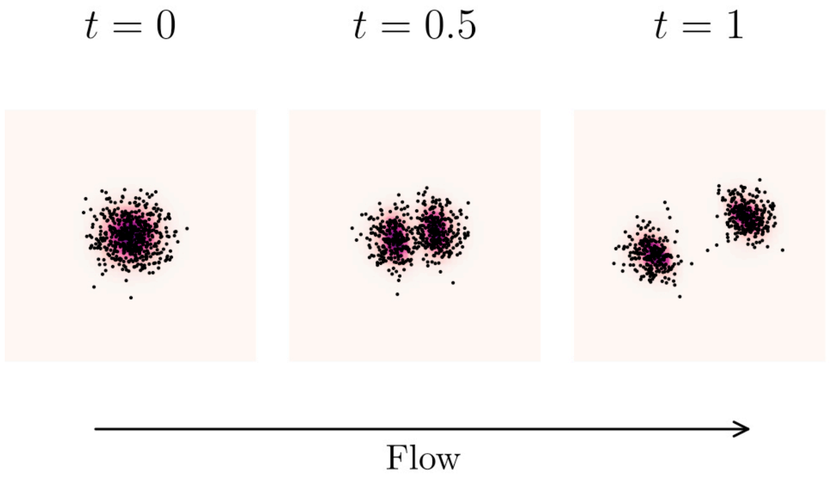
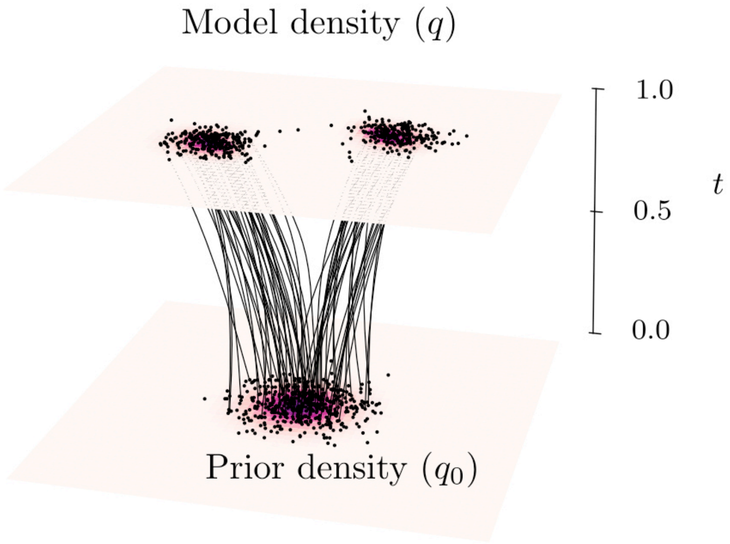
Normalizing flows for Lattice gauge theory





Gurtej Kanwar
Chancellor's Fellow in AI & Datascience
University of Edinburgh

Jul 28 - Aug 1, 2025
MITP Summer School

Lecture 1

Markov chains

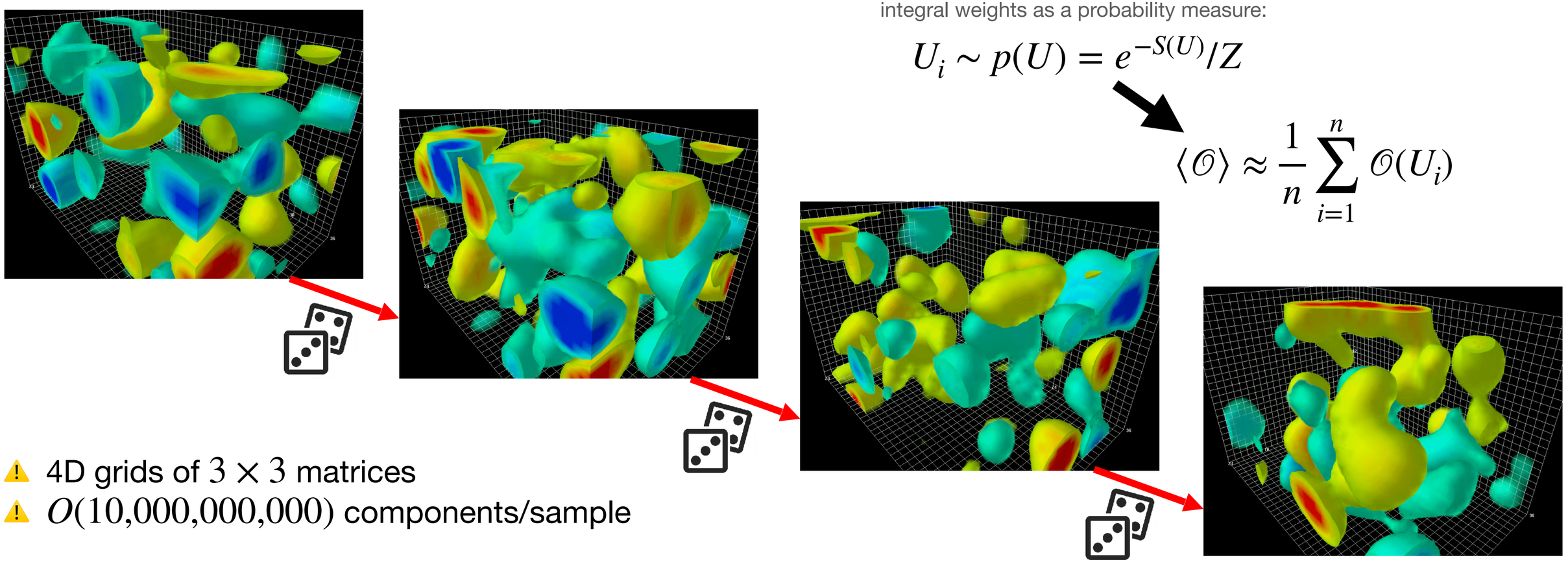
$$\langle \mathcal{O} \rangle = \left[\prod_{x,\mu} \int dU_\mu(x) \right] \mathcal{O}(U)\, e^{-S(U)}/Z$$

Estimate observables by averaging over Markov chain Monte Carlo samples.

Positive integrand allows interpreting path integral weights as a probability measure:

$$U_i \sim p(U) = e^{-S(U)}/Z$$

$$\langle \mathcal{O} \rangle \approx \frac{1}{n} \sum_{i=1}^{n} \mathcal{O}(U_i)$$



- ⚠ 4D grids of 3×3 matrices
- ⚠ $O(10{,}000{,}000{,}000)$ components/sample

Image credit: Leinweber, et al.

Critical slowing down (CSD)

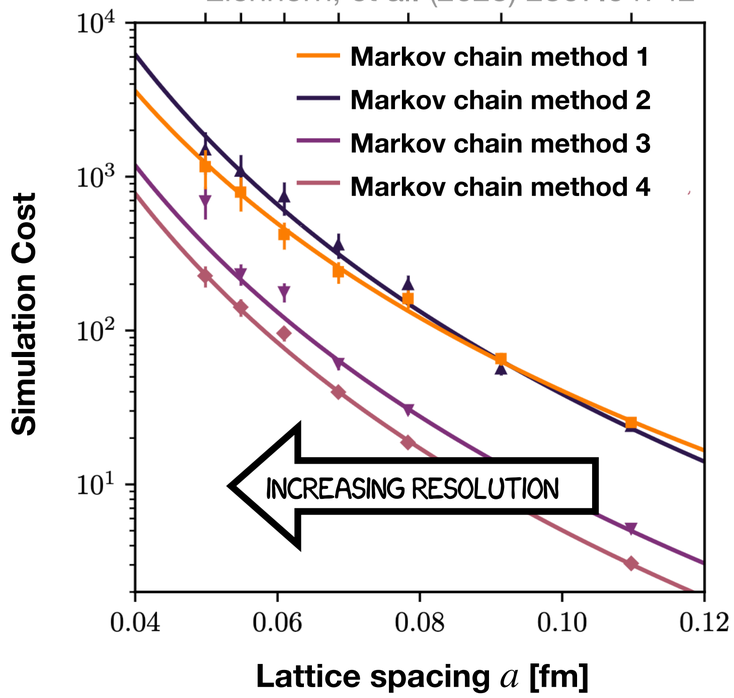
Markov chains are typically **inefficient** as $a \to 0$

- Information transfer is local while correlations grow
- **CSD:** diverging autocorrelation time
- **Topological freezing:** Markov chain gets “stuck”

This **limits the physics results:**

- Heavy quarks (short-range interactions)
- Systematic uncertainties

Eichhorn, et al. (2023) 2307.04742

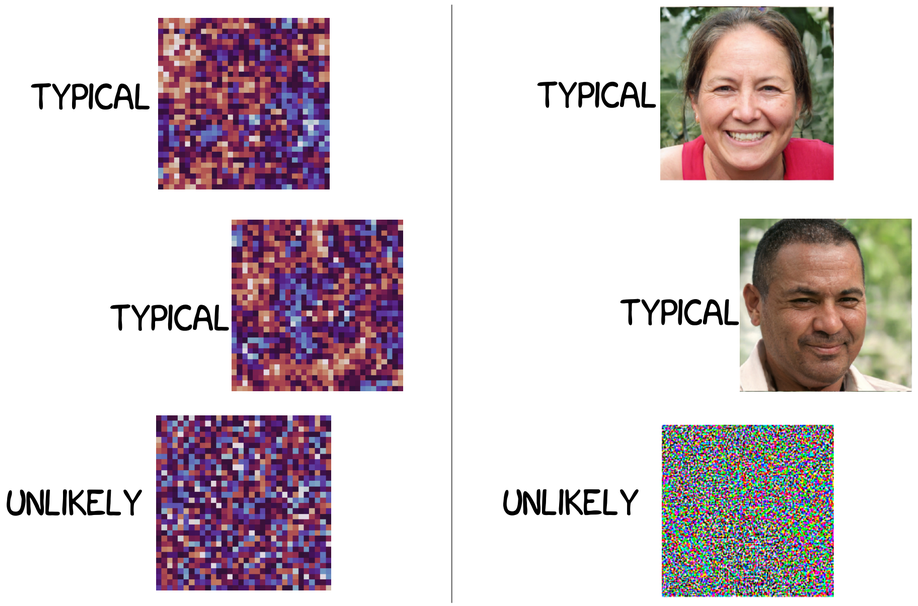


Lattice Monte Carlo ~ Image generation

Generative models can directly produce "typical" samples

Lattice sampling vs. Generative AI

Karras, Lane, Aila (NVIDIA) 1812.04948

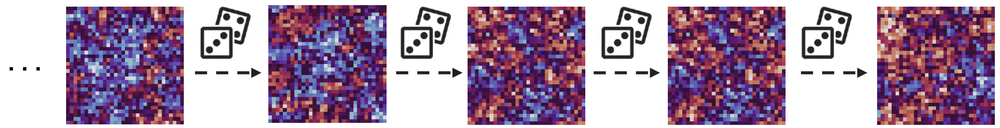


Unique features:

- ⚠ Demand **unbiased results**
- ⚠ Not much **existing training data**
  … $O(1000)$ samples at finest resolutions
  … $O(10{,}000{,}000{,}000)$ components/sample
- ✓ Know **target probability density**
- ✓ Know **physical symmetries**

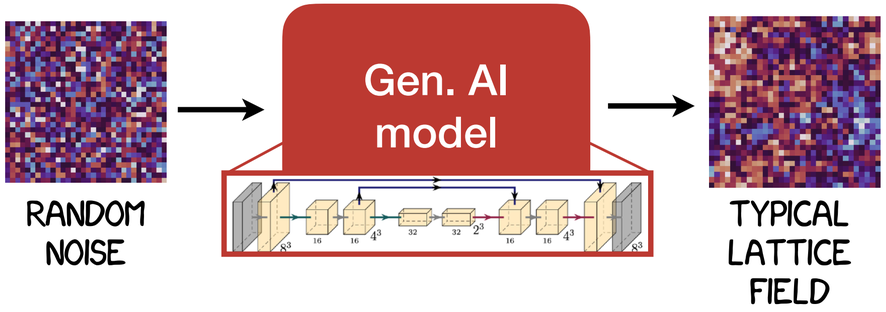
Can we use **Generative AI** to solve critical slowing down?

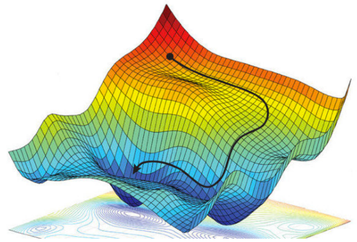
Replace or augment **Markov Chain Monte Carlo**…

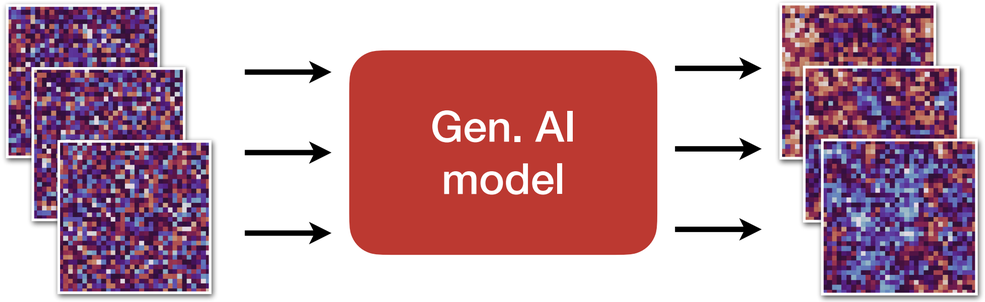


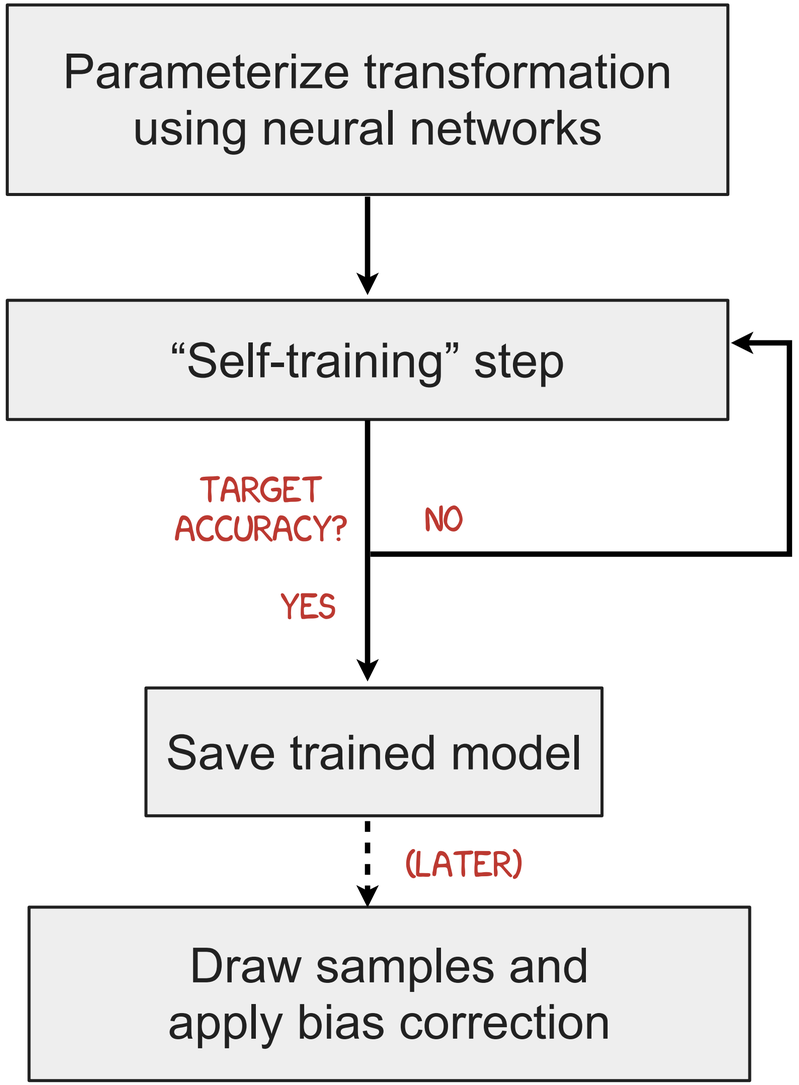
… with generated samples?

Generative AI models for Lattice QCD









Survey of image models

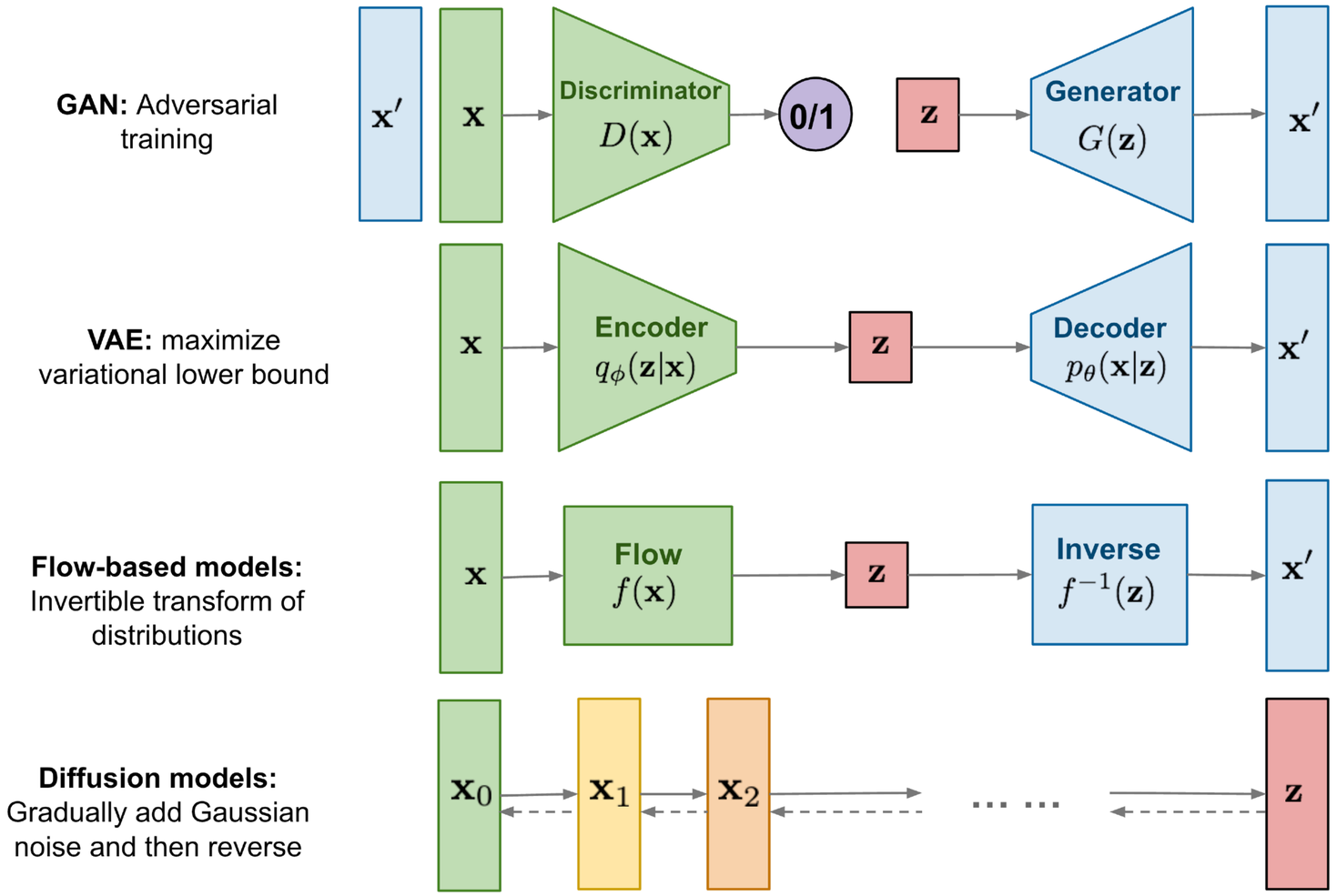


Image credit: Lilian Weng

Survey of image models

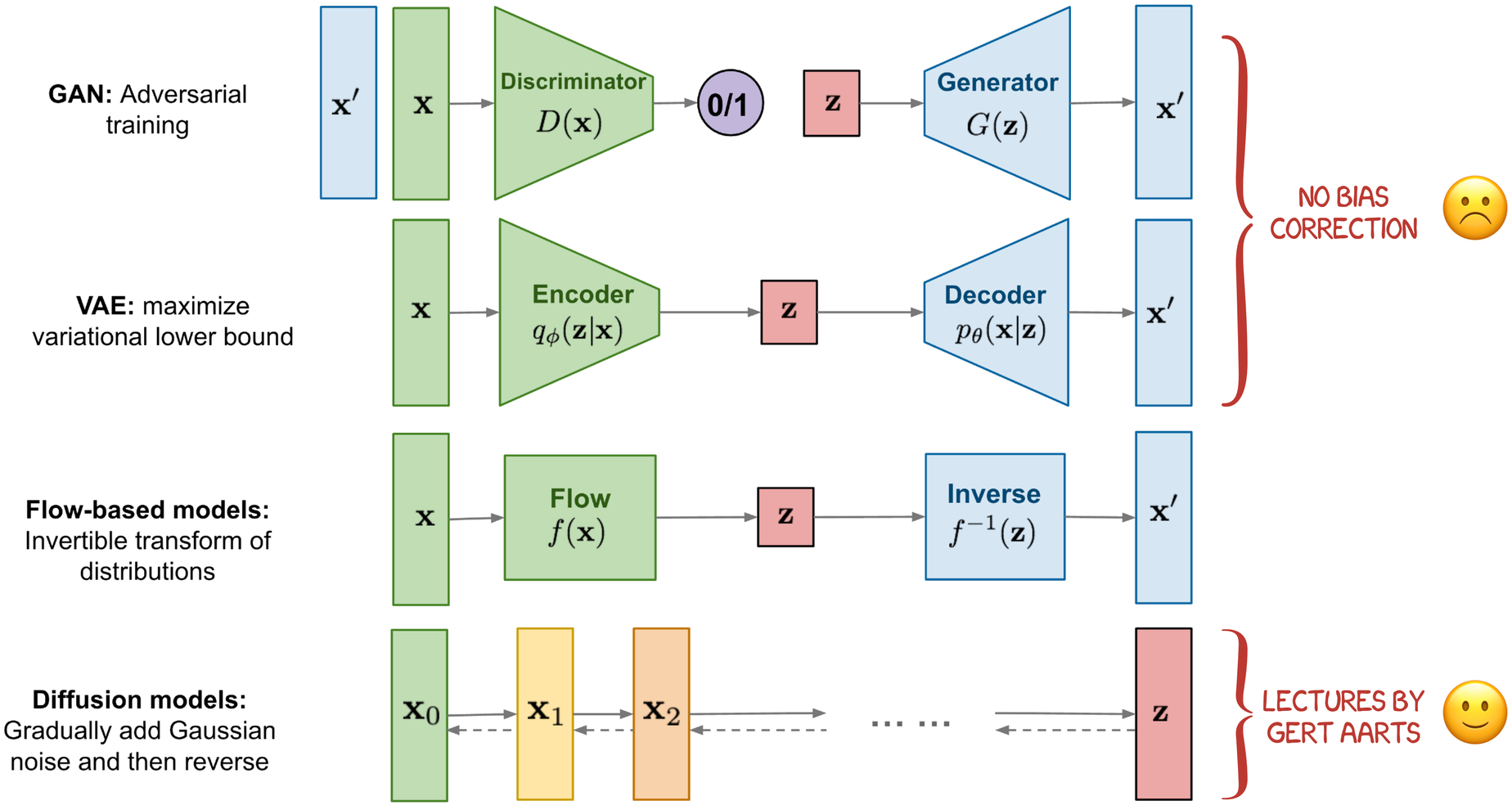


Image credit: Lilian Weng

Survey of image models

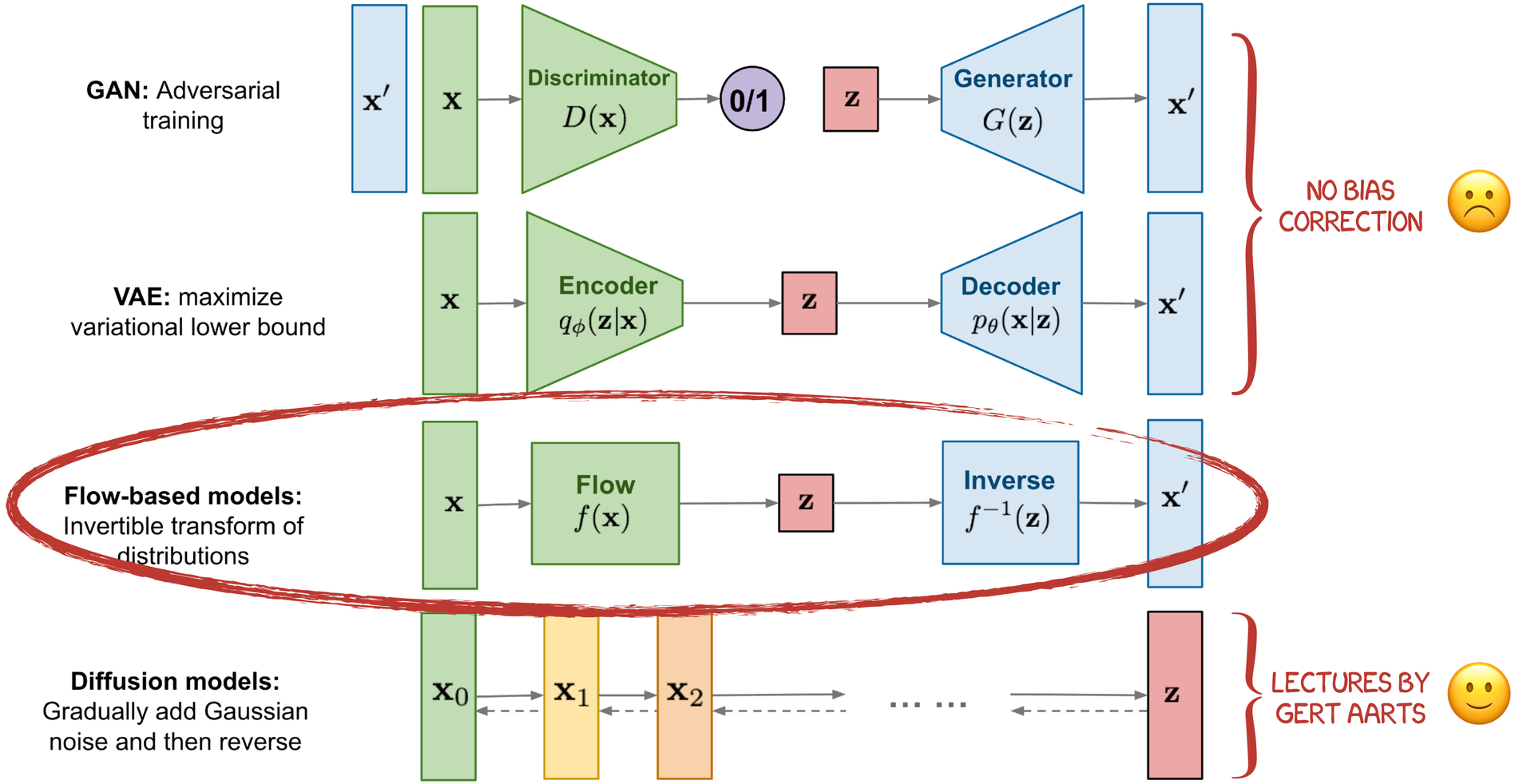
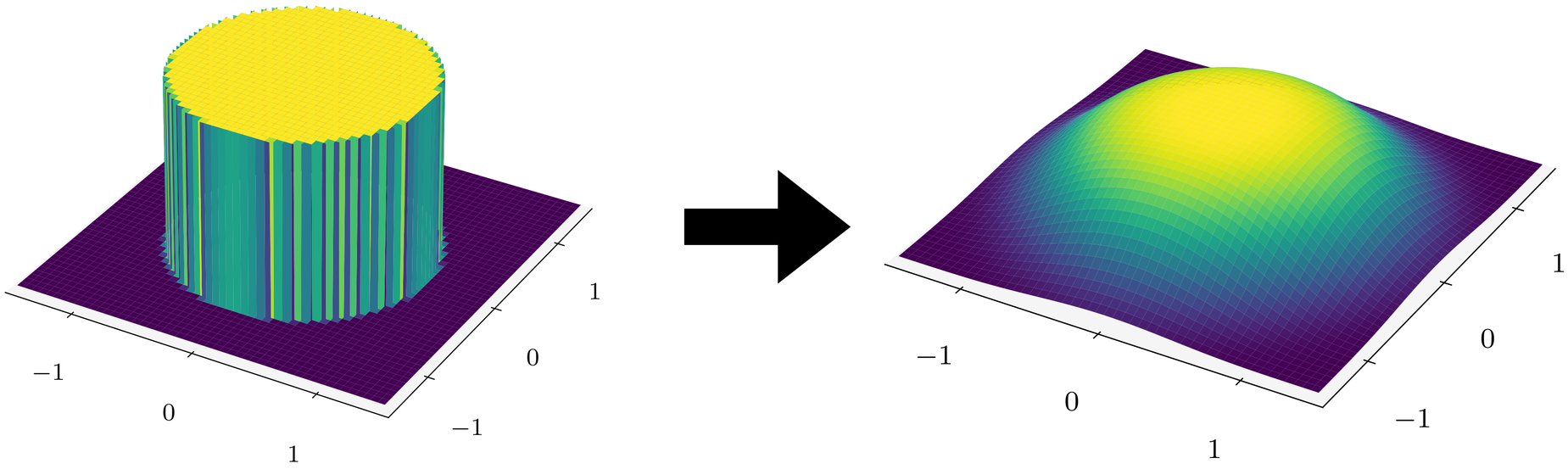


Image credit: Lilian Weng

Sampling using flows

Box-Muller transform to map $\mathrm{Unif}[\mathrm{circle}] \to \mathcal{N}(0,1)^2$

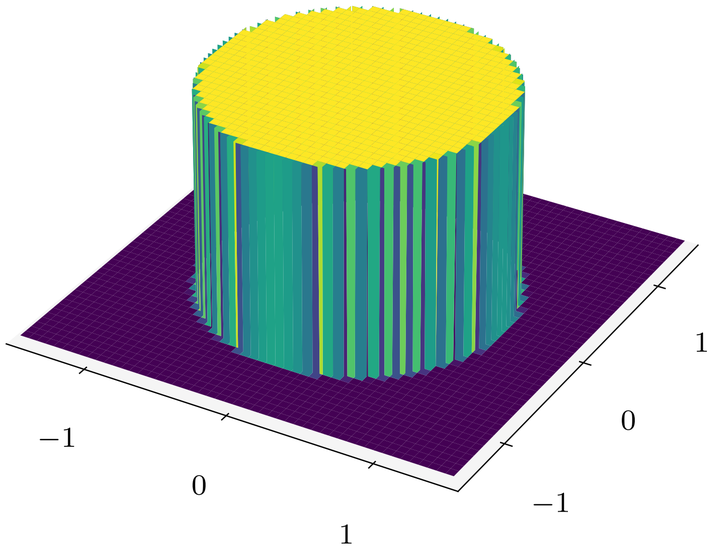
$$x' = \frac{x}{r}\sqrt{-2\ln r^2} \qquad y' = \frac{y}{r}\sqrt{-2\ln r^2}$$



Sampling using flows

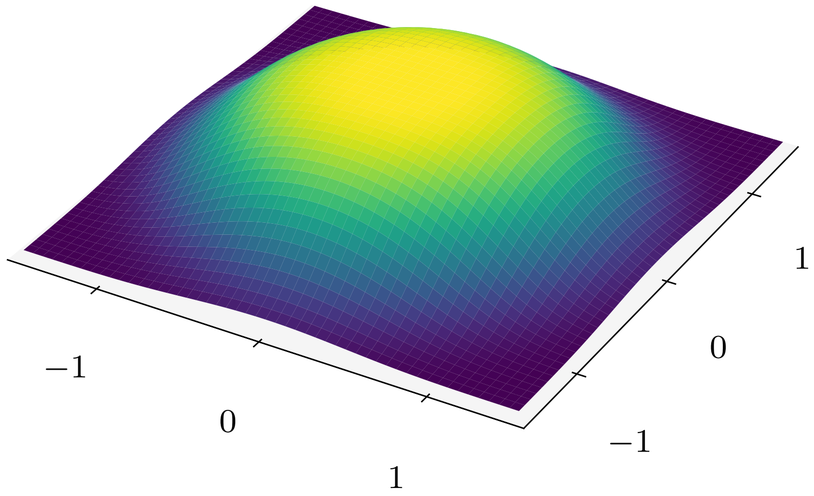
Box-Muller transform to map $\mathrm{Unif}[\mathrm{circle}] \to \mathcal{N}(0,1)^2$

$$x' = \frac{x}{r}\sqrt{-2\ln r^2} \qquad y' = \frac{y}{r}\sqrt{-2\ln r^2}$$



(Simple) Prior density:
$r(x, y)$

Flow f

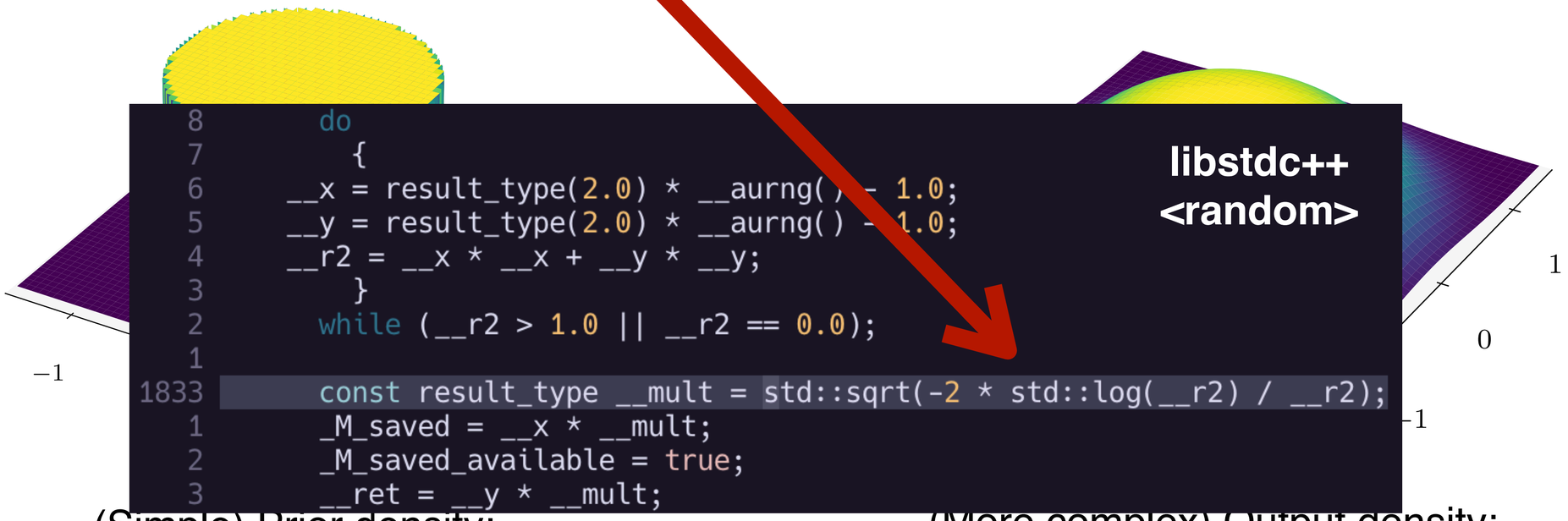


(More complex) Output density:
$q(x', y') = r(x, y)\,|\det J|^{-1}$

Sampling using flows

Box-Muller transform to map $\mathrm{Unif}[\mathrm{circle}] \to \mathcal{N}(0,1)^2$

$$x' = \frac{x}{r}\sqrt{-2\ln r^2} \qquad y' = \frac{y}{r}\sqrt{-2\ln r^2}$$




```
      do
        {
      __x = result_type(2.0) * __aurng() - 1.0;
      __y = result_type(2.0) * __aurng() - 1.0;
      __r2 = __x * __x + __y * __y;
        }
      while (__r2 > 1.0 || __r2 == 0.0);

      const result_type __mult = std::sqrt(-2 * std::log(__r2) / __r2);
      _M_saved = __x * __mult;
      _M_saved_available = true;
      __ret = __y * __mult;
```

libstdc++ <random>

(Simple) Prior density:

$r(x, y)$

(More complex) Output density:

$q(x', y') = r(x, y)\,|\det J|^{-1}$

Flow-based generative models

Tabak & Vanden-Eijnden CMS8 (2010) 217
Tabak & Turner CPA66 (2013) 145
Rezende & Mohamed (2015) PMLR 37, 1530

1. Sample from “easy” prior density

$$V \sim r(V)$$

2. Apply parametrized diffeomorphism (the “flow”)

$$U = f(V)$$

3. Output samples follow new “model density” thanks to **change of measure** formula

$$q(U) = r(V)\,|\det \partial U/\partial V|^{-1} = r(V)\,J_f(V)^{-1}$$

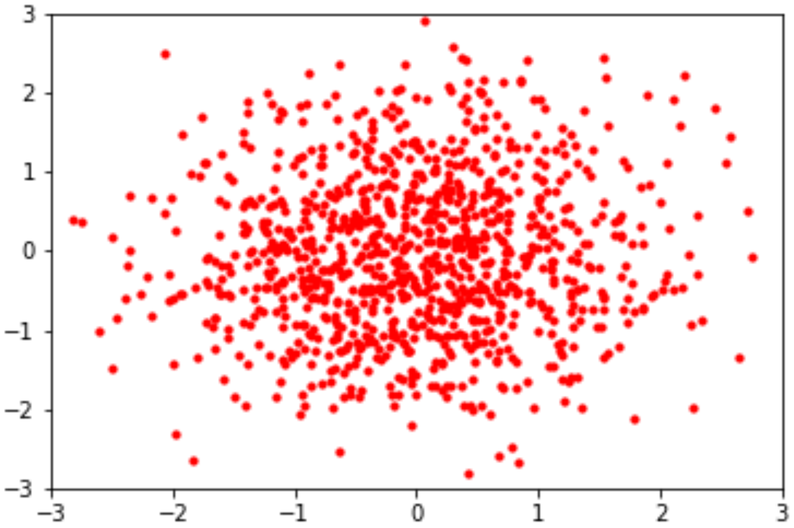


Image credit: blog.evjang.com/2019/07/nf-jax.html

Flow-based generative models

Tabak & Vanden-Eijnden CMS8 (2010) 217
Tabak & Turner CPA66 (2013) 145
Rezende & Mohamed (2015) PMLR 37, 1530

1. Sample from “easy” prior density

$$V \sim r(V)$$

2. Apply parametrized diffeomorphism (the “flow”)

$$U = f(V)$$

3. Output samples follow new “model density” thanks to **change of measure** formula

$$q(U) = r(V)\,|\det \partial U/\partial V|^{-1} = r(V)\,J_f(V)^{-1}$$

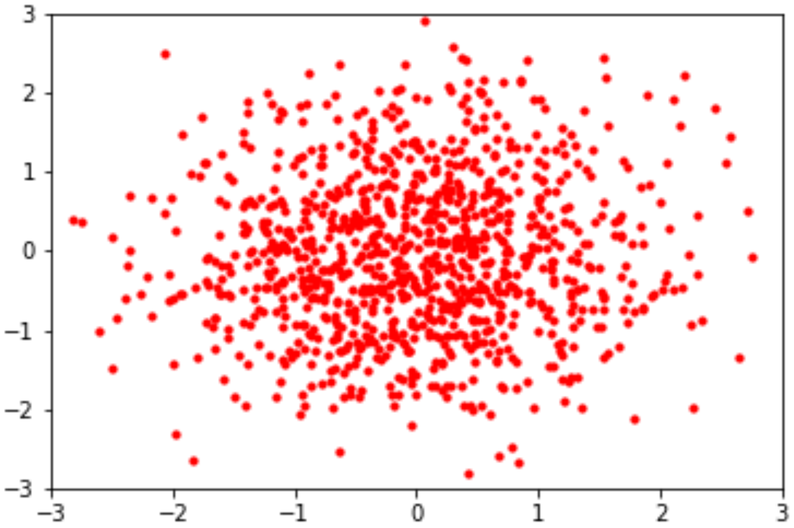


Image credit: blog.evjang.com/2019/07/nf-jax.html

Learned flows

Learned flows

Approximate more complicated distributions by optimizing flexible ansatz f.

(E.G. USING ML METHODS)

Learned flows

Approximate more complicated distributions by optimizing flexible ansatz f.

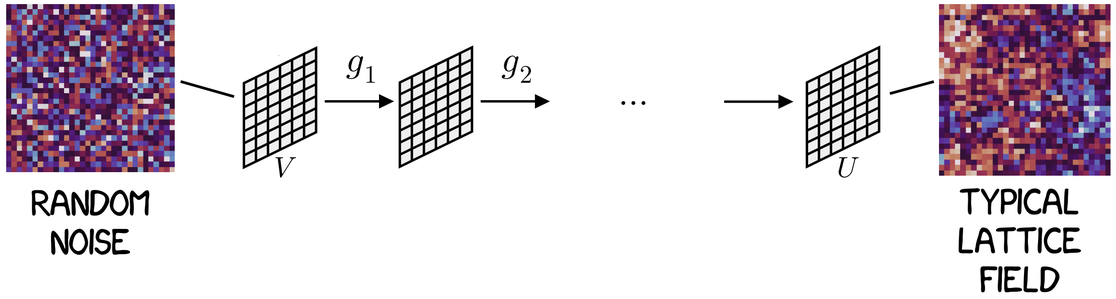
(E.G. USING ML METHODS)

- Discrete coupling layers:

  Dinh, et al (2014) 1410.8516; Dinh, et al (2016) 1605.08803

FLOW $f = g_1 \circ \ldots \circ g_n$

JAC. $\det J = \det J_1 \cdot \ldots \cdot \det J_n$



Learned flows

Approximate more complicated distributions by optimizing flexible ansatz f.

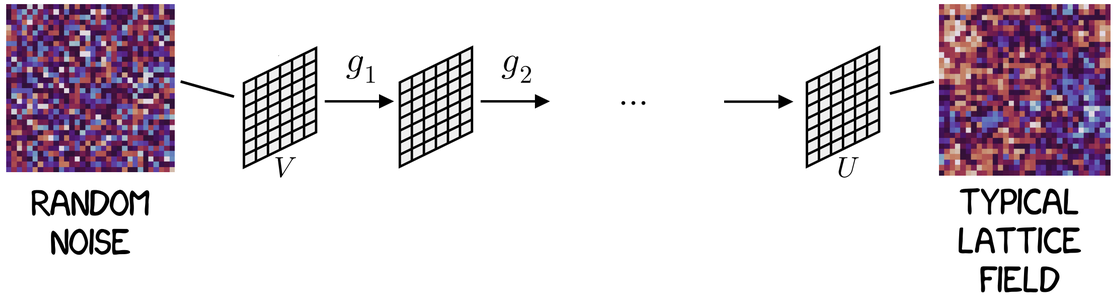
(E.G. USING ML METHODS)

- Discrete coupling layers:

  Dinh, et al (2014) 1410.8516; Dinh, et al (2016) 1605.08803

  FLOW $f = g_1 \circ \ldots \circ g_n$

  JAC. $\det J = \det J_1 \cdot \ldots \cdot \det J_n$



- Continuous flows:

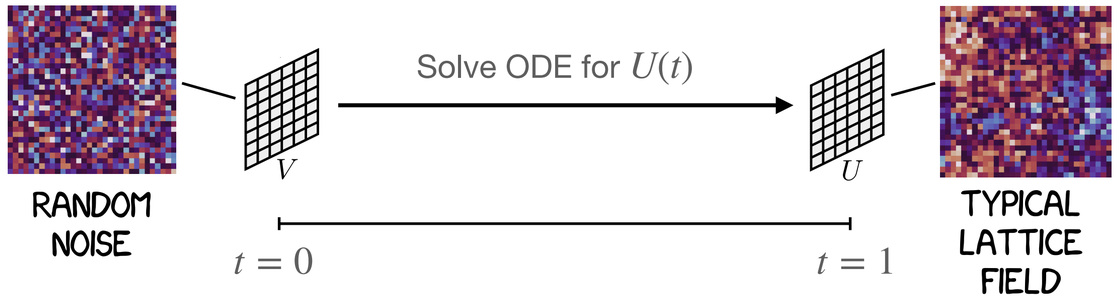
  Chen, et al (2018) 1806.07366; Zhang, et al (2018) 1809.10188

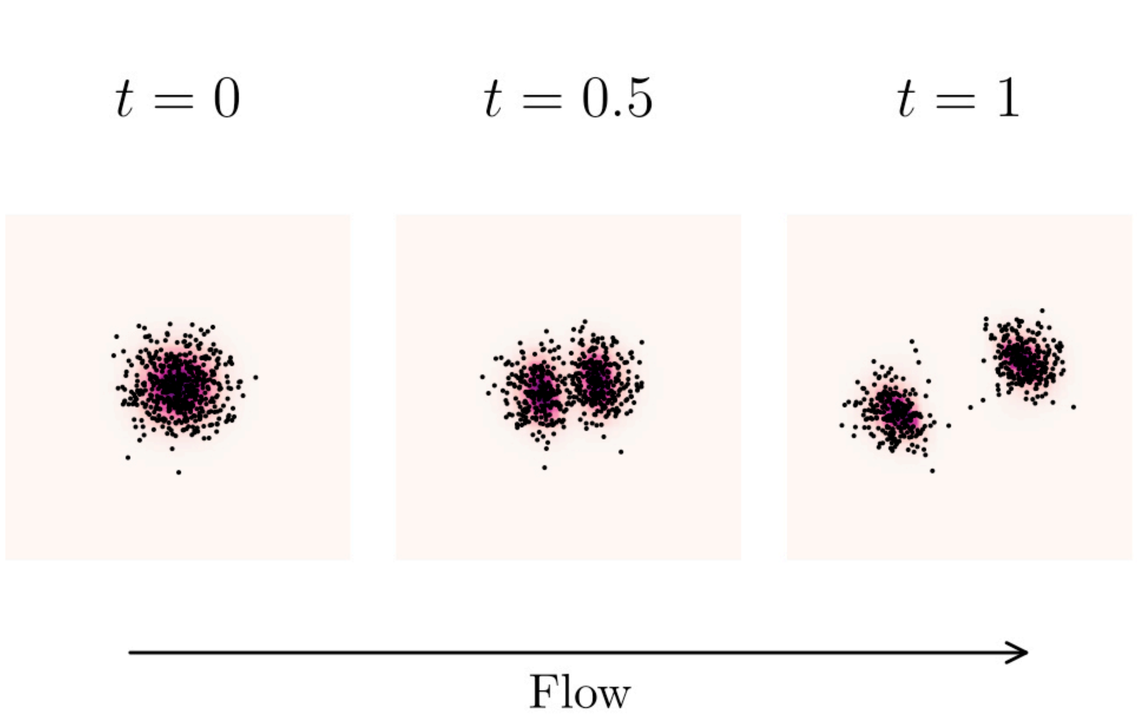
  FLOW $\dot{U}(t) = \nabla\varphi(U(t);t)\,U(t)$

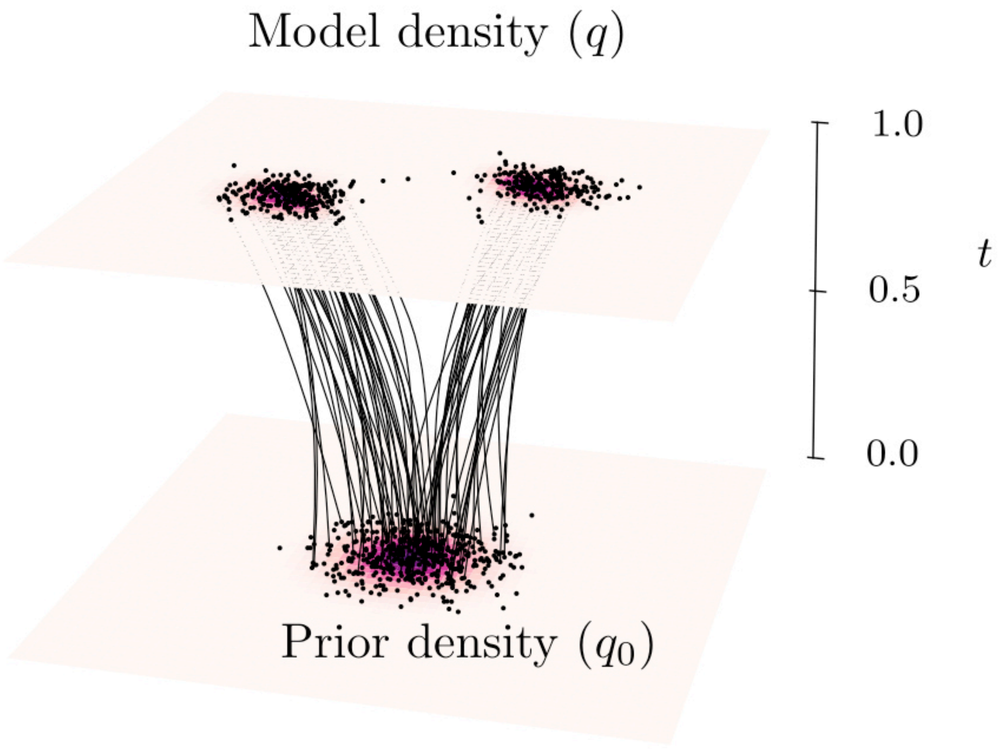
  with $f(V) := U(t=1)$

  JAC. $\ln\det J = -\int_0^1 dt\,\nabla^2\varphi(U(t);t)$

(NOTE: DERIVATIVE ∇ EVALUATED IN LIE ALGEBRA)



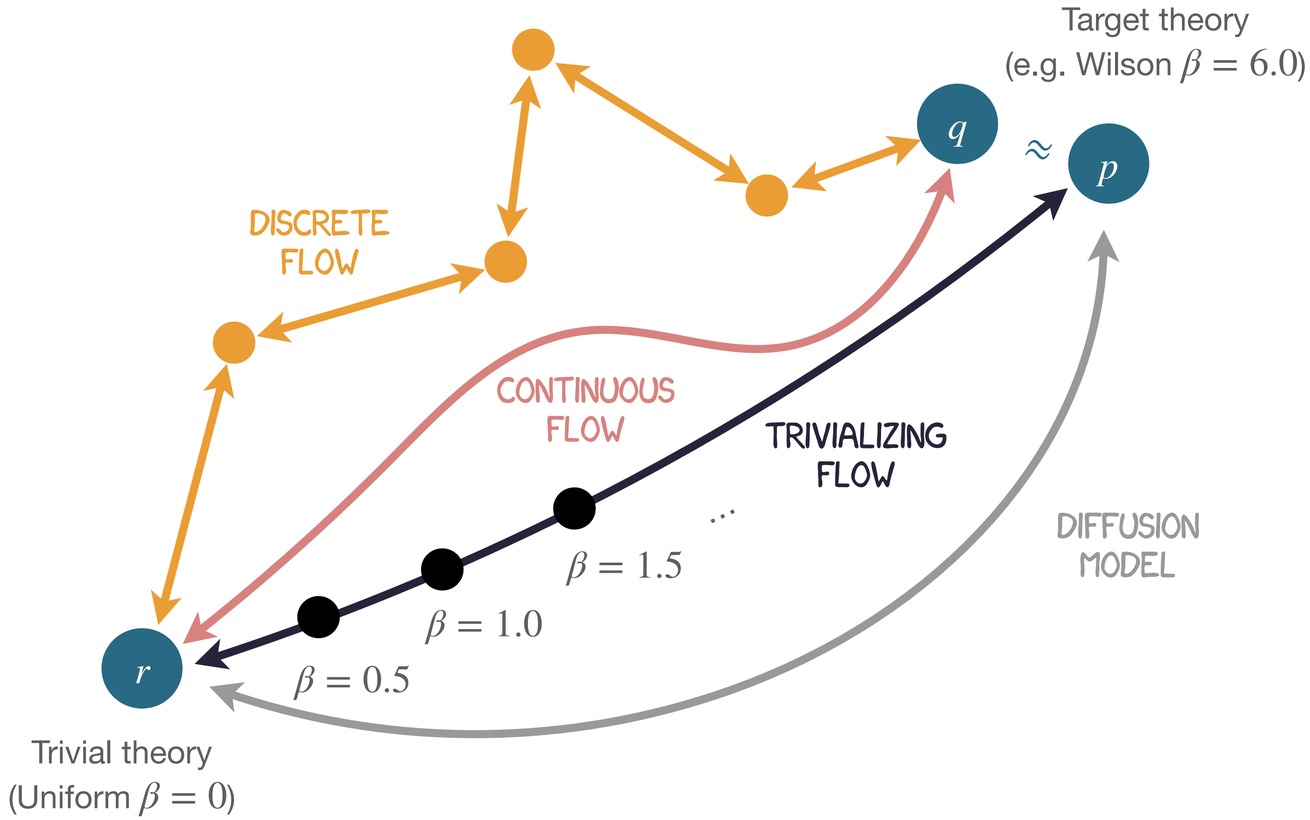

$t = 0$
$t = 0.5$
$t = 1$
Flow


Model density (q)
1.0
0.5
0.0
t
Prior density (q_0)

Flow-based sampling strategies

Various possible paths through space of distributions:

- **Freely learnable**
  - E.g. discrete / continuous flow models
- **Explicit fixed path**
  - Define $S_t(U)$ for $t \in [0,1]$
  - E.g. Lüscher's trivializing flow
- **Implicit fixed path**
  - Interpolation on samples
  - E.g. diffusion models, flow matching



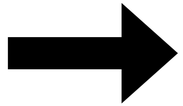
Exactness

Samples from model are from **biased distribution** $q(U) \neq p(U)$, but…

For each U_i drawn from the model, we know $q(U_i)$ and $p(U_i)$

Flow-based models provide this.

Known in terms of the lattice action.



Exact bias correction possible
(e.g. “flow-based MCMC” or reweighting)

$$\langle \mathcal{O} \rangle_p = \frac{\langle \mathcal{O}(U)\, p(U)/q(U) \rangle_q}{\langle p(U)/q(U) \rangle_q}$$

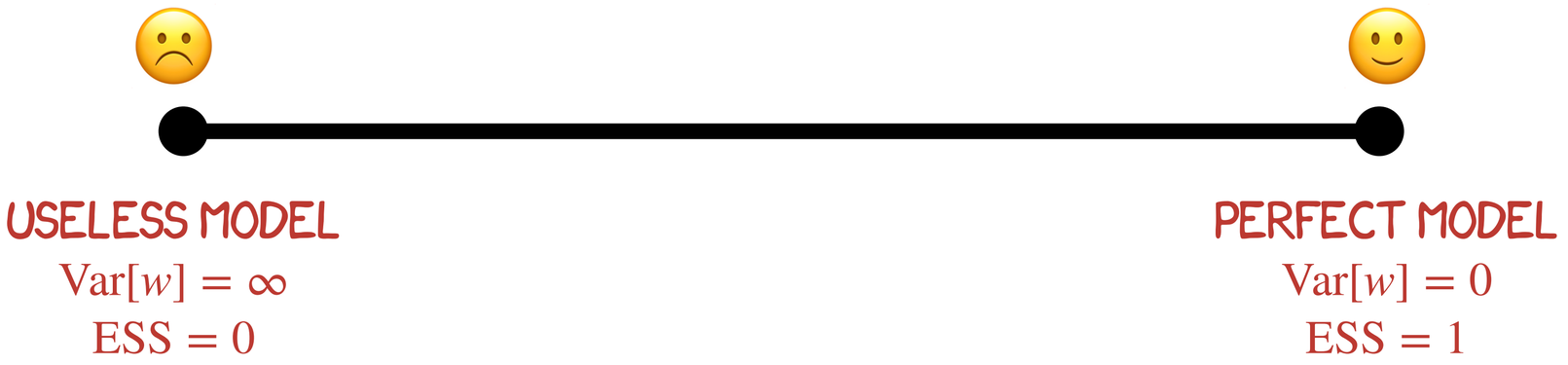
Note: *Efficiency* of bias correction depends on how close q and p are.

Measuring model quality

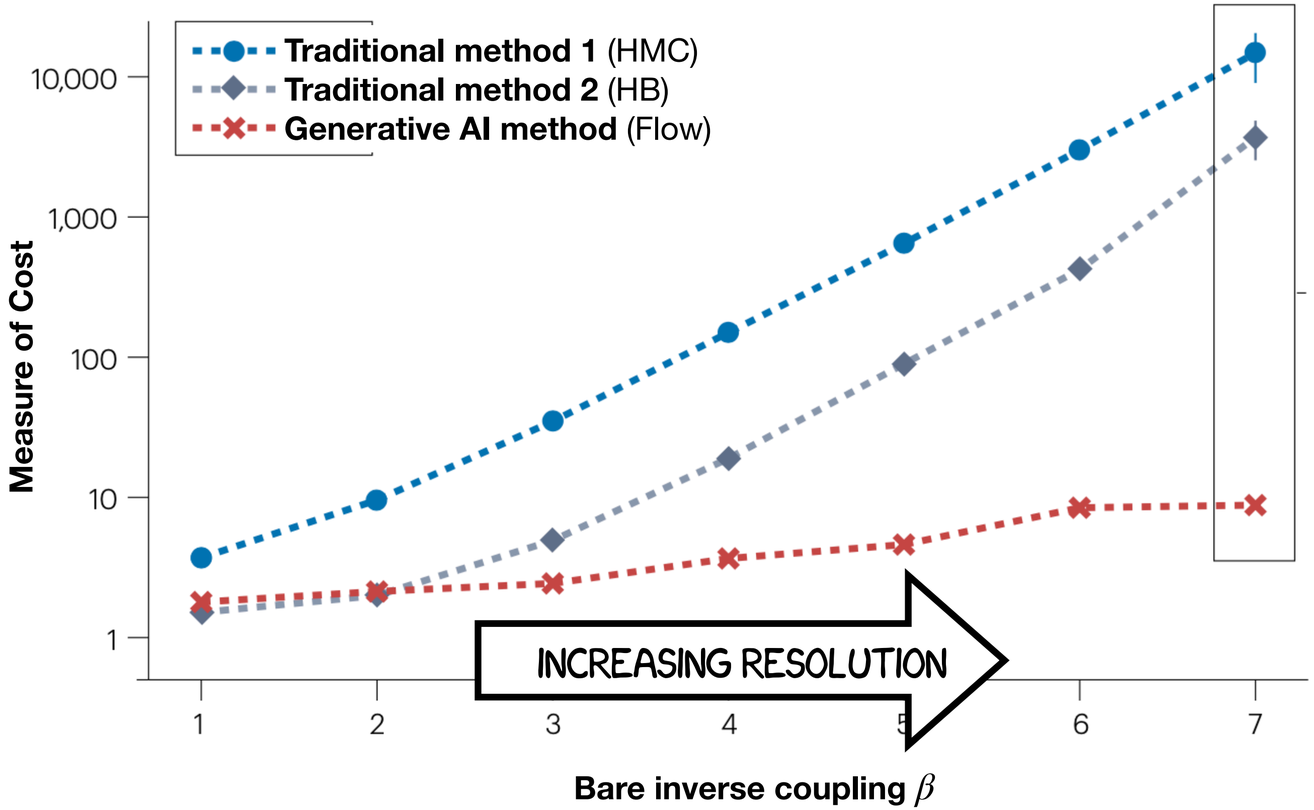
Variance of reweighting factors $\mathrm{Var}[w]$, where $w(U) := p(U)/q(U)$

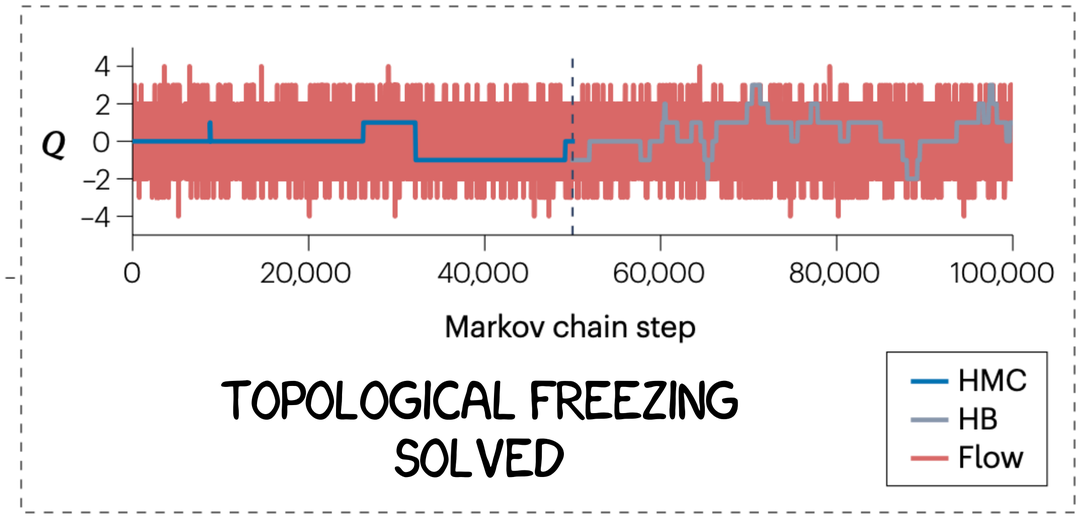
Effective sample size, $\mathrm{ESS} := \langle w \rangle_q^2 / \langle w^2 \rangle_q$

- "Fraction samples from q that are effective samples of p"



Preview: Very effective in some cases!





Cranmer, GK, Racanière, Rezende, Shanahan
Nature Reviews Physics 5 (2023) 526

Hands-on coding

github.com/gkanwar/flow-lectures

- Set up your env and follow along (if you want!)
- Implement HMC for ϕ^4 theory, observe slowing down
- Implement a toy flow-based sampler
- Explore this code further in the exercises