TECHNISCHE UNIVERSITÄT WIEN

# Sampling Non-Abelian Lattice Gauge Theories with Diffusion Models

xQCD 2026 – 14.07.2026

Thomas Ranner

Institute for Theoretical Physics, TU Wien
thomas.ranner@tuwien.ac.at

# Based on



**High Energy Physics - Lattice**

*[Submitted on 27 Jan 2026]*

## Generalizable Equivariant Diffusion Models for Non-Abelian Lattice Gauge Theory

Gert Aarts, Diaa E. Habibi, Andreas Ipp, David I. Müller, Thomas R. Ranner, Lingxiao Wang, Wei Wang, Qianteng Zhu

We demonstrate that gauge equivariant diffusion models can accurately model the physics of non-Abelian lattice gauge theory using the Metropolis-adjusted annealed Langevin algorithm (MAALA), as exemplified by computations in two-dimensional U(2) and SU(2) gauge theories. Our network architecture is based on lattice gauge equivariant convolutional neural networks (L-CNNs), which respect local and global symmetries on the lattice. Models are trained on a single ensemble generated using a traditional Monte Carlo method. By studying Wilson loops of various size as well as the topological susceptibility, we find that the diffusion approach generalizes remarkably well to larger inverse couplings and lattice sizes with negligible loss of accuracy while retaining moderately high acceptance rates.

# Motivation

- Lattice quantum field theory as alternative to perturbative methods in QFT
- Very slow (calculations may take weeks on HPCs)
- Solving the path integral to calculate observables

$$\langle O \rangle = \frac{1}{\mathrm{Z}} \int DU\, O(U)\, e^{-S_E(U)}, \qquad \mathrm{Z} = \int DU\, e^{-S_E(U)}$$

- By using Monte Carlo integration:

$$\langle O \rangle \approx \frac{1}{\mathrm{M}} \sum_{i=1}^{\mathrm{M}} O(U_i), \qquad U_i \sim p_0(U) = \frac{e^{-S_E(U)}}{Z}$$

- Develop diffusion models for these problems to possibly speed up calculations in the future.
  - For now, demonstrate that they can be applied

# Introduction to diffusion models

- Generative machine learning method
- Most famous for image generation
- Based on stochastic processes
- Draw random samples from complicated probability distribution

This summer, he got invited to a Discord chat server where people were testing Midjourney, which uses a complex process known as "diffusion" to turn text into custom images. Users type a series of words in a message to Midjourney; the bot spits back an image seconds later.

The New York Times

THE SHIFT

## *An A.I.-Generated Picture Won an Art Prize. Artists Aren't Happy.*

"I won, and I didn't break any rules," the artwork's creator says.

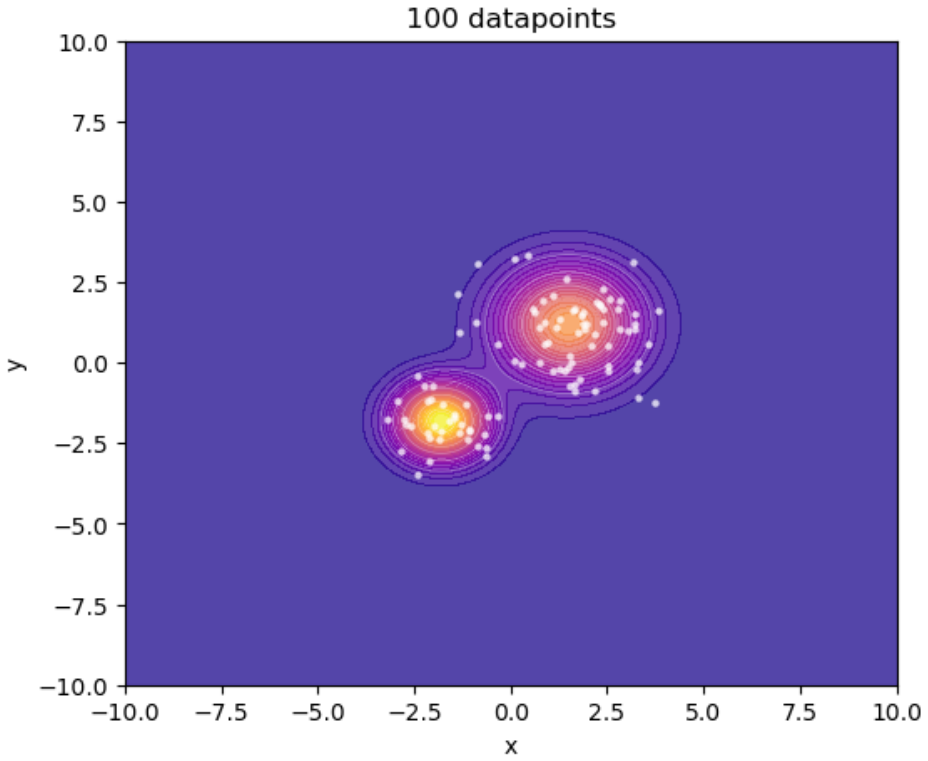


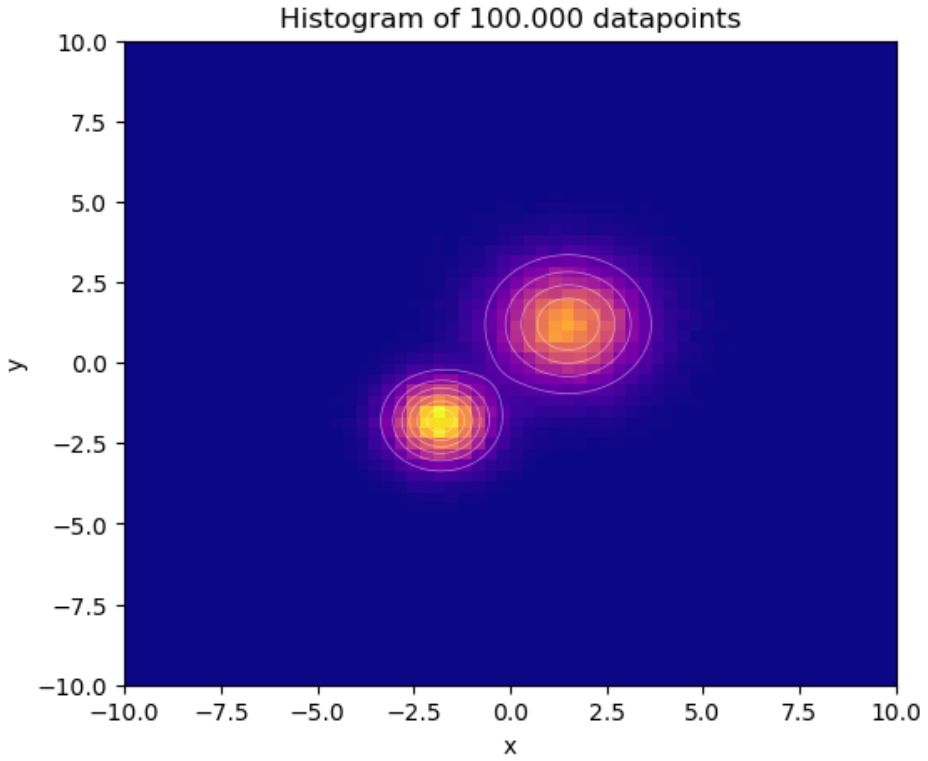
Jason Allen's A.I.-generated work, "Théâtre D'opéra Spatial," took first place in the digital category at the Colorado State Fair. via Jason Allen

By Kevin Roose

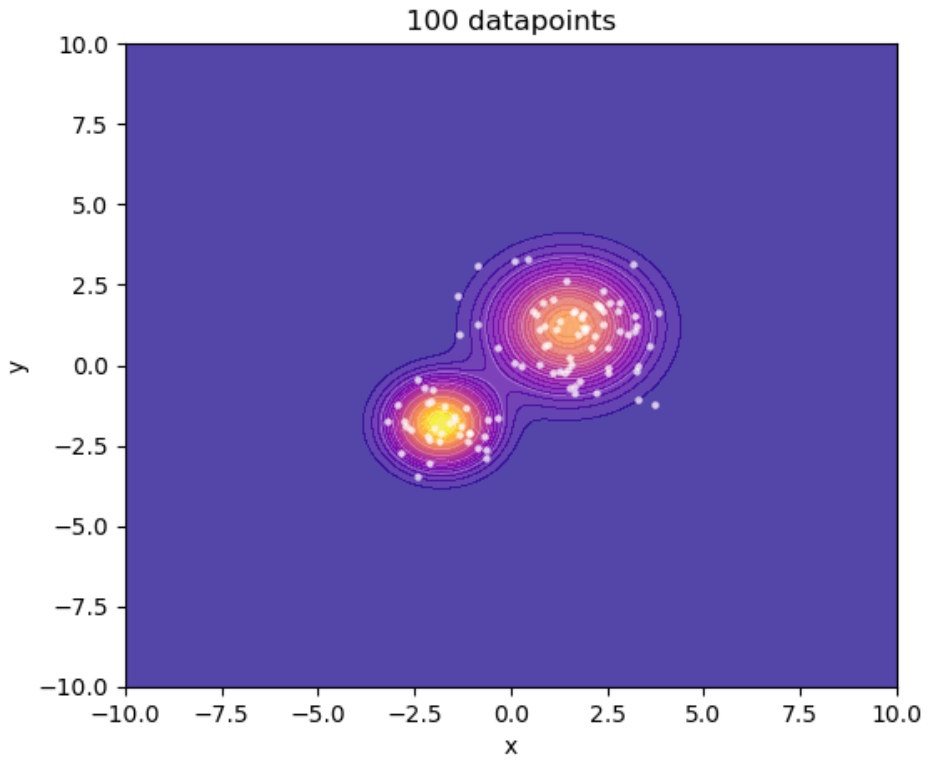
Sept. 2, 2022

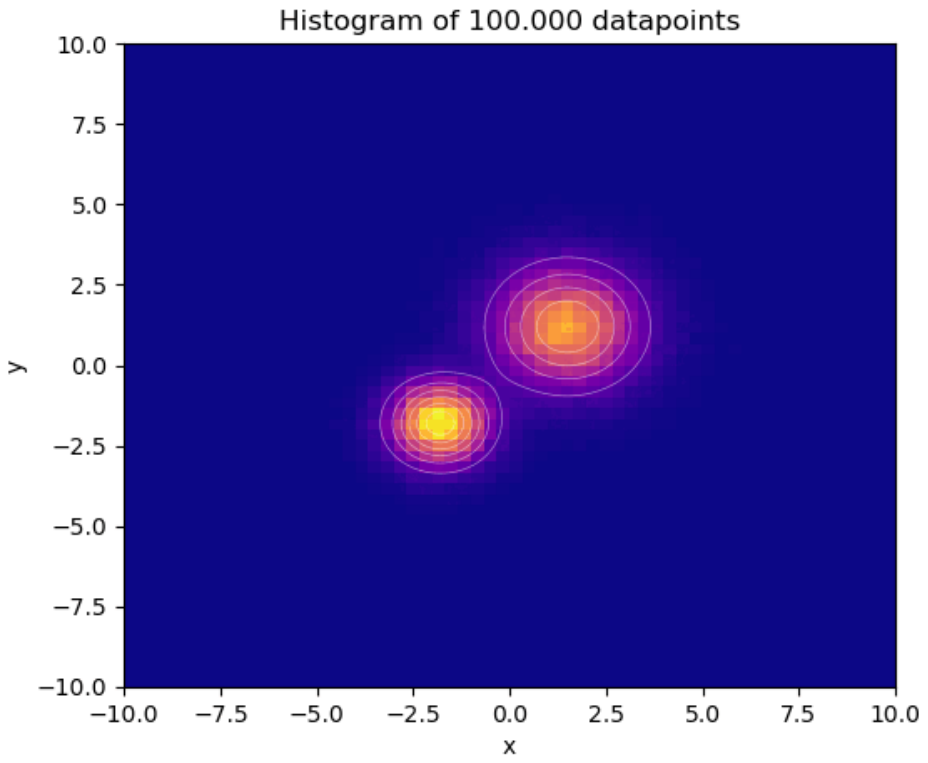
# Introduction to diffusion models



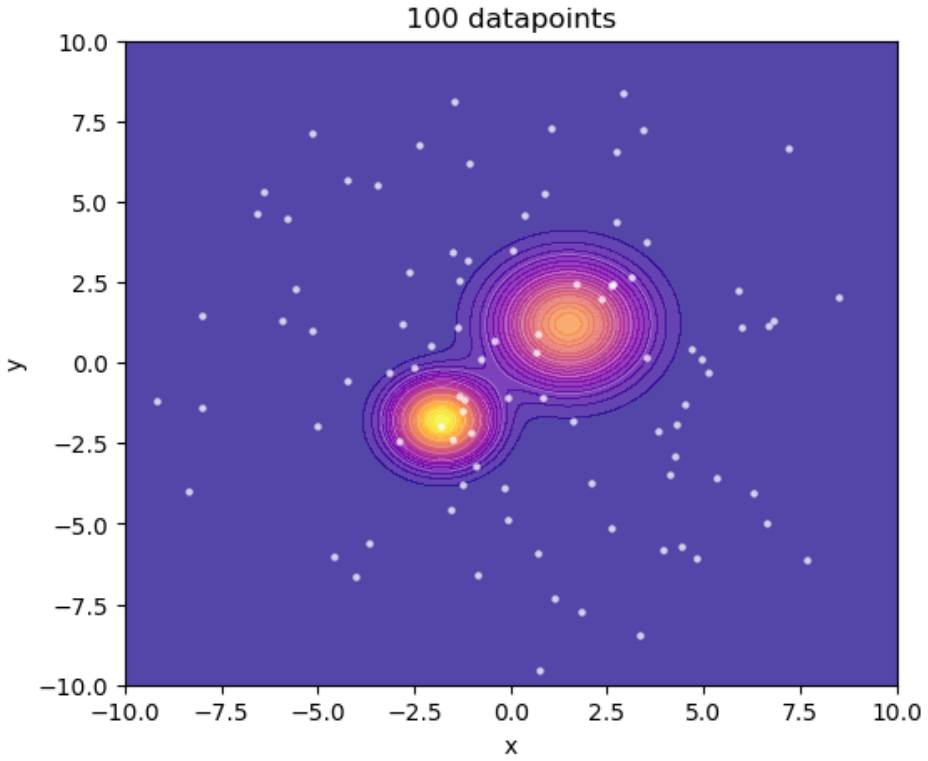



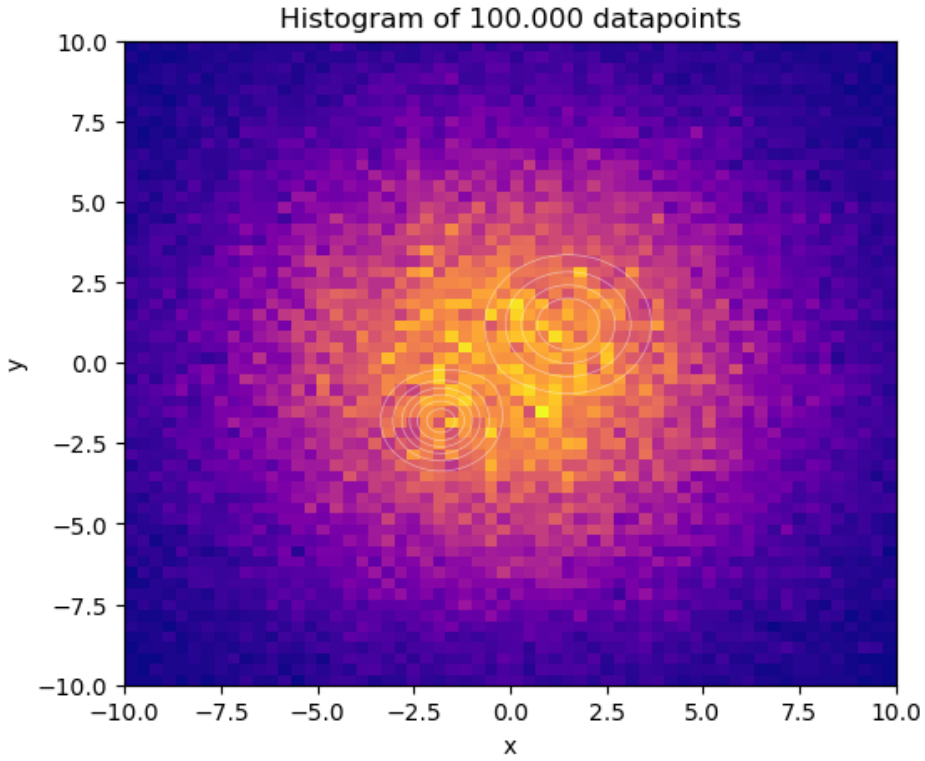
# Introduction to diffusion models



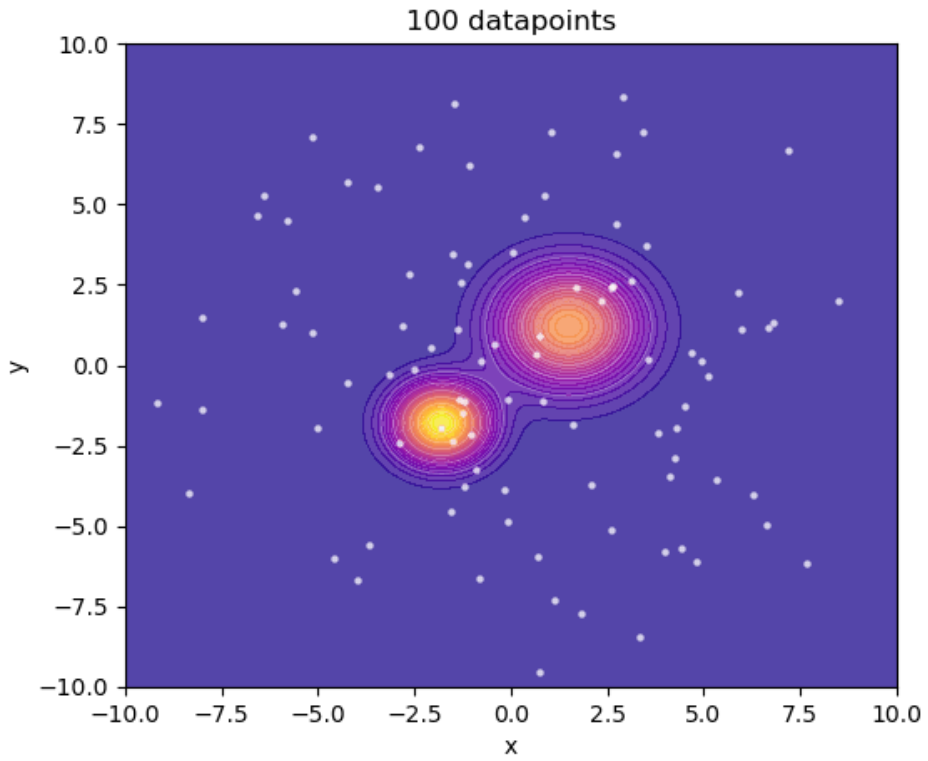



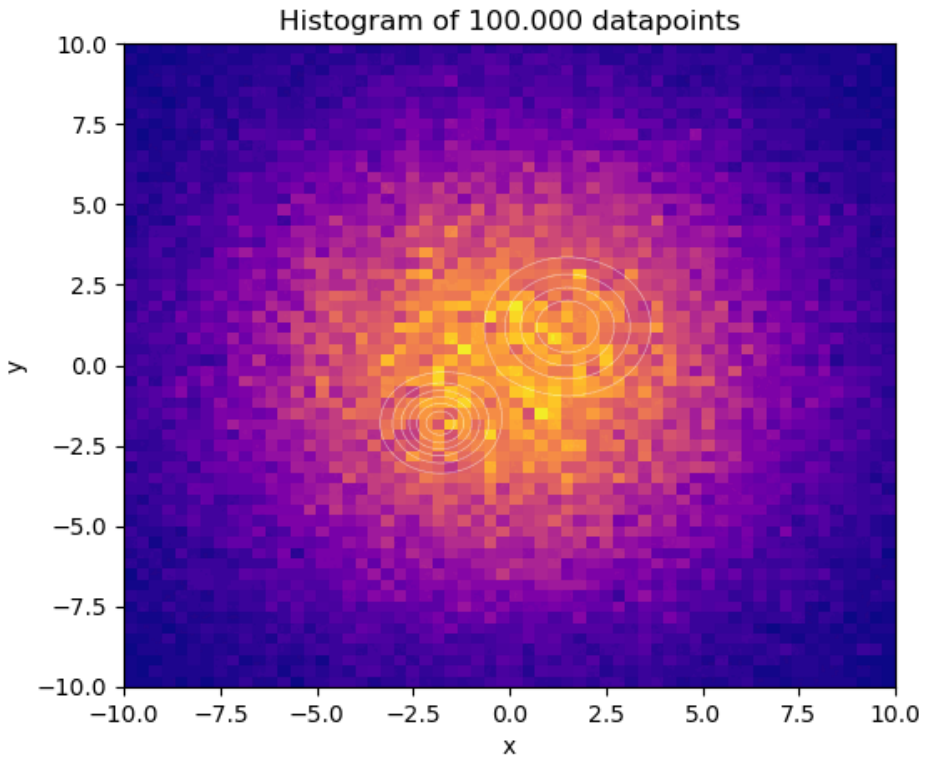
# Introduction to diffusion models



# Introduction to diffusion models



# Diffusion models for non-Abelian gauge theories

- Diffusion on group elements of U(N) or SU(N)
- Complex unitary N×N matrices

$$U = \exp\left(i \sum_j a_j T_j\right)$$

Group element

Scalar coefficients

Generators of the group

e.g. for U(2):

$$\frac{1}{2}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

$$\frac{1}{2}\begin{pmatrix} 0 & -\mathrm{i} \\ \mathrm{i} & 0 \end{pmatrix}$$

$$\frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

$$\frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

- Construct diffusion process that stays in the group

# Diffusion models for non-Abelian gauge theories

- Stratonovich stochastic differential equation (SDE):

$$\mathrm{d}U_t = i\sum_a T^a \left(K^a(U_t)\mathrm{d}t + g(t)\mathrm{d}W_t^a\right) \circ U_t$$

Drift term

Diffusion term

- Reverse process:

$$\mathrm{d}U_t = i\sum_a T^a \left[\left(K^a\left(U^{(t)}\right) - g(t)^2 D^a \ln p_t\left(U^{(t)}\right)\right)\mathrm{d}t + g(t)\mathrm{d}\widetilde{W}_t^a\right] \circ U_t$$

Score function
In general unknown, will be learned by a neural network.

Trained using samples from the target distribution

Derivation via Fokker-Planck equation

# Non-Abelian lattice gauge theory

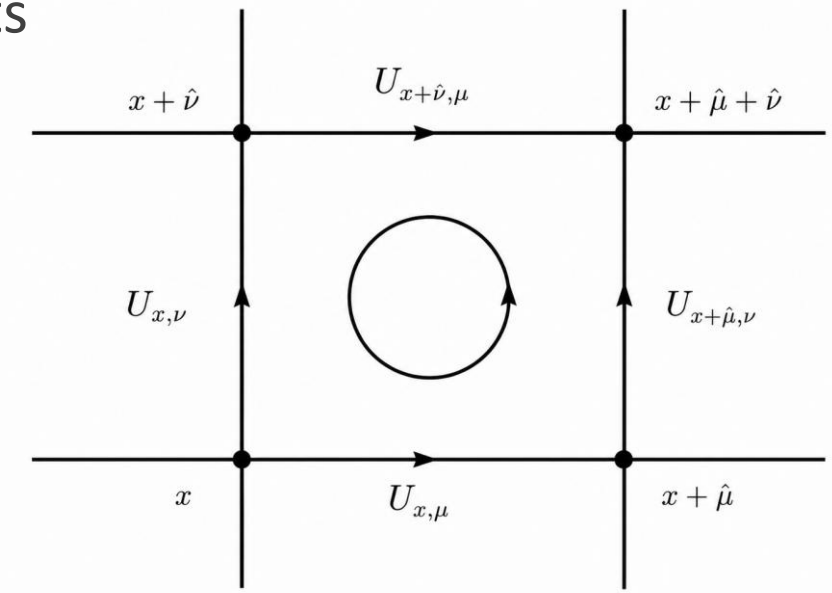
- Discretization of the gauge field on a Hypercubic lattice
- Link variables $U_{x,\mu}$ as connections between lattice points
- Links are elements of the gauge group
- Plaquette, elemental closed loop:

$$\mathrm{P}_{x,\mu\nu} = U_{x,\mu} U_{x+\hat{\mu},\nu} U^{\dagger}_{x+\hat{\nu},\mu} U^{\dagger}_{x,\nu}$$

- Wilson action:

$$S_E(U) = -\beta \sum_{x,\mu<\nu} \mathrm{Re}\,\mathrm{Tr}\left[\mathrm{P}_{x,\mu\nu}\right]$$

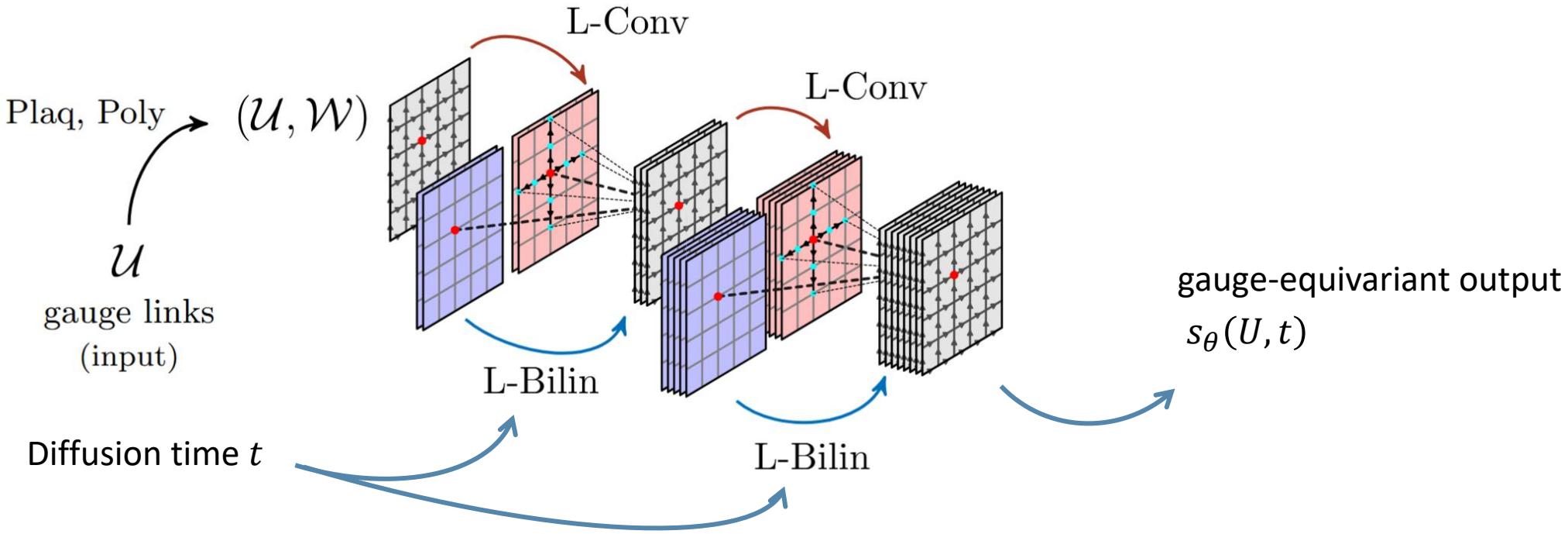
  - $\beta$… inverse coupling constant




Based on: Gattringer and Lang, Quantum chromodynamics on the lattice

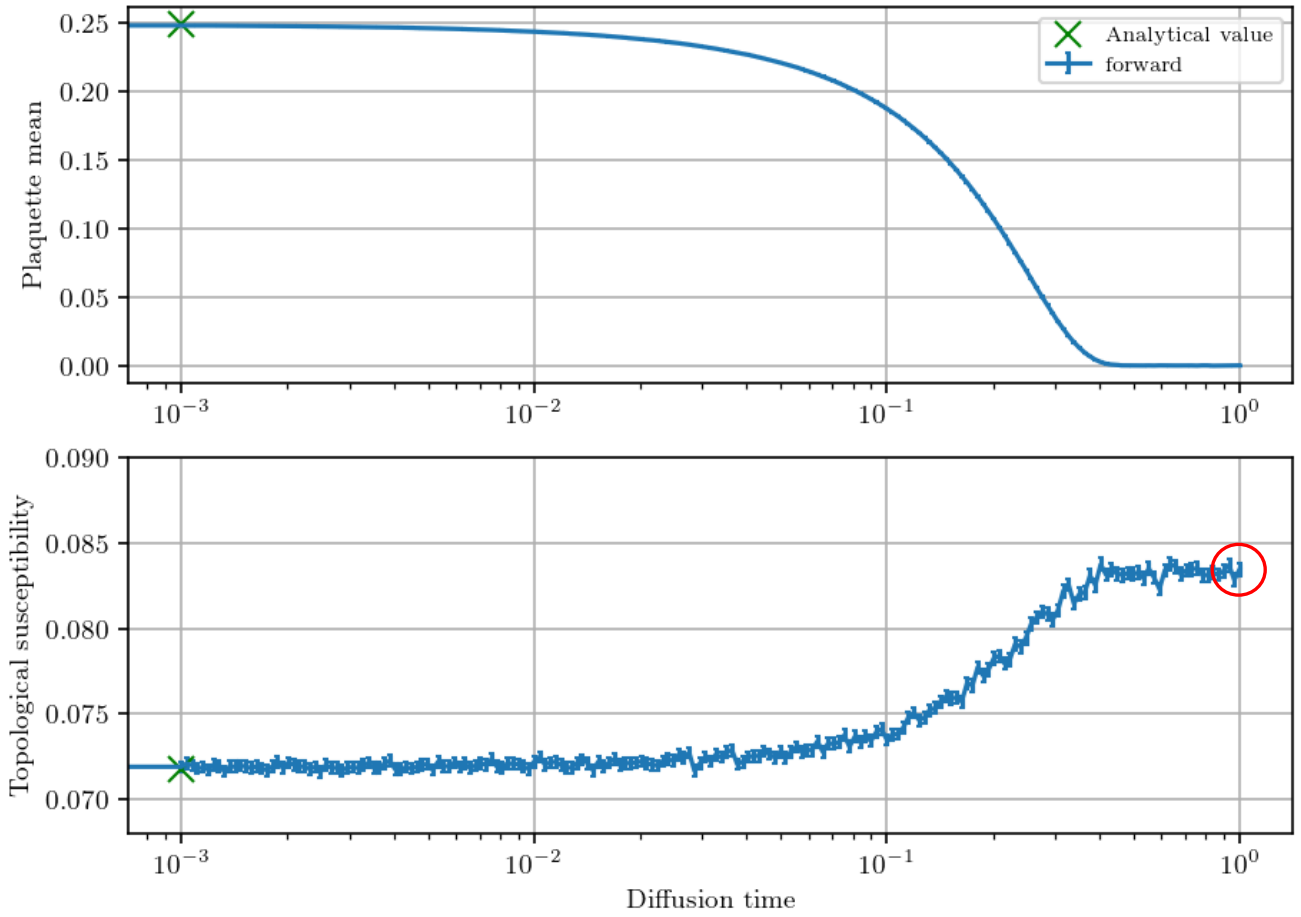
# Lattice gauge-equivariant convolutional neural networks (L-CNNs)

- Adaption of convolutional NNs to lattice gauge theories



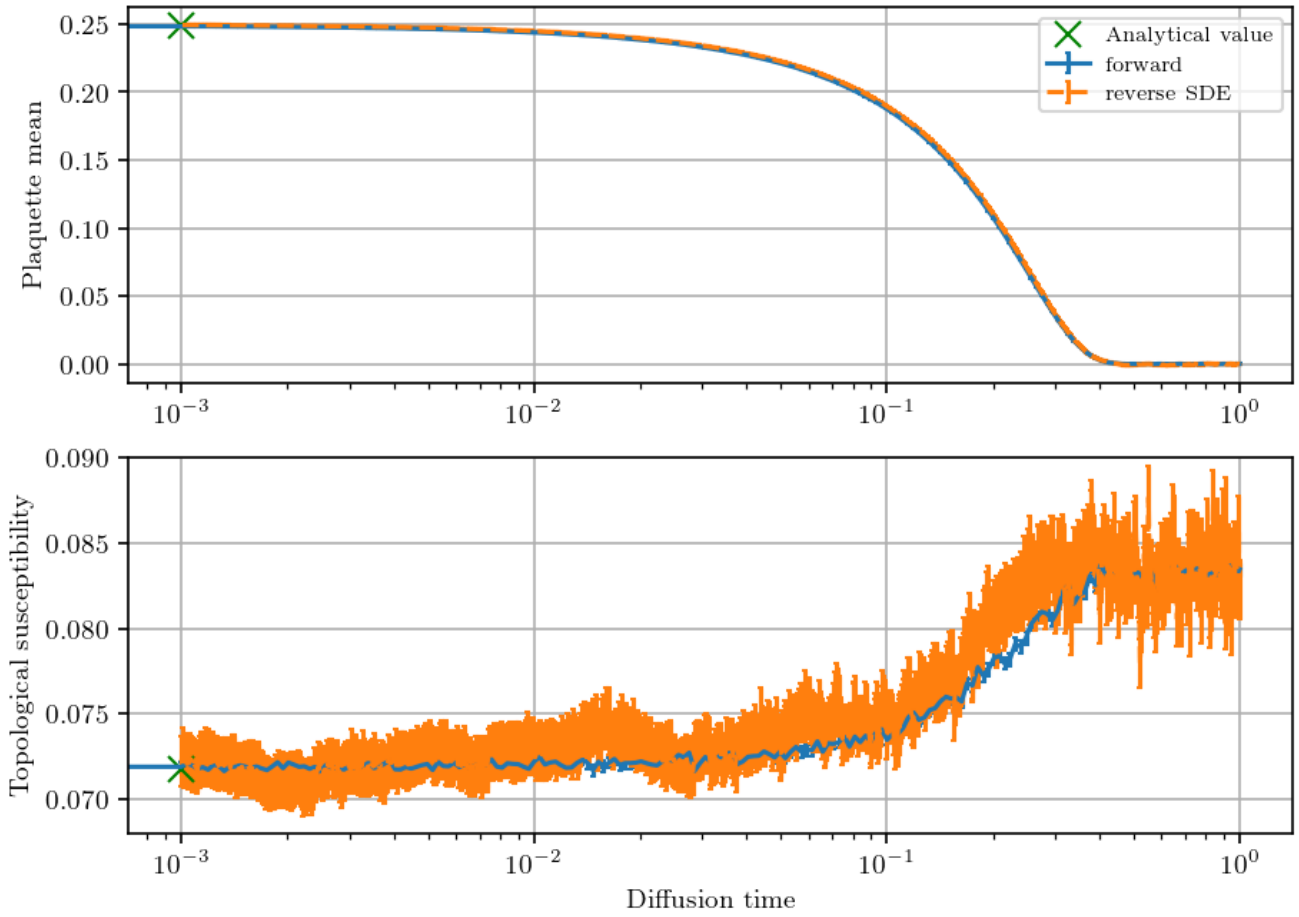

Source: Favoni et al., “Lattice Gauge Equivariant Convolutional Neural Networks”

# Reverse SDE results for 2D U(2)




$\frac{1}{12}$ …strong coupling limit

# Reverse SDE results for 2D U(2)



# But why?

- Goal is to draw configurations from a probability distribution
- But we need configurations to train the diffusion model
- Generate more configurations faster
  - For images it has been shown that the DM learns the underlying probability distribution
- Extrapolate to different lattice sizes
- Interpolate/extrapolate to different couplings
  - Two options: Learn the coupling dependence (not done yet)
  - Alternative sampling method MAALA – advantage: Metropolis adjustment for exact results

For 5.000 samples:
Reverse SDE: 6min
HMC: 55min

# Stochastic Quantization (SQ)

- Langevin equation:

$$\mathrm{d}U_\tau = i\sum_a T^a\left(-\mathrm{D}^a S_E(U_\tau)\mathrm{d}\tau + \sqrt{2}\mathrm{d}W_\tau^a\right)\circ U_\tau$$

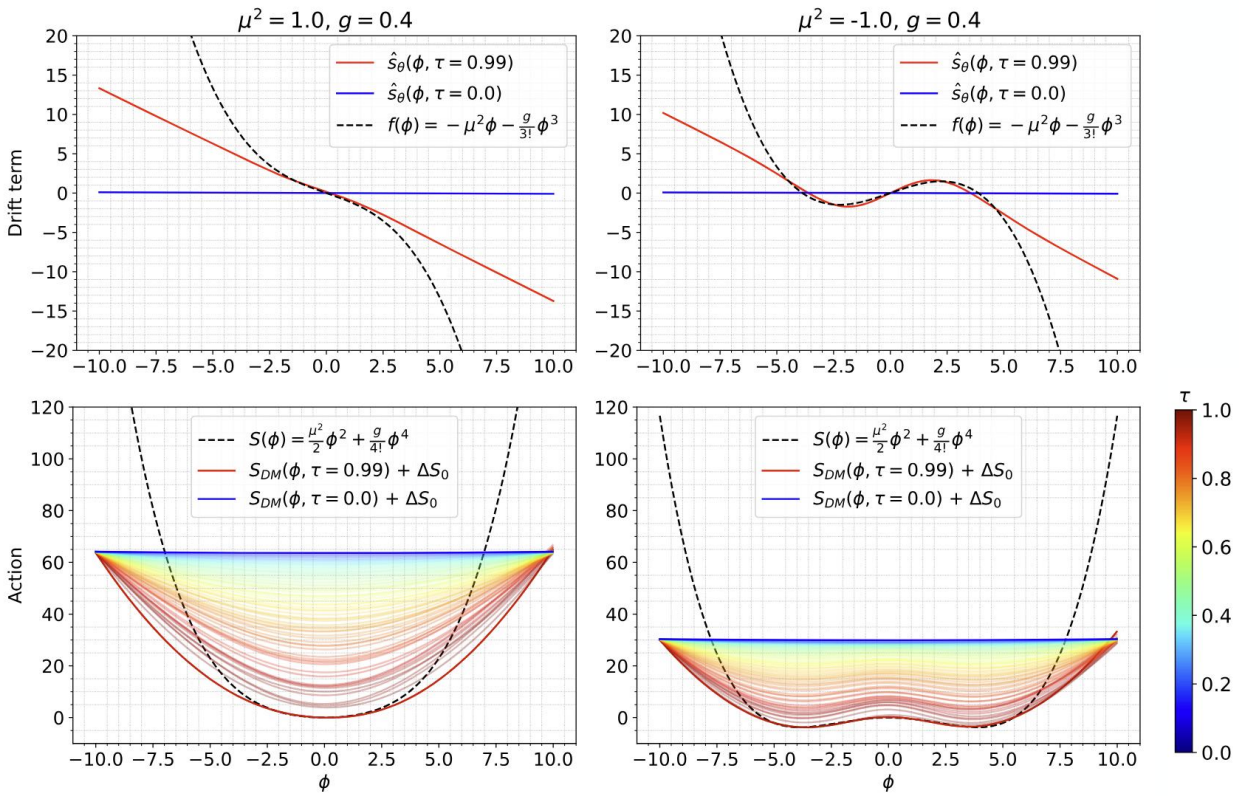
- For $\tau \to \infty$ configurations $U_\tau$ will follow

$$p(U) = \frac{1}{Z}e^{-S_E(U)} \xrightarrow{D^a \ln} -D^a S_E(U)$$

- Score function $s^a(U) = D^a \ln\big(p(U)\big)$ is the drift of SQ

# Score function

- Time dependent score-function
- Learns drift of some evolving action along the diffusion process
- Run SQ for a number of intermediate actions
- Annealed sampling
  - Gradually changing the coupling constant during sampling

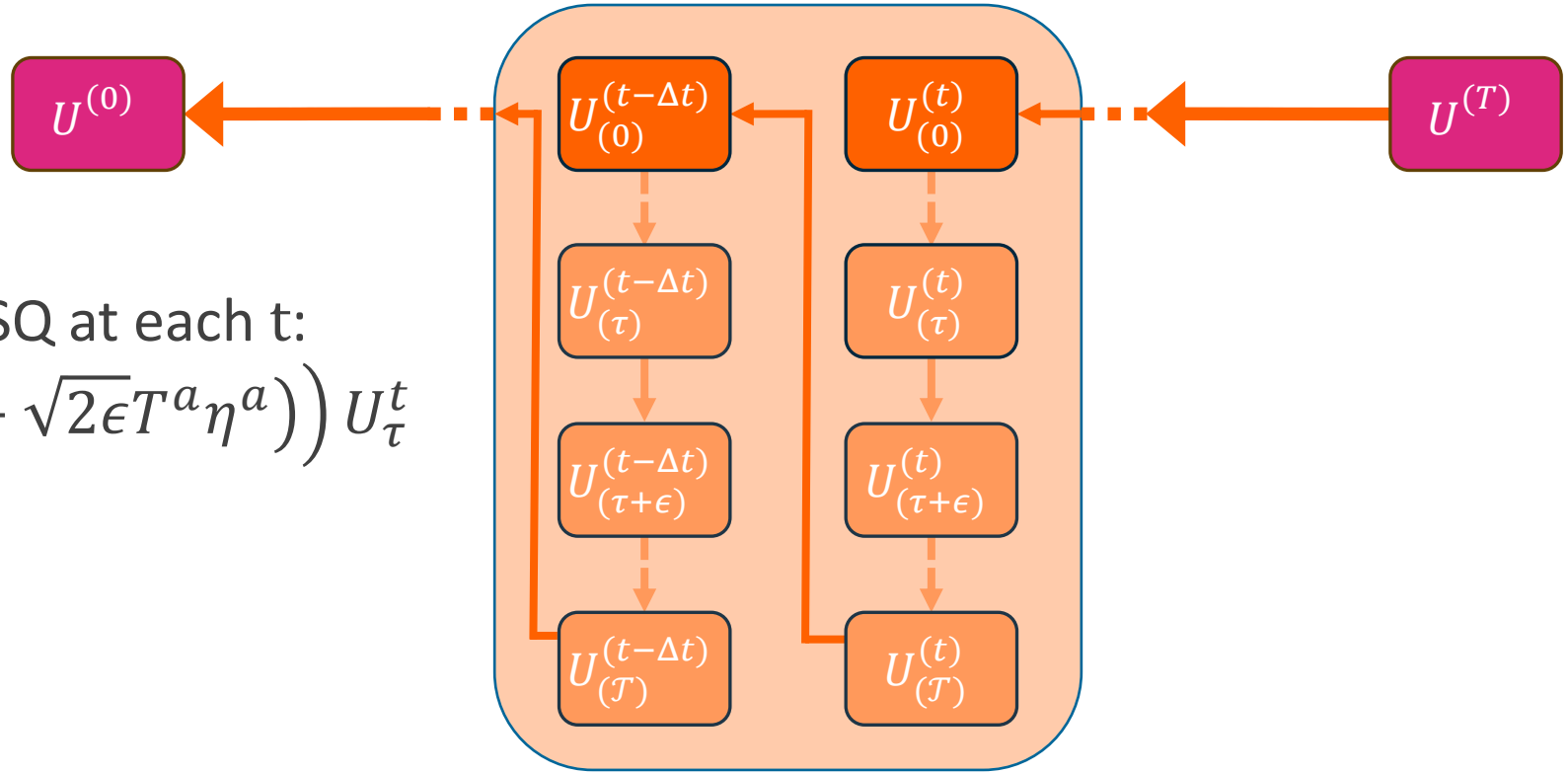



Source: Wang, Aarts, and Zhou, “Diffusion models as stochastic quantization in lattice field theory”

# Metropolis adjusted annealed Langevin (MAALA)

- On a diffusion time grid, run SQ at each t:

$$U_{\tau+\epsilon}^{t} = \exp\left(i\left(-\epsilon s_\theta(U_\tau^t, \mathrm{t}) + \sqrt{2\epsilon}T^a\eta^a\right)\right)U_\tau^t$$



# Metropolis adjusted annealed Langevin (MAALA)

- This diffusion process traces the same path as the reverse SDE
- What is the advantage?
- Scaling with beta

$$U_{\tau+\epsilon}^{t} = \exp\left( i\left( -\epsilon \frac{\beta}{\beta_0} \mathrm{s}_\theta(U_\tau^t, t) + \sqrt{2\epsilon} T^a \eta^a \right) \right) U_\tau^t$$

$$\frac{\beta}{\beta_0} s_\theta^a(U, t=0) = -\frac{\beta}{\beta_0} D^a S_E(\beta_0; U) = -D^a S_E(\beta; U)$$

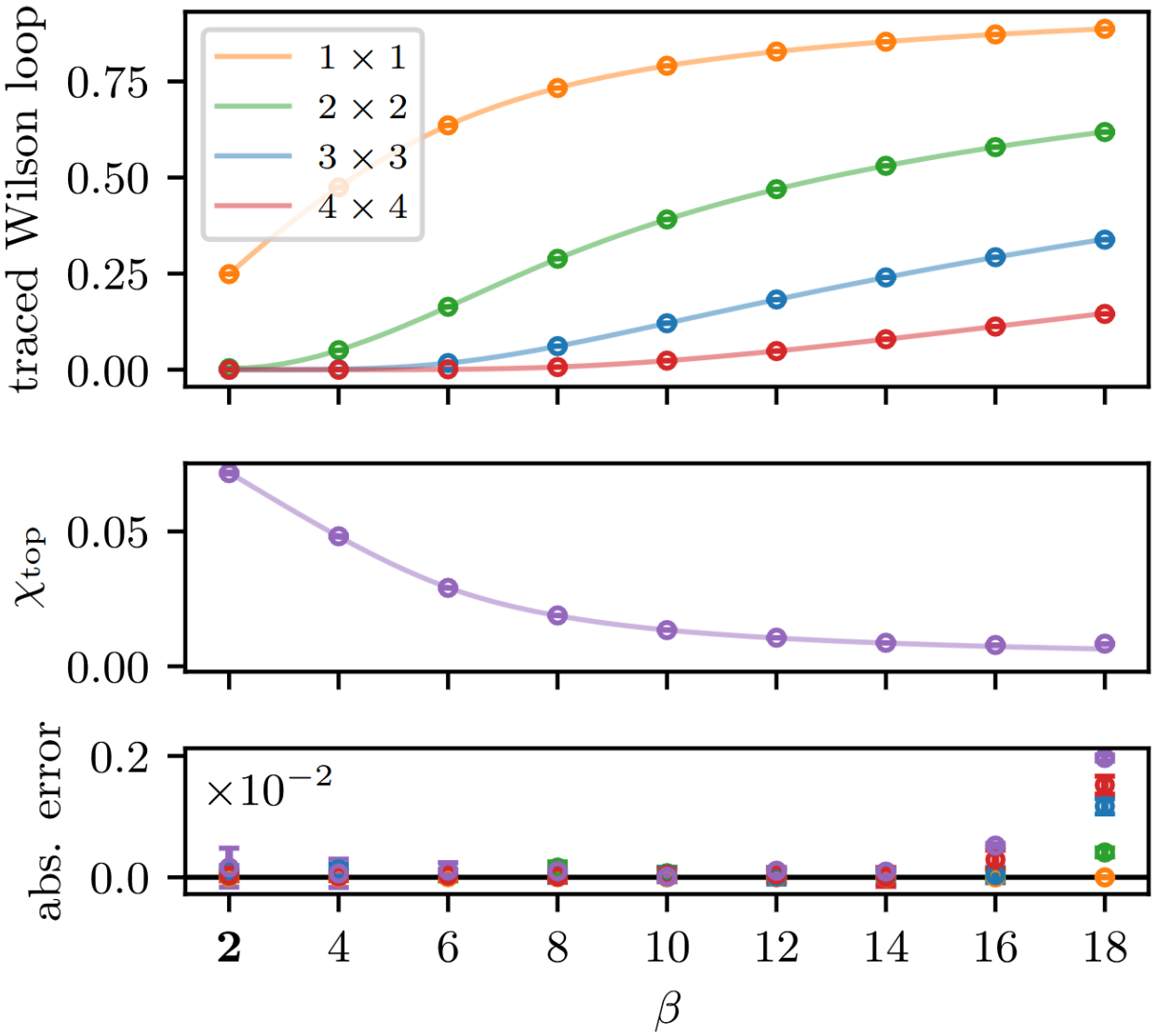
- Metropolis adjustment

# Training setup for 2D U(2)

- $16 \times 16$ lattice
- Using HMC, generate 100.000 training configurations at $\beta = 2$
- L-CNN with two bilinear-convolution layers (15.266 trainable parameters)
- Generate 10.000 configurations at several $\beta's$

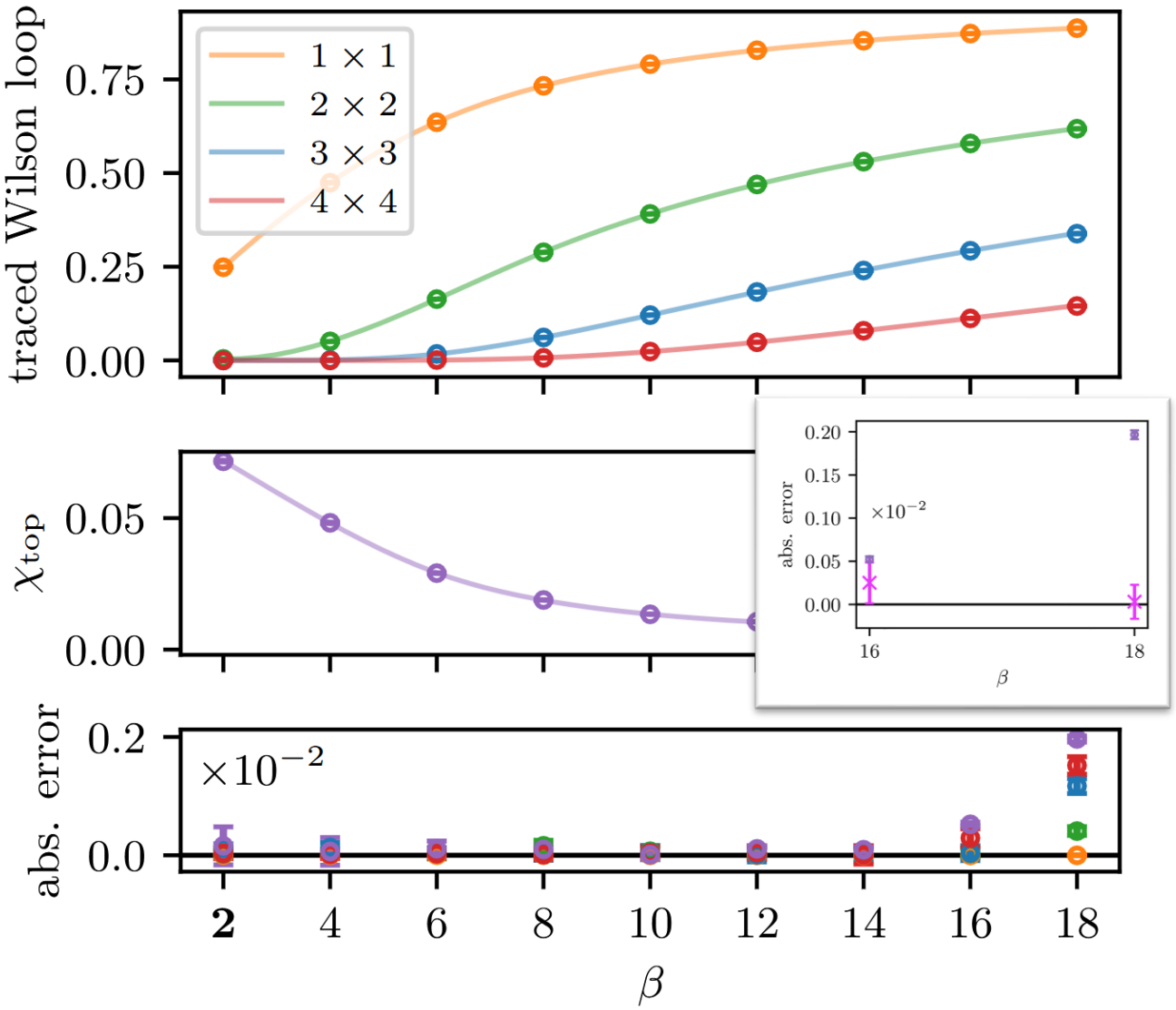
# Results for 2D U(2)

- Extrapolation to higher $\beta$
- Fixed number of diffusion steps
- Excellent agreement up to $\beta = 14$
- Extrapolation to bigger lattices ($L = 64$) shows similar performance



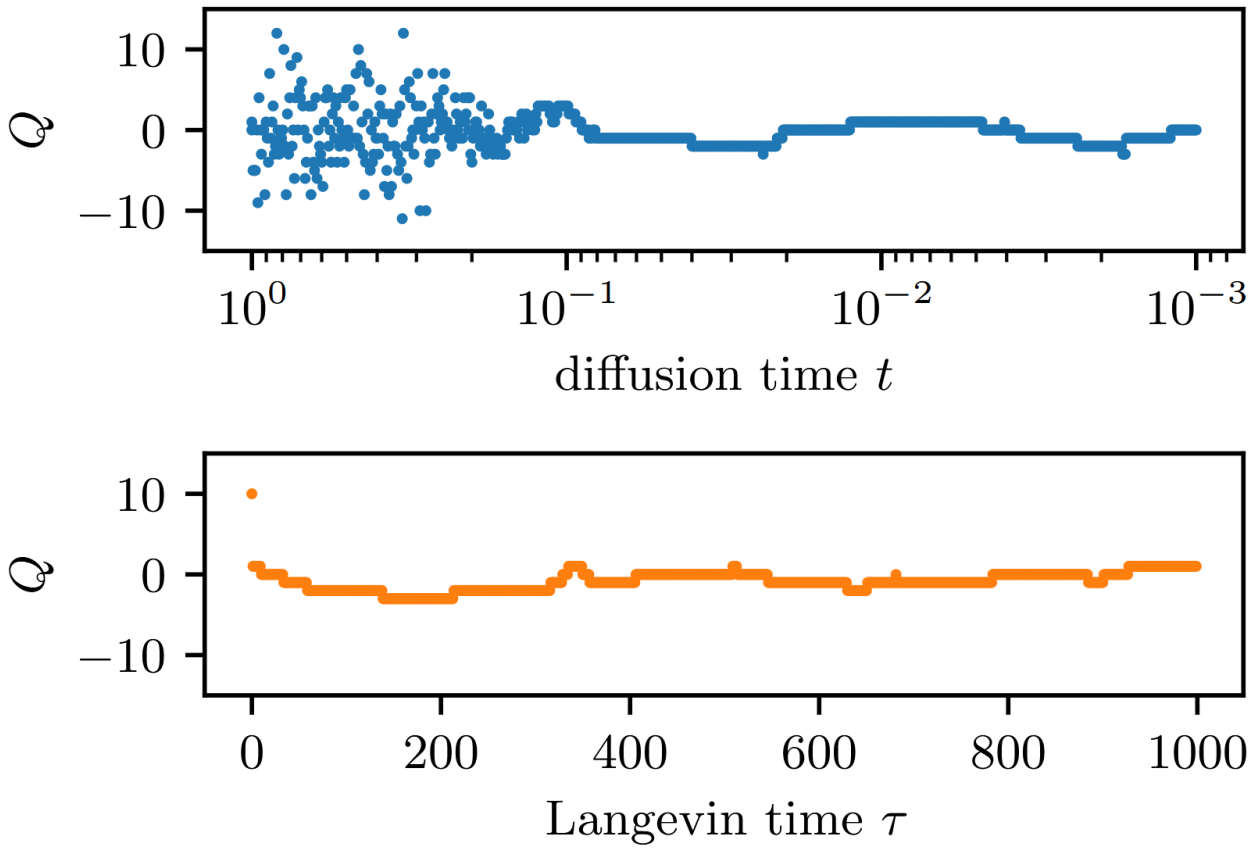
# Results for 2D U(2)

- Extrapolation to higher $\beta$
- Fixed number of diffusion steps
- Excellent agreement up to $\beta = 14$
- Extrapolation to bigger lattices ($L = 64$) shows similar performance
- Perfect results at $\beta = 18$ require 10 times longer diffusion time



# Results for 2D U(2)

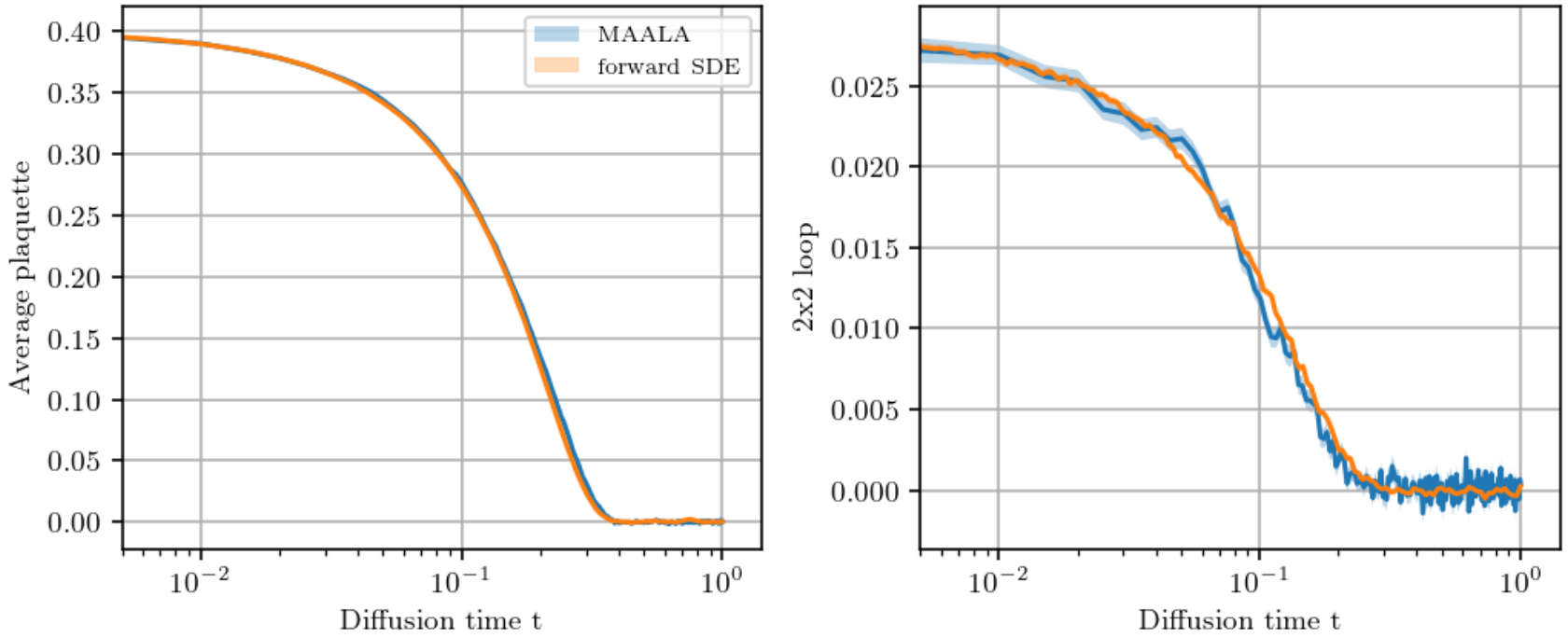
- Topological freezing
  - Markov Chain getting stuck in one topological sector
  - Appears towards high $\beta$
  - Hinders correct sampling over the whole phase space
- Annealing allows for more exploration in the initial sampling phase



# First results for 4D SU(3)

- On a $4^4$ lattice
- $\beta = 5$

PRELIMINARY



# Summary

- Adaptation of diffusion models to non-Abelian lattice gauge theories
- Alternative sampling method MAALA
  - Extrapolation to different couplings
  - Metropolis adjustment
- Excellent extrapolation performance for 2D U(2)
- First promising results for 4D SU(3)

# Thank you for your attention!

# Backup slides

# Introduction to diffusion models

- Stochastic differential equation describing the diffusion process:

$$\mathrm{d}x = f(x,t)\mathrm{d}t + g(t)\mathrm{d}w$$

Drift term

Diffusion term

Derivation via Fokker-Planck equation

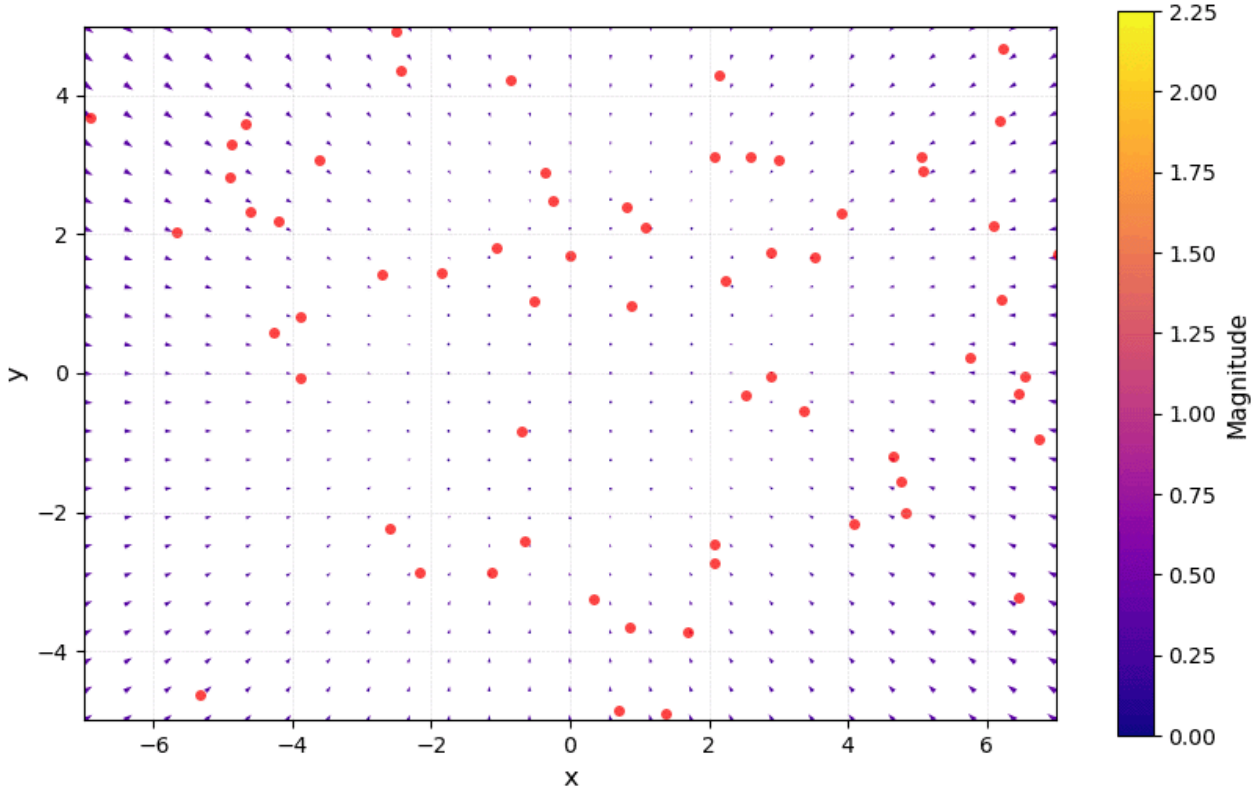
- Corresponding reverse process:

$$\mathrm{d}x = [f(x,t) - g(t)^2 \nabla_x \log p_t(x)]\mathrm{d}t + g(t)\mathrm{d}\bar{w}$$

Score function
In general unknown, will be learned by a neural network.

# Introduction to diffusion models

- Score function:



# Non-Abelian lattice gauge theory

- Observables we look at:
- Average traced square loops:

$$W_{1\times 1} = \left\langle \frac{1}{N} \mathrm{Re}\ \mathrm{Tr}\left(P_{x,\mu\nu}\right) \right\rangle$$

$n \times n$ loops analogously

- Topological susceptibility (for 2D N=2):

$$\chi_{\mathrm{top}} = \frac{\langle Q^2 \rangle}{V}, \qquad Q = \frac{1}{2\pi} \sum_x \arg\left(\det P_{x,01}\right)$$

# Diffusion models for non-Abelian gauge theories

- Solving the SDE numerically:

$$U_{t+\epsilon} = \exp\left( i \sum_a T^a [\epsilon K^a(U_t) + \sqrt{\epsilon} g(t) \eta^a] \right) U_t$$

$$\eta^a \sim \mathcal{N}(0,1)$$

# Diffusion models for non-Abelian gauge theories

- To actually solve the reverse SDE:
  - approximate the score function with a neural network $s_\theta^a(U,t)$
- Train with samples from the target distribution, minimizing the loss function

$$\int_0^T dt \, E_{p_0(U_0)} E_{p_{0t}(U_t|U_0)} \sum_a [s_\theta^a(U_t,t) - D^a \ln p_{0t}(U_t|U_0)]^2$$

- Training algorithm:

Pick a random batch of training samples → Pick a random batch of diffusion times from $\mathcal{U}(0,1)$ → Run diffusion process on samples until t → Calculate difference and update network

# Diffusion models for non-Abelian gauge theories

- The actual SDE we use:

$$\mathrm{d}U_t = i \sum_a T^a \, s^t \, \mathrm{d}W_t^a \circ U_t, \qquad \mathrm{s} = 25$$

- Time interval $t \in [0,1]$
- Trick to solve it in one step:

$$U_t = \exp\left( i\sigma(\mathrm{t}) \sum_a T^a \eta^a \right) U_0, \qquad \sigma(t) = \sqrt{(s^{2t} - 1)/(2\ln(s))}$$

- Analytical expression for the grad log of the transition probability:

$$D^a \ln p_{0t}(U_t|U_0) = -\frac{1}{\sigma(t)} \eta^a$$

# Metropolis adjustment

- At t close to zero, apply Metropolis adjustment
- Langevin step proposes $\widetilde{U_\tau^t}$ given $U_\tau^t$, accept with

$$p_{\text{accept}} = \min\left\{1, \frac{p\left(\widetilde{U_\tau^t}\right)}{p(U_\tau^t)} \frac{q\left(U_\tau^t \middle| \widetilde{U_\tau^t}\right)}{q\left(\widetilde{U_\tau^t} \middle| U_\tau^t\right)}\right\}$$

$p$… target distribution
$q$… transition probability of the diffusion process

$$\exp\left(S_E(U_\tau^t) - S_E\left(\widetilde{U_\tau^t}\right)\right)$$

$$\exp\left(\sum_{x,\mu}\left[\mathrm{Tr}\left(\widehat{\eta_{x,\mu}^2}\right) - \mathrm{Tr}\left(\widehat{\eta_{x,\mu}'^2}\right)\right]\right)$$

$$\widehat{\eta_{x,\mu}} = \frac{1}{i\sqrt{2\epsilon}} \ln\left(\widetilde{U_{\tau;x,\mu}^t} U_{\tau;x,\mu}^{t\dagger}\right) - \sqrt{\frac{\epsilon}{2}} \sum_a T^a s_{\theta;x,\mu}\left(U_{\tau;x,\mu}\right)$$

# Lattice gauge-equivariant convolutional neural networks (LCNNs)

- Input: Links $U$ and Plaquettes $P \rightarrow W$
  - Plus their complex conjugates and unit matrices
- Bilinear convolution layer:

$$\widehat{W_x'^l} = \sum_{k,\mu,m,n} \omega_{k,\mu}^{lmn} \widehat{W_x^m} U_{x,k\hat{\mu}} \widehat{W_{x+k\hat{\mu}}^n} U_{x,k\hat{\mu}}^{\dagger}$$

Ordered product of links starting at $x$ and ending at $x + k\hat{\mu}$

# Lattice gauge-equivariant convolutional neural networks (LCNNs)

- Input: Links $U$ and Plaquettes $P \rightarrow W$
  - Plus their complex conjugates and unit matrices
- Bilinear convolution layer:

$$\widehat{W_x'^l} = \sum_{k,\mu,m,n} \omega_{k,\mu}^{lmn} \widehat{W_x^m}\, U_{x,k\hat{\mu}} \widehat{W_{x+k\hat{\mu}}^n} U_{x,k\hat{\mu}}^{\dagger}$$

Trainable weights

# Lattice gauge-equivariant convolutional neural networks (LCNNs)

- Input: Links $U$ and Plaquettes $P \rightarrow W$
  - Plus their complex conjugates and unit matrices
- Bilinear convolution layer:

$$\widehat{W'^{l}_{x}} = \sum_{k,\mu,m,n} \omega^{lmn}_{k,\mu} \widehat{W^{m}_{x}}\, U_{x,k\hat{\mu}} \widehat{W^{n}_{x+k\hat{\mu}}} U^{\dagger}_{x,k\hat{\mu}}$$

$l \ldots$ output channels

# Lattice gauge-equivariant convolutional neural networks (LCNNs)

- Input: Links $U$ and Plaquettes $P \rightarrow W$
  - Plus their complex conjugates and unit matrices
- Bilinear convolution layer:

$$\widehat{W_x'^l} = \sum_{k,\mu,\boldsymbol{m},\boldsymbol{n}} \omega_{k,\mu}^{l\boldsymbol{mn}} \widehat{W_x^{\boldsymbol{m}}}\, U_{x,k\hat{\mu}} \widehat{W_{x+k\hat{\mu}}^{\boldsymbol{n}}} U_{x,k\hat{\mu}}^{\dagger}$$

m,n… input channels

# Lattice gauge-equivariant convolutional neural networks (LCNNs)

- Input: Links $U$ and Plaquettes $P \rightarrow W$
  - Plus their complex conjugates and unit matrices
- Bilinear convolution layer:

$$\widehat{W_x'^l} = \sum_{\boldsymbol{k},\mu,m,n} \omega_{\boldsymbol{k},\mu}^{lmn} \widehat{W_x^m}\, U_{x,\boldsymbol{k}\hat{\mu}}\, \widehat{W_{x+\boldsymbol{k}\hat{\mu}}^n}\, U_{x,\boldsymbol{k}\hat{\mu}}^{\dagger}$$

$-(K-1) \leq k \leq K-1$

$K$... Kernel size

# Lattice gauge-equivariant convolutional neural networks (LCNNs)

- Input: Links $U$ and Plaquettes $P \rightarrow W$
  - Plus their complex conjugates and unit matrices
- Bilinear convolution layer:

$$\widehat{W_x'^l} = \sum_{k,\mu,m,n} \omega_{k,\mu}^{lmn} \widehat{W_x^m}\, U_{x,k\hat{\mu}}\, \widehat{W_{x+k\hat{\mu}}^n}\, U_{x,k\hat{\mu}}^{\dagger}$$

$\mu$ … Lattice direction

# Lattice gauge-equivariant convolutional neural networks (LCNNs)

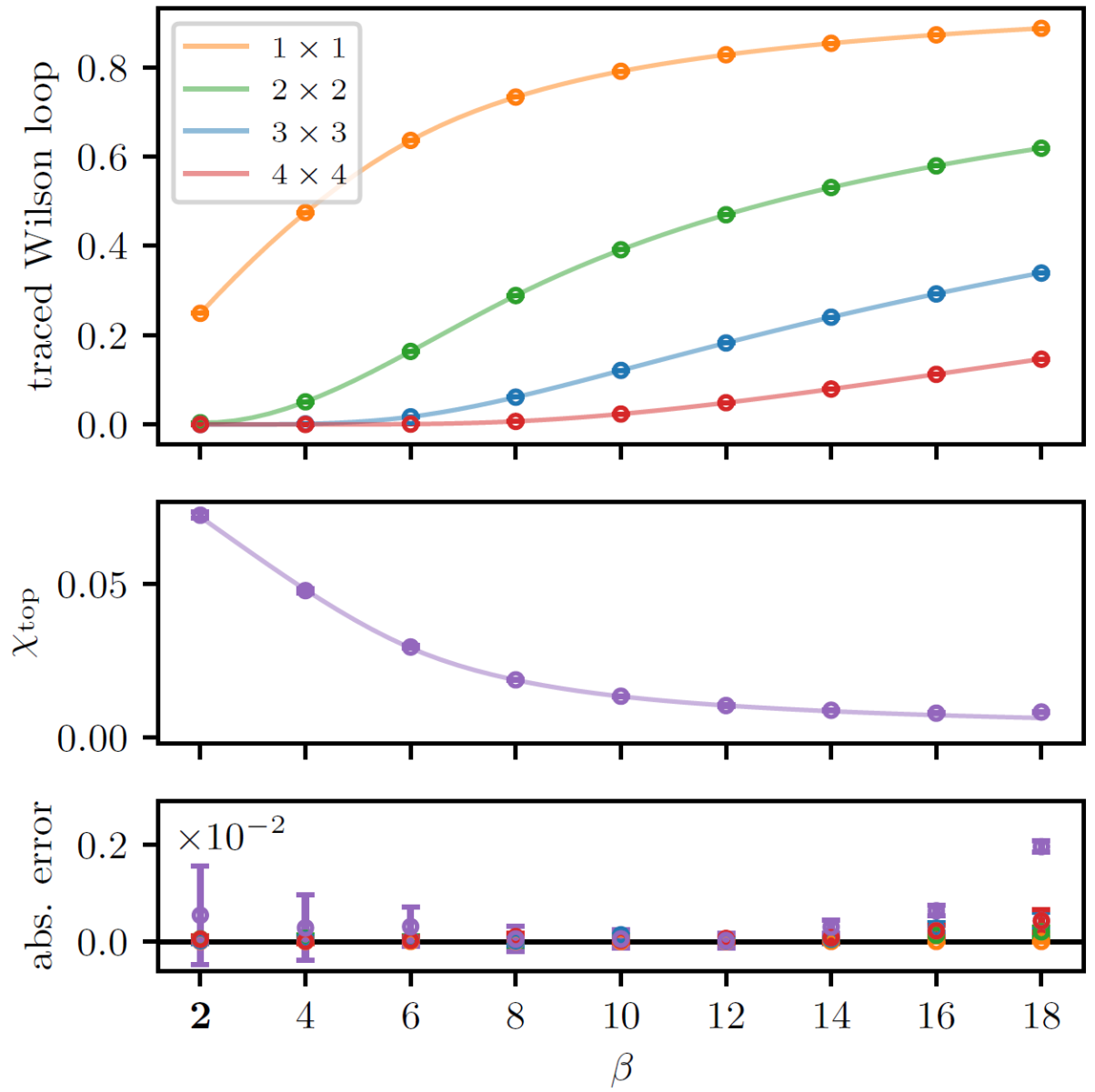
- Time embedding by random Fourier features

$$\widehat{W_x'^l} = \widehat{W_x^l} + \mathbb{I} f^l(t)$$

Time embedding vector

# Results for 2D U(2)

- Extrapolation to a larger lattice
- Model trained at $L = 16$
- Generating configurations at $L = 64$



# Results for 2D U(2)

- Metropolis acceptance rates