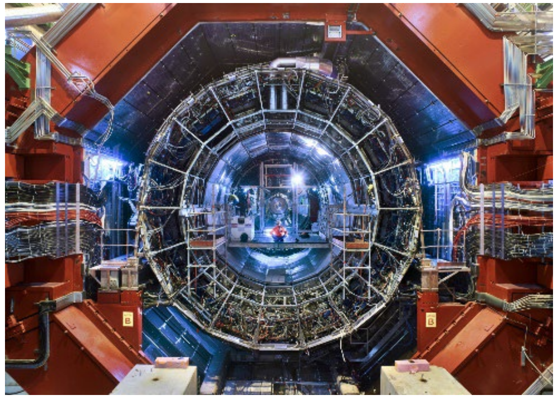
# Computing Topological Field Theories

## Lecture 2: Machine Learning for Lattice Field Theories

**Lena Funcke**

School “Recent Advances in Fundamental Physics”, Tbilisi

# Overview: computing topological field theories

| Numerical computations | Theoretical models | Experiments |
|---|---|---|
| **Lecture 1: Monte Carlo method**<br>High-precision lattice computations<br>Computational issues | **Standard Model**<br>“Real world”<br>Quarks, gluons, Higgs… | **Observables**<br>Spectrum, free energy, entropy, pressure… |
| **Lecture 2: Machine learning**<br>Efficient sampling, thermodynamic observables… | **1+1D $\phi^4$ theory**<br>Higgs toy model<br>Symmetry breaking… | **LHC, cosmology, …**<br>Heavy-ion collisions,<br>Early-universe physics… |
| **Lecture 3: Tensor networks & quantum computing**<br>Topological θ-terms, chemical potentials… | **1+1D Schwinger model**<br>QCD toy model<br>θ-term, confinement… |  |

(Image credit: ALICE Collaboration / CERN)

# Reminder: MCMC is hungry & challenging

**Computational costs of lattice field theory**

Supercomputer usage for different fields (INCITE 2019)

→ **Lattice QCD: ~ 40%**

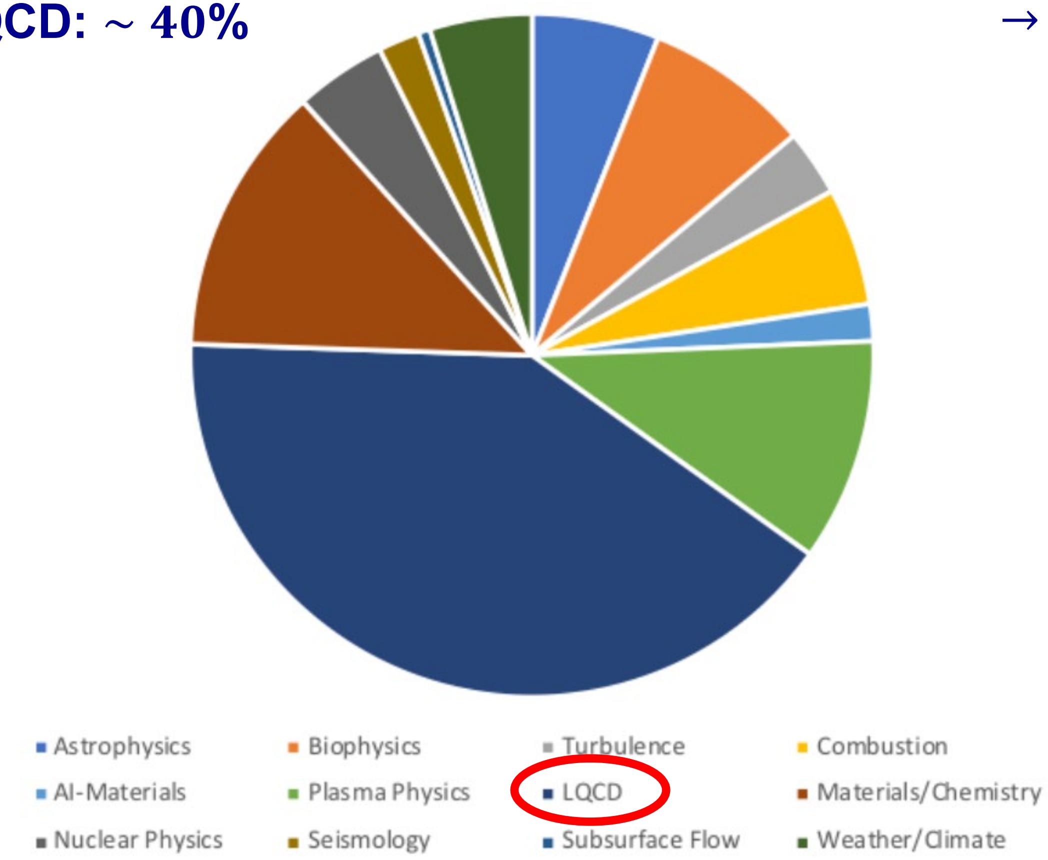
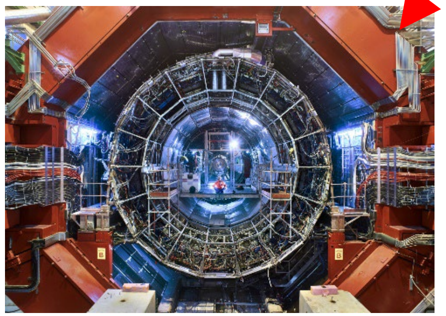



Figure credit: Jack Wells, Kate Clark

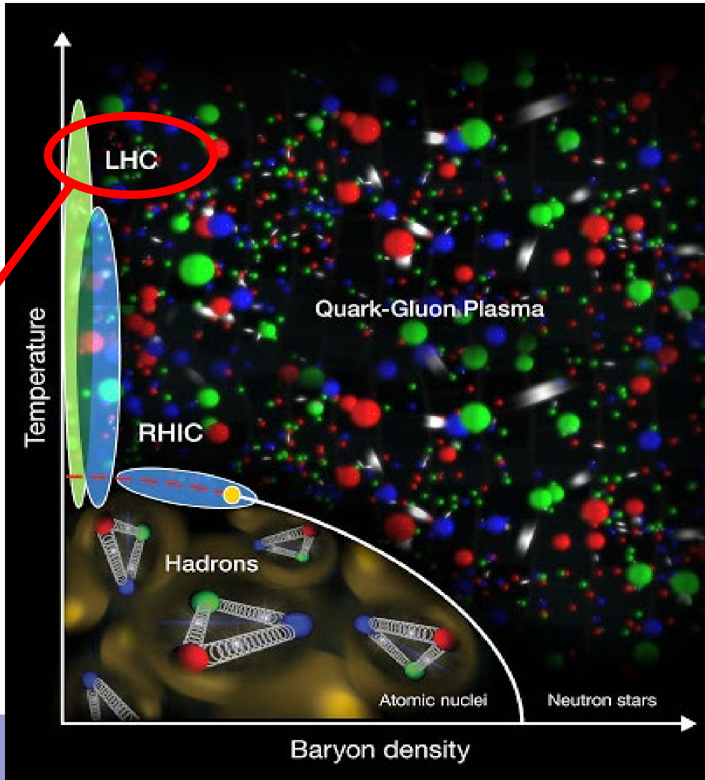
**Computational challenges of lattice field theory**

No direct computation of thermodynamic observables, …

→ **Machine learning (lecture today)**



(ALICE Experiment / CERN)



# Reminder: MCMC is hungry & challenging

**Computational costs of lattice field theory**

Supercomputer usage for different fields (INCITE 2019)

→ **Lattice QCD: ~ $40\%$**

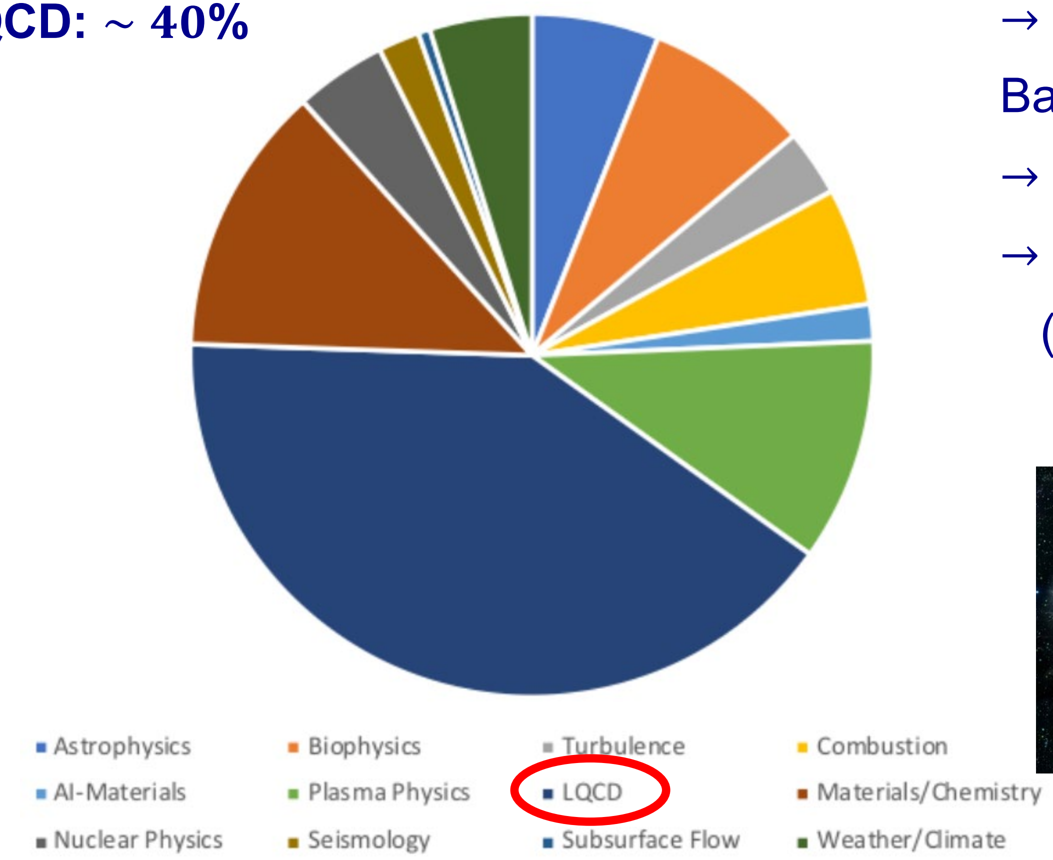



Figure credit: Jack Wells, Kate Clark

**Computational challenges of lattice field theory**

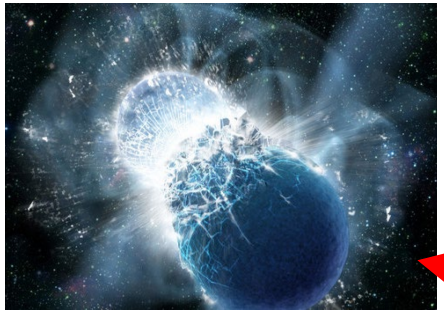
No direct computation of thermodynamic observables, …

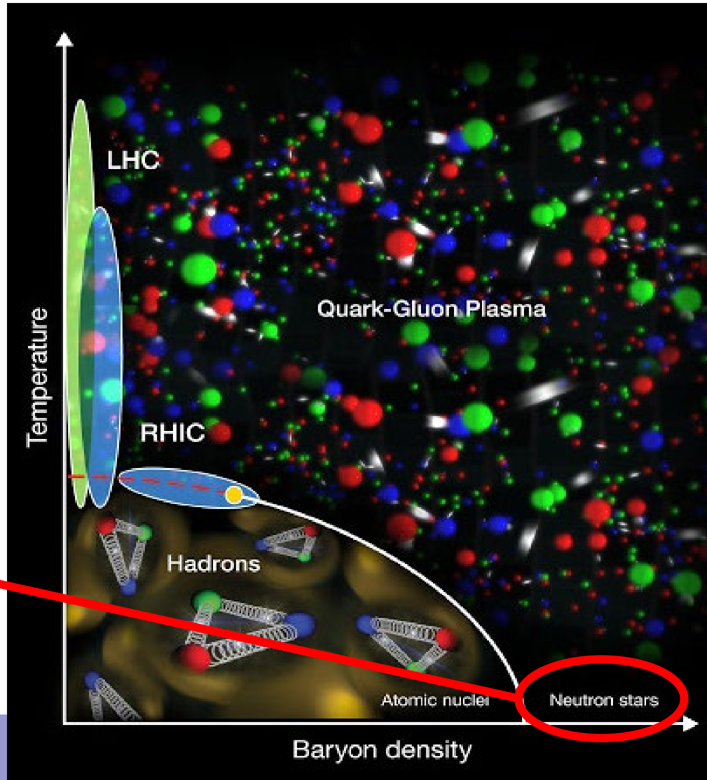
→ **Machine learning (lecture today)**

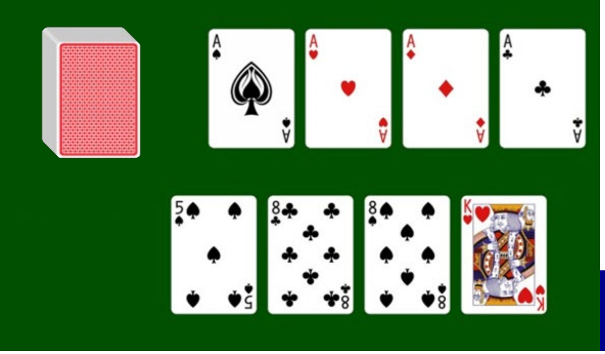
Baryon chemical potential, $\theta$-term, real-time evolution, …

→ Tensor networks

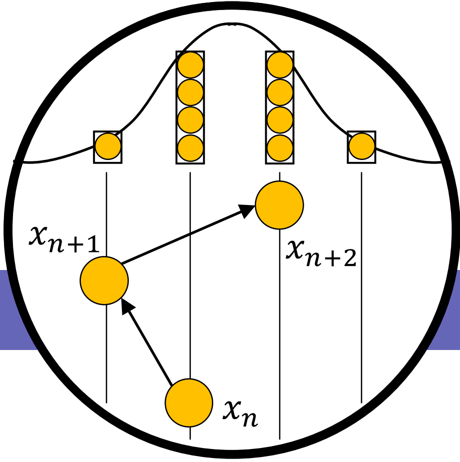
→ Quantum computing (lecture tomorrow)







# Reminder: (MC)MC method




## Monte Carlo method (MC)

### Naïve approach

*Randomly* generate ensemble of "configurations" $\{x\}$

$$\rightarrow \langle x_f|U(\tau',\tau)|x_i\rangle = \underbrace{\frac{1}{N}}\sum_{\{x\}} e^{-S(x)} = \underbrace{\langle e^{-S(x)} \rangle}$$

number of configurations in ensemble

average value within ensemble

### Problem

Generate lots of irrelevant configurations → inefficient!

### Solution

Generate configurations such that probability $p(x_n)$ of obtaining configuration $x_n$ is $p(x) \propto \exp[-S(x)]$

→ configurations have high probability of being relevant!

## Markov Chain … method (MCMC)

### Starting point

Initialization: Choose arbitrary starting point $x_n$

Proposal density: $g(x'|x_n)$ [e.g. Gaussian centered at $x_n$] suggests candidate $x'$ for $x_{n+1}$, given previous value $x_n$

### For each iteration $n$:

Generate $x'$ and calculate $\alpha = \exp[-S(x')] \,/\, \exp[-S(x_n)]$

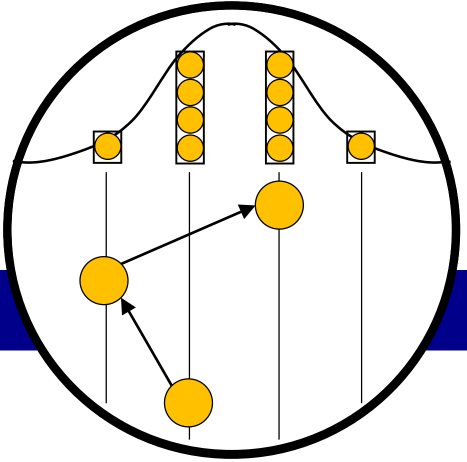
Accept or reject:

Generate uniform random number $u \in [0,1]$

If $u \leq \alpha$, accept candidate by setting $x_{n+1} = x'$

If $u \geq \alpha$, reject candidate and set $x_{n+1} = x_n$ instead

→ **Metropolis-Hastings algorithm**



# Why is this algorithm problematic?

| MCMC problems | Machine learning |
|---|---|

## MCMC problems

**(1) No parallel sampling**

$x_{n+1}$ depends on $x_n$ → *sequential* sampling → slow

**(2) No independent sampling**

$x_{n+1}$ depends on $x_n$ → samples are *correlated*

**(3) Incorrect normalization factor**

MCMC distribution: $p_{\mathrm{M}}(x) \propto \exp[-S(x)]$

Path integral distribution: $p(x) = \exp[-S(x)]\,/Z$

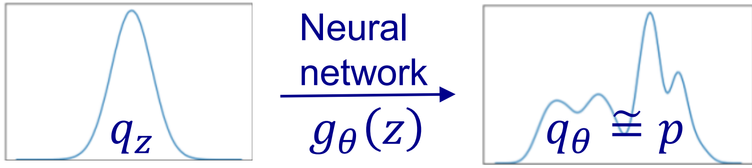
**(4) No direct access to thermodynamic observables**

Unknown partition function $Z = \int D[x]\, \exp[-S(x)]$

→ no direct computation of free energy $F = -T\ln Z$, etc.

## Machine learning

**(1) Parallel sampling**




$x_{n+1}$ is independent of $x_n$ → *parallel* sampling → fast

**(2) Independent sampling**

$x_{n+1}$ is independent of $x_n$ → samples are *uncorrelated*

**(3) Correct normalization factor**

ML distribution: $q_\theta(x) \cong p(x) = \exp[-S(x)]\,/Z$

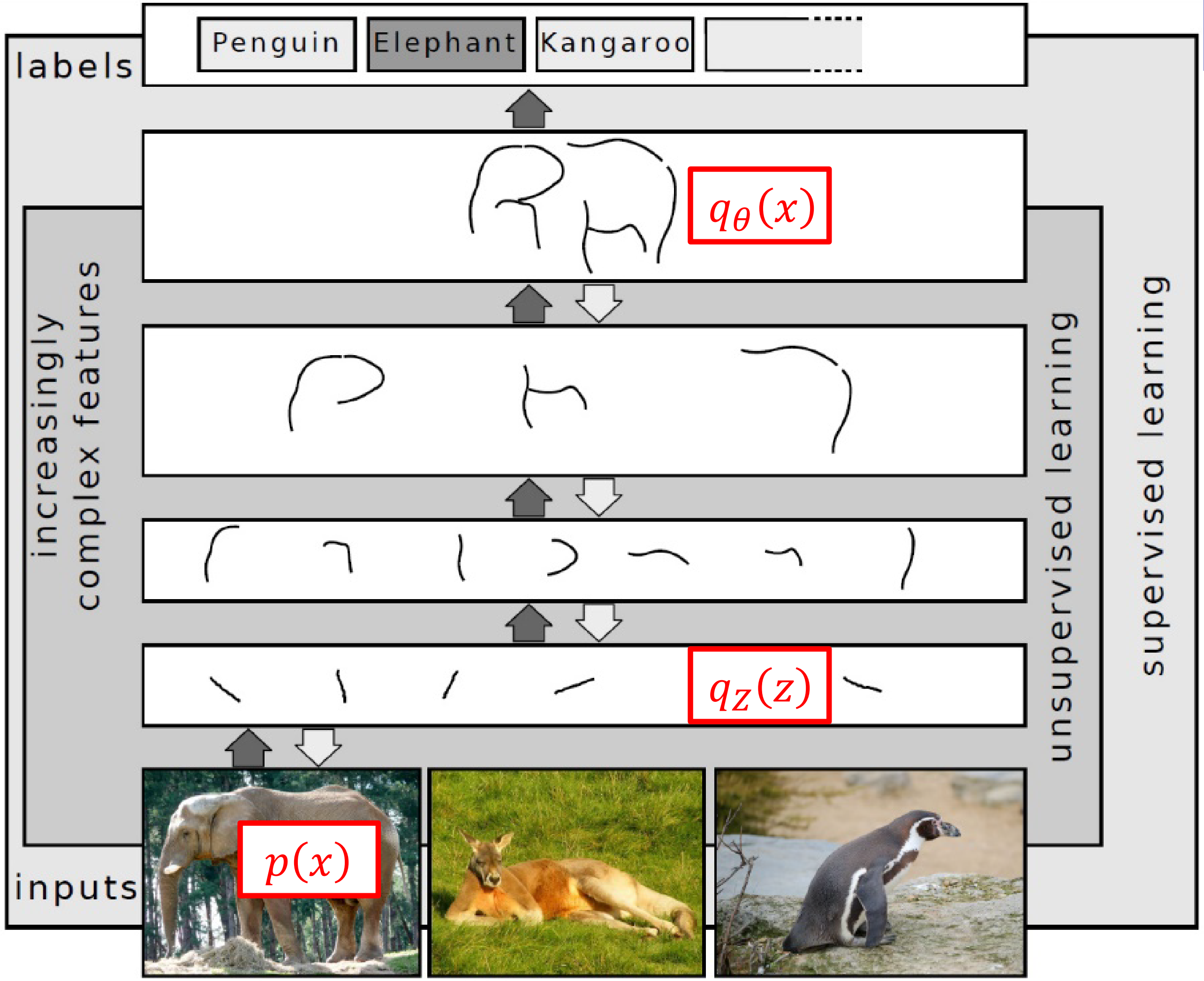
**(4) Direct access to free energy**

Known partition function $Z = \int D[x]\; q_\theta(\mathrm{x})\;\tilde{w} = \frac{1}{N}\sum \tilde{w}(x_i)$,

where $\tilde{w} = \exp[-S(x)]/q_\theta(x)$ and $x_i \sim q_\theta$

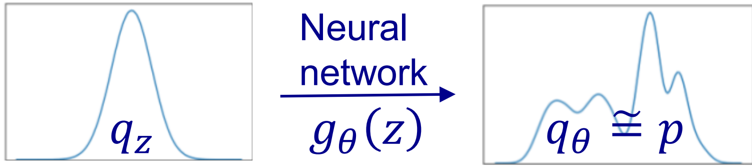
→ direct computation of free energy $F = -T\ln Z$, etc.

# Why is this algorithm problematic?




## Machine learning

**(1) Parallel sampling**




$x_{n+1}$ is independent of $x_n$ → *parallel* sampling → fast

**(2) Independent sampling**

$x_{n+1}$ is independent of $x_n$ → samples are *uncorrelated*

**(3) Correct normalization factor**

ML distribution: $q_\theta(x) \cong p(x) = \exp[-S(x)]\,/Z$

**(4) Direct access to free energy**

Known partition function $Z = \int D[x]\; q_\theta(\mathrm{x})\; \tilde{w} = \frac{1}{N}\sum \tilde{w}(x_i)$,
where $\tilde{w} = \exp[-S(x)]/q_\theta(x)$ and $x_i \sim q_\theta$

→ direct computation of free energy $F = -T \ln Z$, etc.

# Starting point: (deep) neural networks

**Layers**

- $y^L(z) = \sigma(W^L z + b^L)$
- input: $z \in \mathbb{R}^{|\Lambda|}$, output: $y^L \in \mathbb{R}^{|\Lambda|}$
- weights: $W^L \in \mathbb{R}^{|\Lambda|,|\Lambda|}$, bias: $b^L \in \mathbb{R}^{|\Lambda|}$

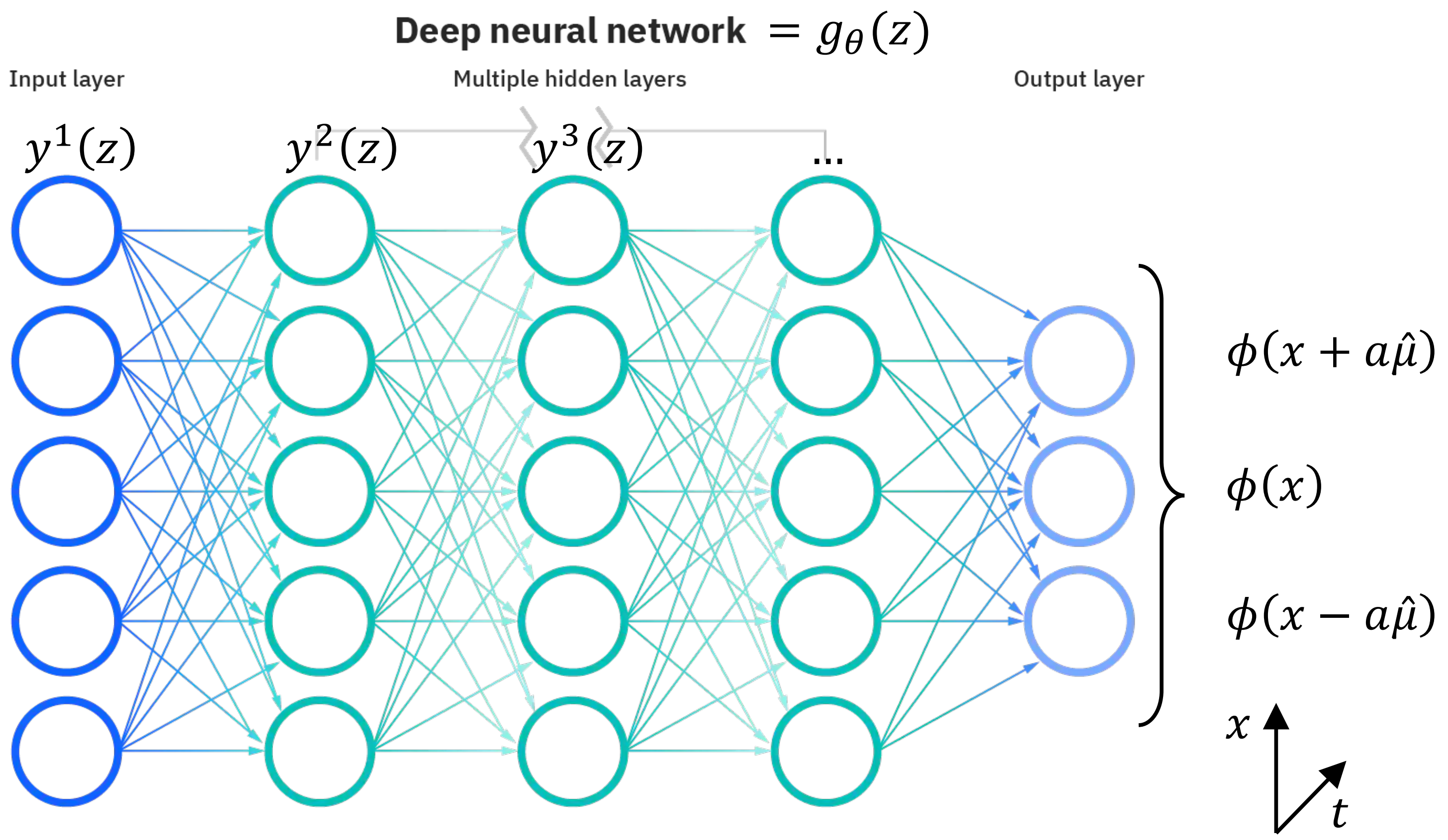
**Network**

- $g_\theta(z) = (y^L \circ y^{L-1} \circ \cdots \circ y^1)(z)$
- "deep": $L \gtrsim 10$
- odd: $b^L = 0$, $\sigma(z) = \tanh(z)$

**Symmetry**

- e.g. $\mathbb{Z}_2$ symmetry: real $\phi \leftrightarrow -\phi$
- prior distribution: $q_z(-z) = q_z(z)$
- odd network: $g_\theta(-z) = -g_\theta(z)$

Generate $\phi$-field configurations with neural network:
$\phi = g_\theta(z) \in \mathbb{R}^{|\Lambda|}$, $z \in \mathbb{R}^{|\Lambda|}$, where $|\Lambda| = N_x \times N_t$



# Starting point: normalizing flow

**Flow**

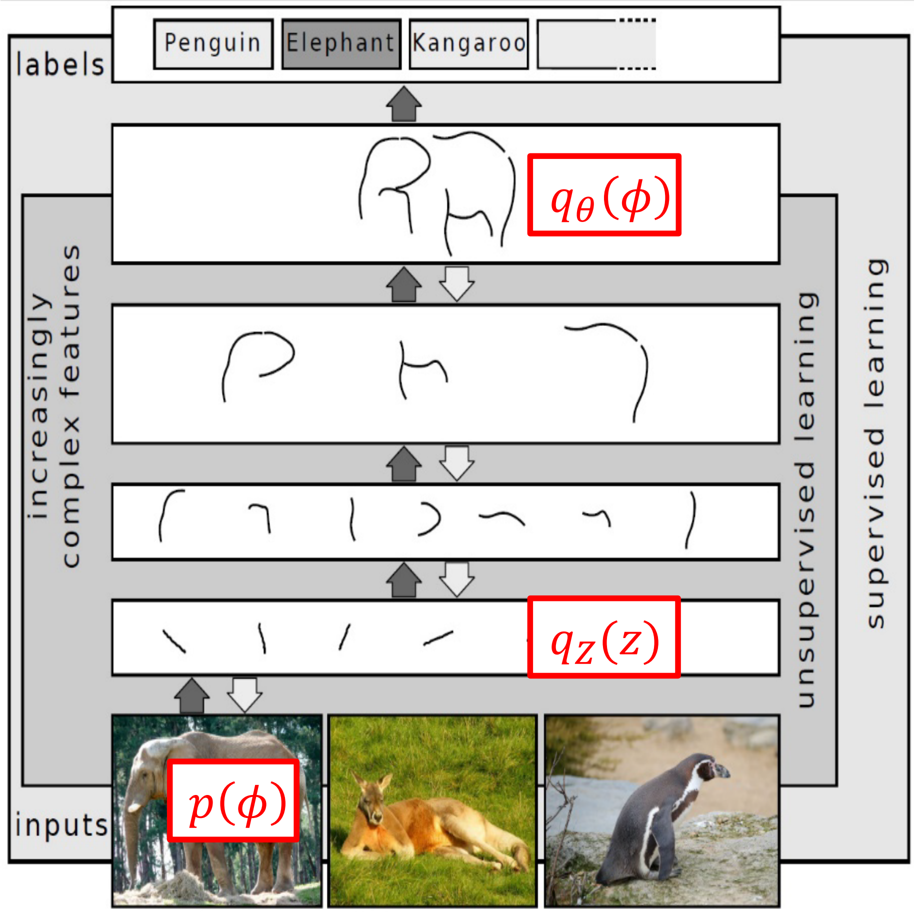
- normalizing flow: "flows" $z \in \mathbb{R}^{|\Lambda|}$ to $\phi \in \mathbb{R}^{|\Lambda|}$
- "generative model": built from neural network

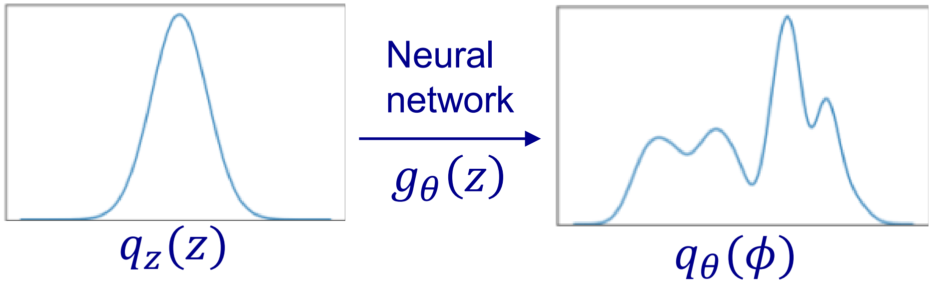
**Input**

- simple prior distribution: $q_z$ (e.g. Gaussian), $z \in \mathbb{R}^{|\Lambda|}$
- neural network: $g_\theta(z) = (y^L \circ y^{L-1} \circ \cdots \circ y^1)(z)$

**Output**

- complicated final distribution: $q_\theta(\phi) = q_z\left(g_\theta^{-1}(\phi)\right)\left|\frac{dg_\theta}{dz}\right|^{-1}$
- samples: $\phi = g_\theta(z) \sim q_\theta$ obtained from applying $g_\theta(z)$ to $z \sim q_z$



# Normalizing flows: applications

**2015**

**normalizing flows**

(Rezende, Mohamed, 2015)

**2018/2019**

- computer graphics

(Mueller, …, 2018)

- image generation

(Kingma, …, 2018, Ho, …, 2019)

**2019/2020**

- condensed matter physics

(Noé, …, 2019)

- collider physics

(Butter, 2020; Gao, …, 2020)

**2020/2021**

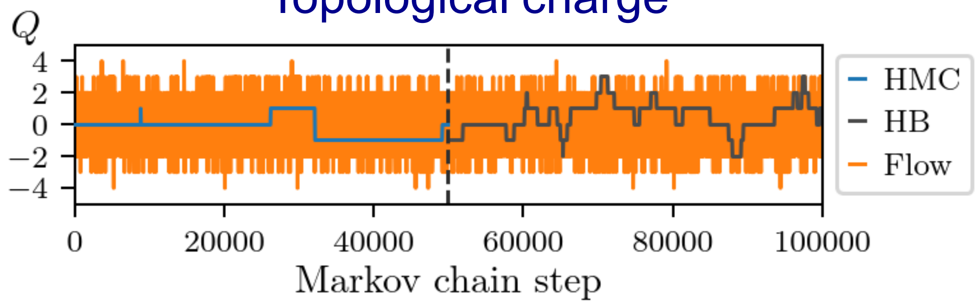
- gravitational waves

(Wong, …, 2020)

- cosmology

(Euclid Collaboration, 2021)

Topological charge




(Albergo, …, 2019)

**2019/2020**

- $\phi^4$ theory in 1+1D

(Albergo, …, 2019, Kim, …, 2020)

- U(1) gauge theories

(Kanwar, …, 2020)

- SU(N) gauge theories

(Boyda, …, 2020)

**2021/2022**

- lattice fermions

(Albergo, …, 2021)

- multimodal distributions

(Hackett, …, 2021)

- 3+1 dimensional "QCD"

(Abbott, …, 2022)

# Neural network: NICE architecture

**Architecture**

- Non-linear Independent Component Estimation (NICE)
- $q_\theta(\phi) = q_z\left(g_\theta^{-1}(\phi)\right)\left|\frac{dg_\theta}{dz}\right|^{-1}$ requires: $g_\theta(z) = (y^L \circ \cdots \circ y^1)(z)$ invertible, $\left|\frac{dg_\theta}{dz}\right|$ efficiently evaluable

**Features**

- split layers: $y^l = (y_u^l, y_d^l)$, where $y_u^l \in \mathbb{R}^{|\Lambda|-k}$, $y_d^l \in \mathbb{R}^k$, $k \in \{1, |\Lambda| - 1\}$
- $y_u^{l+1} = y_u^l$, $y_d^{l+1} = y_d^l + m(y_u^l)$, invertible: $y_u^l = y_u^{l+1}$, $y_d^l = y_d^{l+1} - m(y_u^{l+1})$
- $\det\frac{\partial y^{l+1}}{\partial y^l} = \begin{vmatrix} \mathbb{I} & 0 \\ * & \mathbb{I} \end{vmatrix} = 1 \rightarrow \left|\frac{dg_\theta}{dz}\right| = 1$, trivial

**Parameters**

- $m(y_u^l)$ contains learnable parameters $\theta$ of $g_\theta(z)$
- $g_\theta(z)$ has 6 coupling layers, $m(y_u^l)$ has 5 hidden layers

$\rightarrow$ number of parameters: $6 \times 5 \times (N_x \times N_t)^2$, up to $8 \times 10^6$

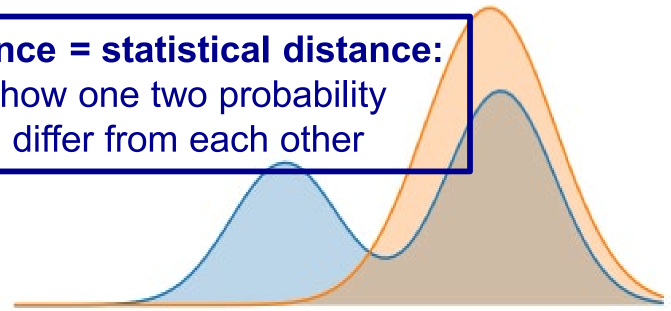
(NICE model trained on Toronto Face Dataset (TFD)
Image credit: Dinh, Krueger, Bengio, 2014)

# Training: approximating path integral distribution

## Goal

- target distribution: $p(\phi) = \frac{1}{Z}\exp\{-S(\phi)\}$
- train neural network: $q_\theta(\phi)$ as close as possible to $p(\phi)$

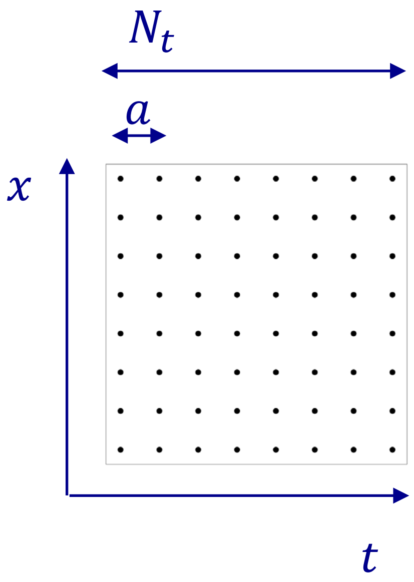
**KL divergence = statistical distance:**
measure of how one two probability distributions differ from each other



## Training

- minimize Kullback-Leibler (KL) divergence between $q_\theta$ and $p$:

$$\mathrm{KL}(q_\theta || p) = \int \mathcal{D}[\phi] q_\theta(\phi) \ln\left(\frac{q_\theta(\phi)}{p(\phi)}\right) = \beta(F_q - F) \xrightarrow{q_\theta(\phi) \to p(\phi)} 0$$




## Energy

- variational free energy: $\beta F_q = \mathbb{E}_{\phi \sim q_\theta}[S(\phi) + \ln q_\theta(\phi)]$
- free energy: $\beta F = -\ln Z$ with temperature $T = \frac{1}{\beta} = \frac{1}{a N_t}$

# Training: approximating path integral distribution

**Minimizing**

- Gibb's inequality: $\mathrm{KL}(q_\theta || p) = \beta\left(F_q - F\right) \geq 0$
- $F$ irrelevant → minimize $F_q$ via *gradient descent*

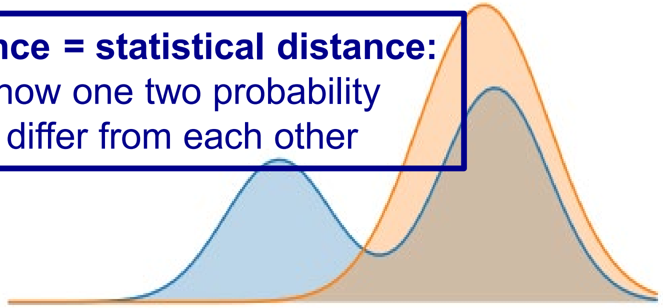
**Result**

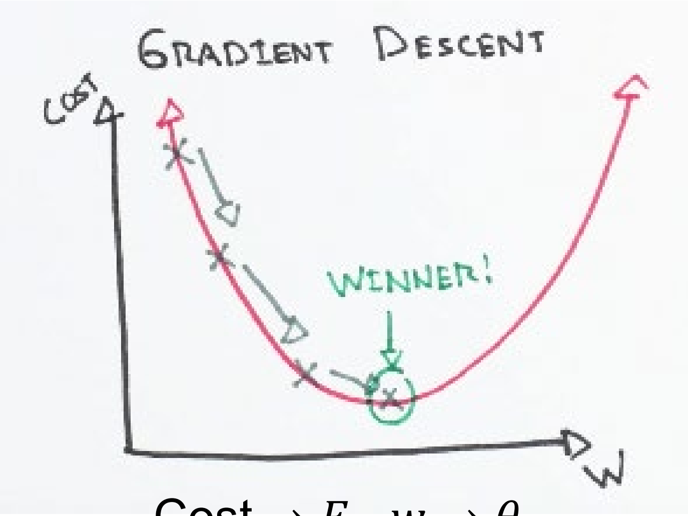
- free energy: $\beta F_q = \mathbb{E}_{\phi \sim q_\theta}[S(\phi) + \ln q_\theta(\phi)] \equiv -\langle \ln \tilde{w}(\phi) \rangle$
- "un-normalized importance weight": $\tilde{w}(\phi) = \exp\{-S(\phi)\} / q_\theta(\phi)$

**Sampling**

- approximated partition function: $Z = \int \mathrm{D}[\phi] q_\theta(\phi) \tilde{w}(\phi)$
- Monte-Carlo estimate: $\hat{Z} = \frac{1}{N} \sum_{i=1}^{N} \tilde{w}(\phi_i),\ \phi_i \sim q_\theta$

**KL divergence = statistical distance:** measure of how one two probability distributions differ from each other






Cost → $F_q$, $w \to \theta$

# The model: $\phi^4$ real scalar field theory

**Action**

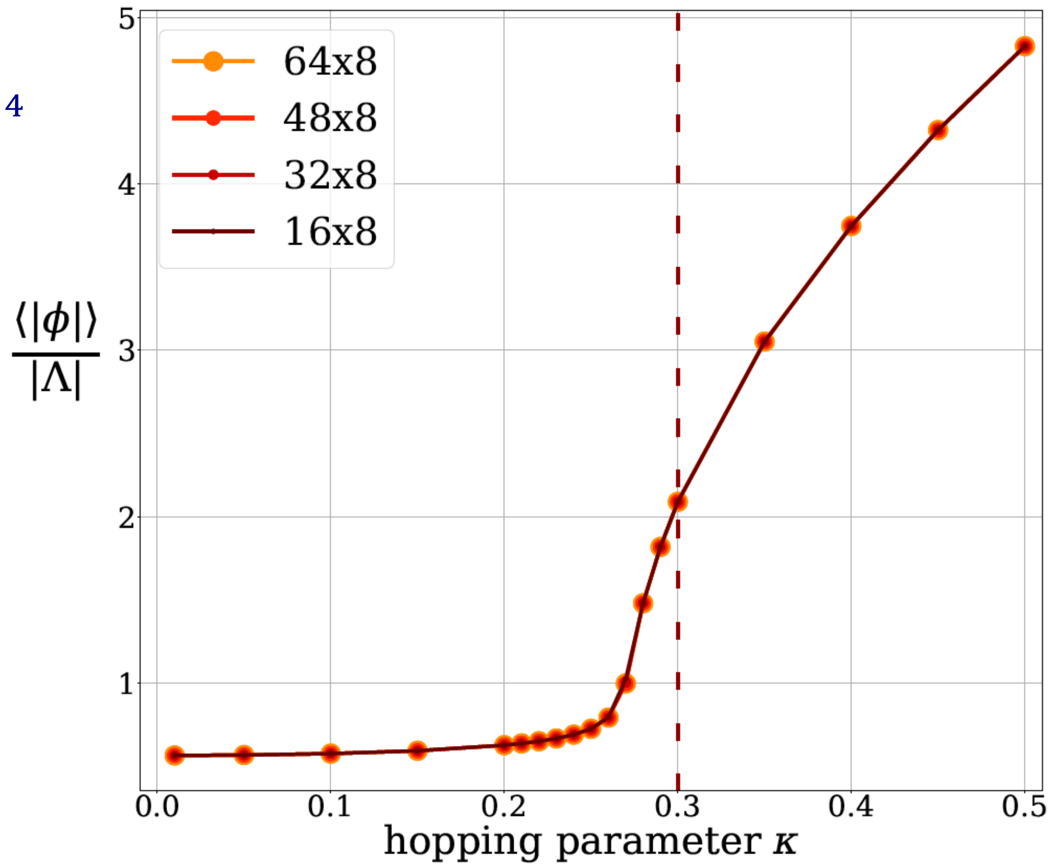
- $S = \sum_{x\in\Lambda} -2\kappa \sum_{\hat{\mu}=1}^{2} \phi(x)\phi(x+\hat{\mu}) + (1-2\lambda)\phi(x)^2 + \lambda\phi(x)^4$
- invariant under $\mathbb{Z}_2$ transformations: $\phi \leftrightarrow -\phi$

**Observables**

- absolute magnetization: $\langle|\phi|\rangle$
- free energy: $F = -T \ln Z = -PV$

**Transition**

- spontaneous symmetry breaking: $\langle|\phi|\rangle \neq 0$
- for $\lambda = 0.022$, $\kappa \lesssim 0.3$, and different $|\Lambda| = N_x \times N_t$



# MCMC: indirect computation of free energy

MCMC

- no *direct* estimate of $Z$ and $F = -T \ln Z$
- only differences: $\Delta F_{t,0} = F_t - F_0 = -T \ln \frac{Z_t}{Z_0}$,

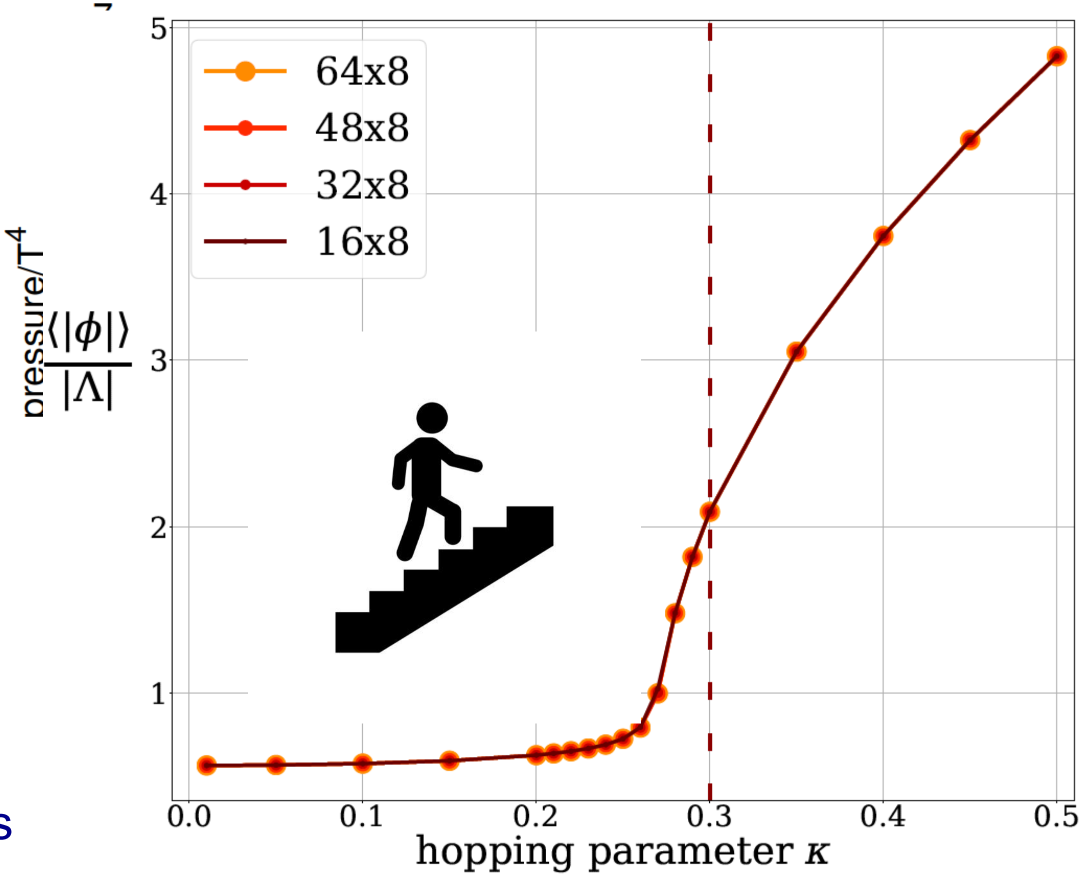
$$\frac{Z_t}{Z_0} = \frac{1}{Z_0} \int D[\phi] e^{-S_0(\phi)} \frac{e^{-S_t(\phi)}}{e^{-S_0(\phi)}} = \mathbb{E}_{p_0}\left[\frac{\exp(-S_t)}{\exp(-S_0)}\right]$$

Trajectory

- starting point: $F_0 \equiv F(\kappa = 0)$ known analytically
- target: $F_t \equiv F(\kappa = 0.3)$ beyond critical point

Problem

- small overlap of $p_0$ and $p_t$: use $\Delta F_{t,0} = \Delta F_{t,i_k} + \cdots + \Delta F_{i_1,0}$
- integration through phase space → large statistical errors




(Image credit: Borsanyi et al., 2016)

# MCMC vs. machine learning: results *before* critical point

**Parameters**

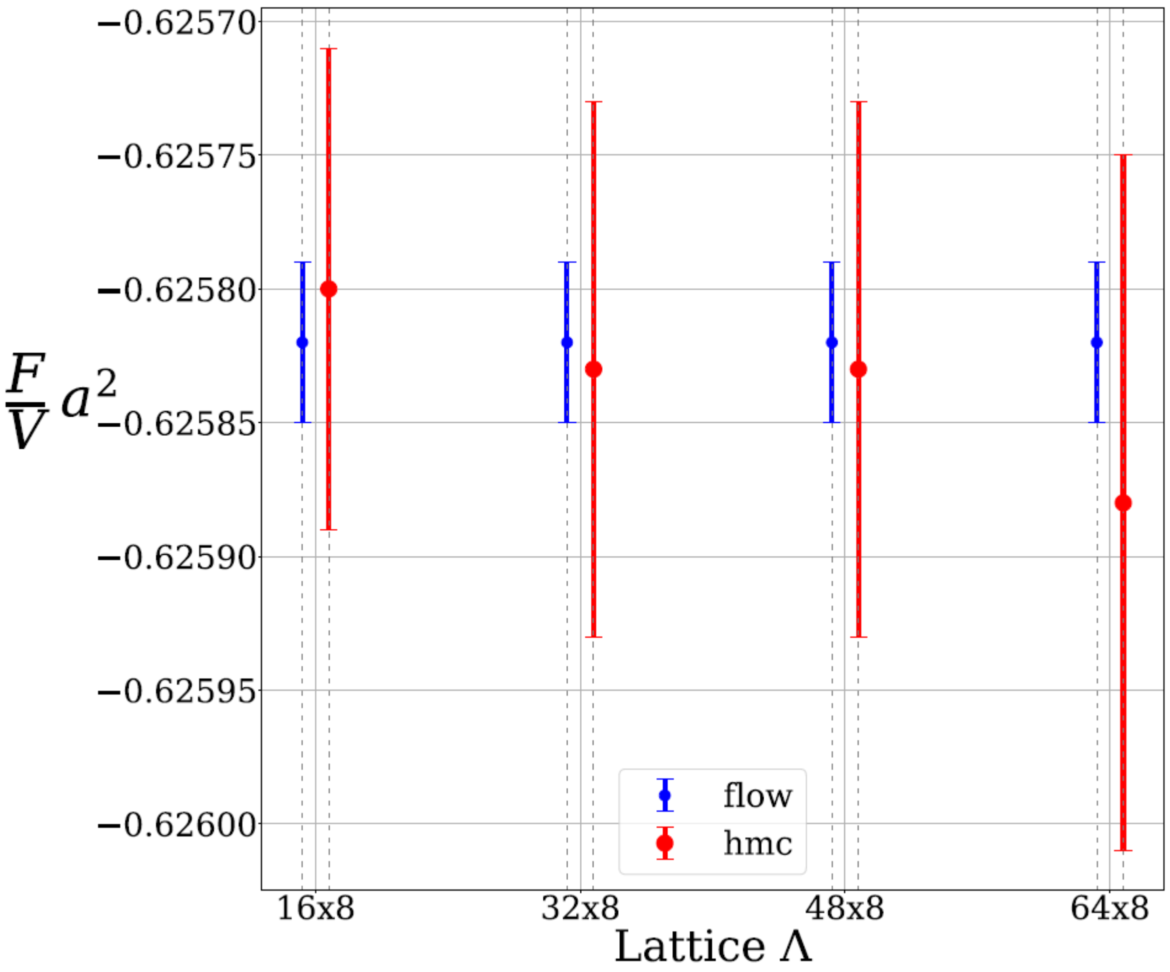
- $\kappa = 0.2,\ \lambda = 0.022$
- different volumes $V = N_x \times N_t$

**MCMC**

- *indirect* computation of free energy $F_t$
- need 4 Markov chains from $F_0$ to $F_t$

**ML**

- *direct* computation of $F_t$ independently of $F_0$
- errors are comparable to MCMC errors



# MCMC vs. machine learning: results *after* critical point

**Parameters**

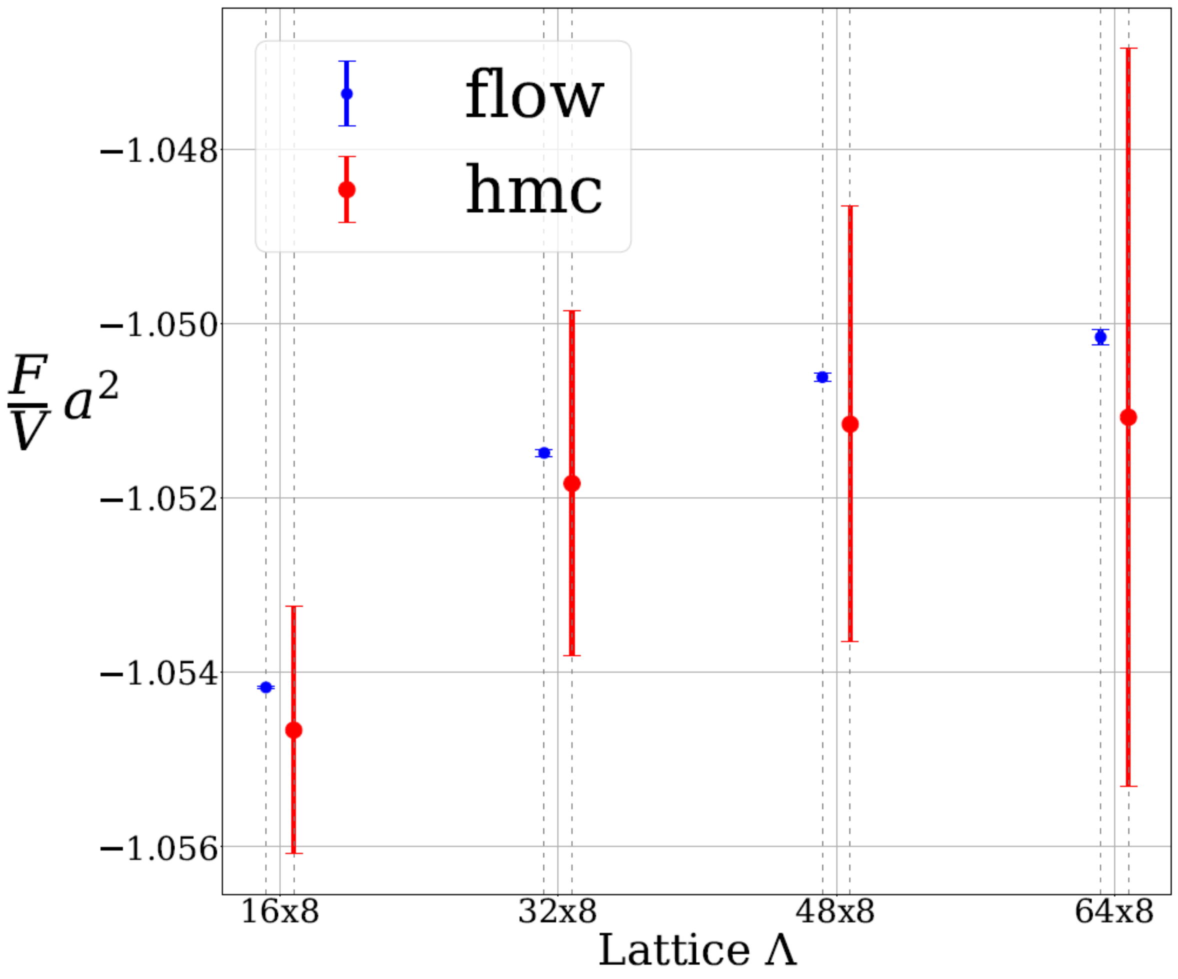
- $\kappa = 0.3$, $\lambda = 0.022$
- different volumes $\mathrm{V} = N_x \times N_t$

**MCMC**

- now 14 Markov chains from $\mathrm{F}_0$ to $\mathrm{F}_\mathrm{t}$
- crossing critical region → larger errors

**ML**

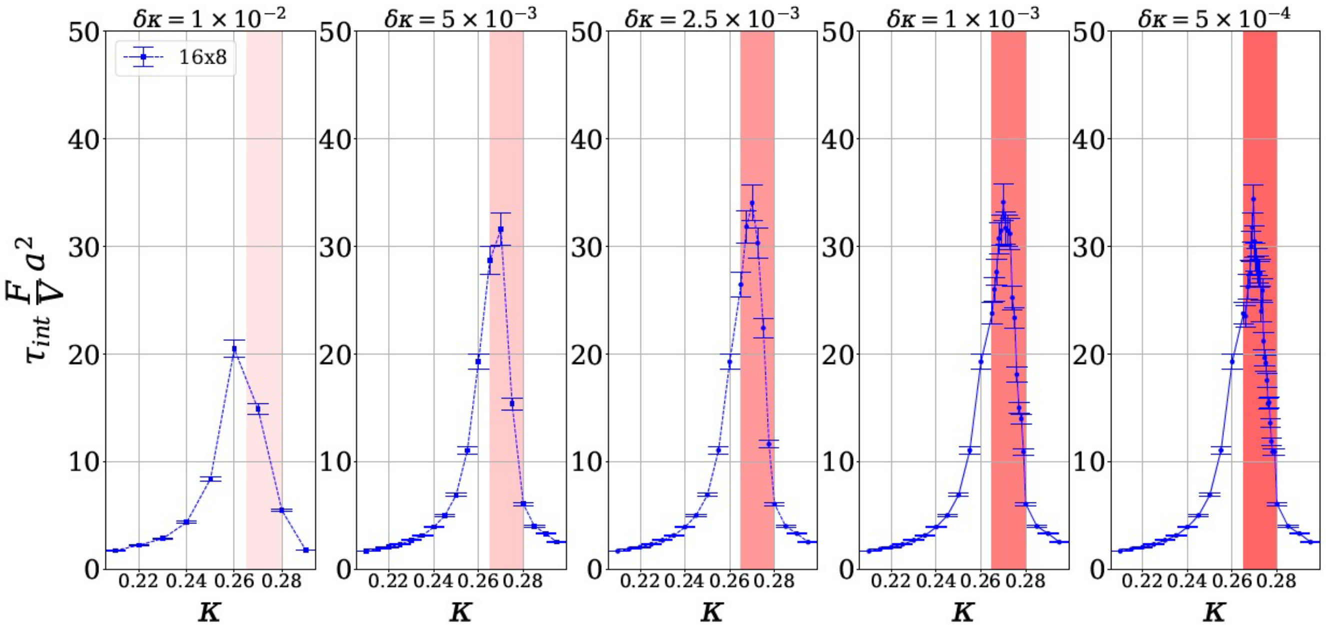
- errors are $\mathcal{O}(10^{-2})$ smaller than MCMC errors!
- precision *independent* of point in parameter space

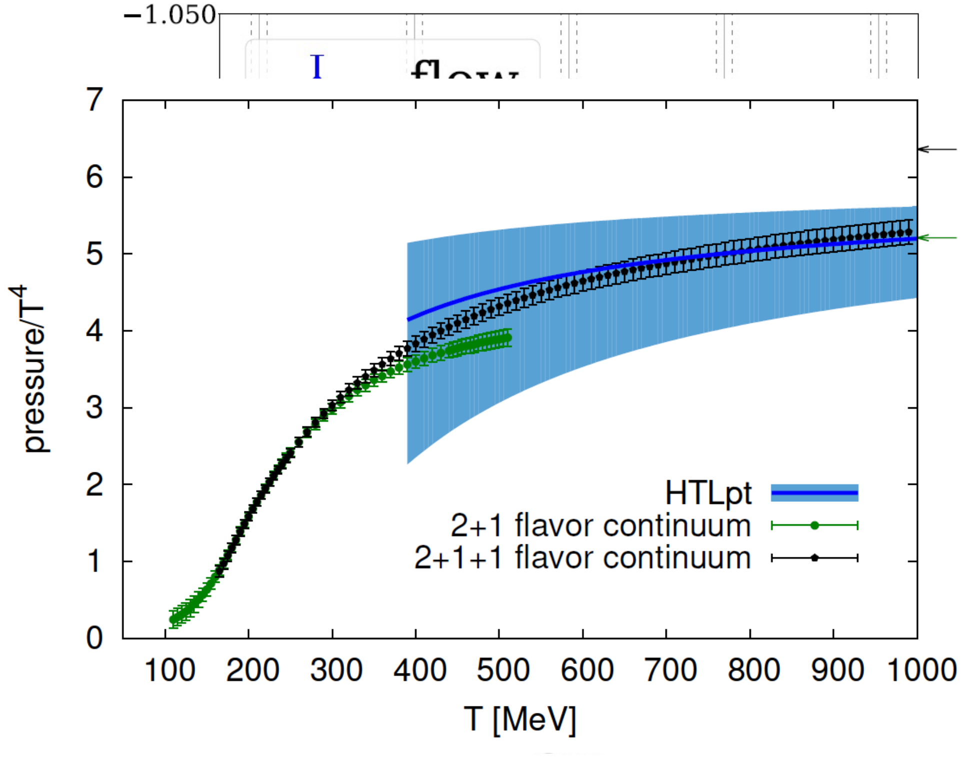


# MCMC results: systematic errors

MCMC

- error increases with step size $\delta\kappa$
- “integrated autocorrelation time” peaks at critical point
- finite-T QCD: $F_0$ approximated → additional error



# Machine-learning results: systematic errors

**Reminder**

- KL divergence ("distance" between $q_\theta(\phi)$ and $p(\phi)$):

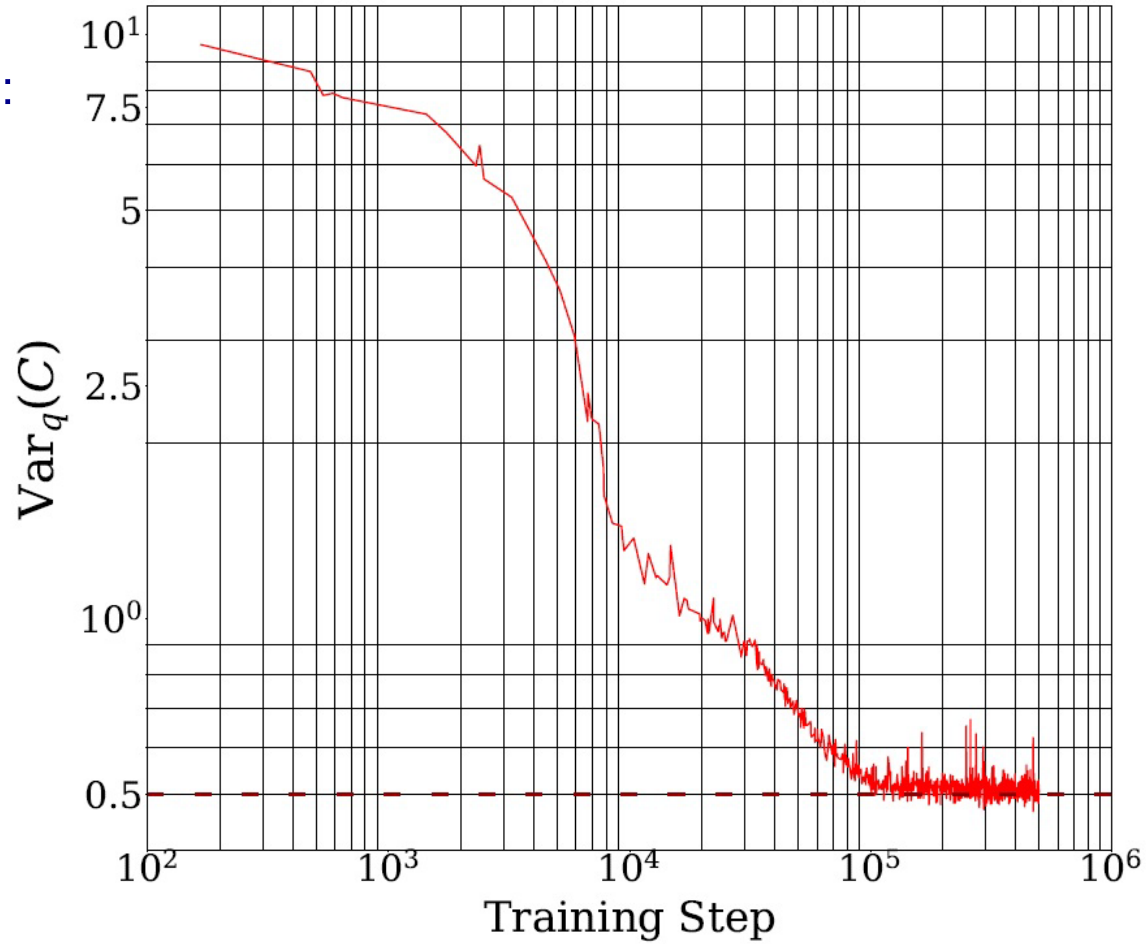
$$\mathrm{KL}(q_\theta||p) = \int \mathcal{D}[\phi] q_\theta(\phi) \ln\left(\frac{q_\theta(\phi)}{p(\phi)}\right) = \beta(F_q - F)$$

**Flow**

- $\beta F_q = -\langle \ln \tilde{w} \rangle_q = -\langle S(\phi) + \ln q_\theta(\phi) \rangle_q \equiv \langle C \rangle_q$
- $\mathrm{KL}(q_\theta||p) = \frac{1}{2}\mathrm{Var}_q(C) + \mathcal{O}\left(\mathbb{E}_q[|w-1|^3]\right) \neq 0$

**Errors**

- bias $\mathrm{KL}(q_\theta||p) \neq 0$ due to imperfect training: $\sim N^{-1}$
- error of estimator $\beta\hat{F} = -\ln \frac{1}{N}\sum_{i=1}^{N} \tilde{w}(\phi_i)$: $\sigma \sim N^{-1/2}$



# MCMC vs. machine learning: computational costs

**MCMC**

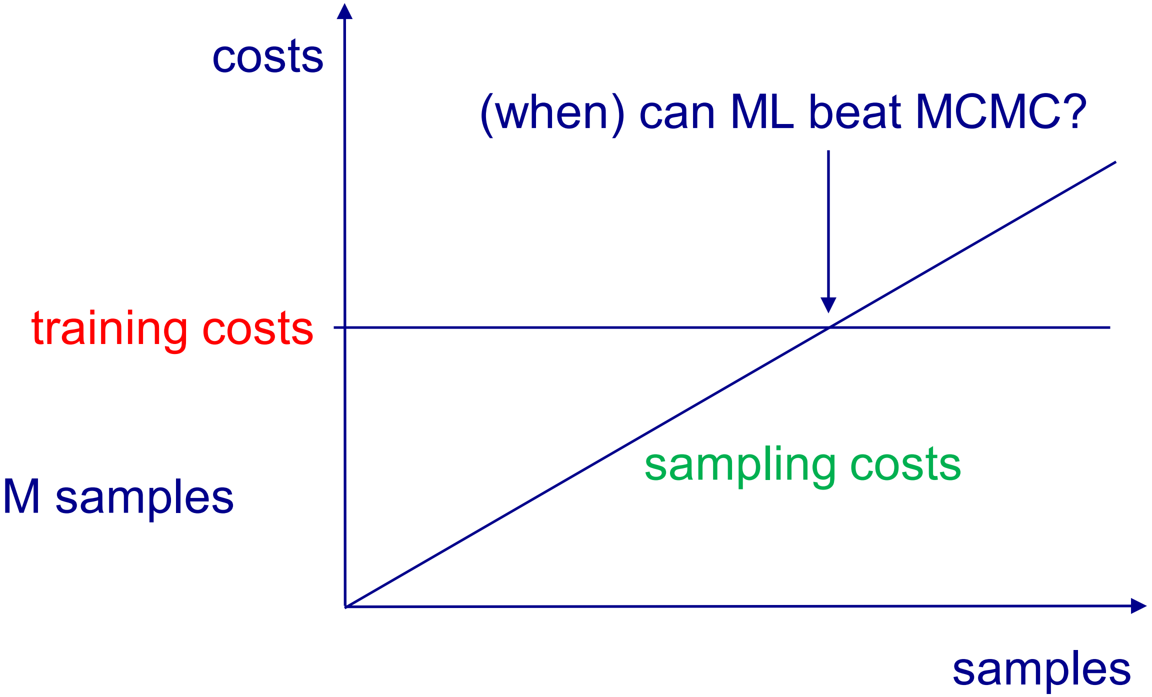
- no training costs
- sequential, dependent sampling

**ML**

- up-front training: $\sim$ 20 hours
- parallel, independent sampling: $<$ 1 minute for 5.6M samples

**Prospects**

- ML could beat MCMC when crossing critical points
- need optimization of network, hyperparameter tuning, etc.



# Summary: machine learning for lattice field theories

**MCMC**

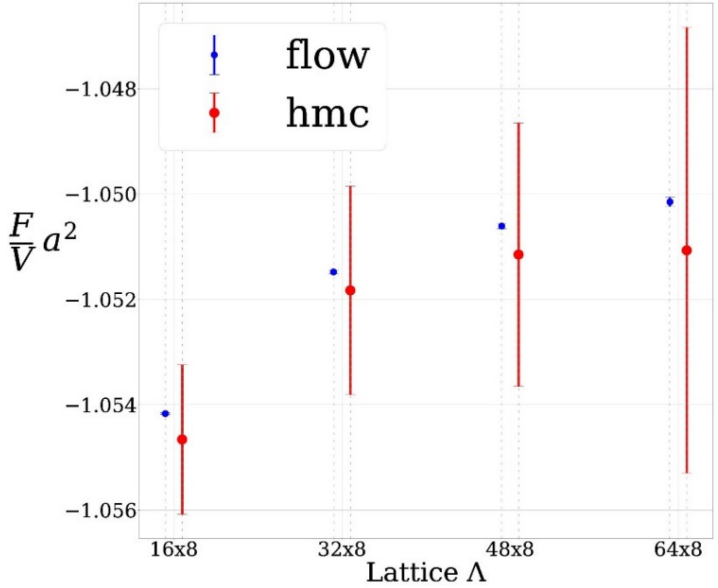
- no access to $Z$ → *indirect* computation of $F = -T \ln Z = -PV$
- integration through phase space → large statistical errors

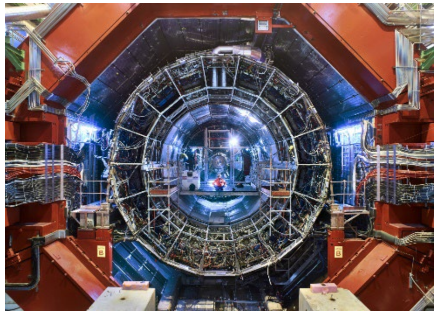
**ML**

- approximate path integral distribution
- efficiently sample topological quantities (Kanwar, …, 2020)
- compute $F = -T \ln Z$ without phase-space integration (Kim, …, 2020)

**Outlook**

- extensions: gauge fields, fermions, higher dimensions, larger lattices
- long-term goal: scaling to lattice QCD calculations



(ALICE Experiment / CERN)

# Outlook: roadmap and obstacles

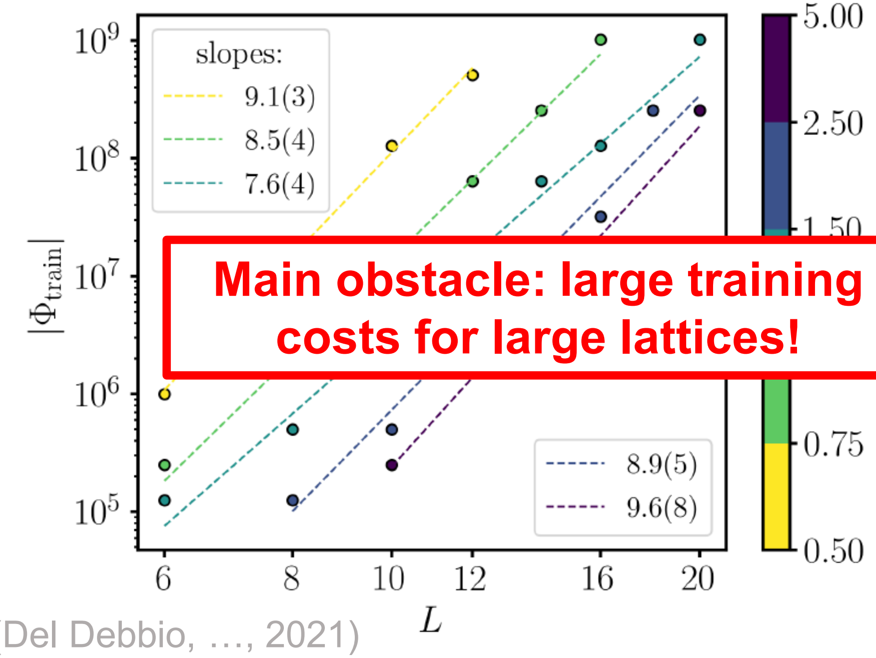
**2015** — **2018/2019** — **2019/2020** — **2020/2021**

**normalizing flows**
(Rezende, Mohamed, 2015)

- computer graphics
  (Mueller, …, 2018)
- image generation
  (Kingma, …, 2018, Ho, …, 2019)

- condensed matter physics
  (Noé, …, 2019)
- collider physics
  (Butter, 2020; Gao, …, 2020)

- gravitational waves
  (Wong, …, 2020)
- cosmology
  (Euclid Collaboration, 2021)




**Main obstacle: large training costs for large lattices!**

(Del Debbio, …, 2021)

**2019/2020** — **2021/2022**

- $\phi^4$ theory in 1+1D
  (Albergo, …, 2019, Kim, …, 2020)
- U(1) gauge theories
  (Kanwar, …, 2020)
- SU(N) gauge theories
  (Boyda, …, 2020)

- lattice fermions
  (Albergo, …, 2021)
- multimodal distributions
  (Hackett, …, 2021)
- 3+1 dimensional “QCD”
  (Abbott, …, 2022)

# Resources: tutorials, codes, talks, slides

**Introduction to Normalizing Flows for Lattice Field Theory**

Michael S. Albergo,[1,*] Denis Boyda,[2,3,4,†] Daniel C. Hackett,[3,4,‡] Gurtej Kanwar,[3,4,§]
Kyle Cranmer,[1] Sébastien Racanière,[5] Danilo Jimenez Rezende,[5] and Phiala E. Shanahan[3,4]

[1] *Center for Cosmology and Particle Physics, New York University, New York, NY 10003, USA*
[2] *Argonne Leadership Computing Facility, Argonne National Laboratory, Lemont IL 60439, USA*
[3] *Center for Theoretical Physics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA*
[4] *The NSF AI Institute for Artificial Intelligence and Fundamental Interactions*
[5] *DeepMind, London, UK*

(Dated: August 9, 2021)

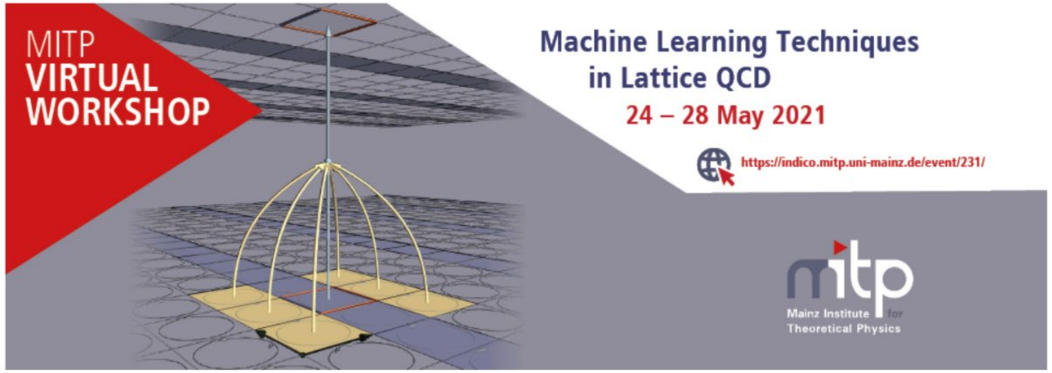
This notebook tutorial demonstrates a method for sampling Boltzmann distributions of lattice field theories using a class of machine learning models known as normalizing flows. The ideas and approaches proposed in arXiv:1904.12072, arXiv:2002.02428, and arXiv:2003.06413 are reviewed and a concrete implementation 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.

```
def train_step(model, action, loss_fn, optimizer, metrics):
    layers, prior = model['layers'], model['prior']
    optimizer.zero_grad()

    or, layers, batch_size=batch_size)
```

**Thanks for listening!**

**Do you have any questions?**




# Backup: derivation of $\mathrm{KL}(q_\theta||p) \sim \frac{1}{2}\mathrm{Var}_q(C)$

**Divergence**

- $\mathrm{KL}(q_\theta||p) = \mathbb{E}_q\left[\ln\frac{p}{q}\right] = -\mathbb{E}_q[\ln w] \geq 0$ due to: $\sum_i^N p_i \log p_i \geq \sum_i^N p_i \log q_i$
- $\mathbb{E}_q\left[\frac{p}{q}\right] = \mathbb{E}_q[w] = \int D[\phi]p(\phi) = 1$

**Expansion**

- $\mathrm{KL}(q_\theta||p) = \mathbb{E}_q[w-1] + \frac{1}{2}\mathbb{E}_q[(w-1)^2] + \mathcal{O}(\mathbb{E}_q[(w-1)^3]) = \frac{1}{2}\mathbb{E}_q[(w-1)^2] + \mathcal{O}(\mathbb{E}_q[(w-1)^3])$
- $\mathbb{E}_q[C] = -\mathbb{E}_q[\ln\tilde{w}] = \mathbb{E}_q[\ln w + \ln Z] = -\ln Z + \mathrm{KL}(q_\theta||p) = -\ln Z + \mathcal{O}(\mathbb{E}_q\,[(w-1)^2])$

**Variance**

- $\mathrm{Var}_q(C) = \mathbb{E}_q\left[\left(C - \mathbb{E}_q[C]\right)^2\right] = \mathbb{E}_q\left[\left(-\ln\tilde{w} + \ln Z + \mathcal{O}(\mathbb{E}_q\,[(w-1)^2])\right)^2\right] = \mathbb{E}_q[(w-1)^2] + \mathcal{O}(\mathbb{E}_q\,[(w-1)^3])$
- Result: $\mathrm{Var}_q(C) = 2\mathrm{KL}(q_\theta||p) + \mathcal{O}(\mathbb{E}_q\,[w-1]^3)$

$-\ln\tilde{w} + \ln Z = -\ln w \approx (w-1) + \cdots$

# Backup: analytical solution at $\kappa = 0$

**$\kappa = 0$**

- $Z = \prod_{x \in \Lambda} \int d\phi(x) \, \exp\left[-(1-2\lambda)\phi(x)^2 - \lambda\phi(x)^4\right]$
- solution: $\int dx \; \exp(-ax^4 - bx^2) = \sqrt{\frac{b}{4a}} \exp\left(\frac{b^2}{8a}\right) K_{\frac{1}{4}}\left(\frac{b^2}{8a}\right)$

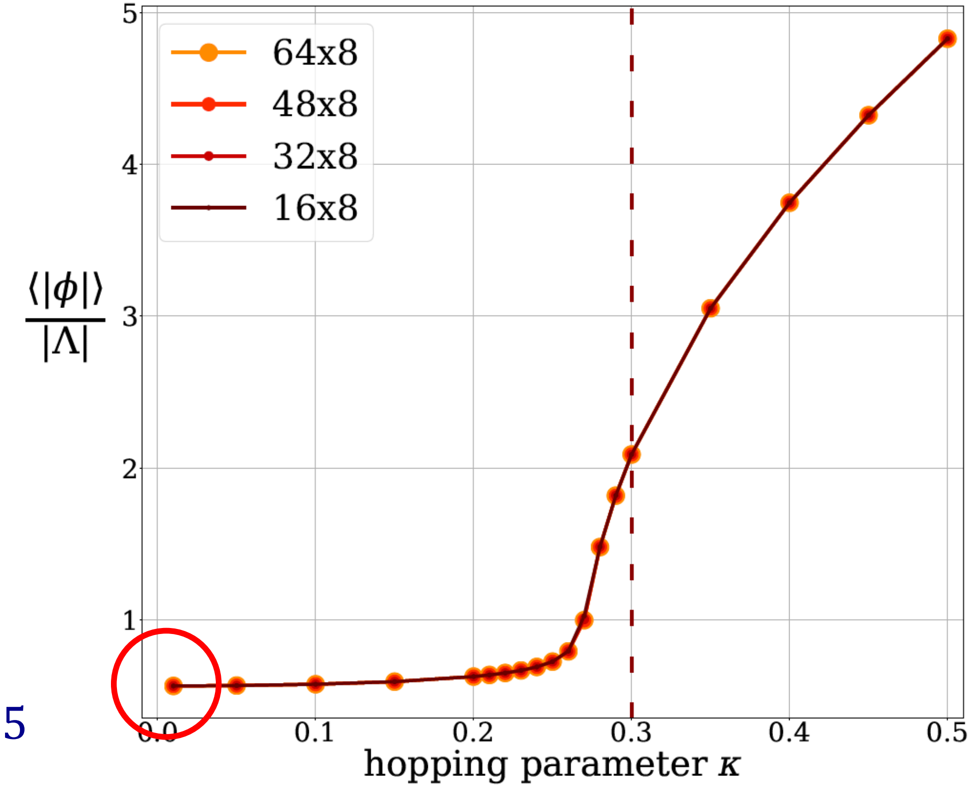
**Free energy**

- analytical solution: $F = -T|\Lambda| \ln z\,(\lambda)$

  with $z(\lambda) = \sqrt{\frac{1-2\lambda}{4\lambda}} \exp\left(\frac{(1-2\lambda)^2}{8\lambda}\right) K_{\frac{1}{4}}\left(\frac{(1-2\lambda)^2}{8\lambda}\right)$

**Magnetiz.**

- analytical solution: $\langle|\phi|\rangle = \int \; dx \; |x| \exp(-ax^4 - bx^2) \,/\, Z$
- for $\lambda = 0.002$: $\langle|\phi|\rangle_{\text{an.}} = 0.5618$ vs. $\langle|\phi|\rangle_{\text{num.}} = 0.5621 \pm 0.0005$



# Backup: analytical solution for magnetization at $\kappa = 0$

**$\kappa = 0$**

- partition function: $Z = \int D[\phi] \exp\left[-S(\phi)\right]$
- $Z = \prod_{x\in\Lambda} \int d\phi(x) \exp\left[-(1-2\lambda)\phi(x)^2 - \lambda\phi(x)^4\right]$

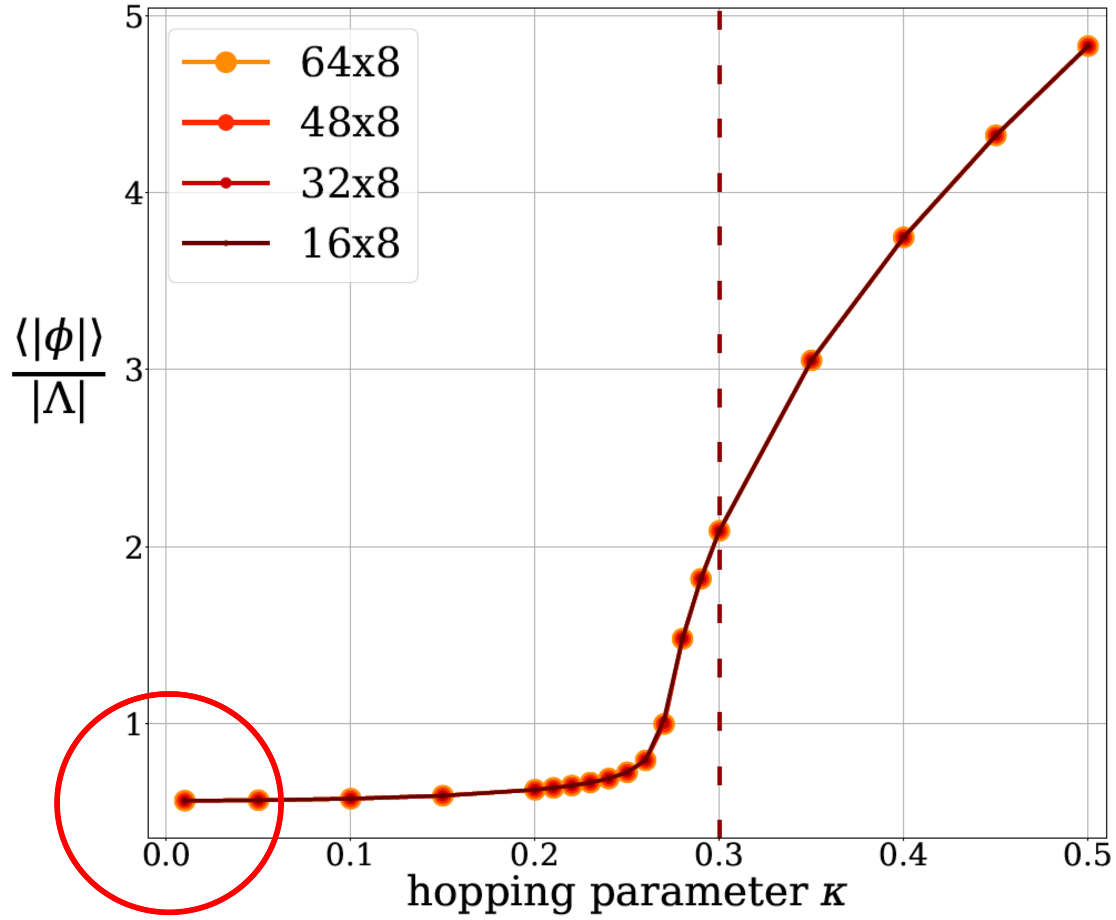
**Integral**

- analytical solution:

$$\int \exp(-ax^4 - bx^2)dx = \sqrt{\frac{b}{4a}} \exp\left(\frac{b^2}{8a}\right) K_{\frac{1}{4}}\left(\frac{b^2}{8a}\right)$$

**Result**

- free energy: $F = -T|\Lambda| \ln z\,(\lambda)$
- with $z(\lambda) = \sqrt{\frac{1-2\lambda}{4\lambda}} \exp\left(\frac{(1-2\lambda)^2}{8\lambda}\right) K_{\frac{1}{4}}\left(\frac{(1-2\lambda)^2}{8\lambda}\right)$



# Backup: no mode-dropping for MCMC

Goal

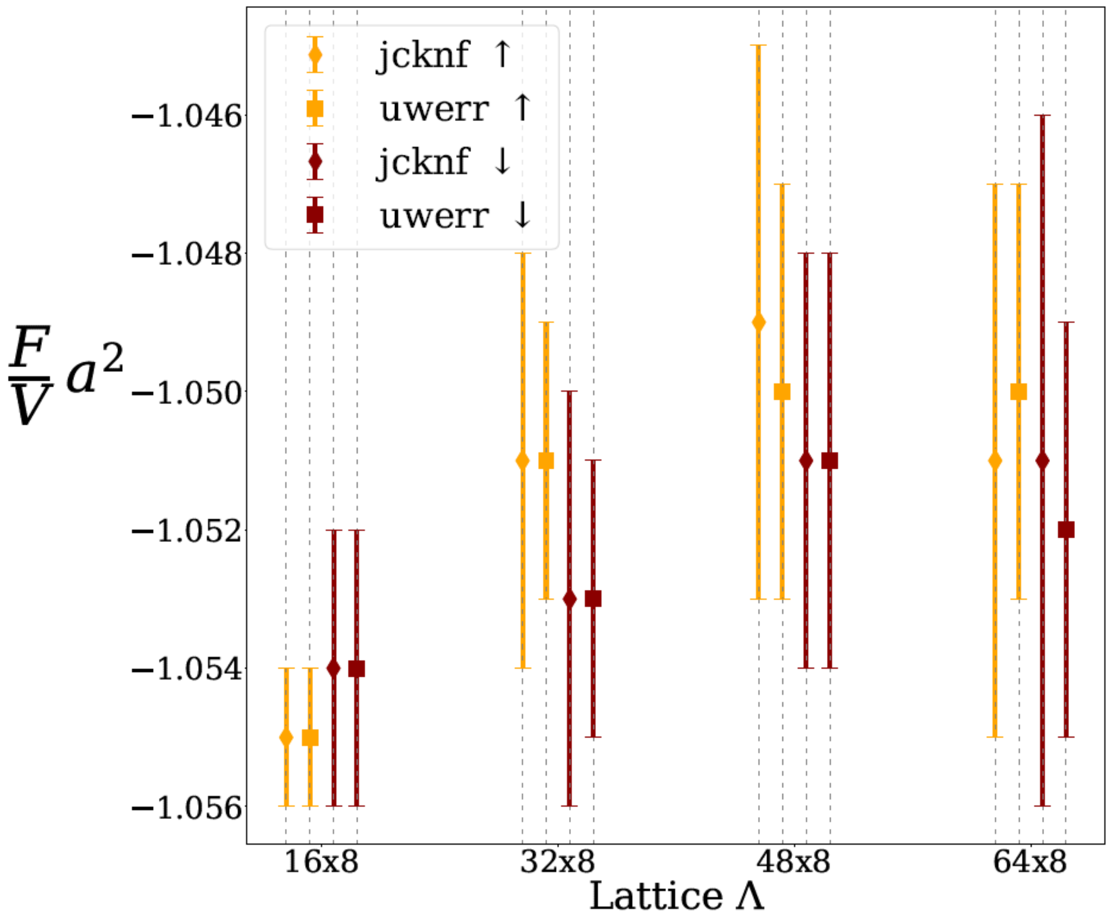
- ensure sufficient overlap of distributions $p_i$ and $p_{i+1}$ in:
- $\mathbb{E}_{p_i}\left[\frac{\exp(-S_{i+1})}{\exp(-S_i)}\right] = \frac{1}{Z_i}\int D[\phi]e^{-S_i(\phi)}\frac{e^{-S_{i+1}(\phi)}}{e^{-S_i(\phi)}} = \frac{Z_{i+1}}{Z_i}$

Method

- estimate $\frac{Z_i}{Z_{i+1}}$ by exchanging $p_i$ with $p_{i+1}$ above
- cheap consistency check: run a single Markov chain

Result

- MCMC-based 1: $\frac{Z_{i+1}}{Z_i}$ (up)
- MCMC-based 2: $\frac{Z_i}{Z_{i+1}}$ (down)



# Backup: no mode-dropping for flow

**Goal**

- all modes of $p(\phi)$ should be captured by $q_\theta(\phi)$
- no mode-dropping → no underestimation of $Z$

**Method**

- use: $\mathbb{E}_{\phi\sim p}[\tilde{w}^{-1}(\phi)] = \frac{1}{Z}\int D[\phi] e^{-S(\phi)} \frac{q_\theta(\phi)}{e^{-S(\phi)}} = \frac{1}{Z}$
- LHS: estimated using a single Markov chain

**Result**

- flow-based 1: plug estimate $\hat{Z}$ into $\hat{F} = -T \ln \hat{Z}$
- flow-based 2: $\hat{F} = -T \ln \frac{1}{N} \sum_{i=1}^{N} \tilde{w}(\phi_i)$ with $\phi_i \sim q_\theta$