Flow-based generative models for Markov chain Monte Carlo in lattice field theory

M. S. Albergo,[1,2,3] G. Kanwar,[4] and P. E. Shanahan[4,1]

[1]*Perimeter Institute for Theoretical Physics, Waterloo, Ontario N2L 2Y5, Canada*
[2]*Cavendish Laboratories, University of Cambridge, Cambridge CB3 0HE, United Kingdom*
[3]*University of Waterloo, Waterloo, Ontario N2L 3G1, Canada*
[4]*Center for Theoretical Physics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA*

(Received 17 May 2019; published 22 August 2019; corrected 21 November 2019)

A Markov chain update scheme using a machine-learned flow-based generative model is proposed for Monte Carlo sampling in lattice field theories. The generative model may be optimized (trained) to produce samples from a distribution approximating the desired Boltzmann distribution determined by the lattice action of the theory being studied. Training the model systematically improves autocorrelation times in the Markov chain, even in regions of parameter space where standard Markov chain Monte Carlo algorithms exhibit critical slowing down in producing decorrelated updates. Moreover, the model may be trained without existing samples from the desired distribution. The algorithm is compared with HMC and local Metropolis sampling for ϕ^4 theory in two dimensions.

DOI: 10.1103/PhysRevD.100.034515

I. INTRODUCTION

A key problem in lattice field theory and statistical mechanics is the evaluation of integrals over field configurations, referred to as path integrals. Typically, such integrals are evaluated via a Markov chain Monte Carlo (MCMC) approach: field configurations are sampled from the desired probability distribution, dictated by the action of the theory, using a Markov chain. A significant practical concern is the existence of correlations between configurations in the chain. Critical slowing down [1] refers to the divergence of the associated autocorrelation time as a critical point in parameter space is approached. This behavior drastically increases the computational cost of simulations in these parameter regions [2,3]. For some models, algorithms have been found which significantly reduce or eliminate this slowing down [4–11], enabling efficient simulation. For field theories, a number of methods have been proposed to circumvent critical slowing down by variations of hybrid Monte Carlo (HMC) techniques [12–15], multiscale updating procedures [16–18], open boundary conditions or nonorientable manifolds [19–21], metadynamics [22], and machine-learning (ML) tools [23,24]. In important classes of theories, however, critical slowing down remains limiting; for example, in lattice formulations of quantum chromodynamics (QCD, the piece of the standard model describing the strong nuclear force) it is a major barrier to simulations at the fine lattice spacings required for precise control of the continuum limit.

Here, a new flow-based MCMC approach is proposed and is applied to lattice field generation. The resulting Markov chain has autocorrelation properties that are systematically improvable by an optimization (training) step before sampling. In this method, samples z are drawn from a simple distribution and then transformed by a change of variables (or "flow") $\phi = f^{-1}(z)$, resulting in samples ϕ with a new effective distribution $\tilde{p}_f$. The mapping f^{-1} is chosen to be efficient to compute, making it easy to draw samples ϕ, and is optimized within a variational family to produce a distribution $\tilde{p}_f$ close to the desired one. To guarantee asymptotic exactness of sampling, a Markov chain is constructed using Metropolis-Hastings steps with $\tilde{p}_f$ taken as a proposal distribution. Since proposed samples are independent of the previous samples in the chain, the autocorrelation time and acceptance rate are coupled; the autocorrelation time drops to 0 as the acceptance rate approaches 1. This is true even in regions of parameter space where standard algorithms exhibit critical slowing down. Under mild conditions (detailed in Sec. II), this approach is guaranteed to generate samples from the desired probability distribution in the limit of a large number of updates.

This method has several features that make it attractive for the evaluation of path integrals in lattice field theories.

(1) The autocorrelation time of the Markov chain can be systematically decreased by training the model.
(2) Each step of the Markov chain requires only the model evaluation and an action computation.
(3) Each update proposal is independent of the previous sample; thus proposals can be generated in parallel and efficiently composed into a Markov chain.
(4) The model is trained using samples produced by the model itself, without the need for existing samples from the desired probability distribution.

Several other machine-learning approaches have been applied to MCMC, for statistical mechanics systems, synthetic distributions, and simple lattice quantum field theories. Self-learning Monte Carlo (SLMC) methods have been applied fairly successfully to one- to three-dimensional Ising and fermionic systems. These methods construct, by a variety of techniques, an effective Hamiltonian for a theory that can be more easily sampled than the original Hamiltonian [25–29]. The effective Hamiltonian is learned using supervised learning techniques based on training data drawn from a combination of existing MCMC simulations, randomly mutated samples, and the accelerated Markov chain itself (hence the term self-learning). Flow-based methods have been used for Monte Carlo sampling in the two-dimensional Ising model [30], many-body systems [31], and synthetic distributions [32,33], and generative adversarial methods have been applied to two-dimensional scalar field theory [34,35].

In contrast to these approaches, the method proposed here focuses on directly generating samples from a close approximation to the true distribution in such a way that the exact likelihood of each produced sample is known. The direct generation allows self-learning as in SLMC, and the known likelihood allows use of a Metropolis-Hastings acceptance step to ensure exactness.

The proposed flow-based MCMC algorithm is detailed in Sec. II. A numerical study of its effectiveness in the context of two-dimensional ϕ^4 theory is presented in Sec. III. Finally, Sec. IV outlines the further development and scaling of the approach that will be required for applications to theories defined in a larger number of spacetime dimensions and to more complicated field theories such as QCD.

II. A FLOW-BASED MARKOV CHAIN MONTE CARLO ALGORITHM

In lattice field theory, a MCMC process is an efficient way to generate field configurations $\phi \in \mathbb{R}^D$ distributed according to a target probability distribution

$$p(\phi) = e^{-S(\phi)}/Z, \quad \text{with} \quad Z = \int \prod_{j=1}^{D} d\phi_j e^{-S(\phi)}, \tag{1}$$

where j indexes the D components of ϕ, $S(\phi)$ is the action that defines the theory, and Z is the partition function. Here, ϕ is defined to be a vector of D real components representing the combined internal (α) and spacetime (x) degrees of freedom of the field $\phi(x,\alpha)$ evaluated on a finite, discrete spacetime lattice (generalizations to gauge fields are discussed in Sec. IV). A MCMC process generates a chain $\phi^{(0)} \to \phi^{(1)} \to \ldots \phi^{(N)}$ by steps through configuration space starting with an arbitrary configuration $\phi^{(0)}$. The steps are stochastic and are determined by the probabilities $T(\phi, \phi')$ associated with each possible transition $\phi \to \phi'$. These probabilities must be non-negative and normalized,

$$T(\phi, \phi') \geq 0 \quad \text{and} \quad \int \prod_{j=1}^{D} d\phi'_j T(\phi, \phi') = 1. \tag{2}$$

They must also satisfy the conditions of *ergodicity* and *balance* to ensure that samples in the chain are drawn from a distribution that converges to $p(\phi)$ after thermalization. For the chain to be ergodic, it must be possible to transition from a starting configuration ϕ to any other configuration ϕ' in a finite number of steps, i.e.,

$$\exists\, n \text{ such that } T^n(\phi, \phi') > 0 \quad \text{for all } \phi, \phi', \tag{3}$$

and the chain must not have a period, for which it is sufficient that a single state has nonzero self-transition probability, i.e.,

$$\exists\, \phi \text{ such that } T(\phi, \phi) > 0. \tag{4}$$

Balance is the condition that $p(\phi)$ is a stationary distribution of the transition,

$$\int \prod_{j=1}^{D} d\phi_j p(\phi) T(\phi, \phi') = p(\phi'). \tag{5}$$

Any procedure which satisfies these conditions will, in the limit of a sufficiently long Markov chain, produce field configurations $\{\phi^{(i)}\}$ distributed according to $p(\phi)$.

A. Metropolis-Hastings with generative models

Given a model that allows sampling from a known probability distribution $\tilde{p}(\phi)$, a Markov chain for a desired probability distribution $p(\phi)$ can be constructed via the independence Metropolis sampler, a specialization of the Metropolis-Hastings method [36]. For each step i of the chain, an *update proposal* ϕ' is generated by sampling from $\tilde{p}(\phi)$, independent of the previous configuration. This proposal is accepted with probability

$$A(\phi^{(i-1)}, \phi') = \min\left(1, \frac{\tilde{p}(\phi^{(i-1)})}{p(\phi^{(i-1)})} \frac{p(\phi')}{\tilde{p}(\phi')}\right). \tag{6}$$

If the proposal is accepted, $\phi^{(i)} = \phi'$; otherwise $\phi^{(i)} = \phi^{(i-1)}$. This procedure defines the transition probabilities of the Markov chain.

The general Metropolis-Hastings algorithm has been proven to satisfy balance [37] for any proposal scheme. For the independence Metropolis sampler, under the further condition that every state ϕ has nonzero proposal density and nonzero desired density,

$$\tilde{p}(\phi) > 0, \qquad p(\phi) > 0 \quad \text{for all } \phi, \tag{7}$$

the Markov chain is also ergodic and thus guaranteed to converge to the desired distribution [36].

This Markov chain can be intuitively considered a method to correct an approximate distribution $\tilde{p}(\phi)$ to the desired distribution $p(\phi)$. The accept/reject statistics of the Metropolis-Hastings algorithm serve as a diagnostic for closeness of the approximate and desired distributions; if the distributions are equal, proposals are accepted with probability 1 and the Markov chain process is equivalent to a direct sampling of the desired distribution. This is made precise in Sec. II C.

B. Sampling using normalizing flows

Here, a normalizing flow model is used to define a proposal distribution $\tilde{p}(\phi)$ for a generative Metropolis-Hastings algorithm. Normalizing flows [38] are a machine-learning approach to the task of sampling from complicated, intractable distributions. They do so by learning a map from an input distribution that is easy to sample to an output distribution that approximates the desired distribution. Normalizing flow models produce both samples and their associated probability densities, allowing the acceptance probability in Eq. (6) to be calculated.

A normalizing flow enacts the transformation between distributions by a change of variables[1]: a smooth, bijective function, $f^{-1}: \mathbb{R}^D \to \mathbb{R}^D$ maps samples z from a prior distribution $r(z)$ to $\phi = f^{-1}(z)$. This mapping defines an output distribution $\tilde{p}_f(\phi)$, by the change-of-variables formula

$$\tilde{p}_f(\phi) = r(f(\phi))\left|\det\frac{\partial f(\phi)}{\partial \phi}\right|. \tag{8}$$

Typically, the prior distribution is a simple and analytically understood distribution (e.g., a normal distribution). While the desired distribution $p(\phi)$ is often complicated and difficult to sample from directly, optimizing the function f allows one to generate samples from $\tilde{p}_f(\phi) \approx p(\phi)$. The function f is chosen to have a tractable Jacobian such that the probability density $\tilde{p}_f(\phi)$ can be computed exactly according to Eq. (8).

To encode a map from a simple distribution $r(z)$ to a complicated distribution $\tilde{p}_f(\phi)$, the map f must be highly expressive while also being invertible and having a computable Jacobian. Here, the real nonvolume-preserving (NVP) flow [39] machine-learning approach is used: f is constructed by the composition of affine coupling layers that scale and offset half of the components of the input at a time; the choice of which components of the data are transformed is part of the layer definition. Splitting the D-dimensional vector ϕ into $(D/2)$-dimensional pieces ϕ_a and ϕ_b according to this choice, a single coupling layer g_i transforms ϕ to $z = g_i(\phi)$ via

$$g_i(\phi) \coloneqq \begin{cases} z_a = \phi_a \\ z_b = \phi_b \odot e^{s_i(\phi_a)} + t_i(\phi_a), \end{cases} \tag{9}$$

where s_i and t_i are neural networks mapping from $\mathbb{R}^{D/2}$ to $\mathbb{R}^{D/2}$ and $\odot$ denotes elementwise multiplication. Importantly, each layer g_i is invertible without inverting the neural networks s_i or t_i,

$$g_i^{-1}(z) \coloneqq \begin{cases} \phi_a = z_a \\ \phi_b = (z_b - t_i(z_a)) \odot e^{-s_i(z_a)}. \end{cases} \tag{10}$$

The Jacobian matrix is lower triangular and its determinant can be easily computed. For coupling layer g_i,

$$\left|\det\frac{\partial g_i(\phi)}{\partial \phi}\right| = \prod_{j=1}^{D/2} e^{[s_i(\phi_a)]_j}, \tag{11}$$

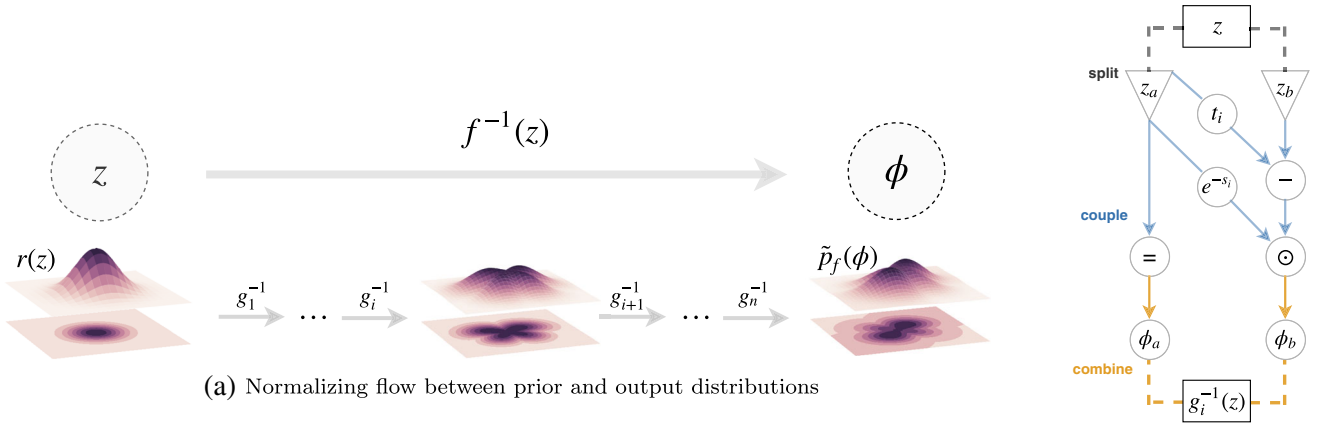
where j indexes the $D/2$ components of the output of s_i. Stacking many coupling layers $g_1, \ldots, g_n$ which alternate which half of the data is transformed, the function f is defined as

$$f(\phi) = g_1(g_2(\ldots g_n(\phi)\ldots)). \tag{12}$$

Using the chain rule, the determinant of the Jacobian of f is a product of the contributions from each g_i. By increasing the number of coupling layers and the complexity of the networks s_i and t_i, f can systematically be made more expressive and general. Figure 1 depicts how composing many coupling layers incrementally modifies a prior distribution which is easy to sample into a more complex output distribution that approximates a distribution of interest.

For a fixed initial distribution $r(z)$, the neural networks within each affine coupling layer of f can be trained to bring $\tilde{p}_f(\phi)$ close to the desired distribution $p(\phi)$. This training is undertaken by minimizing a loss function. Here, the loss function used is a shifted Kullback-Leibler (KL)

[1]The convention of using f^{-1} for the change of variables stems from typical applications of normalizing flows.



(a) Normalizing flow between prior and output distributions

(b) Inverse coupling layer

FIG. 1. In (a), a normalizing flow is shown transforming samples z from a prior distribution $r(z)$ to samples ϕ distributed according to $\tilde{p}_f(\phi)$. The mapping $f^{-1}(z)$ is constructed by composing inverse coupling layers g_i^{-1} as defined in Eq. (10) in terms of neural networks s_i and t_i and shown diagrammatically in (b). By optimizing the neural networks within each coupling layer, $\tilde{p}_f(\phi)$ can be made to approximate a distribution of interest, $p(\phi)$.

divergence[2] between the target distribution of the form $p(\phi) = e^{-S(\phi)}/Z$ and the proposal distribution $\tilde{p}_f(\phi)$,

$$\begin{aligned} L(\tilde{p}_f) &:= D_{KL}(\tilde{p}_f||p) - \log Z \\ &= \int \prod_j d\phi_j \tilde{p}_f(\phi)(\log \tilde{p}_f(\phi) - \log p(\phi) - \log Z) \\ &= \int \prod_j d\phi_j \tilde{p}_f(\phi)(\log \tilde{p}_f(\phi) + S(\phi)). \end{aligned} \tag{13}$$

This loss function has been successfully applied in related generative approaches to statistical lattice models [30,41]. The formal shift by $\log Z$ in Eq. (13) eliminates the need to compute the true partition function, and does not affect the gradients or location of the minima. By non-negativity of the KL divergence, the lower bound on the loss is $-\log Z$, and this minimum is achieved exactly when $\tilde{p}_f = p$. In practice, the loss is stochastically estimated by drawing batches of M samples from the model $\{\phi^{(i)} \sim \tilde{p}_f\}$ and computing the sample mean,

$$\widehat{L(\tilde{p}_f)} = \frac{1}{M}\sum_{i=1}^{M}(\log \tilde{p}_f(\phi^{(i)}) + S(\phi^{(i)})). \tag{14}$$

The loss minimization can then be undertaken using stochastic optimization techniques such as stochastic gradient descent or momentum-based methods including Adam and Nesterov [42,43].

[2]This training paradigm is a specific instance of probability density distillation [30,40].

By construction, the flow model allows sampling from $\tilde{p}_f$ efficiently. The training process can thus be performed by drawing samples from the model itself, rather than using existing samples from the desired distribution as training data. This self-training is a key feature of the proposed approach to Monte Carlo sampling for field theories, where samples from the desired distribution are often computationally expensive to obtain. If samples do exist, they can be used to pretrain the network, although in the numerical studies undertaken here this was not found to be markedly more efficient in network optimization than using only self-training.

Given a trained model with distribution $\tilde{p}_f(\phi) \approx p(\phi)$, samples from $\tilde{p}_f$ can be used as proposals to advance a Markov chain using the generative Metropolis-Hastings algorithm described above. This forms the basis for the flow-based MCMC algorithm proposed here.

(1) A flow-based generative model (here, a real NVP model) is trained using the shifted KL loss given in Eq. (13) to have output distribution $\tilde{p}_f(\phi) \approx p(\phi)$.
(2) N proposals $\{\phi'^{(i)} \sim \tilde{p}_f\}$ are produced by sampling from the flow-based model (this can be done in parallel) and the associated action $S(\phi')$ is computed for each proposal.
(3) Starting from an arbitrary initial configuration, each proposed sample is successively accepted or rejected using the Metropolis-Hastings algorithm given in Eq. (6) to build a Markov chain of length N.

When the prior distribution $r(z)$ is strictly positive, the invertibility and continuity of f guarantees that the generated distribution $\tilde{p}_f(\phi)$ is also strictly positive. For all models with finite action, and thus $p(\phi) > 0$, the resulting Markov chain is then ergodic by the arguments detailed in Sec. II A.

C. Autocorrelation time for a generative Metropolis-Hastings algorithm

For any Markov chain constructed via a generative Metropolis-Hastings algorithm (with independent update proposals), an observable-independent estimator for autocorrelation time can be defined from the accept/reject statistics of the chain. This serves both as a measure of the similarity between the proposal and desired distributions and enables proper error estimation for lattice observables [44].

Precisely, the autocorrelation at Markov chain separation τ, for all observables, is given by the probability of τ rejections in a row,

$$p_{\tau\mathrm{rej}} \equiv \left\langle \prod_{i=1}^{\tau} \mathbb{1}_{\mathrm{rej}}(i) \right\rangle = \rho(\tau)/\rho(0), \tag{15}$$

where $\mathbb{1}_{\mathrm{rej}}(i)$ is an indicator variable taking value 1 when the proposed step from $i-1$ to i was rejected in the Metropolis-Hastings algorithm, and 0 otherwise. In practice, for a near equilibrium, finite Markov chain with length N, a finite-sample estimator provides a good approximation to $\rho(\tau)/\rho(0)$,

$$\widehat{\rho(\tau)/\rho}(0)_{\mathrm{acc}} = \frac{1}{N-\tau}\sum_{j=1}^{N-\tau}\prod_{i=1}^{\tau}\mathbb{1}_{\mathrm{rej}}(i+j). \tag{16}$$

This measure of autocorrelation is consistent with the usual definition; it is shown in Appendix A that the standard estimator for the autocorrelation of any given observable $\mathcal{O}$,

$$\widehat{\rho(\tau)/\rho}(0)_{\mathcal{O}} = \frac{\frac{1}{N-\tau}\sum_{i=0}^{N-\tau-1}(\mathcal{O}_i-\bar{\mathcal{O}})(\mathcal{O}_{i+\tau}-\bar{\mathcal{O}})}{\frac{1}{N}\sum_{i=0}^{N-1}(\mathcal{O}_i-\bar{\mathcal{O}})^2}, \tag{17}$$

also converges to $p_{\tau\mathrm{rej}}$ in the limit $N\to\infty$.

The qualitative relation between acceptance rate and autocorrelations gives a convenient measure of the autocorrelation characteristics of a Markov chain. Precisely, the autocorrelation at distance τ can be bounded in terms of the average acceptance rate $a = 1-\mathbb{E}[p_{\mathrm{rej}}]$,

$$\rho(\tau)/\rho(0) = \mathop{\mathbb{E}}_{\phi\sim p}[p_{\mathrm{rej}}^{\tau}(\phi)] \geq (\mathop{\mathbb{E}}_{\phi\sim p}[p_{\mathrm{rej}}(\phi)])^{\tau} = (1-a)^{\tau}. \tag{18}$$

Increasing the acceptance rate of an independence Metropolis sampler is thus a necessary condition to reduce autocorrelations. Additionally, $a=1$ exactly when the proposal and desired distributions are equal. In this case, $p_{\mathrm{rej}}(\phi)=0$ for each ϕ, and there are no autocorrelations. While bringing a close to 1 does not provide an upper bound on autocorrelation, stochastically improving a loss function that measures distance between distributions is expected to reduce autocorrelations on average. In practice, autocorrelations should be evaluated as a test metric alongside the training loss to confirm improvement over the course of training the model. The correspondence between loss minimization, acceptance rate, and autocorrelations is studied in the context of ϕ^4 theory in Sec. III, where a clear correlation between a and $\rho(\tau)/\rho(0)$ is observed.

D. Critical slowing down

When a distribution is sampled using a Markov chain with large autocorrelation time, many updates are required to produce decorrelated samples. Critical slowing down (CSD) is defined as the divergence of the autocorrelation time of Markov chain sampling as a critical point in parameter space is approached [1]. A numerically stable definition of the characteristic autocorrelation time of a Markov chain is the integrated autocorrelation time,

$$\tau_{\mathcal{O}}^{\mathrm{int}} = \frac{1}{2} + \lim_{\tau_{\max}\to\infty}\sum_{\tau=1}^{\tau_{\max}}\frac{\rho_{\mathcal{O}}(\tau)}{\rho_{\mathcal{O}}(0)}. \tag{19}$$

As a critical point is approached, analysis of standard local-update algorithms for lattice models suggests $\tau_{\mathcal{O}}^{\mathrm{int}}$ is typically well described by a power law in the lattice spacing, or for fixed physical volume, a power law in the lattice sites per dimension, L. A dynamical critical exponent $z_{\mathcal{O}}$ is thus defined by a fit to $\tau_{\mathcal{O}}^{\mathrm{int}} = \alpha_{\mathcal{O}}L^{z_{\mathcal{O}}}$ along a line of constant physics. An update algorithm for which the critical exponent is 0 is unaffected by CSD.

In any generative Metropolis-Hastings simulation, the autocorrelation time is completely fixed by the expected accept/reject statistics, which in turn result from the structure of the proposal and desired distributions. For models trained with a target value of the integrated autocorrelation time used as a stopping criterion, CSD associated with the Markov chain sampling is thus trivially removed at the expense of up-front training costs. The difficulty of CSD is in essence shifted to the training of the model, i.e., to the optimization of the proposal distribution. The cost and scaling of this optimization task for ϕ^4 theory, as studied here, and the prospects of scaling this approach to more complicated theories, are discussed in Sec. III D.

III. APPLICATION OF FLOW-BASED MCMC TO ϕ^4 THEORY

The theory with a massive scalar field $\phi(x)$ and a quartic self-interaction is one of the simplest interacting field theories that can be constructed. It is thus a convenient testing ground for new algorithms for lattice field theory, such as the flow-based MCMC approach proposed here.

In a d-dimensional Euclidean spacetime, a discretized formulation of ϕ^4 theory can be defined on a lattice with sites $x_\mu = an_\mu$, where a denotes the lattice spacing, $\mu\in\{1,\ldots,d\}$ labels spacetime dimension, and $n_\mu\in\mathbb{Z}^d$.

Here, a finite lattice volume $V = (aL)^d$ is considered, with periodic boundary conditions in all dimensions. The lattice action (in units where $a = 1$) can be expressed as

$$S(\phi) = \sum_x \left(\sum_y \phi(x) \Box(x,y) \phi(y) + m^2 \phi(x)^2 + \lambda \phi(x)^4 \right), \tag{20}$$

where the parameters m^2 and λ are the bare mass squared and bare coupling, respectively, and the lattice d'Alembert operator is defined by

$$\sum_y \Box(x,y)\phi(y) = \sum_\mu (2\phi(x) - \phi(x-\hat{\mu}) - \phi(x+\hat{\mu})). \tag{21}$$

By taking expectation values over the distribution $p(\phi) = e^{-S(\phi)}/Z$, observables in the theory can be estimated.

The observables studied here are the connected two-point Green's function

$$G_c(x) = \frac{1}{V} \sum_y (\langle \phi(y)\phi(y+x)\rangle - \langle \phi(y)\rangle \langle \phi(y+x)\rangle) \tag{22}$$

and its momentum-space representation

$$\tilde{G}_c(\vec{p}, t) = \frac{1}{L^{d-1}} \sum_{\vec{x}} e^{i\vec{p}\cdot\vec{x}} G_c(\vec{x}, t), \tag{23}$$

where $x_\mu = (\vec{x}, t)$, as well as the corresponding pole mass

$$m_p = -\partial_t \log \langle \tilde{G}_c(0,t)\rangle, \tag{24}$$

and the two-point susceptibility

$$\chi_2 = \sum_x G_c(x). \tag{25}$$

In the limit $\lambda \to \infty$, with $m^2/\lambda < 0$ fixed, scalar ϕ^4 theory reduces to an Ising model. Another observable of interest is therefore the average Ising energy density [45], defined by

$$E = \frac{1}{d} \sum_{1 \le \mu \le d} G_c(\hat{\mu}), \tag{26}$$

where the sum runs over single-site displacements in all dimensions.

The action of ϕ^4 theory is invariant under the discrete symmetry $\phi(x) \to -\phi(x)$. Depending on the value of the parameters m^2 and λ, this symmetry can be spontaneously broken. The theory thus has two phases: a symmetric phase and a broken-symmetry phase.

A. Model definition and training

For this proof-of-principle study, the flow-based MCMC algorithm detailed in Sec. II was applied to ϕ^4 theory in two dimensions with $L = \{6, 8, 10, 12, 14\}$ lattice sites in each dimension. The parameters m^2 and λ were chosen to fix $m_p L \approx 4$ for each lattice size; their numerical values are given in Table I. For simplicity in this initial work, all parameters were chosen to lie in the symmetric phase. In principle, the flow-based MCMC algorithm can be applied with identical methods to the broken-symmetry phase of the theory, but it remains to be shown that models can be trained for such choices of parameters.

For each set of parameters, real NVP models were defined using 8–12 affine coupling layers (see Sec. II B). The coupling layers were defined to update half of the lattice sites in a checkerboard pattern; successive layers alternately updated the odd and even sites. The neural networks s_i and t_i used in coupling layer g_i [see Eq. (9)] were constructed from two to six fully connected layers, each defined as multiplication by a rectangular matrix followed by pointwise application of a nonlinear function (here, a leaky rectified linear unit [46]). Intermediate vectors (hidden units) had sizes ranging between 100 and 1024. The prior distribution $r(z)$ was chosen to be an uncorrelated Gaussian distribution

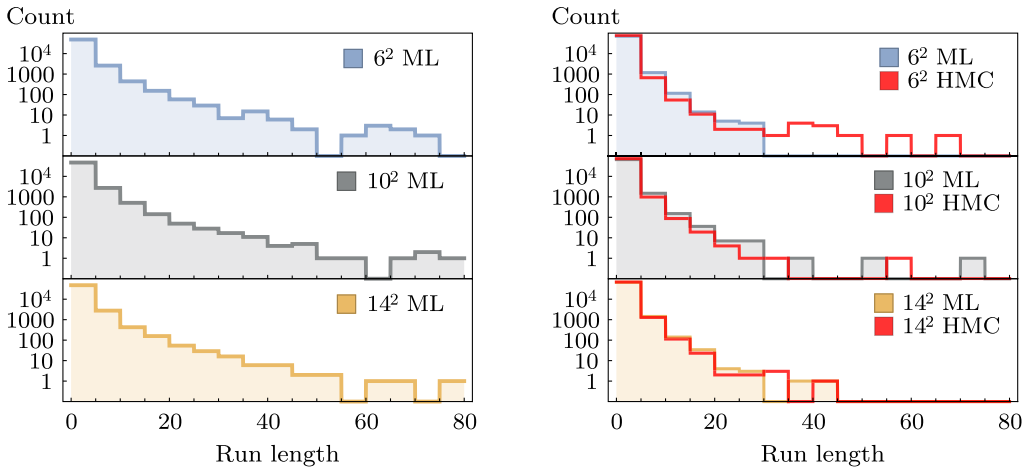
$$r(z) \propto \prod_i e^{-z_i^2/2}. \tag{27}$$

The models were trained to minimize the shifted KL loss between the output distribution $\tilde{p}_f(\phi)$ and the desired distribution $p(\phi) = e^{-S(\phi)}/Z$ using gradient-based updates with the Adam optimizer [42], a specific variety of gradient descent with momentum. A mean absolute error loss, defined in Appendix B, was optimized before training in the case of the 14^2 model where it was found to accelerate convergence to the KL loss minimum.

An exhaustive study of the optimal choice of prior distribution $r(z)$, model depth, architecture, and initialization of the neural networks, and of the mode of coupling of the affine layers, is beyond the scope of this proof-of-principle study. The parameters used here, however, proved to define sufficiently expressive models such that

TABLE I. Parameters $\{m^2, \lambda\}$ of ϕ^4 theory on $L \times L$ lattices used for numerical study of the flow-based MCMC algorithm proposed here. The coupling constants λ have been chosen to approximately maintain constant $m_p L$ as L is varied.

	E1	E2	E3	E4	E5
L	6	8	10	12	14
m^2	−4	−4	−4	−4	−4
λ	6.975	6.008	5.550	5.276	5.113
$m_p L$	3.96(3)	3.97(5)	4.00(4)	3.96(5)	4.03(6)



(a) Flow-based MCMC trained to 50% mean acceptance.

(b) Flow-based MCMC trained to 70% mean acceptance and HMC tuned to 70% mean acceptance.

FIG. 2. Histograms of length of consecutive runs of Metropolis rejections in ML models at both 50% and 70% mean acceptance. Also shown is the same statistic for Markov chains generated via HMC, where mean acceptance was tuned to 70%. The frequency of long runs of rejections is consistently reduced for models trained to reach higher average acceptance. The ML and HMC ensembles at 70% acceptance display very similar distributions of rejection streaks.

the Metropolis-Hastings algorithm applied to output from the trained models easily achieved acceptance rates of well over 50%. With further investment in hyperparameter optimization, higher rates of acceptance could be achieved. In any Markov chain using the Metropolis-Hastings algorithm, there is a tradeoff between computational cost and correlations resulting from low acceptance rates. The optimal acceptance rate minimizes the cost per decorrelated sample from the chain. Here, the cost of training, and not just model evaluation, must be considered, and the optimal level of training in future applications will depend on many factors, such as the desired ensemble size.

For each set of parameters studied, instances of the model were trained to reach both 50% and 70% average Metropolis acceptance. Figure 2 shows histograms of the number of updates between accepted configurations for models at both levels of training. Models trained to reach the higher acceptance rate are seen to have shorter runs of consecutive rejections. Because autocorrelation is related to rejections by $\rho(\tau)/\rho(0) = p_{\tau\text{rej}}$ for independence Metropolis sampling, a reduced frequency of rejection runs with length longer than τ directly implies a reduction in $\rho(\tau)/\rho(0)$. Implications for critical slowing down of the generation of decorrelated configurations are discussed in Sec. III C.

For comparison, ensembles of 10^6 lattice configurations were generated using the machine-learned models in flow-based MCMC as well as standard local Metropolis [47] and HMC [48] algorithms at matched parameters. The local Metropolis algorithm employed a fixed order of sequential updates to each site, with proposed updates to $\phi(x)$ sampled uniformly from the interval $[\phi(x) - \delta, \phi(x) + \delta]$ followed by a Metropolis-Hastings accept/reject step; for all parameters considered, the width δ was tuned to achieve a 70% acceptance rate. The HMC method was implemented using a leapfrog integrator with a fixed division of trajectory length τ into ten steps; the trajectory length τ was also tuned to achieve a 70% acceptance rate. In both the local Metropolis and HMC methods, samples were saved after every 10th update.

B. Tests: physical observables and error scaling

Since the flow-based MCMC algorithm satisfies ergodicity and balance, it is guaranteed to produce samples from the desired probability distribution in the limit of an infinite chain. To test the performance of the algorithm for a finite number of samples, each of the physical observables defined above was computed on ensembles of configurations at the parameters of Table I, generated both using standard HMC and local Metropolis methods, as well as with the trained flow-based MCMC algorithm. Figures 3–5 compare the observables computed on ensembles generated using all three methods.

To estimate the pole mass m_p, an effective mass is defined based on the zero-momentum Green's functions at various time separations,

$$m_p^{\text{eff}}(t) = \operatorname{arccosh}\left(\frac{\tilde{G}_c(0,t-1) + \tilde{G}_c(0,t+1)}{2\tilde{G}_c(0,t)}\right). \tag{28}$$

For all observables, the values computed using the flow-based MCMC ensembles are consistent within statistical uncertainties with those computed using the standard

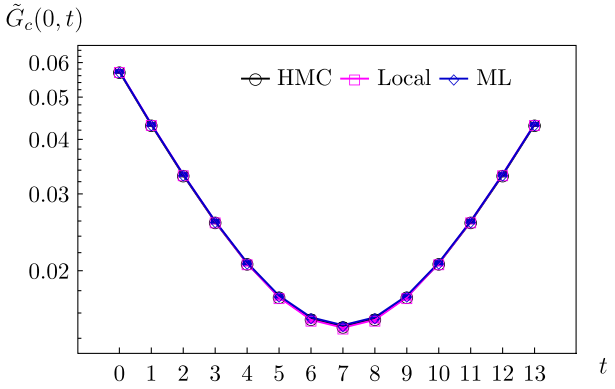


FIG. 3. Zero-momentum Green's functions evaluated for parameter set E5. Results computed using 10^6 configurations from the HMC, local Metropolis, and ML ensembles are consistent within statistical errors. Error bars indicate 68% confidence intervals estimated using bootstrap resampling with bins of size 100.

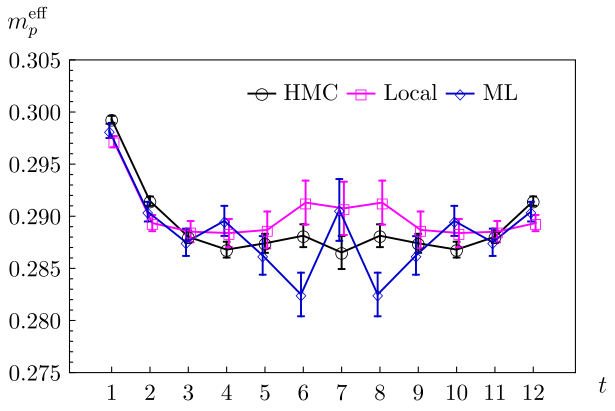


FIG. 4. Effective pole masses evaluated for parameter set E5, defined by the arccosh estimator given in the main text. Results computed using 10^6 configurations from the HMC, local Metropolis, and ML ensembles are consistent within statistical errors. Error bars indicate 68% confidence intervals estimated using bootstrap resampling with bins of size 100.

methods. Moreover, Fig. 6 shows that the statistical uncertainties of the observables scale as $1/\sqrt{N}$ with the number of samples N, as expected for decorrelated samples.

C. Critical slowing down

For ϕ^4 theory, a number of algorithms have been developed that mitigate CSD to various extents, such as worm algorithms [45], multigrid methods [49], Fourier-accelerated Langevin updates [50], and cluster updates via embedded Ising dynamics [51]. The path towards generalizing those algorithms to more complicated theories such as QCD, however, is not clear. Algorithms such as HMC and local Metropolis, which are also used for studies of QCD and pure gauge theory, exhibit CSD for ϕ^4 (as well as more complicated theories) as the continuum limit is approached.

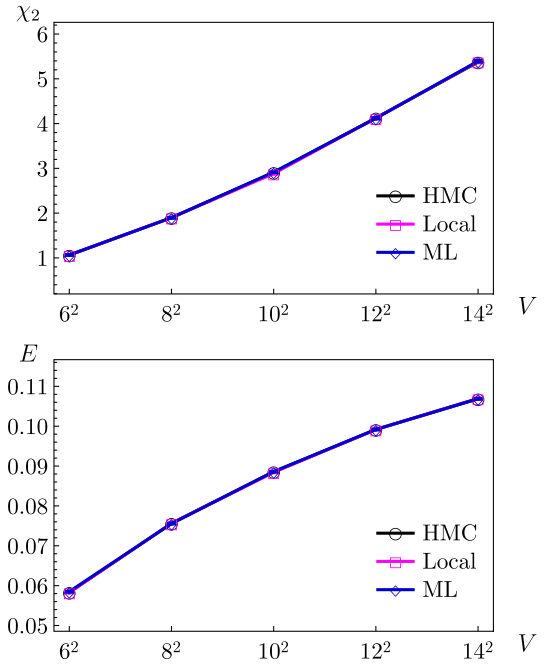


FIG. 5. Susceptibility (χ_2) and Ising energy (E) estimated on all ensembles. Results computed using 10^6 configurations from the HMC, local Metropolis, and ML ensembles are consistent within statistical errors. Errors indicate 68% confidence intervals estimated using bootstrap resampling with bins of size 100.

The parameter sets chosen for the study of ϕ^4 theory in this work (Table I) correspond to a critical line with constant $m_p L$ as $L \to \infty$. For the flow-based MCMC approach proposed here, as well as for ensembles generated using the HMC and local Metropolis algorithms, the autocorrelation times of the set of physical observables discussed previously were fit to leading-order power laws in L to determine the dynamical critical exponents $z_{\mathcal{O}}$ for that observable. Figure 7 shows the autocorrelation times for each observable for each approach to ensemble generation. The absolute values of τ_{int} are not directly comparable between methods because the cost per update differs. The scaling with lattice size, on the other hand, indicates the sensitivity of each method to critical slowing down. For both HMC and local Metropolis, the critical behavior and consequently the performance of the algorithm was found to depend on the observable. In each case, the critical exponent was $0.3 \lesssim z_{\mathcal{O}} \lesssim 2.0$. In comparison, for the flow-based MCMC ensembles at a fixed acceptance, the critical exponent was found to be consistent with 0, with the autocorrelation time being observable independent and in agreement with the acceptance-based estimator defined in Sec. II C.

Since the mean acceptance rate was used as the stopping criterion for training these models, it was not guaranteed *a priori* that the measured integrated autocorrelation time would be constant across the different models used. The results in Fig. 7, however, suggest that beyond the simple

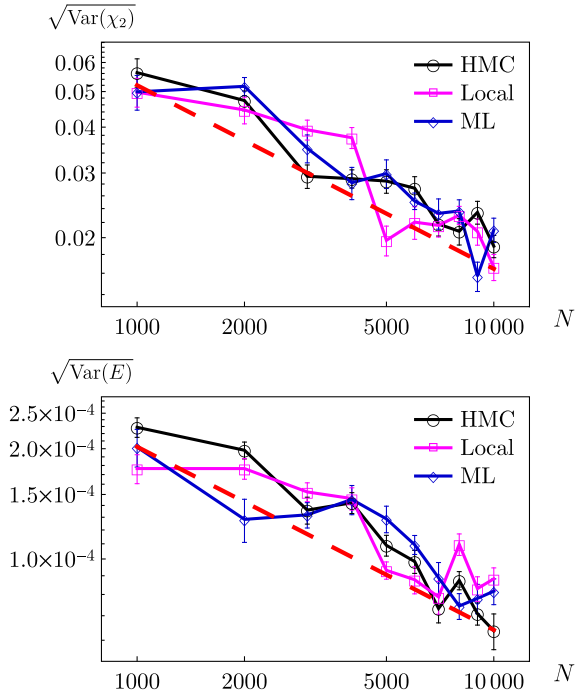


FIG. 6. Statistical error varying with the number of samples N in two candidate observables, χ_2 and E, for the HMC, local Metropolis, and ML ensembles. The red dashed line shows a $1/\sqrt{N}$ curve normalized by the average error estimate of the three approaches at $N = 1000$. Central values were estimated as 68% confidence intervals on each observable by bootstrap resampling ensemble subsets of size N. Error bars indicate 68% confidence intervals estimated using an external bootstrap resampling step.

lower bound from Eq. (18) there is a strong correlation between the mean acceptance rate and integrated autocorrelation time for models trained using a shifted KL loss. This is further confirmed by the similarity of the rejection run histograms across lattice sizes for flow-based MCMC, as shown in Fig. 2.

D. Training costs

While CSD in the sampling step for the flow-based MCMC is eliminated, training the generative model introduces an additional up-front cost, as discussed in Sec. II D. Since this cost is amortized over the ensemble, this approach will naturally be computationally advantageous in the limit of generating a large number of samples. For a finite target ensemble size, the potential acceleration offered depends crucially on the training time.

In this work, all models were trained using one to two GPU weeks, with the larger lattices incurring the most computational cost. For the simple fully connected architecture used in this work, the scaling of both the sampling and training time is controlled by dense matrix-vector multiplications which require $O(V^2)$ floating point operations each. The number of epochs used to train the largest lattice was also roughly 10× that of the smallest lattice. This asymptotic scaling is a result of the simple model architecture used in this proof-of-principle study. For related methods applied to image generation, using convolutional neural networks and a multiscale architecture reduced training and sampling costs significantly and

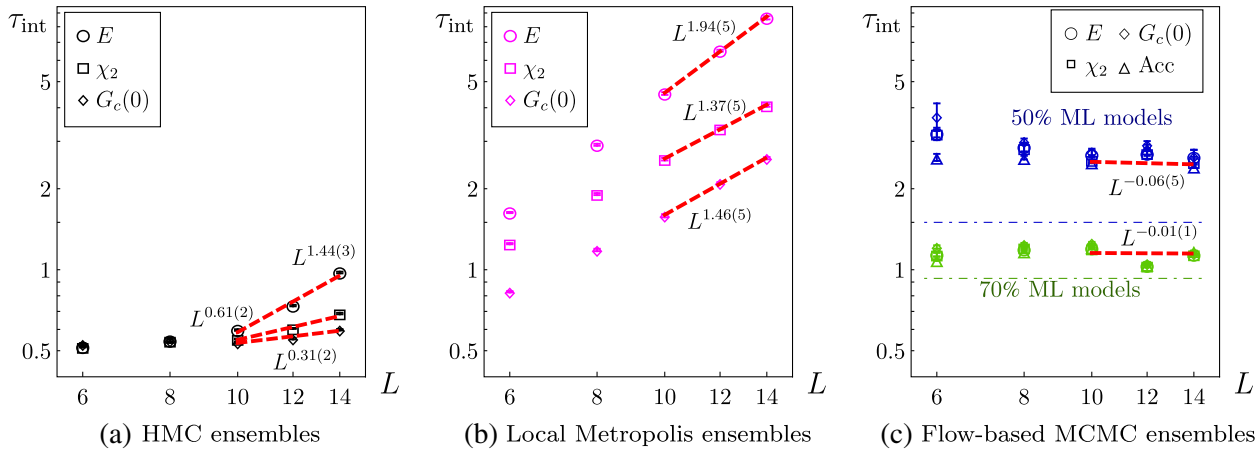


FIG. 7. Scaling of integrated autocorrelation time with respect to lattice size for HMC, local Metropolis, and flow-based MCMC. In (c) the upper sets of points in blue correspond to models trained to a mean acceptance rate of 50%, while the lower sets of points in green correspond to models trained to a mean acceptance rate of 70%. Dashed red lines display power-law fits to $L = \{10, 12, 14\}$ with labels L^z specifying the scaling. The HMC and local Metropolis methods demonstrate power-law growth of τ_{int}, while τ_{int} for the flow-based MCMC is consistent with a constant in L and decreases as mean acceptance rate increases. Dot-dashed blue and green lines for the flow-based ensembles display lower bounds in terms of mean acceptance rate based on Eq. (18). Error bars indicate 68% confidence intervals estimated by bootstrap resampling and error propagation.

improved scaling to $O(V)$ [39]. There are physical grounds to expect these tools to apply equally well to the present application. Convolutional networks use only local information to update values in each layer, exploiting locality in the system, and use identical weights for each point on the lattice, manifestly preserving translational invariance. A multiscale architecture learns coarse-grained distributions and fine-graining procedures in separate layers; this is an effective division of tasks for renormalizable quantum field theories, where simple coarse-grained descriptions are expected to arise. Generative models, and in particular flow-based models, are also rapidly evolving towards more efficient representation capacity. Complex coupling layers have been implemented [39,52], as have generalized convolutions [53,54] and transformations with continuous dynamics that are not dependent on restricted coupling layers [55]. These developments allow models to better capture a distribution within a given number of training steps.

For complex applications, it is also critical that larger models with many coupling layers can be trained without exceeding memory bounds. The algorithm proposed here can be trained with constant memory cost as the number of layers is increased [56], alleviating the storage limitations that can arise in gradient-based optimization. Memory costs can be further reduced by distributing samples within each training batch across many machines.

Finally, typical applications seek to produce ensembles at many different choices of parameters, and often require parameter tuning. Training costs can therefore by amortized further; models trained with respect to an action at a given set of parameter values can either be used to initialize training or as a prior distribution for models targeting that action at nearby parameter values.

IV. SUMMARY

This work defines a flow-based MCMC algorithm to sample lattice field configurations from a desired probability distribution.

(1) A real NVP flow model is trained to produce approximately the desired distribution.
(2) Samples are proposed from the trained model.
(3) Starting from an arbitrary configuration, each proposal is accepted or rejected to advance a Markov chain using the Metropolis-Hastings algorithm.

The approach is shown to define an ergodic and balanced Markov chain, thus guaranteeing convergence to the desired probability distribution in the limit of a long Markov chain. In essence, the flow-based MCMC algorithm combines the expressiveness of normalizing flows based on neural networks with the theoretical guarantees of Markov chains to create a trainable and asymptotically correct sampler. Since these flows are applicable for arbitrary configurations with continuous, real-valued degrees of freedom, one can generically apply this method to any of a broad class of lattice theories. Here, the algorithm is implemented in practice for ϕ^4 theory, and is demonstrated to produce ensembles of configurations that are indistinguishable from those generated using standard local Metropolis and HMC algorithms, based on studies of a number of physical observables.

A key feature of the approach is that models trained to a fixed acceptance rate do not experience critical slowing down in the sampling stage. In particular, the autocorrelation time for all observables is dictated entirely by the accuracy with which the flow model has been trained; perfect training corresponds to decorrelated samples and 100% acceptance in the Metropolis-Hastings step of the MCMC process. Nevertheless, the efficiency with which the training step of this approach can be scaled to larger model sizes, and to more complicated theories such as QCD, remains to be studied. Recent advances in the training and scaling of flow models provide reasons for optimism on this front. Further, incorporating symmetries generally improves data efficiency of training, and implementing spacetime and gauge symmetries [57] may be a natural next step to practically train these flow models for lattice gauge theories like QCD.

In moving towards lattice gauge theories such as QCD, several theoretical developments are also required. The real NVP model chosen to parametrize the normalizing flows here is described in terms of vectors of variables $\phi \in \mathbb{R}^D$. Gauge configurations, however, live in a compact manifold arising from the Lie group structure. Extending this method requires a normalizing flow model that can act on this manifold while remaining sufficiently expressive. The choice of prior likewise will need to be extended to a distribution over the manifold of lattice gauge configurations which can be easily sampled. A uniform distribution, for example, may be a candidate for a prior, but this choice must be tested in the context of a specific flow model.

If the flow-based MCMC algorithm proposed here can be implemented for a complex theory such as QCD, the advantages would be significant; arbitrarily large ensembles of field configurations could be generated at minimal cost. The independence of the proposal step from any previous configuration allows parallel generation of proposals, and the continually improving support in hardware and software for neural network execution suggests future practical gains for this style of ensemble generation. Given efficient sample generation from a trained model, ensembles would not need to be stored long term. Moreover, a model trained for one action could either be retrained or used as a prior for another flow model targeting an action with nearby parameter values. This would allow efficient tuning of parameters and generation of additional ensembles interpolating between and extrapolating from existing models.

ACKNOWLEDGMENTS

We thank J.-W. Chen, K. Cranmer, W. Detmold, R. Melko, D. Murphy, A. Pochinsky, and B. Trippe for helpful

discussions. This work is supported in part by the U.S. Department of Energy, Office of Science, Office of Nuclear Physics under Contract No. DE-SC0011090. This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract No. DE-AC02-06CH11357, under the ALCF Aurora Early Science Program. Some work was undertaken at the Kavli Institute for Theoretical Physics, supported by the National Science Foundation under Grant No. NSF PHY-1748958. P. E. S. is supported by the National Science Foundation under CAREER Award No. 1841699, G. K. is supported by the U.S. Department of Energy under the SciDAC4 Award No. DE-SC0018121, and P. E. S. and M. S. A. are supported in part by NSERC and the Perimeter Institute for Theoretical Physics. Research at the Perimeter Institute is supported by the Government of Canada through the Department of Innovation, Science and Economic Development and by the Province of Ontario through the Ministry of Research and Innovation.

APPENDIX A: ACCEPTANCE RATE ESTIMATOR FOR AUTOCORRELATION

Here it is shown that the standard estimator for the autocorrelation of an observable $\mathcal{O}$ converges in the limit of infinite path length to $p_{\tau\mathrm{rej}}$, as claimed in Sec. II C. The standard estimator is defined by

$$\lim_{N\to\infty} \mathbb{E}[\widehat{\rho(\tau)/\rho(0)}_O] \tag{A1}$$

$$= \lim_{N\to\infty} \mathbb{E}\left[\frac{\frac{1}{N-\tau}\sum_{i=0}^{N-\tau-1}(\mathcal{O}_i-\bar{\mathcal{O}})(\mathcal{O}_{i+\tau}-\bar{\mathcal{O}})}{\frac{1}{N}\sum_{i=0}^{N-1}(\mathcal{O}_i-\bar{\mathcal{O}})^2}\right] \tag{A2}$$

$$= \lim_{N\to\infty} \mathbb{E}\left[\frac{(\mathcal{O}_i-\bar{\mathcal{O}})(\mathcal{O}_{i+\tau}-\bar{\mathcal{O}})}{\frac{1}{N}\sum_{i=0}^{N-1}(\mathcal{O}_i-\bar{\mathcal{O}})^2}\right], \tag{A3}$$

where the final equality is true assuming the Markov chain is initialized with a sample from the stationary distribution $p(\phi)$ (i.e., it is assumed that enough prior iterations were discarded such that the chain is thermalized). The expectation value can then be split by cases and, conditioning on the fixed accept/reject pattern, the expectation values can be computed by identifying the distributions of observables $\mathcal{O}_i$ and $\mathcal{O}_{i+\tau}$,

$$\lim_{N\to\infty} \mathbb{E}[\widehat{\rho(\tau)/\rho(0)}_O] = \lim_{N\to\infty} p_{\tau\mathrm{rej}}\mathbb{E}\left[\frac{(\mathcal{O}_i-\bar{\mathcal{O}})(\mathcal{O}_{i+\tau}-\bar{\mathcal{O}})}{\frac{1}{N}\sum_{i=0}^{N-1}(\mathcal{O}_i-\bar{\mathcal{O}})^2}\middle|\text{all proposals } i+1,\ldots,i+\tau \text{ rejected}\right] + \tag{A4}$$

$$(1-p_{\tau\mathrm{rej}})\mathbb{E}\left[\frac{(\mathcal{O}_i-\bar{\mathcal{O}})(\mathcal{O}_{i+\tau}-\bar{\mathcal{O}})}{\frac{1}{N}\sum_{i=0}^{N-1}(\mathcal{O}_i-\bar{\mathcal{O}})^2}\middle|\text{some proposal } i+1,\ldots,i+\tau \text{ accepted}\right] \tag{A5}$$

$$= p_{\tau\mathrm{rej}}\lim_{N\to\infty}\mathop{\mathbb{E}}_{\phi\sim p}\left[\frac{(\mathcal{O}[\phi]-\bar{\mathcal{O}})^2}{\frac{1}{N}\sum_{j\notin[i,i+\tau]}(\mathcal{O}_j-\bar{\mathcal{O}})^2+O(\tau/N)}\right] + \tag{A6}$$

$$(1-p_{\tau\mathrm{rej}})\lim_{N\to\infty}\mathop{\mathbb{E}}_{\phi\sim p}\mathop{\mathbb{E}}_{\phi'\sim\tilde{p}}\left[\frac{(\mathcal{O}[\phi]-\bar{\mathcal{O}})(\mathcal{O}[\phi']-\bar{\mathcal{O}})}{\frac{1}{N}\sum_{j\notin[i,i+\tau]}(\mathcal{O}_j-\bar{\mathcal{O}})^2+O(\tau/N)}\right] \tag{A7}$$

$$= p_{\tau\mathrm{rej}}\frac{\rho(0)}{\rho(0)} + (1-p_{\tau\mathrm{rej}})\frac{\cancel{\mathop{\mathbb{E}}_{\phi\sim p}[\mathcal{O}[\phi]-\bar{\mathcal{O}}]}\mathop{\mathbb{E}}_{\phi'\sim\tilde{p}}[\mathcal{O}[\phi']-\bar{\mathcal{O}}]}{\rho(0)} = p_{\tau\mathrm{rej}}. \tag{A8}$$

The limit $N\to\infty$ is used to drop biases arising from conditioning on behavior within the region $[i, i+\tau]$.

APPENDIX B: MEAN ABSOLUTE ERROR LOSS

The mean absolute error (MAE) loss optimized before training some models is defined by

$$L_{\mathrm{MAE}}(\tilde{p}_f) \coloneqq \int \prod_j d\phi_j \tilde{p}_f(\phi)|\log\tilde{p}_f(\phi)-\log p(\phi)|. \tag{B1}$$

It is bounded below by the KL divergence $D_{KL}(\tilde{p}_f||p)$ and has global minima exactly where the KL loss does, when $\tilde{p}_f = p$.

In practice, the loss is stochastically estimated by drawing batches of M samples from the model $\{\phi^{(i)} \sim \tilde{p}_f\}$ and computing the sample mean,

$$\widehat{L_{\mathrm{MAE}}(\tilde{p}_f)} = \frac{1}{M}\sum_{i=1}^{M} |\log \tilde{p}_f(\phi^{(i)}) - \log p(\phi^{(i)})| = \frac{1}{M}\sum_{i=1}^{M} |\log \tilde{p}_f(\phi^{(i)}) + S(\phi^{(i)}) + \log Z|. \quad \text{(B2)}$$

To employ this loss, the partition function must either be estimated ahead of time, or initialized as a trainable parameter. In this study, a multistage method [58] was used to estimate and fix the partition function value used while optimizing L_{MAE}.

This loss is appealing due to the point-by-point potential driving the distribution towards the correct one. Any errors in computing $\log Z$, however, result in a minimum at which the model distribution $\tilde{p}_f(\phi)$ does not necessarily agree with the desired distribution $p(\phi)$. This loss was therefore only used prior to training with the shifted KL loss.

[1] U. Wolff, Nucl. Phys. B, Proc. Suppl. **17**, 93 (1990).
[2] L. Del Debbio, G. M. Manca, and E. Vicari, Phys. Lett. B **594**, 315 (2004).
[3] H. B. Meyer, H. Simma, R. Sommer, M. Della Morte, O. Witzel, and U. Wolff, Comput. Phys. Commun. **176**, 91 (2007).
[4] D. Kandel, E. Domany, D. Ron, A. Brandt, and E. Loh, Phys. Rev. Lett. **60**, 1591 (1988).
[5] C. Bonati and M. D'Elia, Phys. Rev. E **98**, 013308 (2018).
[6] M. Hasenbusch and S. Schaefer, Phys. Rev. D **98**, 054502 (2018).
[7] R. H. Swendsen and J.-S. Wang, Phys. Rev. Lett. **58**, 86 (1987).
[8] U. Wolff, Phys. Rev. Lett. **62**, 361 (1989).
[9] N. Prokof'ev and B. Svistunov, Phys. Rev. Lett. **87**, 160601 (2001).
[10] N. Kawashima and K. Harada, J. Phys. Soc. Jpn. **73**, 1379 (2004).
[11] W. Bietenholz, A. Pochinsky, and U. J. Wiese, Phys. Rev. Lett. **75**, 4524 (1995).
[12] A. Ramos, *Proc. Sci.*, LATTICE2012 (2012) 193.
[13] A. S. Gambhir and K. Orginos, Proc. Sci. LATTICE2014 (**2015**) 043.
[14] G. Cossu, P. Boyle, N. Christ, C. Jung, A. Jttner, and F. Sanfilippo, EPJ Web Conf. **175**, 02008 (2018).
[15] X.-Y. Jin and J. C. Osborn, Proc. Sci. LATTICE2018 (**2019**) 027.
[16] M. G. Endres, R. C. Brower, W. Detmold, K. Orginos, and A. V. Pochinsky, Phys. Rev. D **92**, 114516 (2015).
[17] W. Detmold and M. G. Endres, Phys. Rev. D **94**, 114502 (2016).
[18] W. Detmold and M. G. Endres, Phys. Rev. D **97**, 074507 (2018).
[19] M. Luscher and S. Schaefer, Comput. Phys. Commun. **184**, 519 (2013).
[20] S. Mages, B. C. Toth, S. Borsanyi, Z. Fodor, S. D. Katz, and K. K. Szabo, Phys. Rev. D **95**, 094512 (2017).
[21] Y. Burnier, A. Florio, O. Kaczmarek, and L. Mazur, EPJ Web Conf. **175**, 07004 (2018).
[22] A. Laio, G. Martinelli, and F. Sanfilippo, J. High Energy Phys. 07 (2016) 089.
[23] A. Tanaka and A. Tomiya, arXiv:1712.03893.
[24] P. E. Shanahan, D. Trewartha, and W. Detmold, Phys. Rev. D **97**, 094506 (2018).
[25] L. Huang and L. Wang, Phys. Rev. B **95**, 035105 (2017).
[26] J. Liu, Y. Qi, Z. Y. Meng, and L. Fu, Phys. Rev. B **95**, 041101(R) (2017).
[27] J. Liu, H. Shen, Y. Qi, Z. Y. Meng, and L. Fu, Phys. Rev. B **95**, 241104 (2017).
[28] Y. Nagai, H. Shen, Y. Qi, J. Liu, and L. Fu, Phys. Rev. B **96**, 161102(R) (2017).
[29] H. Shen, J. Liu, and L. Fu, Phys. Rev. B **97** 205140 (2018).
[30] S. H. Li and L. Wang, Phys. Rev. Lett. **121**, 260601 (2018).
[31] F. Noé and H. Wu, arXiv:1812.01729.
[32] J. Song, S. Zhao, and S. Ermon, in Proceedings in Neural Information Processing Systems (2017), http://arxiv.org/abs/1706.07561.
[33] D. Levy, M. D. Hoffman, and J. Sohl-Dickstein, arXiv:1711.09268.
[34] J. M. Urban and J. M. Pawlowski, arXiv:1811.03533.
[35] K. Zhou, G. Endrdi, L.-G. Pang, and H. Stcker, Phys. Rev. D **100**, 011501 (2019).
[36] L. Tierney, Ann. Stat. **22**, 1701 (1994).
[37] W. K. Hastings, Biometrika **57**, 97 (1970).
[38] D. J. Rezende and S. Mohamed, arXiv:1505.05770.
[39] L. Dinh, J. Sohl-Dickstein, and S. Bengio, arXiv:1605.08803.
[40] A. v. d. Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. v. d. Driessche, E. Lockhart, L. C. Cobo, F. Stimberg *et al.*, arXiv:1711.10433.
[41] L. Zhang, W. E, and L. Wang, arXiv:1809.10188.
[42] D. P. Kingma and J. L. Ba, in *International Conference on Learning Representations (ICLR)* (2015) [arXiv:1412.6980].
[43] Y. E. Nesterov, Dokl. Akad. Nauk SSSR **269**, 543 (1983).
[44] U. Wolff (ALPHA Collaboration), Comput. Phys. Commun. **156**, 143 (2004); **176**, 383(E) (2007).

[45] I. Vierhaus, Ph.D. thesis, Humboldt University of Berlin, 2010.
[46] V. Nair and G. E. Hinton, in *ICML* (Omnipress, Madison, WI, 2010).
[47] U. Wolff, *Numerical Simulation in Quantum Field Theory* (Springer, Berlin, Heidelberg, 1996), pp. 245–257.
[48] S. Duane, A. Kennedy, B. J. Pendleton, and D. Roweth, Phys. Lett. B **195**, 216 (1987).
[49] J. Goodman and A. D. Sokal, Phys. Rev. D **40**, 2035 (1989).
[50] G. G. Batrouni and B. Svetitsky, Phys. Rev. B **36**, 5647 (1987).
[51] R. C. Brower and P. Tamayo, Phys. Rev. Lett. **62**, 1087 (1989).
[52] L. Dinh, D. Krueger, and Y. Bengio, arXiv:1410.8516.
[53] D. P. Kingma and P. Dhariwal, in *Neural Information Processing Systems* (2018), pp. 1–15.
[54] E. Hoogeboom, R. V. D. Berg, and M. Welling, arXiv:1901.11137.
[55] W. Grathwohl, R. T. Q. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud, arXiv:1810.01367.
[56] X. Li and W. Grathwohl, Training Glow with Constant Memory Cost, pp. 1–7 (2018), http://bayesiandeeplearning.org/2018/papers/37.pdf.
[57] T. S. Cohen, M. Weiler, B. Kicanaoglu, and M. Welling, arXiv:1902.04615.
[58] J. P. Valleau and D. N. Card, J. Chem. Phys. **57**, 5457 (1972).

Correction: Equations (20) and (23) contained minor errors and have been fixed.