Introduction to normalizing flows for lattice field theory

Dan Hackett (MIT)

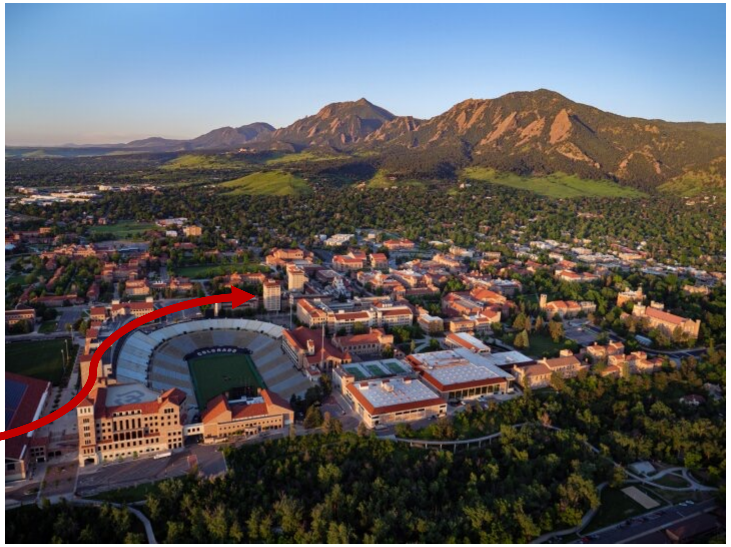
March 15, 2023

About Me

Dan from here

Undergrad: University of Virginia (UVA)

PhD: University of Colorado Boulder



Dan's office (former)

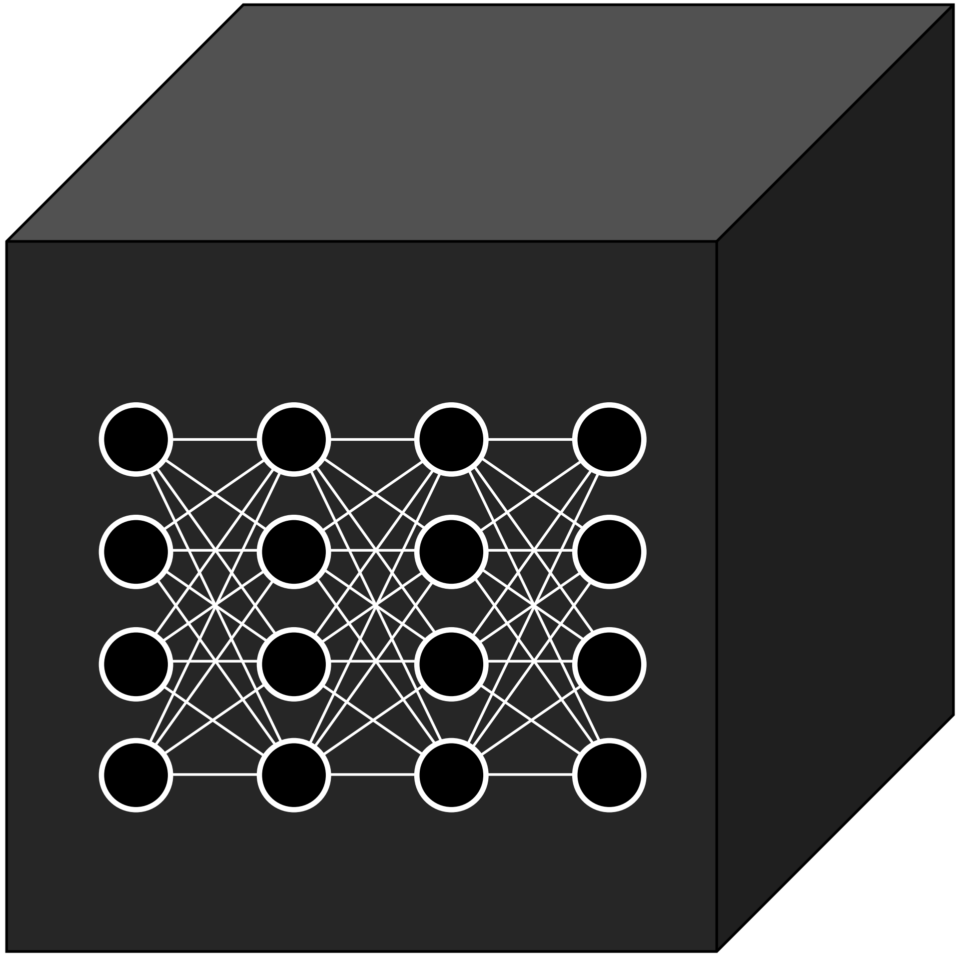
Postdoc (present): MIT

Dan's office

Machine learning?

Can we use black boxes
to do lattice field theory?

Yes, but must be careful!



Outline

- Lattice field theory
- Machine learning
- Generative models (for LQFT)
- Example: affine coupling flow (RealNVP)
- The road to QCD
- Symmetries and equivariance
- Closing thoughts

References

Based on [2101.08176]

Tutorial Jupyter notebook!

[.ipynb link]

Introduction to Normalizing Flows for Lattice Field Theory

Michael S. Albergo, Denis Boyda, Daniel C. Hackett, Gurtej Kanwar, Kyle Cranmer, Sébastien Racanière, Danilo Jimenez Rezende, Phiala E. Shanahan

This notebook tutorial demonstrates a method for sampling Boltzmann distributions of lattice field theories using a class of machine learning models known as normalizing flows. The ideas and approaches proposed in arXiv:1904.12072, arXiv:2002.02428, and arXiv:2003.06413 are reviewed and a concrete implementation of the framework is presented. We apply this framework to a lattice scalar field theory and to U(1) gauge theory, explicitly encoding gauge symmetries in the flow-based approach to the latter. This presentation is intended to be interactive and working with the attached Jupyter notebook is recommended.

See also: [GomalizingFlow.jl]

by Satoshi Terasaki, Akio Tomiya

[GitHub link]

GomalizingFlow.jl: A Julia package for Flow-based sampling algorithm for lattice field theory

Akio Tomiya, Satoshi Terasaki

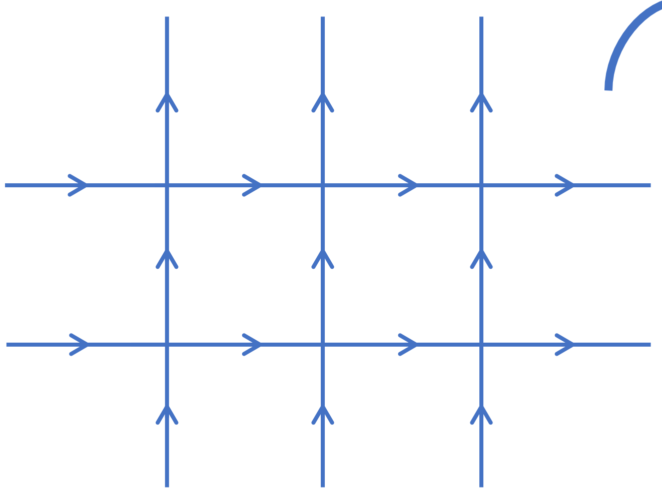
GomalizingFlow.jl: is a package to generate configurations for quantum field theory on the lattice using the flow based sampling algorithm in Julia programming language. This software serves two main purposes: to accelerate research of lattice QCD with machine learning with easy prototyping, and to provide an independent implementation to an existing public Jupyter notebook in Python/PyTorch. GomalizingFlow.jl implements, the flow based sampling algorithm, namely, RealNVP and Metropolis-Hastings test for two dimension and three dimensional scalar field, which can be switched by a parameter file. HMC for that theory also implemented for comparison. This package has Docker image, which reduces effort for environment construction. This code works both on CPU and NVIDIA GPU.

Lattice field theory

Lattice field theory (is just integration)

Want to compute:

$$\langle \mathcal{O} \rangle = \int d\phi \, \frac{e^{-S(\phi)}}{Z} \, \mathcal{O}(\phi)$$



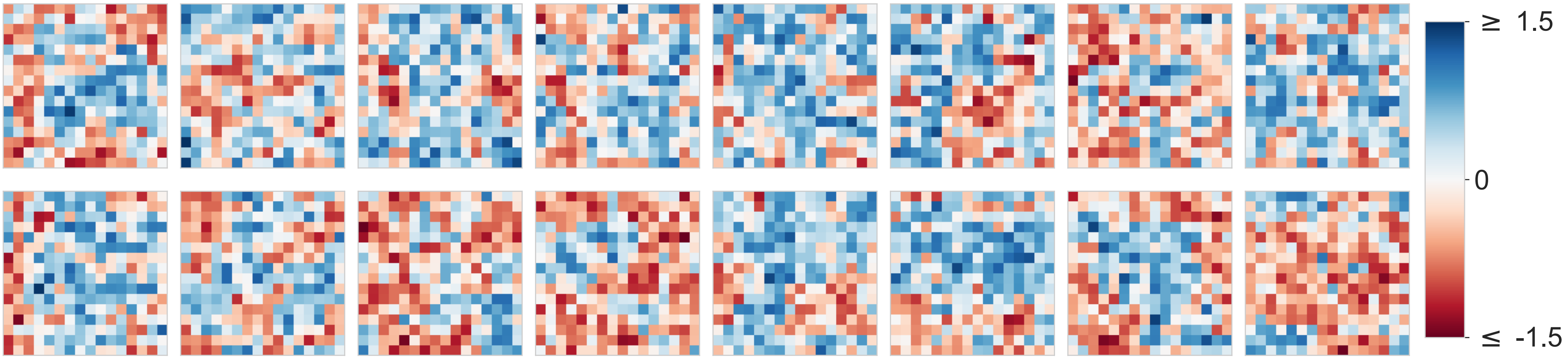
$$p(\phi) = \frac{e^{-S(\phi)}}{Z}$$

"Probability of ϕ"

Example: 2D ϕ^4 theory

$$S(\phi) = \sum_{\mathbf{x}} \left[\underbrace{\frac{1}{2} \sum_{\mu \in 0,1} [\phi(\mathbf{x}+\hat{\mu}) - \phi(\mathbf{x})]^2}_{\sim\, (\partial_\mu \phi)^2} + \underbrace{\frac{1}{2} m^2 \phi(\mathbf{x})^2 + \lambda \phi(\mathbf{x})^4}_{\text{Potential } V[\phi]} \right]$$

$\phi(\mathbf{x}) \sim$ monochromatic images



Lattice field theory with Monte Carlo

Monte Carlo estimate

$$\langle \mathcal{O} \rangle = \int d\phi \, p(\phi) \, \mathcal{O}(\phi) \approx \frac{1}{N} \sum_{i=1}^{N} \mathcal{O}(\phi_i)$$

$$\phi_i \sim p$$

Average over configurations *sampled* from p

Examples:

Magnetization: $\mathcal{O} = \bar{\phi} = \frac{1}{L^2} \sum_{\boldsymbol{x}} \phi(\boldsymbol{x})$

2-point correlator / propagator / Green's function:

$$\mathcal{O} = G(\mathbf{x}, \mathbf{y}) = \phi(\boldsymbol{x}) \, \phi(\boldsymbol{y})$$

Problem: MCMC autocorrelations

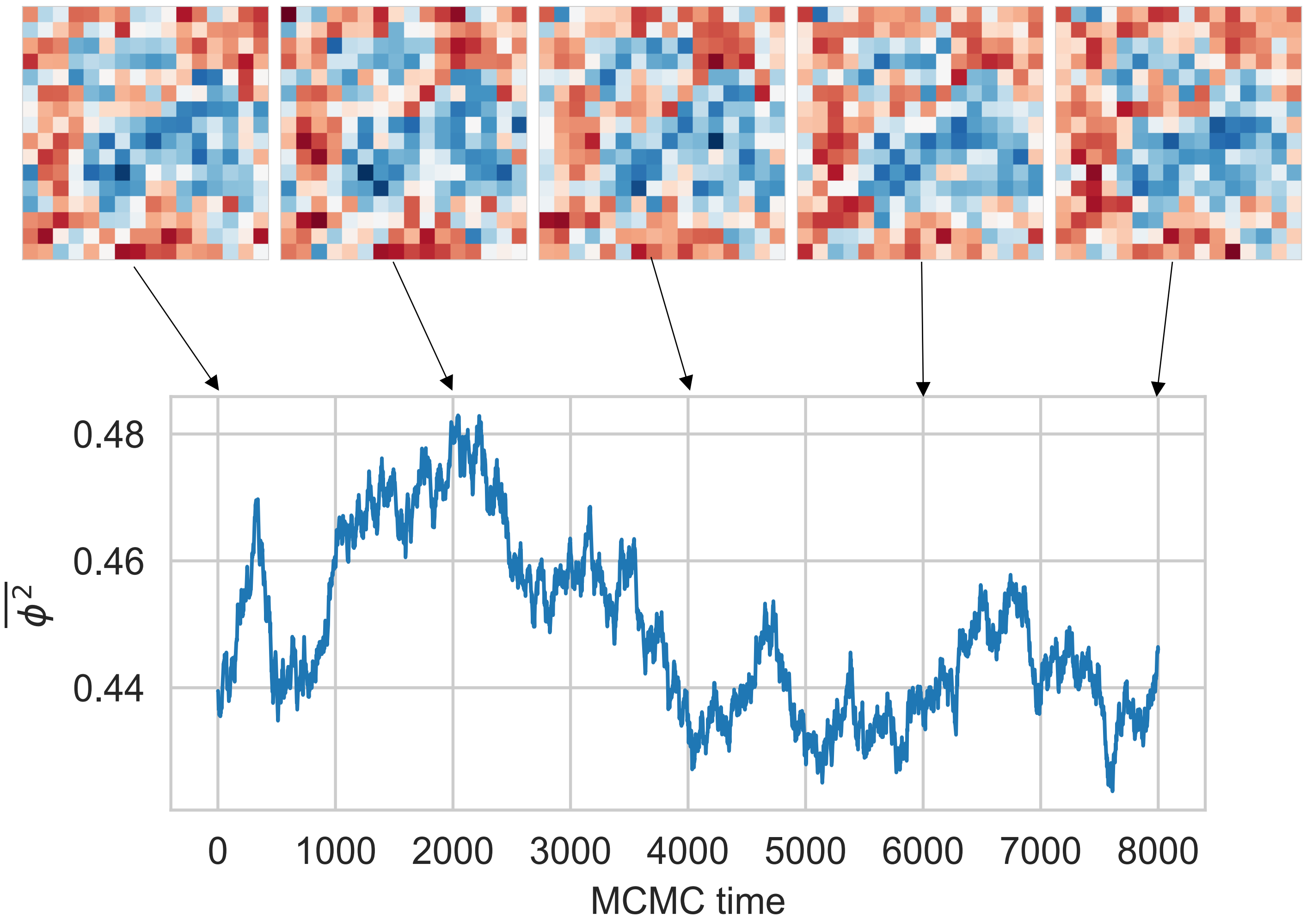
Markov chain Monte Carlo (MCMC) samples are not *independent*

More autocorrelations

↔ Less precise estimates

Better algorithms

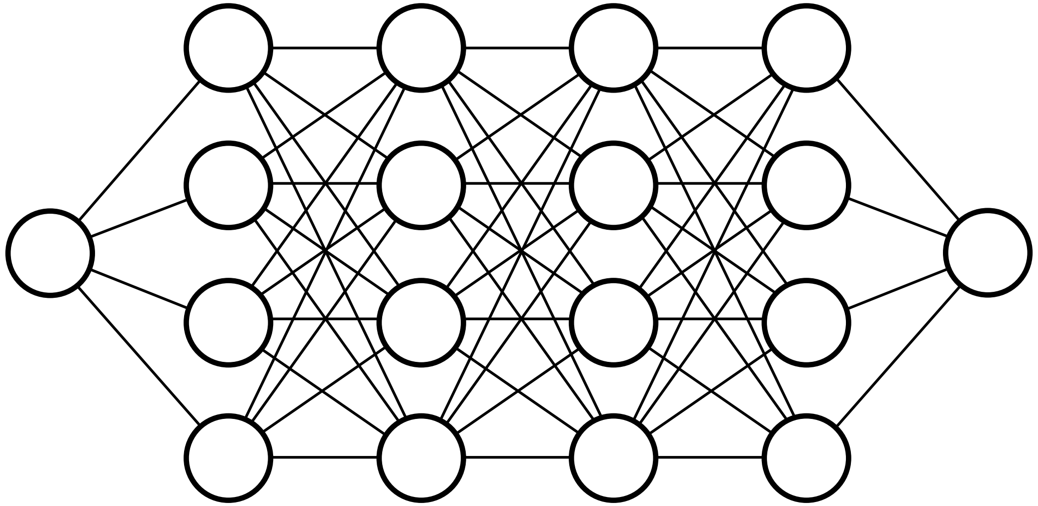
→ More precision with available computers

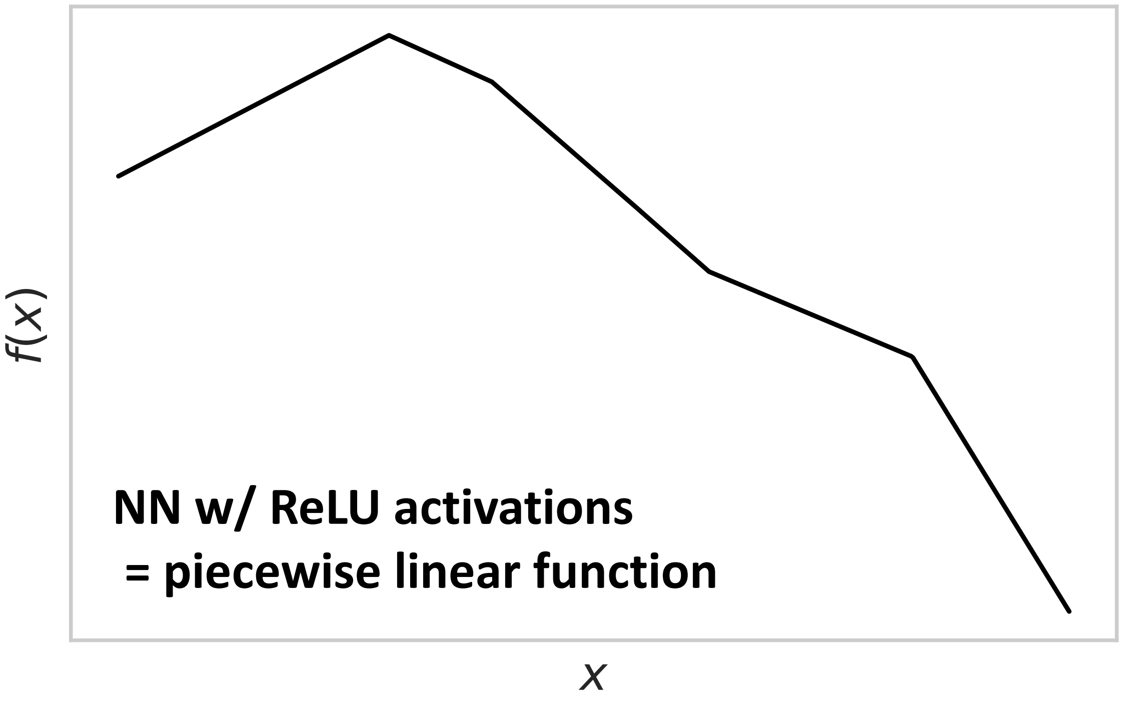


Machine learning

Neural networks are just functions

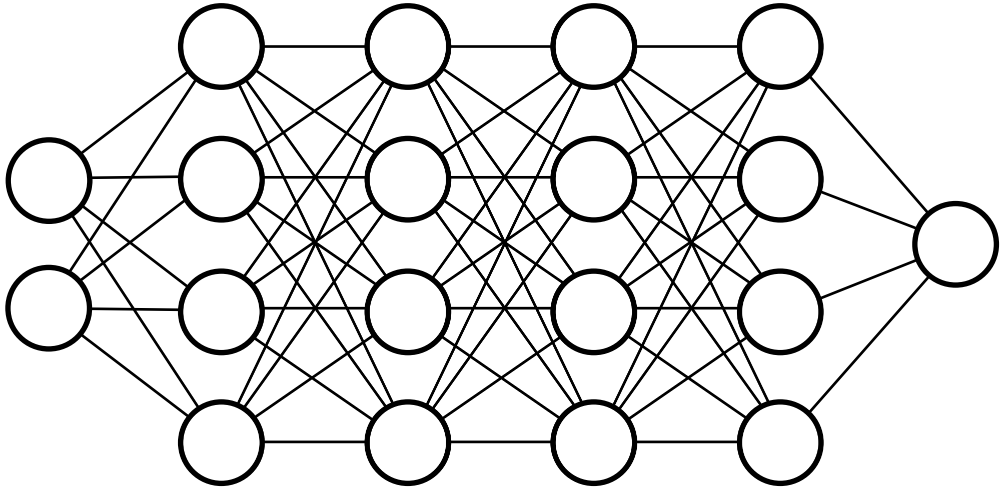
$$f: \mathbb{R} \to \mathbb{R}$$



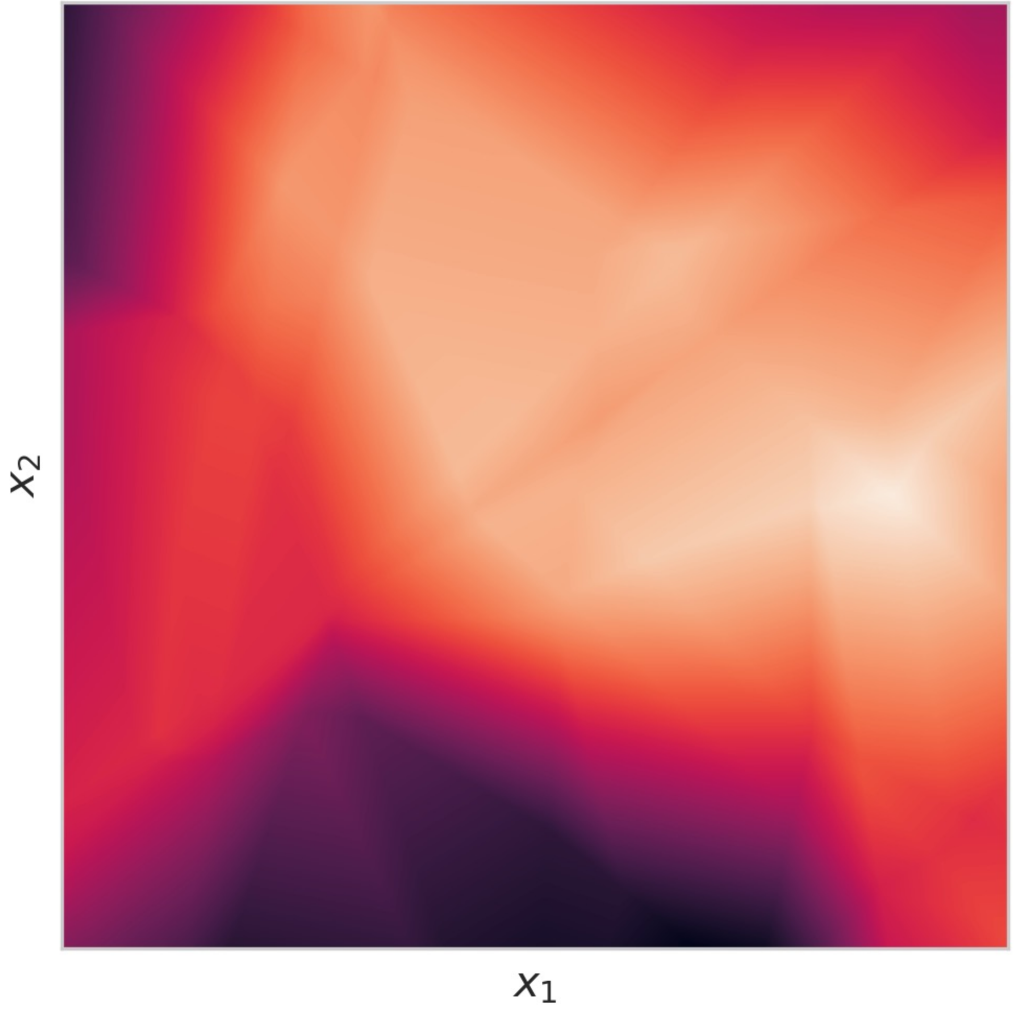


Neural networks are just functions

$f\colon\ \mathbb{R}^2 \to \mathbb{R}$



$f(x_1, x_2)$



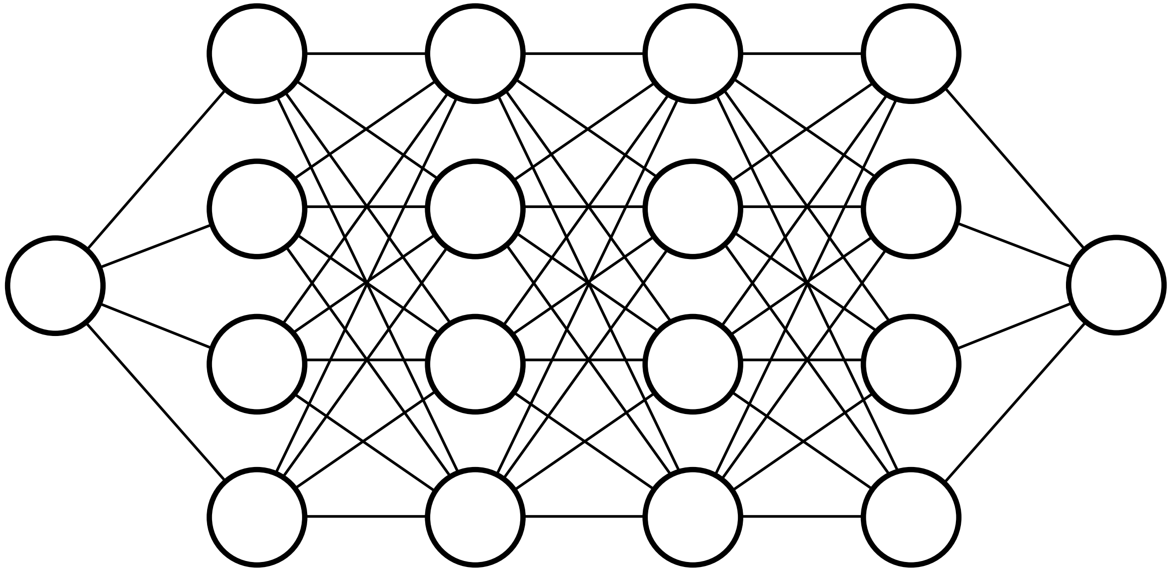
Generative models for LQFT

Mapping between distributions

Using a function to transform samples from a distribution gives samples from another distribution

x

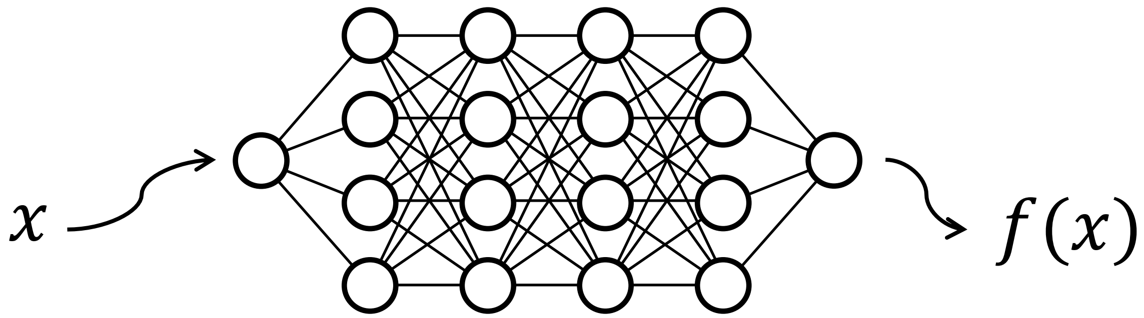
from "base" dist

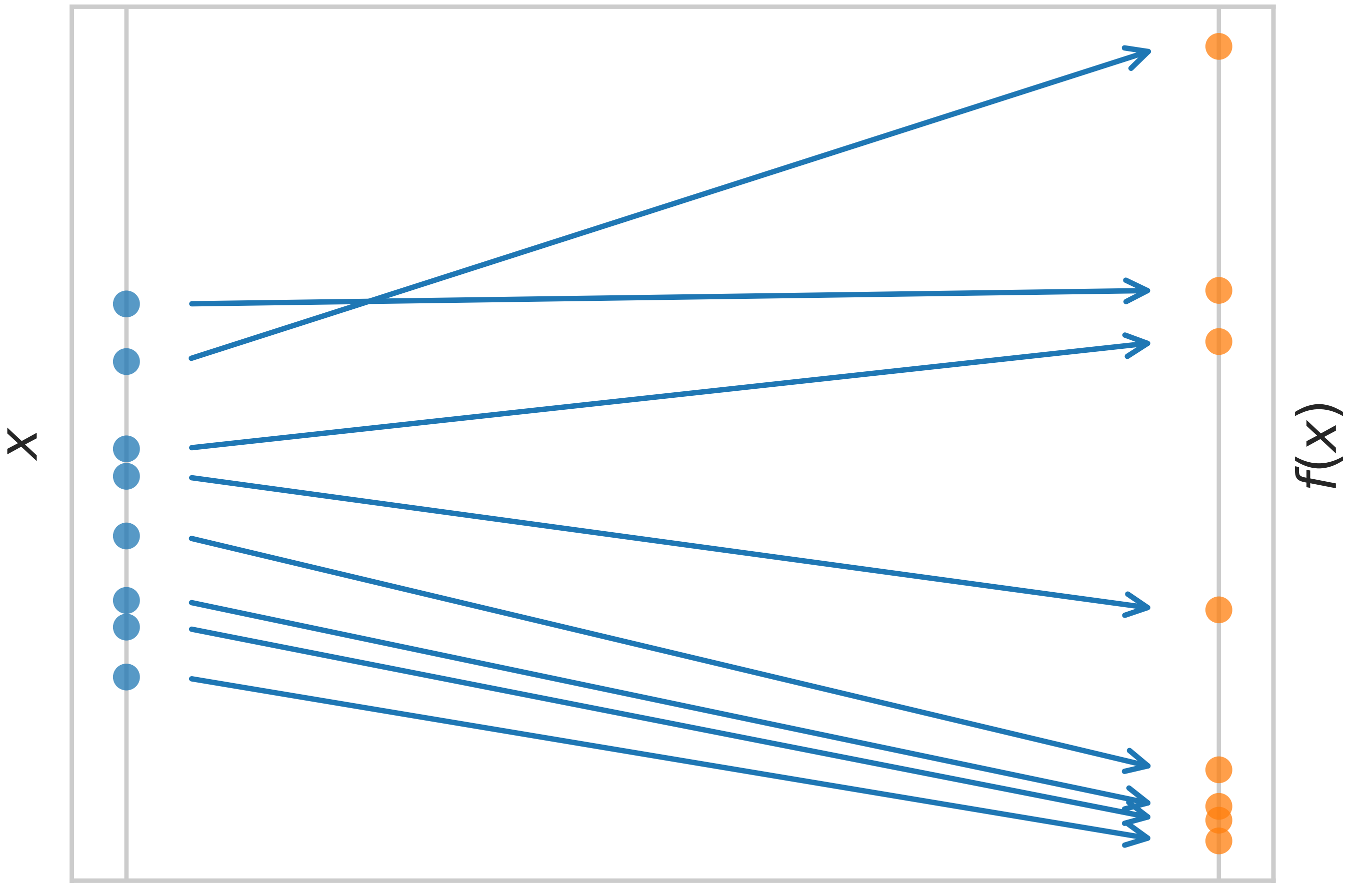


$f(x)$

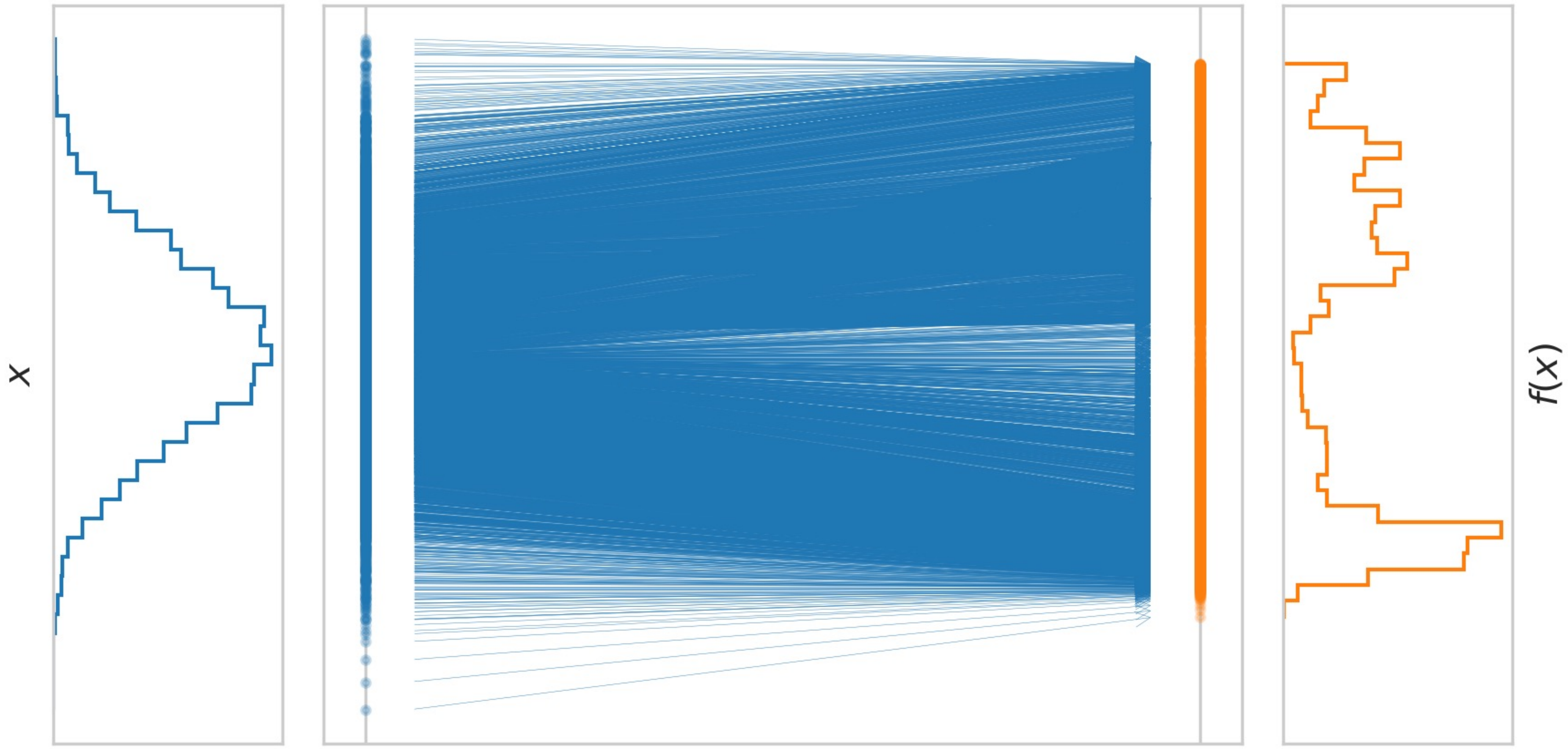
from "model" dist

Mapping between distributions



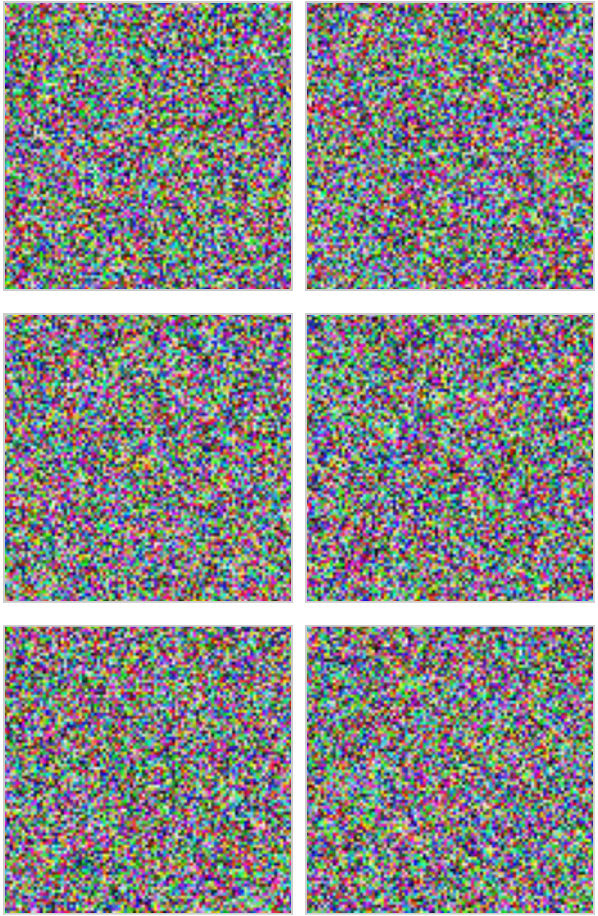


Mapping between distributions

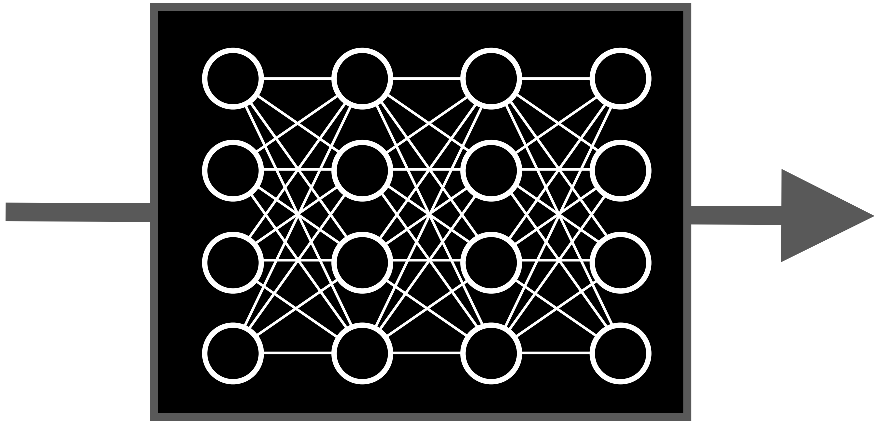


Generative models for image generation?

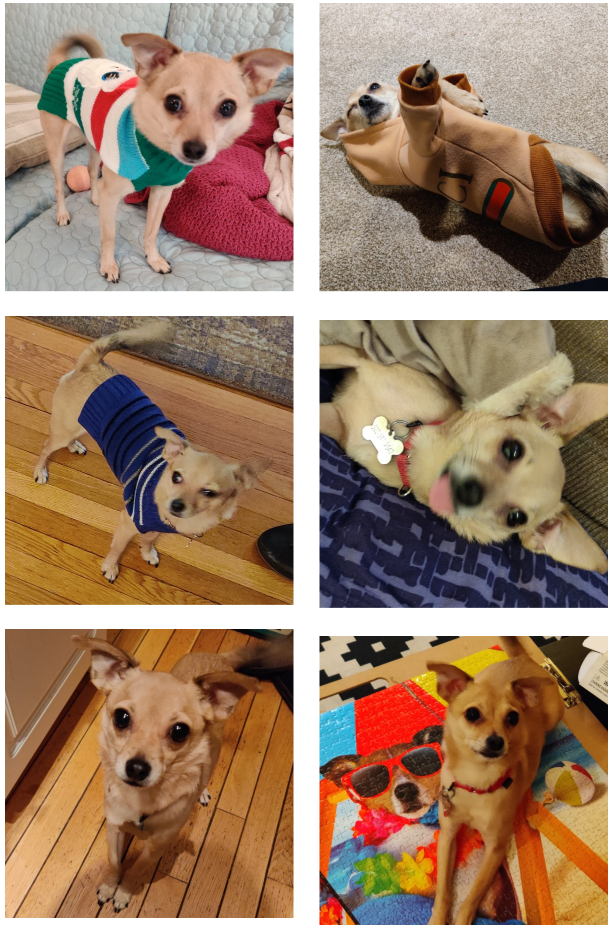
$z \sim$ Gaussian noise



NN transforms noise
$\phi = f(z)$



$\phi \sim$ distribution
of dog pictures

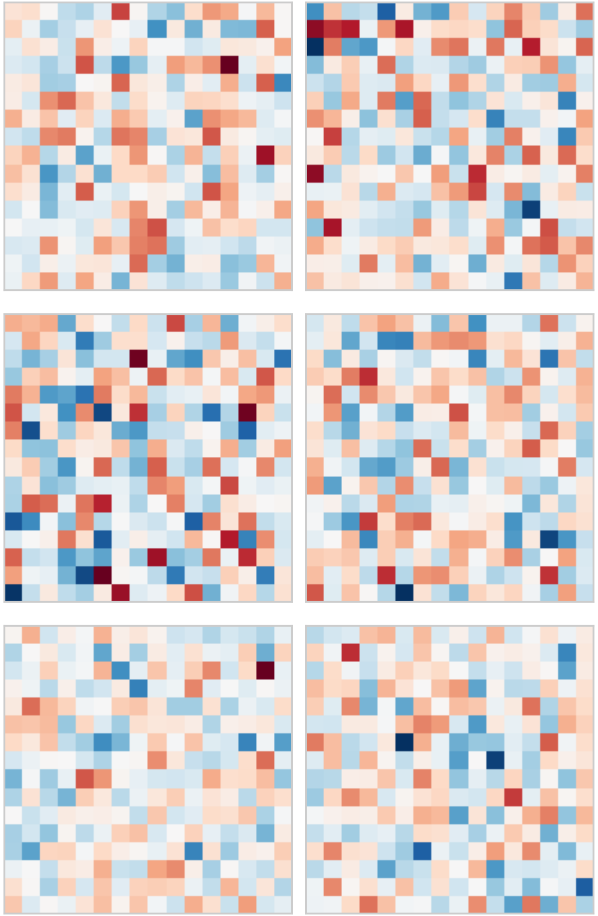


Generative models for LQFT sampling?

$\phi \sim$ model for distribution of lattice fields

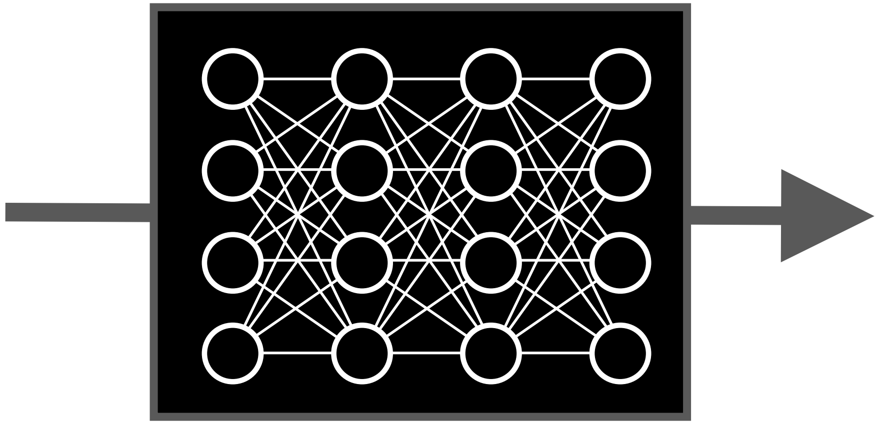
$$q \approx \frac{1}{Z} e^{-S(\phi)}$$

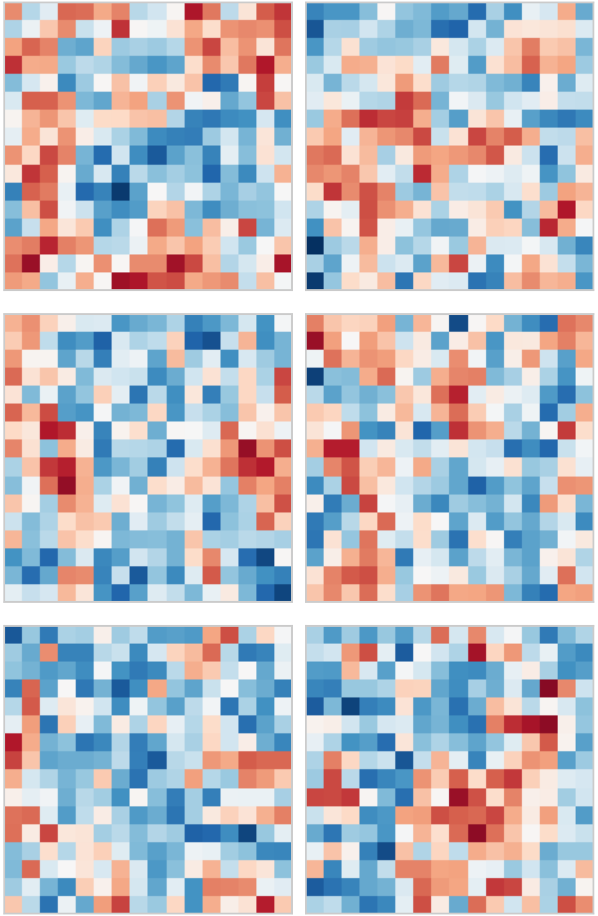
$z \sim$ Gaussian noise



NN transforms noise

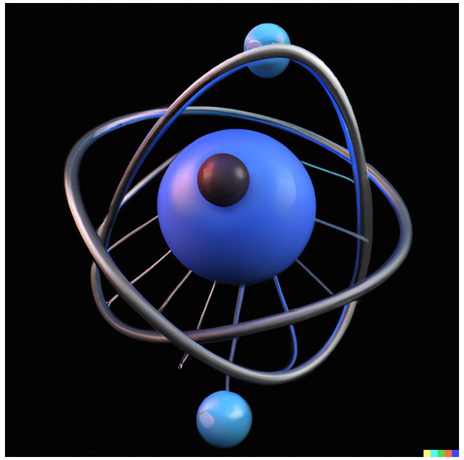
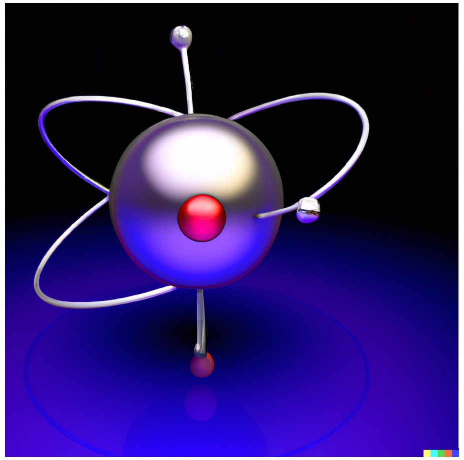
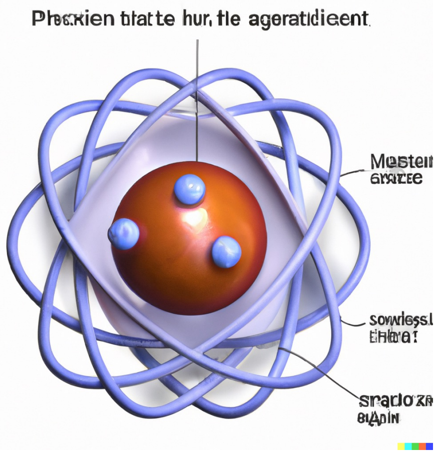
$$\phi = f(z)$$



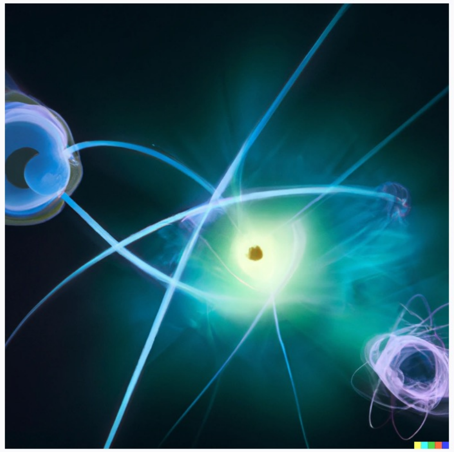
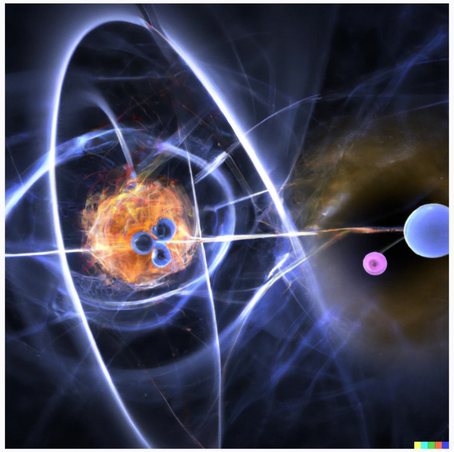
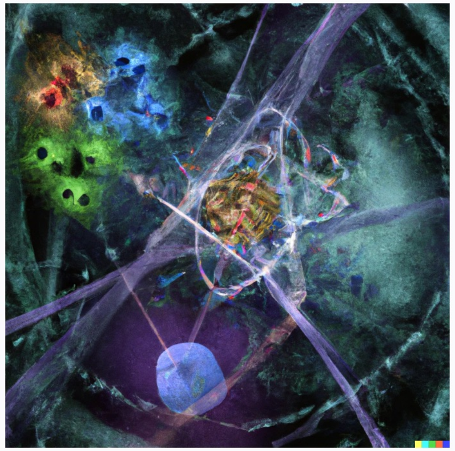


Physics according to DALL-E 2

What does a proton look like?

The interior of a proton

Cool!

…**but** we don't learn any physics

Better models don't help:
Stable Diffusion, Midjourney, etc still won't teach us physics

Requirements for exactness:

Must be able to compute model density q

$$\langle \mathcal{O} \rangle = \int d\phi\, q(\phi)\, \frac{p(\phi)}{q(\phi)}\, \mathcal{O}(\phi) \approx \frac{1}{N} \sum_{i=1}^{N} w(\phi_i)\, \mathcal{O}(\phi_i)$$

Reweighting factors

$$w(\phi) \equiv \frac{p(\phi)}{q(\phi)}$$

$$\phi_i \sim q$$

Reweighted average over configurations sampled from the “model” q

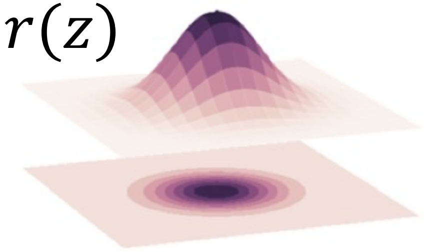
Normalizing flow models

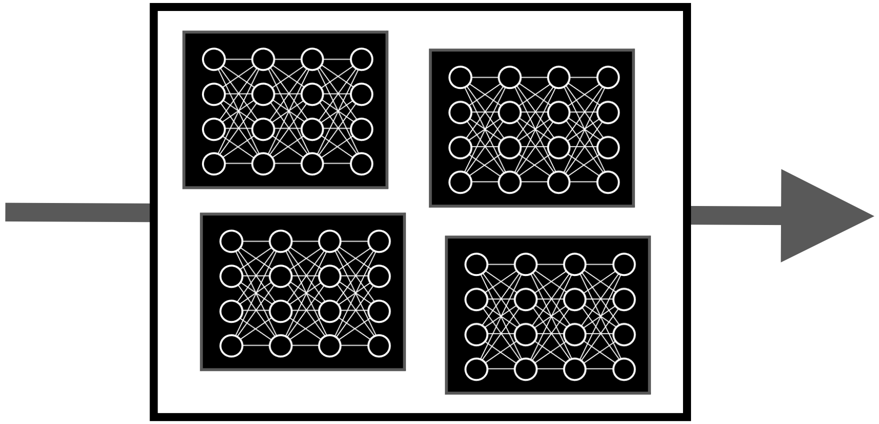
Simple **base distribution** r
- Tractable $r(z)$
- $r(z) > 0$
- Easy to sample

Learned **model distribution** q

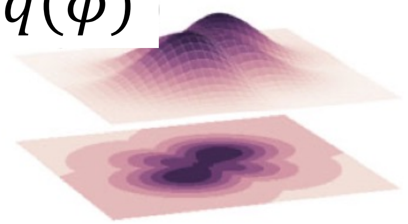
$$q(\phi) = \left|\det \frac{\partial \phi}{\partial z}\right|^{-1} r(z)$$

$r(z)$



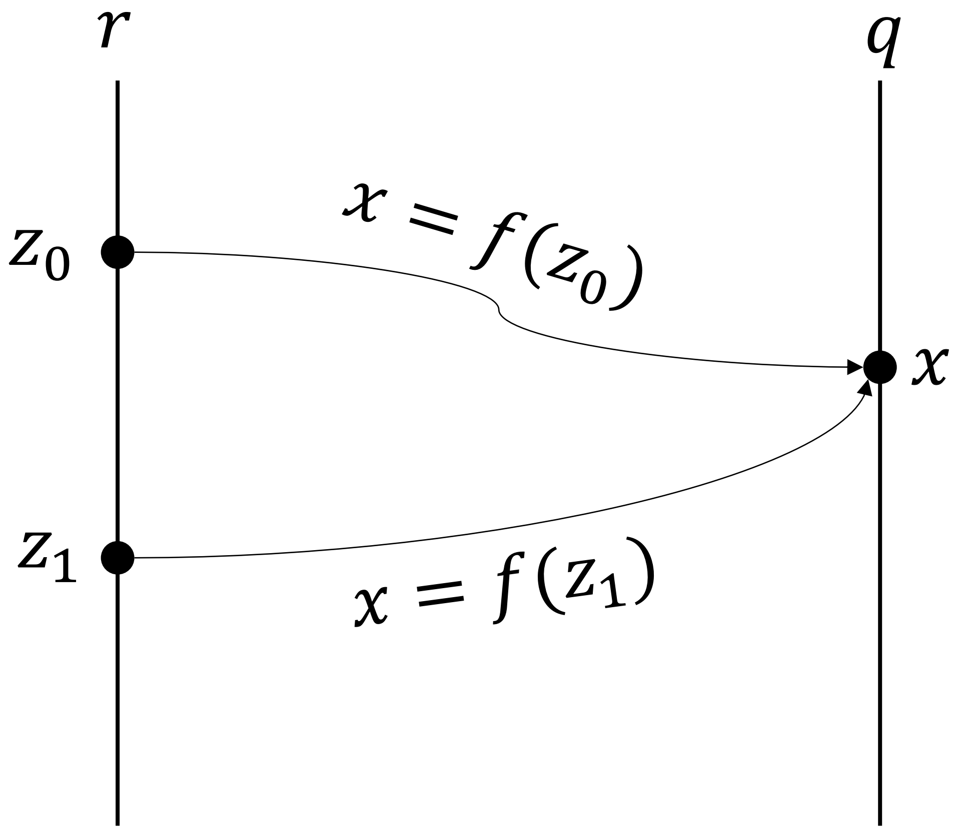


$q(\phi)$



"Flow" transformation
- Invertible
- Tractable Jacobian determinant
- *Parametrized* by NNs

Need for invertibility



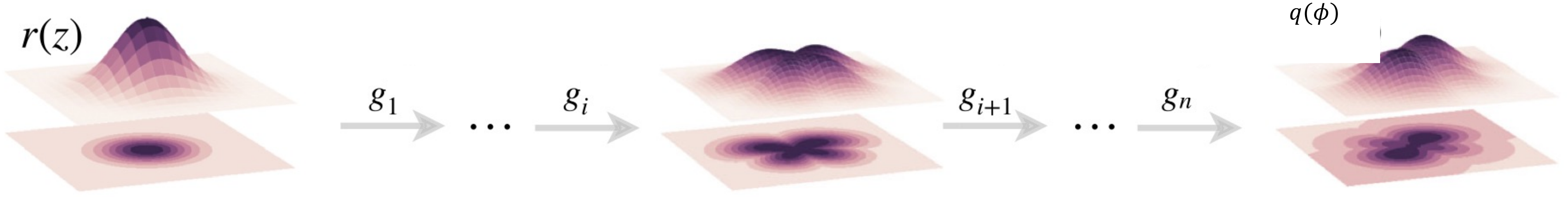
Non-invertible map:

To compute $q(x)$, must know all z such that $f(z) = x$

Intractable search problem!

Strategy: “coupling layers”

Composition: build flow by stacking many simple sub-transformations

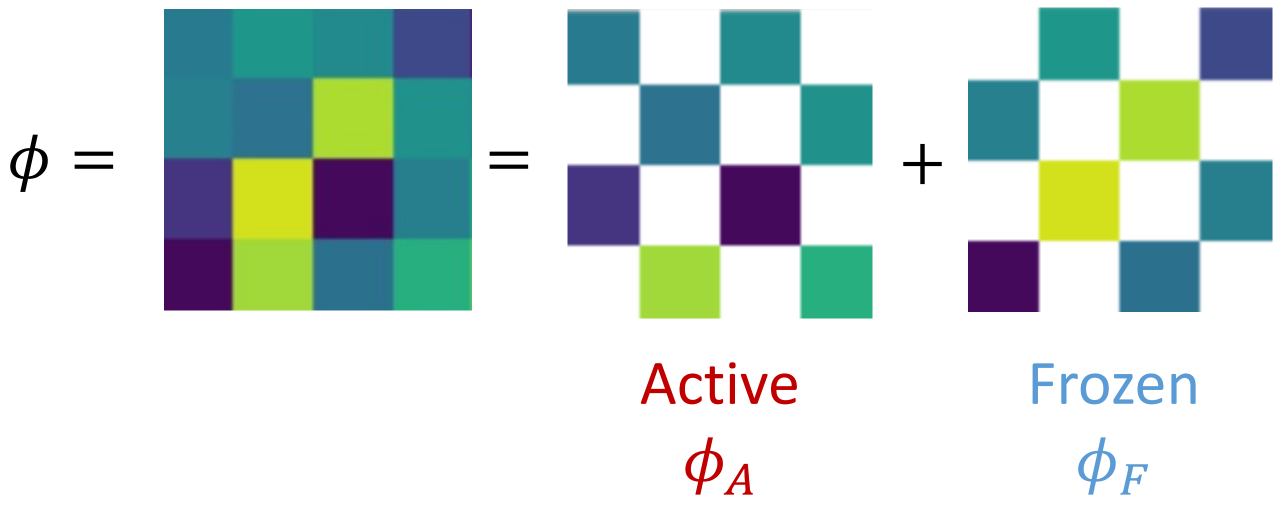


Variable partitioning:

- Freeze some variables
- Update others (active vars)

  …independently

  …conditioned on frozen vars



Example: affine coupling flow (RealNVP)

Affine coupling transformation (2 variables)

Draw ϕ_0 and ϕ_1 from 2d Gaussian

$$r(\phi_0, \phi_1) = \frac{1}{2\pi}\, e^{-\frac{1}{2}(\phi_0^2+\phi_1^2)}$$

Freeze ϕ_1, update ϕ_0

$$\phi_1' = \phi_1$$

$$\phi_0' = e^{s(\phi_1)}\phi_0 + t(\phi_1)$$

→ triangular Jacobian

$$J = \frac{\partial(\phi_0', \phi_1')}{\partial(\phi_0, \phi_1)} = \begin{bmatrix} e^{s(\phi_1)} & \partial\phi_0'/\partial\phi_1 \\ 0 & 1 \end{bmatrix} \quad \Rightarrow \quad |\det J| = e^{s(\phi_1)}$$

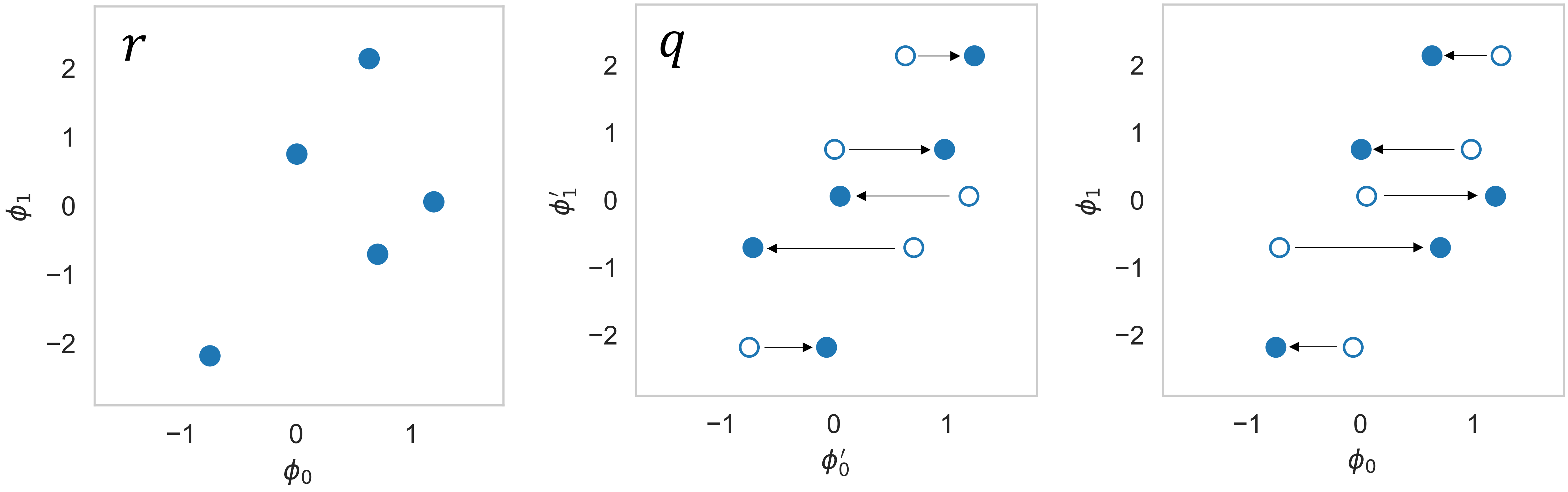
⇒ **Model density:** $q(\phi_0', \phi_1') = e^{-s(\phi_1)}\, r(\phi_0, \phi_1)$

Affine coupling transformation is invertible

Invertible by construction:

$$\phi_0' = e^{s(\phi_1)}\phi_0 + t(\phi_1)$$
$$\phi_1' = \phi_1$$

$$\phi_0 = e^{-s(\phi_1)}\left(\phi_0' - t(\phi_1)\right)$$
$$\phi_1 = \phi_1'$$



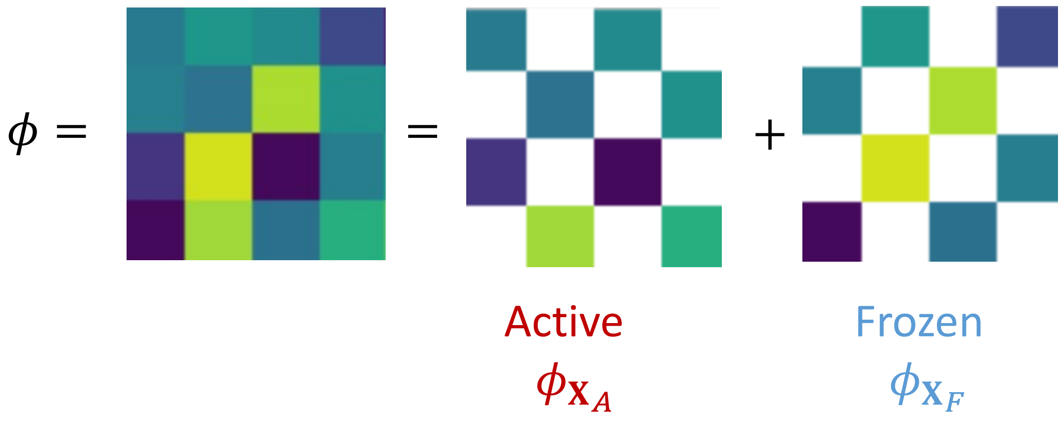
Affine coupling (scalar field theory)

For scalar field $\phi_{\mathbf{x}}$ on sites $\mathbf{x}$

Partition by site into $\mathbf{x}_A$, $\mathbf{x}_F$

Transform as $\phi'_{\mathbf{x}_F} = \phi_{\mathbf{x}_F}$

$$\phi'_{\mathbf{x}_A} = e^{s(\phi_F)_{\mathbf{x}_A}}\, \phi_{\mathbf{x}_A} + t(\phi_F)_{\mathbf{x}_A}$$



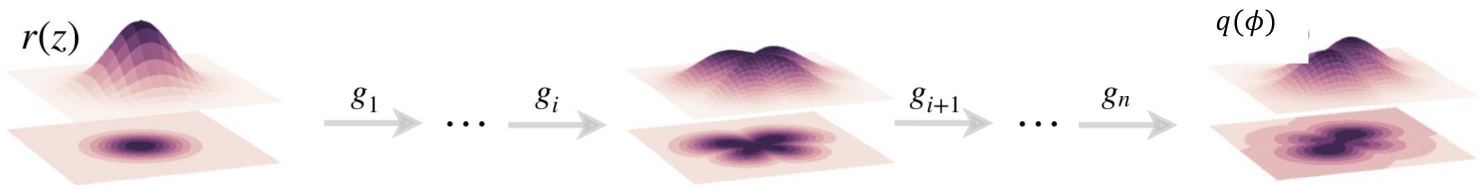
Triangular Jacobian:

$$J = \frac{\partial(\phi'_{\mathbf{x}_A}, \phi'_{\mathbf{x}_F})}{\partial(\phi_{\mathbf{y}_A}, \phi_{\mathbf{y}_F})} = \left[\begin{array}{c|c} \begin{matrix} \cdots & & \\ & e^{s(\phi_F)_{\mathbf{x}_A}} & \\ & & \cdots \end{matrix} & \text{(nonzero)} \\ \hline 0 & \begin{matrix} 1 & & \\ & \cdots & \\ & & 1 \end{matrix} \end{array}\right] \Rightarrow \;|\det J| = \prod_{\mathbf{x}_A} e^{s(\phi_F)_{\mathbf{x}_A}}$$

Machine-learned flows

Model: stack layers, alternating which variables are updated

Independent s, t in each layer



Model quality: need $q \approx p$, or reweighting is **very** noisy

→ Use NNs for s, t (expressive!)

→ Train NNs

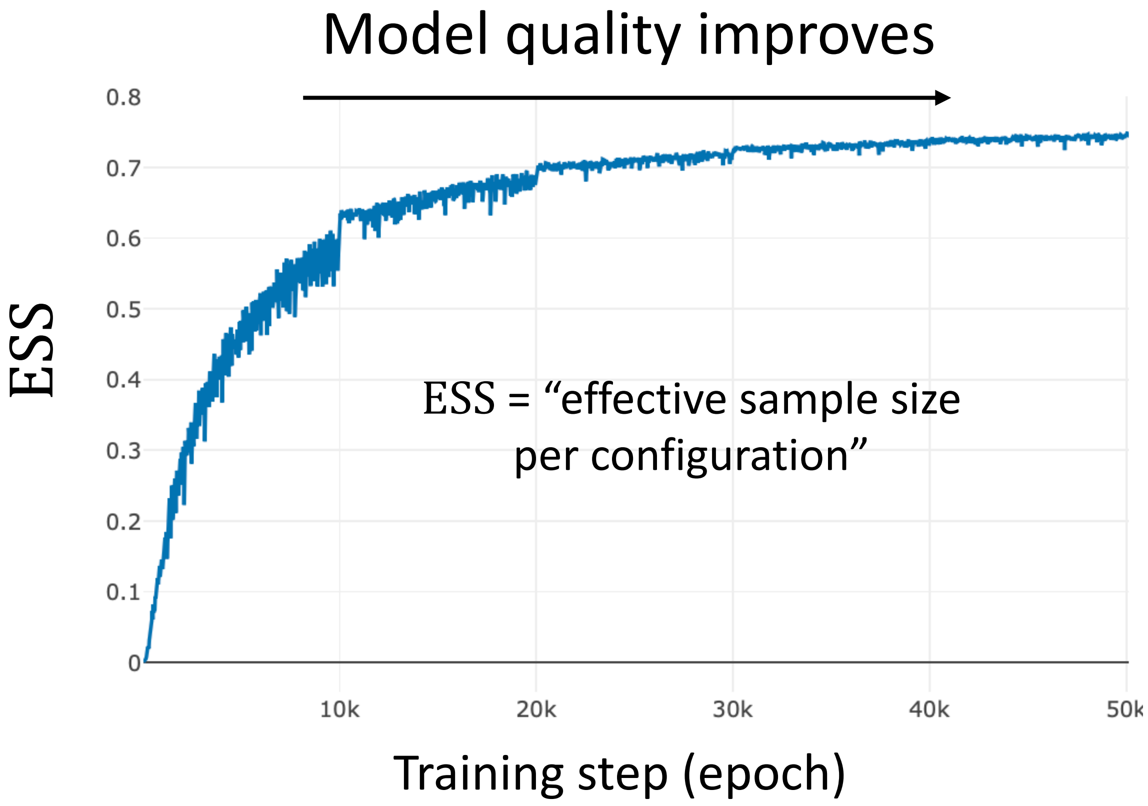
(Reverse KL) self-training

1. Draw a batch $\phi_i \sim q$
2. Compute $p(\phi_i)$, $q(\phi_i)$
3. Minimize “reverse” KL divergence

$$D_{KL}(q||p) = \int d\phi\; q(\phi) \log \frac{q(\phi)}{p(\phi)}$$

$$\approx \frac{1}{N} \sum_i \log \frac{q(\phi_i)}{p(\phi_i)}$$

Reverse KL self-training with Adam optimizer (batch size 16384):



Kullback-Leibler (KL) divergence

$D_{KL}(q||p) \geq 0$

$D_{KL}(q||p) = 0$ when $q = p$

The road to QCD

Progress to date (just my group)

Flows for LQFT (scalar field theories)

[Albergo, Kanwar, Shanahan 1904.12072]

[DH, Hsieh, Albergo, Boyda, JW Chen, KF Chen, Cranmer, Kanwar, Shananan 2107.00734]

Gauge-equivariant flows

U(1) [Kanwar, Albergo, Boyda, Cranmer, DH, Racanière, Rezende, Shanahan 2003.06413]

SU(N) [Kanwar, Albergo, Boyda, Cranmer, DH, Racanière, Rezende, Shanahan 2003.06413]

Flows for fermionic theories

Yukawa model [Albergo, Kanwar, Racanière, Rezende, Urban, Boyda, Cranmer, DH, Shanahan 2106.05934]

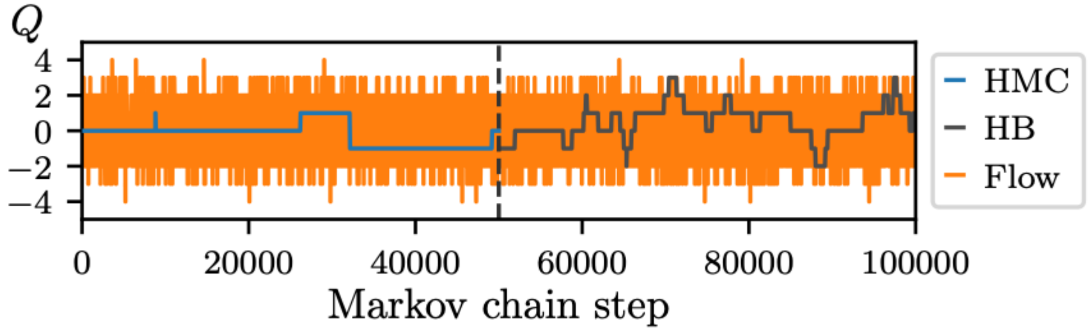
Schwinger model [Albergo, Boyda, Cranmer, DH, Kanwar, Racanière, Rezende, Romero-López, Shanahan, Urban 2202.11712]

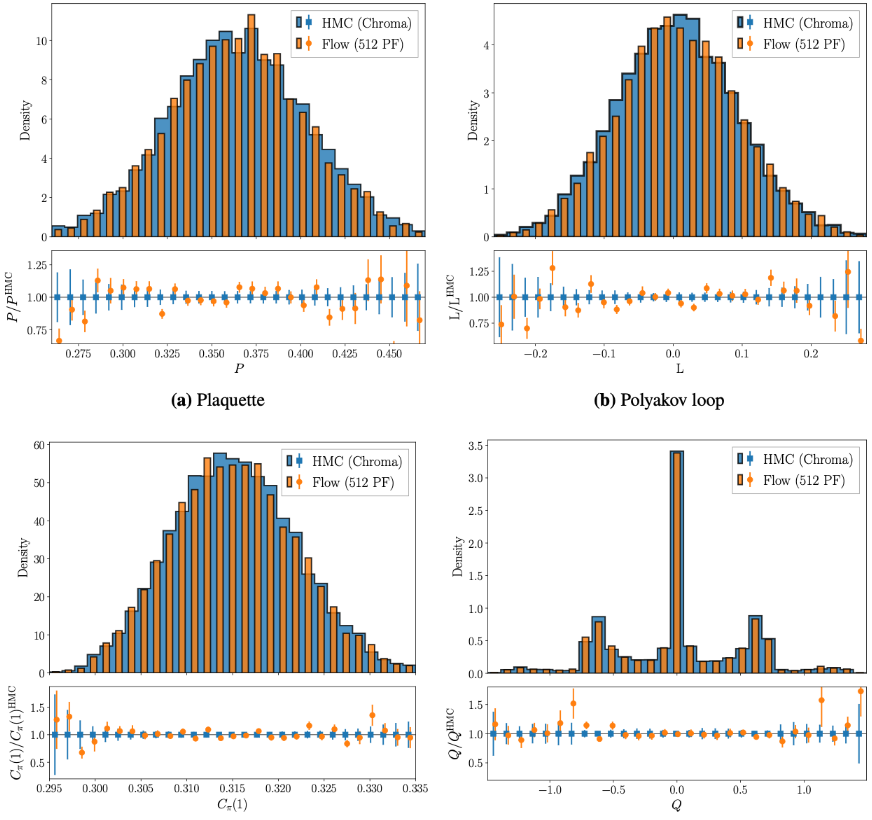
Schwinger with pseudofermions [Abbott, Albergo, Boyda, Cranmer, DH, Kanwar, Racanière, Rezende, Romero-López, Shanahan, Tian, Urban 2207.08945]

QCD!

Early demonstration [Abbott, Albergo, Botev, Boyda, Cranmer, DH, Kanwar, Matthews, Racanière, Razavi, Rezende, Romero-López, Shanahan, Urban 2208.03832]

Result: Improved sampling in (1+1)d U(1) gauge theory





(a) Plaquette

(b) Polyakov loop

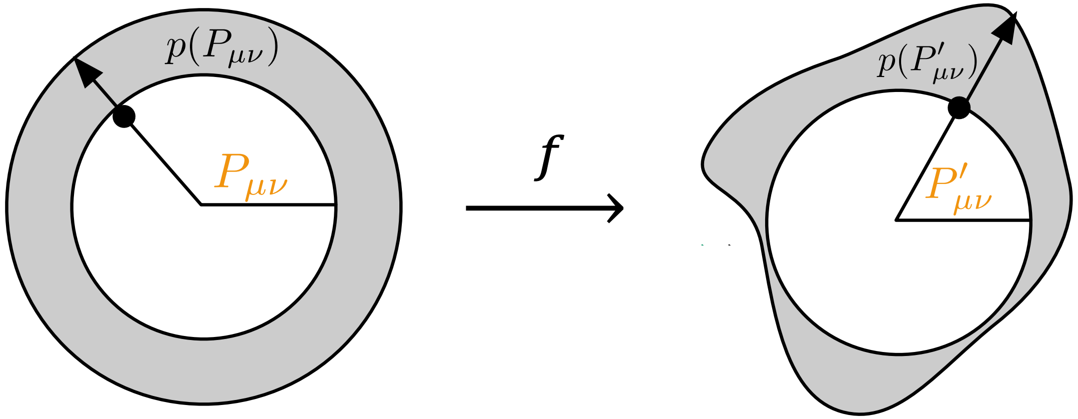
(c) Pion correlation function at $x_0 = 1$

(d) Topological charge at $t/a^2 = 4$

Flows on compact variables

$\mathrm{U}(1)$: phases (live on circles)

$\mathrm{SU}(N)$: Lie group manifold



Base distribution: Haar uniform

Flows on circles and torii

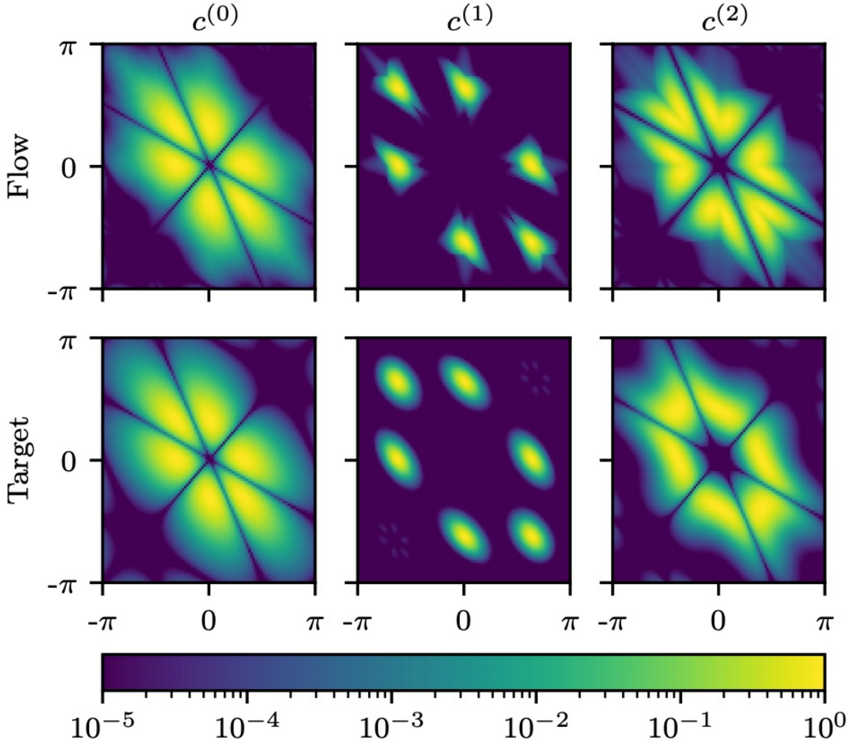
Non-compact projections, **splines**, …

[Rezende, Papamakarios, Racanière, Albergo, Kanwar, Shanahan, Cranmer 2002.02428]

Flows on SU(N) variables

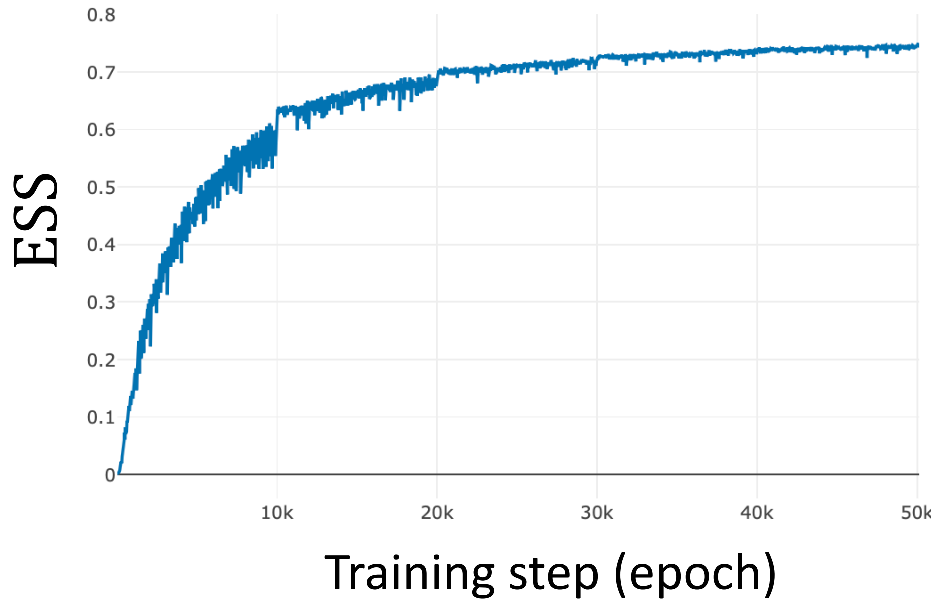
Flow eigenvalue spectrum

[Kanwar, Albergo, Boyda, Cranmer, DH, Racanière, Rezende, Shanahan 2003.06413]



Need physics-informed ML

- Training or increasing model size provides diminishing returns
- Need more qualitative algorithm improvements

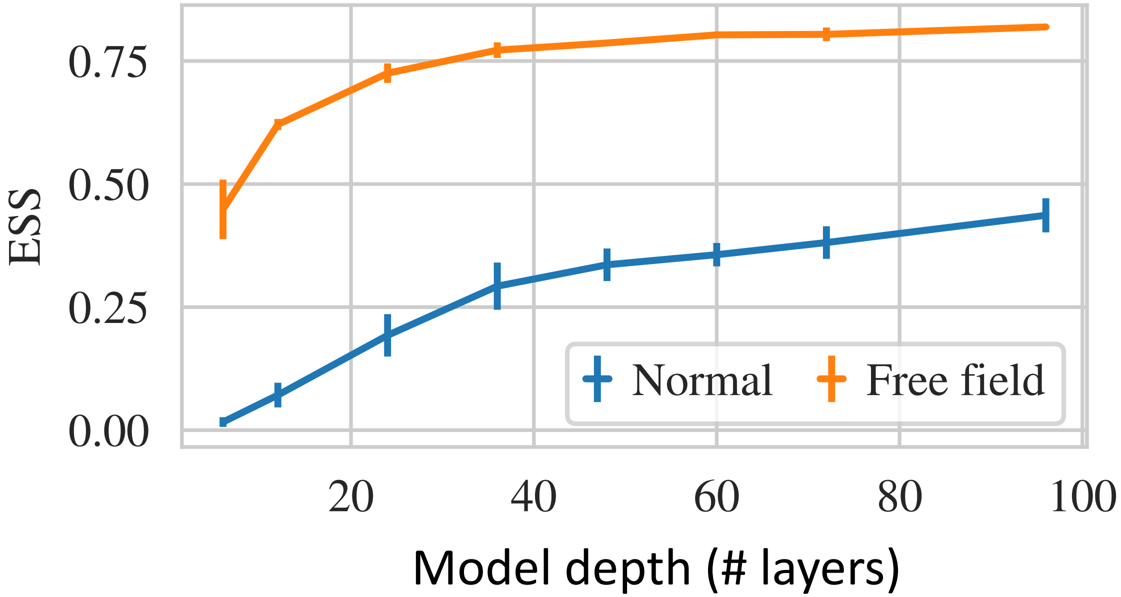


Ex: different base distributions for ϕ^4 models

Independent Gaussians on e/a site

Free field theory

Incorporating physics → better model!



Symmetries and equivariance

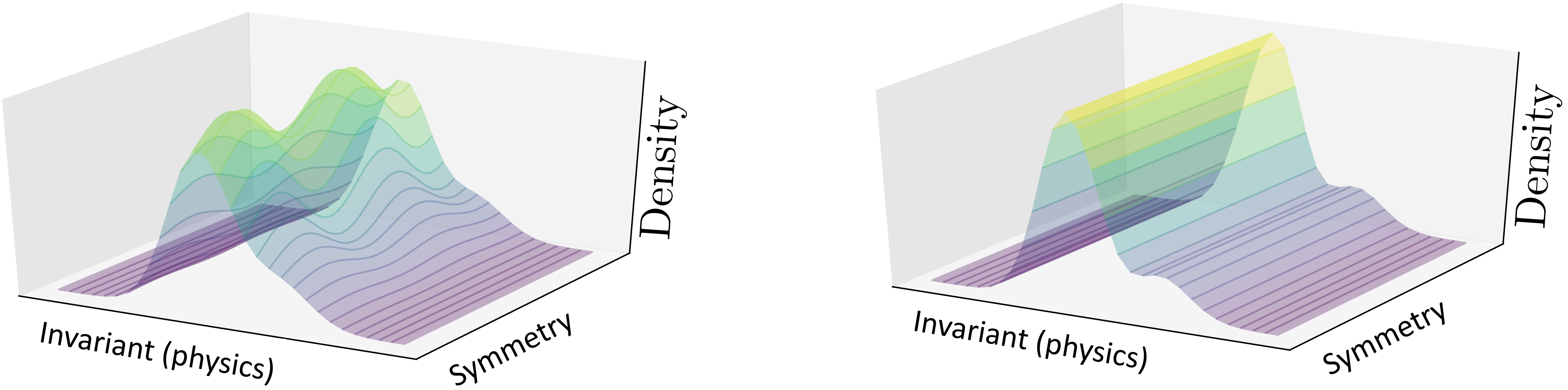
Symmetry

Invariance under $U \to g(U) \quad \Leftrightarrow \quad p(g(U)) = p(\phi)$

Invariant model:

1. Invariant base $\quad r(g(U)) = r(U)$
2. Equivariant flow $\quad f(g(U)) = g(f(U))$

   i.e. flow commutes with symmetry



Non-invariant model must learn approximate symmetry

Invariant model has exact symmetry built in

Discrete example: global Z_2 in ϕ^4 theory

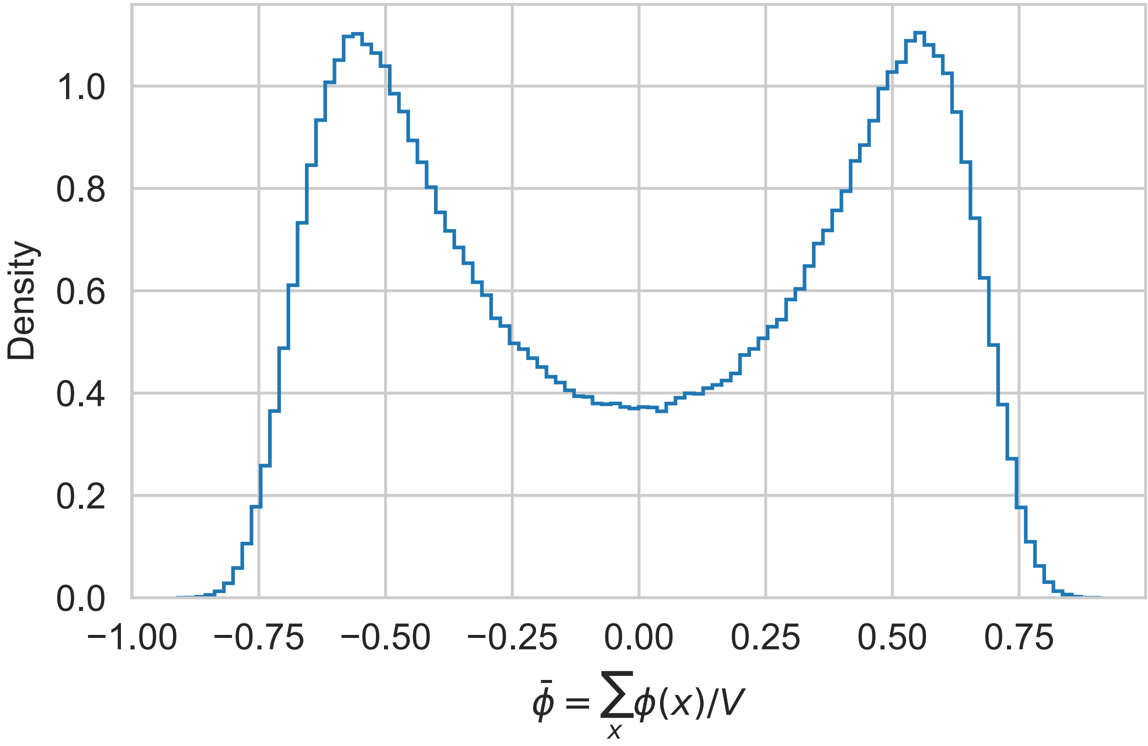
$$S(\phi) = \sum_{\mathbf{x}} \left[\frac{1}{2} \sum_{\mu \in 0,1} [\phi(\mathbf{x}+\hat{\mu}) - \phi(\mathbf{x})]^2 \;+\; \frac{1}{2} m^2 \phi(\mathbf{x})^2 + \lambda \phi(\mathbf{x})^4 \right]$$

Symmetric under $\phi \to -\phi$

$$S(\phi) = S(-\phi) \Leftrightarrow p(\phi) = p(-\phi)$$

How to model?

Magnetization in Z_2-broken phase



Example: global Z_2 in ϕ^4 theory

Ex1: Restrict NN architectures

$$\phi_A' = e^{s(\phi_F)}\,\phi_A + t(\phi_F)$$

but: $s(-\phi_F) = s(\phi_F)$

$t(-\phi_F) = -t(\phi_F)$

No bias in linear terms

Odd/even activations

[Nicoli et al. 2007.07115] [Del Debbio et al. 2105.12481]

Check:

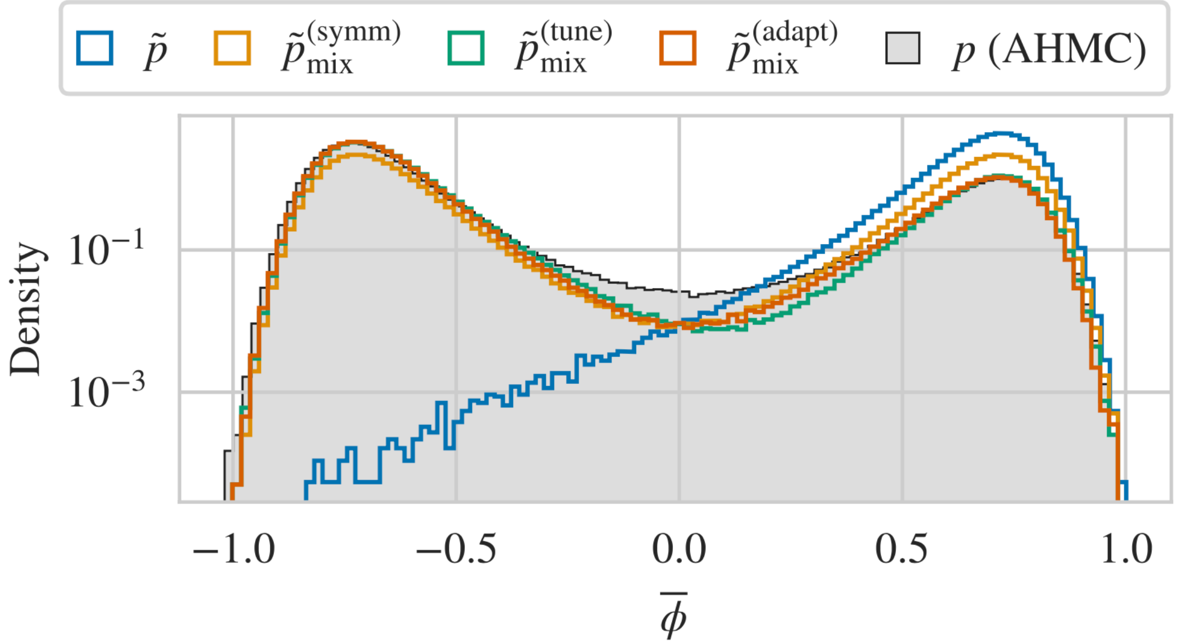
$$e^{s(-\phi_F)}\,(-\phi_A) + t(-\phi_F)$$
$$= -e^{s(\phi_F)}\phi_A - t(\phi_F)$$
$$= -\phi_A'$$

Ex2: Z_2-symmetrized self-mixture

Apply random sign after flow: $\phi = \pm f(z)$

$$\Rightarrow q_{\text{mix}}(\phi) = \frac{1}{2}\left(q(\phi) + q(-\phi)\right)$$

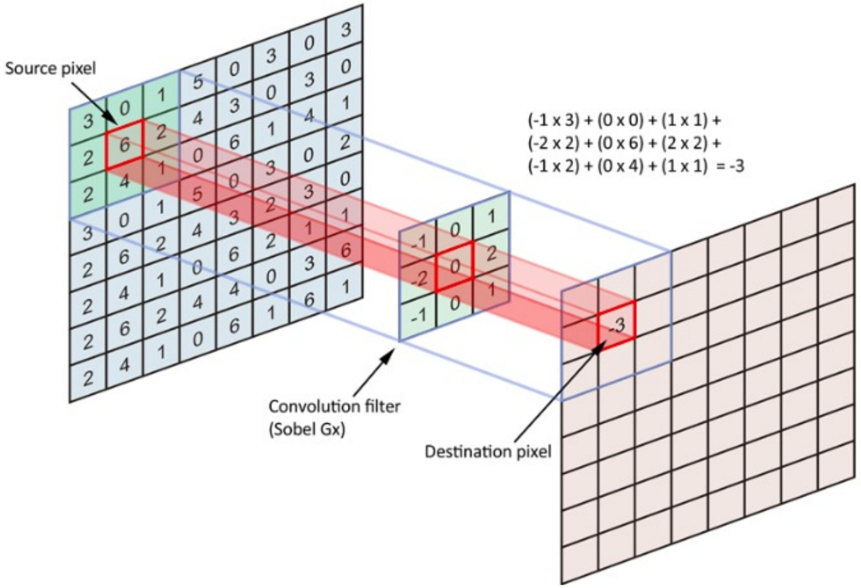
[DH, Hsieh, Albergo, Boyda, JW Chen, KF Chen, Cranmer, Kanwar, Shananan 2107.00734]



[Images from freecodecamp.org]

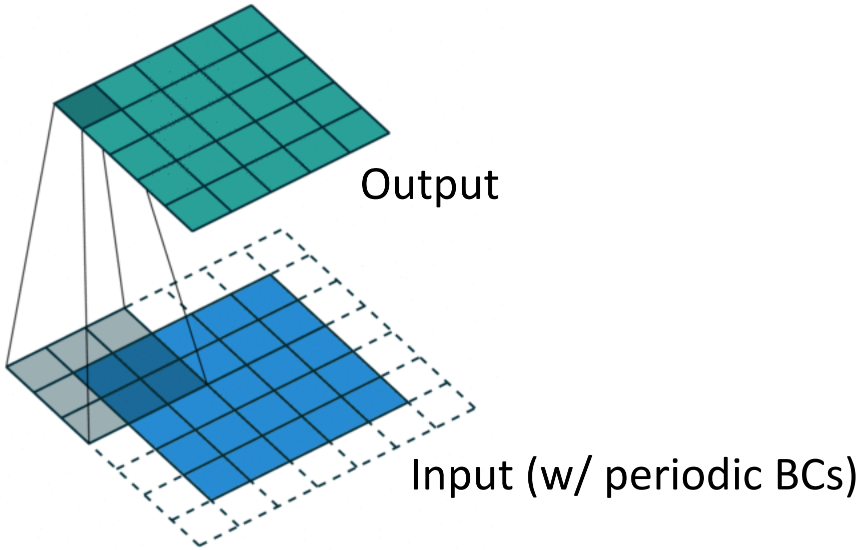
Example: translation symmetry

Translations commute with convolutions

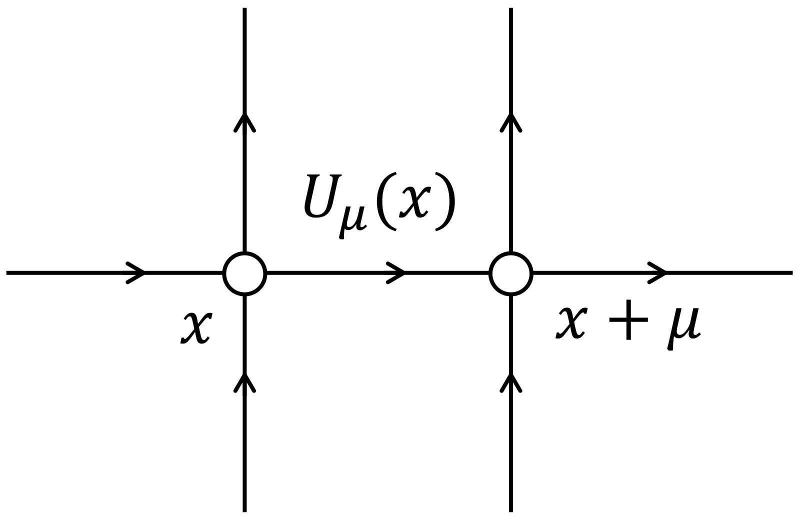


Invariant model:

1. Invariant base: same for each site
2. Equivariant flow: parametrize w/ CNNS

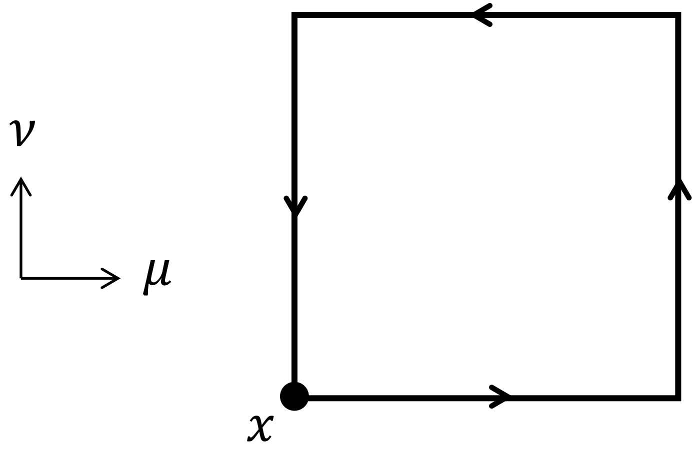


Lattice gauge symmetry



$$U_\mu(x) \to \Omega^\dagger(x+\mu)\, U_\mu(x)\, \Omega(x)$$

…with $U_\mu(x), \Omega(x) \in SU(3)$



$$P_{\mu\nu}(x) = U_\mu(\vec{x})\, U_\nu(\vec{x}+\hat{\mu})\, U_\mu^\dagger(\vec{x}+\hat{\nu})\, U_\nu^\dagger(\vec{x})$$
$$P_{\mu\nu}(x) \to \Omega^\dagger(x)\, P_{\mu\nu}(x)\, \Omega(x)$$
$$\mathrm{Tr}\, P_{\mu\nu}(x) \to \mathrm{Tr}\, P_{\mu\nu}(x)$$

Gauge equivariant flows

In general:

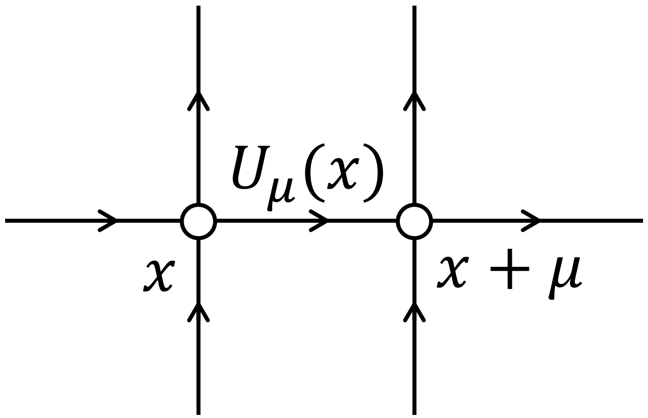
$$U_\mu(x) \to U'_\mu(x) = [f(U)]_\mu(x)$$

Want:

$$U'_\mu(x) \to \Omega^\dagger(x+\mu)\, U'_\mu(x)\, \Omega(x)$$

$\Rightarrow$ Must construct f such that:

$$\left[f\left(\Omega^\dagger U \Omega\right)\right]_\mu(x) = \Omega^\dagger(x+\mu)[f(U)]_\mu(x)\Omega(x)$$



$$U_\mu(x) \to \Omega^\dagger(x+\mu)\, U_\mu(x)\, \Omega(x)$$

No unique way!

See tutorial notebook for one example: [2101.08176](2101.08176)

Many gauge equivariant architectures developed already

- Gauge equivariant CNNs (parallel transport)

  [Favoni, Ipp, Müller, Schuh 2012.12901]

  [Abbott, Albergo, Boyda, Cranmer, DH, Kanwar, Racanière, Rezende, Romero-López, Shanahan, Tian, Urban 2207.08945]

  [Lehner, Wettig 2302.05419]

- Gradient flows w/ learned potentials

  [Bacchio, Kessel, Schaefer, Vaitl 2212.08469]

- Learned smearing

  [Tomiya, Nagai 2103.11965] "Gauge covariant neural network for 4 dimensional non-abelian gauge theory"

- Spectral flows

  U(1) [Kanwar, Albergo, Boyda, Cranmer, DH, Racanière, Rezende, Shanahan 2003.06413]

  SU(N) [Boyda, Kanwar, Racanière, Rezende, Albergo, Cranmer, Hackett, Shanahan 2008.05456]

Not competing options: can be combined!

Closing thoughts

Flows are a promising new approach to LQFT configuration generation

- *Provably exact* physics with ML
- Different properties from traditional sampling algorithms
- Early results suggest qualitative advantage in some cases

On the way to QCD, but work remains

- Model architectures and training schemes are not unique
- Lots of ML engineering required, especially to scale up

Many other sampling approaches to explore

- e.g. alternating flow and traditional updates → complementarity
- e.g. accelerating traditional algos with ML

Only beginning to explore what is possible!