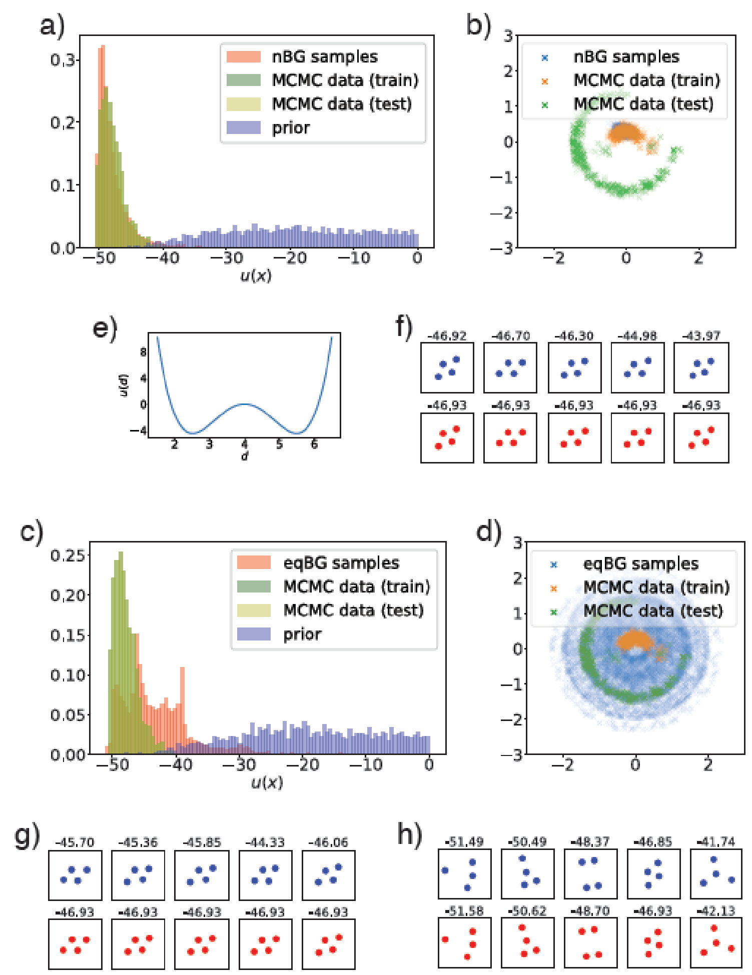




Wu, et al., NeurIPS).

Equivariant flows





Köhler, Klein and Noé, ICML 2020.

THE END

Thanks! Questions?