# Efficient modeling of trivializing maps for lattice $\phi^4$ theory using normalizing flows: A first look at scalability

Luigi Del Debbio, Joe Marsh Rossney[*], and Michael Wilson

*Higgs Centre for Theoretical Physics, School of Physics and Astronomy, The University of Edinburgh, Edinburgh EH9 3FD, United Kingdom*

General-purpose Markov chain Monte Carlo sampling algorithms suffer from a dramatic reduction in efficiency as the system being studied is driven toward a critical point through, for example, taking the continuum limit. Recently, a series of seminal studies suggested that *normalizing flows*—a class of deep generative models—can form the basis of a sampling strategy that does not suffer from this "critical slowing down." The central idea is to use machine learning techniques to build (approximate) trivializing maps, i.e., field transformations that map the theory of interest into a "simpler" theory in which the degrees of freedom decouple. These trivializing maps provide a representation of the theory in which all its nontrivial aspects are encoded within an invertible transformation to a set of field variables whose statistical weight in the path integral is given by a distribution from which sampling is easy. No separate process is required to generate training data for such models, and convergence to the desired distribution is guaranteed through a reweighting procedure such as a Metropolis test. From a theoretical perspective, this approach has the potential to become more efficient than traditional sampling since the statistical efficiency of the sampling algorithm is decoupled from the correlation length of the system. The caveat to all of this is that, somehow, the costs associated with the highly nontrivial task of sampling from the path integral of an interacting field theory are transferred to the training of a model to perform this transformation. In a proof-of-principle demonstration on two-dimensional $\phi^4$ theory, Albergo, Kanwar, and Shanahan [Phys. Rev. D **100**, 034515 (2019)] modeled the trivializing map as a sequence of pointwise affine transformations. We pick up this thread, with the aim of quantifying how well we can expect this approach to scale as we increase the number of degrees of freedom in the system. We make several modifications to the original design that allow our models to learn more efficient representations of trivializing maps using much smaller neural networks, which leads to a large reduction in the computational cost required to train models of equivalent quality. After making these changes, we find that sampling efficiency is almost entirely dictated by how extensively a model has been trained while being unresponsive to further alterations that increase model flexibility. However, as we move toward the continuum limit the training costs scale extremely quickly, which urgently requires further work to fully understand and mitigate.

## I. INTRODUCTION

Lattice field theory involves the computation of expectation values of the form

$$\langle \mathcal{O} \rangle = \frac{1}{\mathcal{Z}} \int \mathcal{D}\phi e^{-S(\phi)} \mathcal{O}(\phi), \qquad (1)$$

[*]Corresponding author.
Joe.Marsh-Rossney@ed.ac.uk

where

$$\mathcal{D}\phi = \prod_{x \in \Lambda} d\phi_x, \qquad (2)$$

resulting from the discretization of Euclidean path integrals onto a space-time lattice $\Lambda$. $\mathcal{O}(\phi)$ is a generic observable defined for the field configuration $\phi$, and the (Euclidean) action $S(\phi)$ encodes all of the dynamics and interactions of the fields.

In most interesting scenarios integrals of this form are not tractable and we are forced to resort to sampling. Concretely, estimating (1) by sampling means evaluating a statistical average,

$$\bar{\mathcal{O}} = \frac{1}{N} \sum_{\{\phi\}} \mathcal{O}(\phi), \tag{3}$$

over a representative sample $\{\phi\}$ comprising $N$ field configurations drawn from a statistical ensemble with Boltzmann factor $e^{-S(\phi)}$. The error on this estimator scales as $1/\sqrt{N_{\text{eff}}}$, where the effective sample size $N_{\text{eff}}$ reaches a maximum value of $N$ in the absence of correlations between configurations.

Markov chain Monte Carlo (MCMC) methods are the best known tool for sampling from high-dimensional distributions. However, the configurations in the resulting sequence are indeed correlated, and the effective sample size is diminished by a factor of twice the *integrated autocorrelation time*,[1]

$$\tau_{\mathcal{O}} = \frac{1}{2} + \sum_{t=1}^{\infty} \frac{\Gamma_{\mathcal{O}}(t)}{\Gamma_{\mathcal{O}}(0)}, \tag{4}$$

defined for each observable in terms of its autocorrelation function $\Gamma_{\mathcal{O}}(t) = \langle \mathcal{O}(\phi^{(n+t)}) \mathcal{O}(\phi^{(n)}) \rangle - \langle \mathcal{O} \rangle^2$, where $t$ represents a number of steps separating pairs of configurations in a Markov chain and $n$ is arbitrary provided the process has equilibrated to its stationary distribution [1].

Under normal conditions this issue is manageable. Most MCMC algorithms, however, suffer from an acute condition known as *critical slowing down* associated with a quite catastrophic reduction in their sampling efficiency as the system under study approaches a critical point [2]. Critical slowing down typically manifests as a power-law scaling of the integrated autocorrelation time with the system's correlation length $\xi$:

$$\tau_{\mathcal{O}} \propto \xi^{z_{\mathcal{O}}}. \tag{5}$$

This is rather unfortunate since $\xi$ (in lattice units) diverges as we take the continuum limit of our lattice field theory.

Algorithms based on random walks or classical molecular dynamics [3–5] have a lower limit of $z_{\mathcal{O}} = 1$ owing to the maximum speed of information propagation, but in the absence of very careful tuning [6] they typically exhibit $z_{\mathcal{O}} = 2$ scaling, corresponding to diffusive information transport. Furthermore, there is substantial evidence that the picture is even worse when considering theories which possess nontrivial topology in the continuum limit [7–11], including QCD itself [12,13]. As the continuum limit is approached, the rapid increase in energy barriers between topological sectors can result in $z_{\mathcal{O}} > 2$ for topological observables and potentially even exponential scaling [14].

Collective update algorithms can, in principle, fare much better, since they need not be restricted to local dynamics.

Indeed, several celebrated algorithms based on collective updates have been devised for certain systems [15–19], but as yet a general-purpose collective update algorithm is lacking, and critical slowing down remains an unsolved problem in the majority of cases, including lattice QCD.

Given this state of affairs, there has recently been a great deal of interest in augmenting the MCMC "toolkit" with novel techniques from the rapidly maturing field of machine learning. Perhaps the most eye-catching new additions to the toolkit are *deep generative models* [20,21], which can be viewed as a category of highly flexible statistical models that, using stochastic optimization techniques (also known as "training"), can approximate complicated probability densities. The unique property of generative models is that, once trained, they can be directly sampled from to generate samples of potentially "realistic" data. The relevance of this becomes clear upon a change of terminology: let "data" mean "field configurations" and "probability density" refer to their statistical weight in the path integral. Already, a number of prototypical hybrid algorithms have been proposed in which deep generative models either guide or replace traditional MCMC update procedures [22–37]. Deep generative models have also been used to automatically identify relevant variables, leading to novel algorithms for characterizing phase diagrams [30,38–41] and enacting renormalization group transformations [42–46].

*Normalizing flows* [47–49] are a class of deep generative model which approximate the distribution of interest by learning an invertible map from a set of "latent" variables whose distribution is much easier to sample from. Typically, the map is built out of a sequence of relatively simple pointwise transformations. The capacity to model complex, correlated probability distributions such as those corresponding to Euclidean lattice field theories arises due to the fact that the parameters of these transformations are generated by neural networks that take the field variables themselves as inputs [50,51]. Put another way, the process of training such a model is an encoding of the correlations between the degrees of freedom in the path integral into the weights and biases of these neural networks.

In spirit, normalizing flows are very similar to Lüscher's *trivializing maps* [52], which are field transformations that map an interacting theory to a limit where the field variables decouple, at which point sampling becomes extremely efficient. Reference [52] provides a power-series expression for the generators of a class of flows which trivialize lattice gauge theories, although only the first two terms in this series are tractable in practice, and additional finite-step errors are accumulated through numerical integration of the flow. Unfortunately, the degree to which the result of this procedure approximates a trivializing map proved insufficient to improve the scaling of integrated autocorrelation times [53]. However, there has recently been a renewed interest in the potential to construct (approximate)

---

[1]The variance of $\bar{\mathcal{O}}$ is $(2\tau_{\mathcal{O}}/N)\Gamma_{\mathcal{O}}(0)$, with $N/(2\tau_{\mathcal{O}})$ taken to be the definition of $N_{\text{eff}}$.

trivializing maps with additional leverage provided by modern machine learning techniques [22,31–35].

As noted in Ref. [52], the existence of a trivializing map[2] implies that it is theoretically possible to evaluate Eq. (1) without MCMC, by simply generating uncorrelated random fields. In Secs. II B and III we outline a more practical procedure in which a normalizing flow generates statistically independent field configurations that act as proposals for the Metropolis-Hastings algorithm. From a theoretical perspective, this has the potential to become more efficient than traditional sampling, since the statistical efficiency of the sampling algorithm is decoupled from the correlation length of the system. The caveat is that, somehow, the costs associated with the highly nontrivial task of sampling from the path integral of an interacting field theory are transferred to the training of the model. Therefore, to answer the question of whether a generative sampling algorithm can be expected to outperform traditional methods is a matter of understanding how these training costs scale as the continuum limit is approached.

Albergo, Kanwar, and Shanahan [22] first demonstrated that the procedure just described is a viable approach to sampling in lattice field theory. In their proof of principle study, which focused on two-dimensional scalar $\phi^4$ theory on lattices with up to $14^2$ sites, the normalizing flow was a sequence of pointwise affine transformations parameterized by neural networks. Here, we continue the same thread, with the aim of establishing how well this approach scales to lattices with up to $20^2$ sites. Finding that the recipe used in Ref. [22] yields relatively inefficient representations of trivializing maps for the particular theory of interest, we make a number of adjustments; most importantly, we bring in a more expressive transformation based on a *spline* and replace the deep neural networks by networks with a single hidden layer that is rather narrow. We quantify the performance and scaling of our models using hardware-independent metrics: the Metropolis-Hastings acceptance rate, the number of trainable parameters in the models, and the total number of field configurations generated during the training phase.

## II. SAMPLING IN LATTICE FIELD THEORY

The problem we are trying to solve can be phrased as follows: we would like to generate samples $\{\phi^{(1)}, \phi^{(2)}, …, \phi^{(N)}\}$ of the discrete random field $\{\phi_x | x \in \Lambda\} \equiv \phi \in \mathcal{M}^{|\Lambda|}$, where $|\Lambda|$ is the number of sites on the lattice and $\mathcal{M}^{|\Lambda|}$ is a direct product called the field space, that are representative of the lattice field theory we seek to study. By "representative" we mean that the

---

[2]Proof of existence is stated in Ref. [52] for the case of compact, connected gauge fields. See Ref. [54] for an alternative, diagrammatic construction of trivializing maps which holds to all orders for scalar theories.

probability of a particular configuration appearing in the sample is to be proportional to its Boltzmann weight:

$$p(\phi) \equiv \frac{e^{-S(\phi)}}{\mathcal{Z}}. \tag{6}$$

We will refer to $p(\phi)$ as the *target density*.

### A. Markov chain Monte Carlo

MCMC sampling methods work by generating a sequence of transitions $\phi^{(n)} \to \phi^{(n+1)}$ which together comprise a Markov chain $(\phi^{(n)})_{n=1}^N$. Thus, implicit in any MCMC method is a transition kernel $W(\phi \to \phi')$, which is required to have a stationary distribution that is equal to the distribution from which we wish to sample, implying the following:

$$\int \mathcal{D}\phi\, p(\phi) W(\phi \to \phi') = p(\phi'). \tag{7}$$

If $W(\phi \to \phi')$ is also ergodic, then the stationary distribution is unique and the Markov chain is guaranteed to converge to $p(\phi)$ [55]. However, this does not imply that any finite section of the chain is representative of $p(\phi)$, since the configurations will be correlated. In practice, this results in statistical errors that scale as $(2\tau_{\mathcal{O}}N)^{-1/2}$ rather than $N^{-1/2}$, leading to a trade-off between algorithmic efficiency—the amount of effort taken to generate a transition—and statistical efficiency—how many transitions are required to produce a statistically independent configuration.

To guarantee Eq. (7) it is sufficient to impose detailed balance:

$$p(\phi) W(\phi \to \phi') = p(\phi') W(\phi' \to \phi). \tag{8}$$

Generating transitions that satisfy Eq. (8) is fairly straightforward when the new state differs from the old at only one lattice site, since they amount to sampling from low-dimensional distributions conditioned on the current state of the rest of the lattice. Collective updates are a different matter entirely; the famous examples [15–19] involve contrived update procedures which are only applicable within certain classes of models.

A simple and robust alternative due to Metropolis *et al.* [3,4] is to generate configurations $\phi'$ via a distribution $q(\phi'|\phi)$ which is easy to sample from and accept or reject these proposals based on an acceptance probability $A(\phi \to \phi')$ such that $W(\phi \to \phi') = q(\phi'|\phi)A(\phi \to \phi')$ and detailed balance is satisfied. The standard choice is the "Metropolis test":

$$A(\phi \to \phi') = \min\left(1, \frac{q(\phi|\phi')}{q(\phi'|\phi)} \frac{p(\phi')}{p(\phi)}\right), \tag{9}$$

which, importantly, does not require the calculation of normalizing factors. A rejection of the proposal corresponds to a duplication of the current state in the chain. Thus, the Markov chain can be seen as a reweighting of the set of proposals in which the configurations pick up integer weights. The proposal distribution $q(\phi'|\phi)$ can be anything which guarantees ergodicity of $W(\phi \to \phi')$, and it is sufficient for it to have nonzero density everywhere on $\mathcal{M}^{|\Lambda|}$ [56].

In the following snippet of PYTHON code, which implements the Metropolis-Hastings algorithm, generator yields proposals drawn from $q(\phi'|\phi)$, and acceptance is a function which evaluates Eq. (9).

---

**Metropolis-Hastings Algorithm (Python)**

```python
chain = []
current = next(generator) # initialise
for n in range(N):
    proposal = next(generator)
    prob = acceptance(current, proposal)
    if rand() < prob:
        chain.append(proposal)
        current = proposal
    else:
        chain.append(current)
```

---

The Metropolis-Hastings algorithm is completely agnostic toward the process through which proposals are generated—be it a single spin flip, a molecular dynamics trajectory or an independent configuration explicitly drawn from some proposal distribution—provided any "selection bias" is properly accounted for by the factor $q(\phi|\phi')/q(\phi'|\phi)$. This makes it very appealing as a kernel around which to construct collective updates algorithms.

The difficulties arise due to the rapidly increasing sparsity of $p(\phi)$ as the number of degrees of freedom increases and as we move toward the continuum, which puts extremely stringent constraints on how proposals may be generated if we are to sample the path integral in an acceptable amount of time. Two main approaches to this problem are the following.

(i) *Local updates.*—Generate proposals that are close to the current configuration by updating individual lattice sites. Changes in $p(\phi)$ can be made arbitrarily small by tuning the step size so as to yield a desired acceptance rate.

(ii) *Hybrid Monte Carlo.*—Generate proposals by numerically integrating a fictitious Hamiltonian system and by doing so update all of the lattice sites. In this case it is the number of integration steps that must be balanced against the acceptance rate.

Both of these methods become less efficient when we take the continuum limit, as we are forced to trade down on step size to keep the acceptance rate reasonably high, meaning each statistically independent configuration requires more steps to produce. This is what is meant by critical slowing down, and the decline in statistical efficiency is quantified by the dynamical critical exponent in Eq. (5).

## B. A generative approach to global updates

Though it might seem extremely ambitious, the simplest possible generative sampling algorithm would be one in which a deep generative model generates entire, statistically independent field configurations with a probability close to their true weight in the path integral. Let us entertain this ambition. For reasons which will shortly become clear, we will focus the following discussion on parametric models with an explicit probability density $\tilde{p}(\phi)$. The intention is to construct a model and identify a set of model parameters $\theta$ such that the approximation $\tilde{p}(\phi) \approx p(\phi)$ is a "good" one. The notion of a "good approximation" is made quantitative through the Kullbach-Leibler divergence [57]

$$D_{\mathrm{KL}}(p\|\tilde{p}) = \int \mathcal{D}\phi\, p(\phi) \log \frac{p(\phi)}{\tilde{p}(\phi)}. \qquad (10)$$

When we speak of "training" such a model, what we really mean is optimizing a particular function with respect to the model's parameters. In the present work, this "objective function" will be (a variant of) the Kullbach-Leibler divergence, and the goal of training will be to find the set of parameters $\theta^*$ which satisfy

$$\theta^* = \underset{\theta}{\mathrm{argmin}}\, D_{\mathrm{KL}}(p\|\tilde{p}). \qquad (11)$$

Successfully training a generative model to generate uncorrelated samples of field configurations with a probability close to their true weight in the path integral certainly appears to be an auspicious starting point for constructing an efficient sampling algorithm. Of course, the problem is that anything less than a perfect fit, i.e., $\tilde{p}(\phi) = p(\phi)$, implies that the samples generated by the model are not truly representative of the field theory, with discrepancies between $\tilde{p}(\phi)$ and $p(\phi)$ manifesting as biases in expectation values. Yet it is possible to exactly correct for these biases through reweighting or a Metropolis step, provided we have access to $\tilde{p}(\phi)$. Hence, as well as restricting ourselves to models with an *explicit* density function, we will also demand that $\tilde{p}(\phi)$ is *tractable*, by which we mean it is given exactly by a closed-form expression computable in polynomial time (in order to be scalable) and whose repeated evaluation (for us, $10^5$–$10^9$ times during training) does not constitute an unacceptably large overhead.[3]

---

[3]One might think that likelihood-based training [i.e., solving (11)] *requires* $\tilde{p}(\phi)$ to be tractable, but approximate training schemes based on a variational upper bound of (10) have proved successful.

Consider a variant of the Metropolis-Hastings algorithm in which generator is a generative model equipped with an explicit and tractable density that is capable of generating independent configurations with probability given by

$$q(\phi'|\phi) = \tilde{p}(\phi'). \tag{12}$$

If the model were a perfect approximation, such that $\tilde{p}(\phi) = p(\phi)$ for all $\phi$, then $A(\phi \to \phi') = 1$ identically and 100% of proposals would be accepted. In a more realistic situation where there are discrepancies, the inefficiency of generating proposals with a probability proportional to $\tilde{p}(\phi)$ rather than $p(\phi)$ manifests itself through multiplicities in the Markov chain due to rejections, which are in turn measurable as autocorrelations. However, if proposals are drawn independently, then rejections are the *only* source of autocorrelation. As explained in Ref. [22], the autocorrelation at separation $t$ is given, *for all observables*, by

$$\frac{\Gamma_{\mathcal{O}}(t)}{\Gamma_{\mathcal{O}}(0)} = \Pr(t \text{ consecutive rejections})$$
$$= \mathbb{E}_{\phi \sim p}[(\mathbb{E}_{\phi' \sim \tilde{p}}[1 - A(\phi \to \phi')])^t]. \tag{13}$$

This is an extremely appealing feature that is not present in traditional algorithms, where local dynamics combined with energy barriers can lead to the decoupling of autocorrelation times for topological and nontopological observables [9]. An estimate of Eq. (13) is trivial to obtain from the accept-reject history of a Metropolis-Hastings simulation, with which we can compute an estimate for the integrated autocorrelation time that we denote $\tau_{\text{rej}}$.

Since Eq. (13) is strictly larger than the average rejection rate $1 - \mathbb{E}_{\phi \sim p}\mathbb{E}_{\phi' \sim \tilde{p}}[A(\phi \to \phi')]$ raised to the $t$th power, a lower bound on the integrated autocorrelation time can be given in closed form by a geometric series:

$$\tau_{\mathcal{O}} \geq \frac{1}{\mathbb{E}_{\phi \sim p}\mathbb{E}_{\phi' \sim \tilde{p}}[A(\phi \to \phi')]} - \frac{1}{2}. \tag{14}$$

This expression is not particularly useful *per se*, but we will be interested in how close to this lower bound the actual integrated autocorrelation falls.

Although this is not the approach we will take, reweighting can instead be done at the level of computing ensemble averages through a change of measure in Eq. (1) to $\mathcal{D}\phi\tilde{p}(\phi)w(\phi)$, where the reweighting factor $w(\phi) \equiv p(\phi)/\tilde{p}(\phi)$ is the same factor used in the Metropolis test [32]. The mean estimator from Eq. (3) then reads

$$\bar{\mathcal{O}} = \frac{\sum_{\phi \in \Phi} w(\phi)\mathcal{O}(\phi)}{\sum_{\phi \in \Phi} w(\phi)}. \tag{15}$$

Of course, while this approach makes use of all of the generated configurations, there is still a price to be paid for drawing samples from $\tilde{p}(\phi)$ rather than $p(\phi)$; the weights ensure that the number of configurations yielding nonnegligible contributions to the sum drops rapidly as the approximation $\tilde{p}(\phi) \approx p(\phi)$ degrades. As remarked on in Ref. [33], this *a posteriori* reweighting approach is appealing if $\mathcal{O}(\phi)$ is cheap to compute relative to the cost of generating configurations from the model.

## III. NORMALIZING FLOWS

For our purposes, we define a normalizing flow as a both-directions continuously differentiable bijective mapping,[4]

$$f_\theta \colon \mathcal{M}^{|\Lambda|} \to \mathcal{M}^{|\Lambda|}$$
$$z \mapsto \phi = f_\theta(z),$$

between "latent" random variables $z \sim r(z)$ and "candidate[5] field configurations" $\phi = f_\theta(z) \sim \tilde{p}(\phi)$.

We will immediately restrict ourselves to the special case of $\mathcal{M} = \mathbb{R}$, which applies to scalar $\phi^4$ theory.[6] Hence, the density associated with the candidate field configurations is given by the familiar formula for a change of variables, involving the Jacobian determinant:

$$\tilde{p}(f_\theta(z)) = r(z)\left|\frac{\partial f_\theta(z)}{\partial z}\right|^{-1}. \tag{16}$$

In practice, we only every require the logarithm of this equation.

We will draw latent variables from an uncorrelated Gaussian distribution:

$$r(z) = \prod_{x \in \Lambda} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-z_x^2/(2\sigma^2)}, \tag{17}$$

which one may interpret, in the spirit of Refs. [52,53], as a trivial limit of $\phi^4$ theory.[7] In principle one could put more effort into generating latent variables that reduce the workload for the flow.[8] However, even with *a priori* knowledge of some features of the target density this comes with a high risk of overengineering the problem, i.e., leading to marginal improvements in the model's approximation to the target, which are completely negated by the increased costs of generating samples and computing the density.

---

[4]This is precisely the definition of a $C^1$ diffeomorphism.
[5]We refer to the configurations generated by the model are referred to as "candidate" since they might be rejected by the Metropolis test.
[6]For discussion and examples of flows on non-Euclidean manifolds, see Refs. [58,59].
[7]Cf. Eq. (40) with $\lambda \to 0$ and $\beta \to 0$.
[8]In Sec. VIII B we briefly discuss the use of free fields as the latent variables.

Although the bijective construction is not the most flexible *a priori*, normalizing flows have several advantages over other generative models. Crucially, it is straightforward in principle to ensure that the density is tractable, by choosing a map whose Jacobian determinant $|\partial f_\theta(z)/\partial z|$ is tractable. It is this feature that provides us with a means of guaranteeing convergence to the correct target density through the Metropolis test. Additional benefits relate to the training, which is discussed in the next subsection. Finally, as an added bonus, the intermediate states of the flow correspond to valid probability densities in their own right, from which we can draw samples. In this sense, normalizing flows are more "interpretable" than other generative models which can behave more like a "black box."

### A. Training a flow model

Since normalizing flows are differentiable by construction, they can be trained using standard gradient-based optimization algorithms such as stochastic gradient descent. The algorithm used in this work is a variant that incorporates momentum, called ADAM [60]. The conventional approach to training is to expose the model to a set of data drawn from the distribution of interest, $p(\phi)$, via a separate process and tune the parameters of the model in order to optimize some objective function. If we were to take the conventional approach here, the set of training configurations would be divided into "batches," passed through the layers of the flow model in the reverse direction, and the resulting variables $f_\theta^{-1}(\phi)$ used to estimate the following objective function by averaging over the batch:

$$\hat{D}_{KL}(p\|\tilde{p}) = \mathbb{E}_{\phi \sim p(\phi)}\left[ -\log r(f_\theta^{-1}(\phi)) - \log \left| \frac{\partial f_\theta^{-1}(\phi)}{\partial \phi} \right| \right]$$
$$+ \text{irrelevant terms.} \qquad (18)$$

Equation (18) is an estimator for the Kullbach-Leibler divergence defined in Eq. (10), up to an unknown self-information term $\mathbb{E}_{\phi \sim p(\phi)}[\log p(\phi)]$ that does not depend on the model's parameters and is therefore irrelevant for the purposes of optimization.

However, for our purposes this strategy is clearly not satisfactory since the problem has gone full circle; the ability to train models would then be tied to the ability to generate a large representative samples of configurations to act as training data, which is exactly what we are prevented from doing by critical slowing down. Thankfully, an alternative path presents itself in the typical scenario where we are interested in sampling from a theory for which $S(\phi)$ is completely specified. In this training paradigm, favored by Ref. [22] and many subsequent studies, one considers the alternative definition of Kullbach-Leibler divergence following a reversal of the arguments with respect to Eq. (10):

$$D_{KL}(\tilde{p}\|p) = \int \mathcal{D}\phi\, \tilde{p}(\phi) \log \frac{\tilde{p}(\phi)}{p(\phi)}. \qquad (19)$$

This allows us to define an objective function that can be minimized using estimates based on configurations generated exclusively by the model:

$$\hat{D}_{KL}(\tilde{p}\|p) = \mathbb{E}_{z \sim r(z)}\left[ S(f_\theta(z)) - \log \left| \frac{\partial f_\theta(z)}{\partial z} \right| \right]$$
$$+ \text{irrelevant terms.} \qquad (20)$$

In Eq. (20), the irrelevant terms that do not depend on the model's parameters are $\mathbb{E}_{z \sim r(z)}[\log r(z)]$ and the normalizing factor in the path integral, $\log \mathcal{Z}$.

One could be forgiven for thinking that the difference between optimizing $\hat{D}_{KL}(p\|\tilde{p})$ and optimizing $\hat{D}_{KL}(\tilde{p}\|p)$ is no more than a matter of exchanging an preexisting training set for configurations drawn from the model. In practice, however, the two modes of training have distinct quirks which are important to appreciate. A recent contribution (Ref. [61]) includes a comparison of the two training schemes in situations where $p(\phi)$ is multimodal.

In fact, by insisting on not having to obtain training inputs from an external process, we have actually side-stepped several of the major difficulties normally faced during training. In particular, since each batch of training inputs is stochastically generated on demand, and never recycled, "overfitting" of training data is relegated to a nonissue. Furthermore, we need not be concerned about bias in the training inputs, since they are obtained through exact sampling from the latent distribution. The problem that we are most likely to encounter is one of insufficient flexibility to resolve all of the features in the target density, leading to a model which "underfits" the target. This is an important example of qualitatively different results arising from the choice of training scheme. Optimizing $\hat{D}_{KL}(p\|\tilde{p})$ typically results in "smoothed" approximations to the target, whereas the approach we take, optimizing $\hat{D}_{KL}(\tilde{p}\|p)$ by sampling from the model, has a tendency to fit (not necessarily all of) the modes of the target and set $\tilde{p}(\phi) \approx 0$ elsewhere. This behavior is explained and its implications discussed in Secs. VII B and VII C.

### B. Building flexible models

When considering potential transformations for the layers $g_i$, there are two conflicting requirements that will need to be met with a potentially very delicate compromise. Firstly, the flow will need to be highly flexible in order to start with uncorrelated Gaussian variables and distill the complex features of a system near to criticality, which will include nontrivial correlations on multiple scales. On the other hand, the Jacobian determinant in Eq. (16) must be tractable since we are still required to evaluate $\tilde{p}(\phi)$ in order to perform the reweighting that guarantees

convergence to the correct target. Furthermore, the speed at which models can be trained and sampled from will depend on the efficiency with which the (logarithm of) the Jacobian determinant can be computed. This is a significant constraint and one that is very much at odds with the goal of using invertible transformations to map simple densities to complex ones; it is challenging to define sufficiently expressive transformations without rendering the Jacobian term intractable.

The key component that initially enabled normalizing flows to become competitive with more flexible generative models at performing benchmark tasks (such as image synthesis) was a particular type of transformation now widely referred to as a *coupling layer* [50,51,62]. Coupling layers are essentially a template for building flexible, pointwise, invertible transformations that are guaranteed to have a triangular Jacobian matrix. One divides the inputs to a coupling layer into two groups, only one of which will actually undergo a nontrivial transformation that is conditioned on information derived from the remaining, non-transformed variables. Since we are interested in $\phi^4$ theory with a single degree of freedom at each lattice site, this splitting equates to an (arbitrary) partitioning of the lattice into $\Lambda^A$ and $\Lambda^P$, which we refer to as the "active" and "passive" partitions, respectively. The normalizing flow is then built out of several couplings layers by function composition: $f_\theta \equiv g_I \circ g_{I-1} \circ \cdots \circ g_2 \circ g_1$. Defining $v_1 \equiv z$ and $v_{I+1} \equiv \phi$ lets us write the action of the $i$th coupling layer as

$$g_i: \mathbb{R}^{|\Lambda|} \to \mathbb{R}^{|\Lambda|}$$
$$v_i \mapsto v_{i+1} = g_i(v_i),$$

where

$$v_{i+1,x} = \begin{cases} v_{i,x}, & x \in \Lambda_i^P, \\ C_{i,x}(v_{i,x}; \mathbf{N}_{i,x}(v_i^P)), & x \in \Lambda_i^A. \end{cases} \quad (21)$$

$v_i^P \in \mathbb{R}^{|\Lambda_i^P|}$ is a vector built out of elements $\{v_{i,x} | x \in \Lambda_i^P\}$, with $v_i^A \in \mathbb{R}^{|\Lambda_i^A|}$ defined analogously. The set of functions $C_i$ (which are as yet unspecified) transform the active partition and are conditioned on a set of parameters $\mathbf{N}_i(v_i^P)$ that are themselves functions of the passive variables. In the examples that we will consider here, these parameters are the output layer of one or more fully connected feed-forward neural networks.[9] Throughout this paper, neural network outputs, exclusively, will be denoted by bold letters, and it will be left to the presence or absence of

---

[9]There is an important distinction to be made between the neural network outputs $\mathbf{N}_{i,x}$ that parametrized a function $C_{i,x}$ and the "model parameters" $\theta$ that are tuned during training, which are the aggregation of all of the parameters (weights and biases) from each neural network, from each coupling layer.

indices (e.g., $i$ for the layer index and $x$ for the lattice sites) to specify the cardinality of sets (for example, $v_{i,x}$ is a number while $v_i$ is a vector with $|\Lambda|$ components).

The Jacobian for a coupling layer is, in block notation,

$$\frac{\partial g_i}{\partial v_i} = \begin{pmatrix} \mathbb{I} & 0 \\ \frac{\partial C_i}{\partial v_i^P} & \frac{\partial C_i}{\partial v_i^A} \end{pmatrix}, \quad (22)$$

where the lower-right block is understood to be a diagonal matrix whose diagonal elements are $\partial C_{i,x}/\partial v_{i,x}^A$ for each $x \in \Lambda_i^A$. As promised, this matrix is triangular, so the determinant is simply equal to the product of terms on the leading diagonal:

$$\left| \frac{\partial g_i}{\partial v_i} \right| = \left| \prod_{x \in \Lambda^A} \frac{\partial C_{i,x}}{\partial v_{i,x}} \right|. \quad (23)$$

By swapping the active and passive partition after every coupling layer ($\Lambda_{i+1}^{A/P} = \Lambda_i^{P/A}$) and composing at least three layers, we ensure that each lattice site is updated using information from every other one. Thus, coupling layers allow us to sample from correlated target densities using uncorrelated latent variables *at no additional expense in the computation of the Jacobian determinant*—the calculation is the same as it would be if we replaced the neural networks $\mathbf{N}_i$ with parameters that had no dependence on $v_i^P$.

Since $\det AB = \det A \det B$, a sequence of $I$ coupling layers induces the following Jacobian determinant:

$$\log \left| \frac{\partial f_\theta}{\partial z} \right| = \sum_{i=1}^{I} \log \left| \frac{\partial g_i}{\partial v_i} \right|. \quad (24)$$

These terms can be accumulated alongside the transformations of field variables, so that a single pass through all of the coupling layers yields both a set of candidate field configurations and the left-hand side of Eq. (24) for each configuration in the batch, ready to evaluate $\log \tilde{p}(\phi)$ (if sampling) or the objective function (if training).

Given freedom to divide the lattice in whichever way seems fit, we will implement a "checkerboard" partitioning featured in Fig. 1, which ensures that each lattice site is directly influenced by its closest neighbors. We will often refer to a pair of coupling layers, which together transform every degree of freedom once, as a "coupling block." From here on, we will drop the $A$ and $P$ superscripts and assume we are always talking about transforming a set of variables $v_i$ belonging to the active partition. Furthermore, we will denote the neural networks without an explicit dependence on the passive partition.
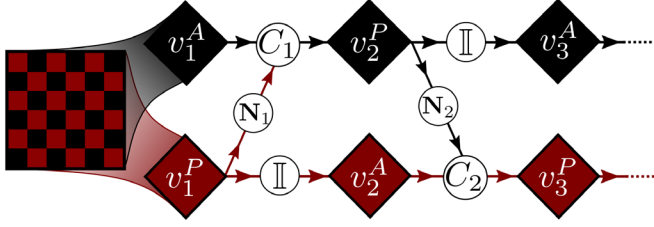
FIG. 1. Graphical representation of the first coupling block under the "checkerboard" partitioning scheme. The latent Gaussian variables $v_1 \equiv z$ are split into two partitions (the red and black nodes). The $i$th coupling layer transforms the active partition $C_i: v_i^A \mapsto v_{i+1}^P$ using information from the passive partition $v_i^P$ via the neural network(s) $\mathbf{N}_i(v_i^P)$. $\mathbb{I}$ denotes the identity transformation. Note that there is no need to merge the active and passive partitions until after the final coupling layer.

## C. Affine and additive transformations

Affine coupling layers were introduced by Dinh, Sohl-Dickstein, and Bengio [51] as part of the *Real NVP* (non-volume-preserving) architecture. The pointwise transformation multiplies and shifts each degree of freedom and is commonly written in vector form:

$$C_i^{\text{aff}}(v_i; \mathbf{s}_i, \mathbf{t}_i) = (v_i - \mathbf{t}_i) \odot e^{-\mathbf{s}_i}, \qquad (25)$$

where $\mathbf{s}_i$ and $\mathbf{t}_i$ are modeled by neural networks with $|\Lambda^A|$ outputs and $\odot$ is the elementwise product.

Using Eqs. (23) and (24), a single affine coupling layer contributes

$$\log \left| \frac{\partial g_i^{\text{aff}}}{\partial v_i} \right| = -\sum_{x \in \Lambda^A} \mathbf{s}_{i,x} \qquad (26)$$

to the logarithm of the Jacobian determinant. The precursor to Real NVP uses volume-preserving "additive" coupling layers [50], such that Eq. (25) reduces to the shift by $\mathbf{t}_i$ only:

$$C_i^{\text{add}}(v_i; \mathbf{t}_i) = v_i - \mathbf{t}_i. \qquad (27)$$

In our implementation of these coupling layers, we standardize the inputs to the neural networks such that they have unit variance and do not apply activation functions to the output layer of these neural networks. We also append a global rescaling transformation after all of the coupling layers have acted, which can have a learnable scale parameter.

As an inexpensive yet remarkably expressive flow architecture, Real NVP has achieved widespread success and is frequently taken as a benchmark model to which new flow models are compared. However, more sophisticated flows using more flexible transformations have since achieved superior results on a number of standard datasets

(mostly images)—see e.g., Refs. [62–70]. This motivates us to explore one of the prominent alternatives.

## D. Rational quadratic splines

Splines are functions defined piecewise by polynomials. Coupling layers using spline-based transformations were introduced in Ref. [66] and further developed by Durkan *et al.* in Refs. [68,69]. We will focus on the most flexible member of the family as described in Ref. [69], which is based on a continuously differentiable spline interpolant first considered in Ref. [71].

A rational quadratic spline (RQS) transformation $C_{i,x}^{\text{rqs}}$ is defined for a single degree of freedom by $K$ rational quadratics, referred to as the "segments" of the spline. These segments are joined end to end at a set of "knots" $\{(V_{i,x}^k, C_{i,x}^{\text{rqs}}(V_{i,x}^k)) | k = 0, \dots, K\}$ such that the result is a strictly monotonic, $C^1$-differentiable function on the interval $[-a, a]$, which will be chosen in order to contain essentially all of the probability mass.

Given a reference point, the parameterization provided by Ref. [71] requires $3K + 1$ strictly positive parameters to uniquely specify this function: the side lengths $(\mathbf{w}_{i,x}^k, \mathbf{h}_{i,x}^k)$ of the $K$ rectangles which have adjacent knots on their opposing corners (often referred to as "widths" and "heights") and the derivatives $\mathbf{d}_{i,x}^k$ at the $K + 1$ knots. Note that our choice of labeling, featured in Fig. 2, means that the end points of the $k$th segment are the $(k-1)$th and $k$th knots. This gives us a set of $|\Lambda^A| \times (3K + 1)$ parameters for the coupling layer:

$$\mathbf{N}_i = \{\mathbf{w}_{i,x}^k, \mathbf{h}_{i,x}^k, \mathbf{d}_{i,x}^k, \mathbf{d}_{i,x}^0 | k = 1, \dots, K; x \in \Lambda^A\}.$$

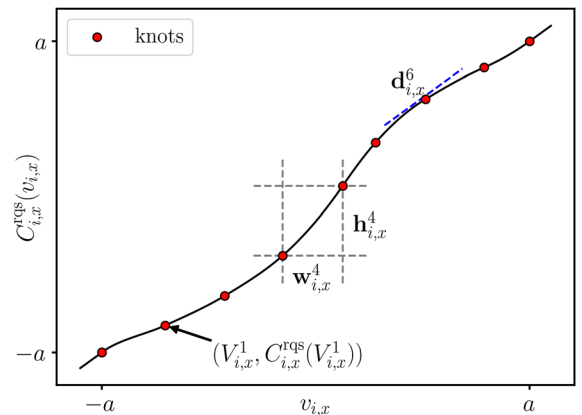For later convenience, define the slopes of the straight lines connecting adjacent knots as



FIG. 2. An example eight-segment rational quadratic spline transforming the degree of freedom at lattice site $x$. $\mathbf{w}_{i,x}^k$ and $\mathbf{h}_{i,x}^k$ are the widths and heights, respectively, of the rectangle containing the $k$th polynomial segment. $\mathbf{d}_{i,x}^k$ is the derivative at the $k$th knot.

$$\frac{C_{i,x}^{\mathrm{rqs}}(V_{i,x}^k) - C_{i,x}^{\mathrm{rqs}}(V_{i,x}^{k-1})}{V_{i,x}^k - V_{i,x}^{k-1}} = \frac{\mathbf{h}_{i,x}^k}{\mathbf{w}_{i,x}^k} \equiv \mathbf{s}_{i,x}^k \geq 0, \qquad (28)$$

and reexpress the variables being transformed as

$$\frac{v_{i,x} - V_{i,x}^{\ell-1}}{\mathbf{w}_{i,x}^\ell} \equiv \alpha_{i,x} \in [0,1], \qquad (29)$$

which corresponds to the fractional position of $v_{i,x}$ within the specific segment in which it is located, whose index we

label $\ell$. Note that each degree of freedom $v_{i,x}$ must first be sorted into the appropriate segment (i.e., the value of $\ell$ determined) using e.g., bisection search, which is not too expensive since the knots are already sorted into ascending order.

Using the 0th knot at $(V_{i,x}^0, C_{i,x}^{\mathrm{rqs}}(V_{i,x}^0)) = (-a, -a)$ as the reference point, the RQS transformation and its gradient can then be written, for each degree of freedom, using Eqs. (30) and (31):

$$C_{i,x}^{\mathrm{rqs}}(v_{i,x}; \mathbf{N}_{i,x}) = \underbrace{-a + \sum_{k=1}^{\ell-1} \mathbf{h}_{i,x}^k}_{V_{i,x}^{\ell-1}} + \frac{\mathbf{h}_{i,x}^\ell[\mathbf{s}_{i,x}^\ell \alpha_{i,x}^2 + \mathbf{d}_{i,x}^{\ell-1}\alpha_{i,x}(1-\alpha_{i,x})]}{\mathbf{s}_{i,x}^\ell + (\mathbf{d}_{i,x}^{\ell-1} + \mathbf{d}_{i,x}^\ell - 2\mathbf{s}_{i,x}^\ell)\alpha_{i,x}(1-\alpha_{i,x})}, \qquad (30)$$

$$\frac{1}{\mathbf{w}_{i,x}^\ell}\frac{dC_{i,x}^{\mathrm{rqs}}}{d\alpha_{i,x}} = \frac{(\mathbf{s}_{i,x}^\ell)^2[\mathbf{d}_{i,x}^\ell \alpha_{i,x}^2 + 2\mathbf{s}_{i,x}^\ell \alpha_{i,x}(1-\alpha_{i,x}) + \mathbf{d}_{i,x}^{\ell-1}(1-\alpha_{i,x})^2]}{[\mathbf{s}_{i,x}^\ell + (\mathbf{d}_{i,x}^{\ell-1} + \mathbf{d}_{i,x}^\ell - 2\mathbf{s}_{i,x}^\ell)\alpha_{i,x}(1-\alpha_{i,x})]^2}. \qquad (31)$$

Taking the logarithm of Eq. (31) and summing over all $x$ in the active partition yields the contribution to the logarithm of the Jacobian determinant from one RQS coupling layer.

The advantage of using this parameterization should now be clear; all that is required to guarantee that the gradient is strictly positive is that every parameter in $\mathbf{N}_i$ is also strictly positive. It is also particularly simple to enforce the desired normalization:

$$\sum_{k=1}^K \mathbf{w}_{i,x}^k = \sum_{k=1}^K \mathbf{h}_{i,x}^k = 2a. \qquad (32)$$

The function defined by Eq. (30) is an interpolant for the set of knots,[10] so the problem of representing complicated transformations reduces to one of generating a sufficient number of knots with sufficient accuracy.

After fixing $\mathbf{d}_i^0 = \mathbf{d}_i^K = \{1\}^{|\Lambda^A|}$, we let a *single* neural network generate the remaining $|\Lambda^A| \times (3K-1)$ parameters in the RQS layer, using the $|\Lambda^P|$ field variables in the passive partition as inputs. We take the unconstrained outputs of the neural net—denoted below with a hat— and split them into widths, heights, and derivatives. Positivity of the $\mathbf{h}_{i,x}^k$ and $\mathbf{w}_{i,x}^k$, as well as the correct normalization, is enforced by passing the unconstrained widths and heights through a "softmax" activation function:

$$\mathbf{h}_{i,x}^k = \frac{e^{\hat{\mathbf{h}}_{i,x}^k}}{\sum_{k'=1}^K e^{\hat{\mathbf{h}}_{i,x}^{k'}}} \times 2a. \qquad (33)$$

The derivatives are instead passed through a "softplus":

_____

[10]To see this, substitute $\alpha_{i,x} = 0$ or 1.

$$\mathbf{d}_{i,x}^k = \log(1 + e^{\hat{\mathbf{d}}_{i,x}^k}). \qquad (34)$$

To ensure that the inputs of a spline layer fall within the interval $[-a, a]$, we generally chose $a = 5$ and standardized the inputs before the first RQS layer by dividing them by the standard deviation, taken over both the batch and the lattice sites. To catch the edge cases of inputs falling outside of this interval, we extended the definition of the transformation to be the identity outside of $[-a, a]$ while fixing the derivatives at the external knot points to be unity to ensure that the transformation remains everywhere differentiable.

### E. Neural networks

The principle behind this entire approach is that correlations in the target density can be encoded in the weights and biases of neural networks. Once these weights and biases have been fixed (i.e., once we have finished training), a neural network is simply a function. In this case, the role of these functions is to take a set of field variables (the passive partition) as inputs and use this information to return a set of parameters that govern the transformation of a different set of field variables (the active partition).

There are theoretical grounds to believe that this works; specifically, there is a universal approximation theorem implying that any well-behaved function defined on a compact subspace of $\mathbb{R}^{|\Lambda^P|}$ can be approximated with arbitrary accuracy by functions of the form

$$\mathbf{N}_x(v^P) = \sum_{j=1}^H A_{xj}^{(2)} \sigma\left(\sum_{x' \in \Lambda^P} A_{jx'}^{(1)} v_{x'}^P + b_j^{(1)}\right) \qquad (35)$$

which represent feed-forward neural networks with a single hidden layer containing $H$ elements and a

nonlinear activation function $\sigma$ [72]. $A^{(1)}$ and $A^{(2)}$ are matrices containing the network weights, and $b^{(1)}$ is a vector of biases.

This theorem does not tell us how large the hidden layer ought to be in order to represent the desired function with a given accuracy, though the number will be related to the degree of nonlinearity and sensitivity with which the function must respond to variations in its inputs. Coupling layers will need to be able to dramatically alter the field variables in response to their closest neighbors while also accounting for more subtle corrections due to field at larger separations, which implies that the functions being modeled by the neural networks must be extremely sensitive. However, there is no hard-and-fast rule dictating the optimal size and depth of neural networks for our specific problem; this must be discovered through experimentation.

Universal approximation theorems have also been proven, quite recently, for so-called (deep) convolutional neural networks [73–75]. The building blocks of these networks are convolutions, denoted by $\star$, with a set of "kernels," $W$, whose weights are trainable parameters:

$$W \star v = \sum_{k_1,k_2=0}^{L-1} W(k) v((x-k)_{\mathrm{mod}\ L}). \qquad (36)$$

In the above, $k = (k_1, k_2)$ and lattice coordinates have temporarily been written as arguments rather than the usual subscripts. A "stride" size of one is also implied, meaning that the convolution is applied at every lattice site. In practice, $W$ is usually taken to be nonzero only inside a small window $-K \leq k_1, k_2 \leq K$, where $K$ tends to be rather

small indeed (often just one or two). Since $K$ represents the largest distance over which correlations can be encoded into the trainable weights of any given convolutional layer, theories with long correlation lengths $\xi \gg K$ require convolution-based models to be sufficiently *deep* in order to indirectly model correlations over large scales.

Rather than a number of nodes $H$ in the hidden layers of a fully connected network, we may specify a number of input and output "channels," each of which has its own convolutional kernel. Since we are dealing with a single-component scalar field, the first layer will have one input channel. The number of output channels will depend on how many numbers are required to parametrized the transformation of each degree of freedom, i.e., one for additive layers, two for affine layers, and $3K - 1$ for the splines. Note that, unlike the fully connected case, the geometry of the objects being convolved must remain intact; one cannot simply pass the passive elements $v^P$ into a convolutional network as an arbitrarily ordered one-dimensional vector, as we have previously been doing. Following Ref. [34], we pass two-dimensional configurations into the convolutional networks, but with zeros as placeholder values for the active partition to maintain the diagonal Jacobian structure of the coupling layer.

Let $M_{\Lambda^P}$ denote an $L \times L$ matrix with ones at the positions corresponding to the *passive* partition and zeros at the *active* lattice sites. Further, let $c_i = 1, \ldots, H_i$ label the channels in the $i$th layer of the network, and $\sigma$ and $b^{(i)}$ once again represent activation functions and vectors of biases, respectively. The depth-$D$ convolutional networks tested here can then be written as

$$\mathbf{N}_{c_D}(v) = \sum_{c_{D-1}=1}^{H_{D-1}} W_{c_D,c_{D-1}}^{(D)} \star \sigma \left( \ldots \sum_{c_1=1}^{H_1} W_{c_2,c_1}^{(2)} \star \sigma (W_{c_1}^{(1)} \star (M_{\Lambda^P} \odot v) + b_{c_1}^{(1)}) + b_{c_2}^{(2)} \ldots \right) + b_{c_D}^{(D)}. \qquad (37)$$

In the present work we have limited the scope of our quantitative study to models using fully connected networks as described by Eq. (35), supplying only indicative examples of models using convolutional networks as described above.

### F. Enforcing equivariance

Convolutions are frequently favored over linear transformations because they possess the very desirable property that translations of the inputs induce nothing more than translations of the outputs, a trait often referred to as *equivariance* (with respect to translations).[11] Hence,

convolutional networks are not required to "learn" that inputs related by a global translation should be considered equivalent.

In addition to the symmetries of the lattice, $\phi^4$ theory possesses a $\mathbb{Z}_2$ symmetry corresponding to invariance of the action under a global sign reversal of the field:

$$p(-\phi) = p(\phi). \qquad (38)$$

Given that equivariant maps are those which commute with the symmetry transformation, it is not difficult to show that, for the $\mathbb{Z}_2$ symmetry, the equivariant maps are *odd* functions and that equivariance of $f_\theta$ requires the coupling layers $g_i(v_i)$ to be equivariant, meaning that the transformations satisfy

$$C_{i,x}(-v_{i,x}^A; \mathbf{N}_{i,x}(-v_i^P)) = -C_{i,x}(v_{i,x}^A; \mathbf{N}_{i,x}(v_i^P)). \qquad (39)$$

---

[11]The translational equivariance of the convolutional networks described by Eq. (37) deserves comment. These networks are really equivariant with respect to the discrete group of translations that are isometries of the checkerboard *sublattices*. A consequence of the checkerboarding is that we sacrifice equivariance under the full group of lattice translational isometries.

In our models, $\mathbf{N}_i(v^P)$ are (almost always) fully connected feed-forward networks as defined by Eq. (35), which are odd functions if we drop the biases and use odd activation functions (e.g., tanh) [32]. If we make these choices for the neural networks in the affine coupling layers $\mathbf{s}_i$ and $\mathbf{t}_i$, then Eq. (39) is trivially satisfied by implementing one additional step, that is to take the absolute value of the output of the $\mathbf{s}_i$ network.

Enforcing $\mathbb{Z}_2$ equivariance in the RQS transformations is less straightforward; the terms in Eq. (30) cannot all be simultaneously odd. We implemented a rather crude workaround that involves splitting the batch of latent variables according to sgn $\sum_{x \in \Lambda} z_x$ (i.e., the initial "magnetization" of each configuration) and treating the two groups slightly differently within the transformation. The key observation is that if we take $a$ as 0th knot (instead of $-a$) and construct the spline in the reverse direction, then this is equivalent to taking $C_{i,x}^{\mathrm{rqs}} \mapsto -C_{i,x}^{\mathrm{rqs}}$. Practically speaking, it is simpler to simply reverse the ordering of the $k$ indices in the network outputs $\mathbf{h}_i$, $\mathbf{w}_i$ and $\mathbf{d}_i$ for one of the two groups.

Unfortunately, although the equivariance condition is satisfied, this approach is not entirely legitimate, since the result is a transformation that is not a continuous function of the inputs, thereby failing to satisfy the conditions required for Eq. (16). To see this, observe that $C_{i,x}(v_{i,x}; \mathbf{N}_{i,x}(v_i^P))$ may take two possible values for a fixed $v_{i,x}$ and $v_i^P$, experiencing a discontinuous jump when changes in the *active* partition cause the overall magnetization of $z$ to flip sign. So, while we include results using these "equivariant splines," we do not recommend using this prescription in future.

It is worth bearing in mind that enforcing symmetries is not absolutely necessary; the Metropolis-Hastings algorithm is guaranteed to converge to the correct target, and therefore reproduce all of its symmetries, as long as the transition kernel is ergodic. We remind the reader that a sufficient condition is $\tilde{p}(\phi) > 0 \; \forall \; \phi \in \mathbb{R}^{|\Lambda|}$ [56] (so that every configuration has a finite probability of being generated) and that this is guaranteed [for a sensible choice of $r(z)$] since $f_\theta$ is a bijection. Nevertheless, a guiding principle of optimization is that it is generally more efficient to enforce known constraints by construction, and benefits of doing so for normalizing flows have been reported in Refs. [31,33,76].

## IV. RELATED WORK

The first demonstration of a normalizing flow forming the basis of a sampling algorithm for lattice field theory was provided by Albergo, Kanwar, and Shanahan [22] for two-dimensional $\phi^4$ theory, using the Real NVP architecture described in Sec. III C. Still with $\phi^4$ as the target theory, Nicoli *et al.* [32] used an even more bare-bones flow where the coupling layers simply shift the field variables in such a way that the symmetry under $\phi \to -\phi$ is preserved. Our work draws on ideas from both of these studies, though we pivot in the opposite direction with respect to Ref. [32] by using coupling layers that are *more* flexible than those in Real NVP. More recent work along these lines has been undertaken by Hackett *et al.*, who compared several optimization strategies for flow-based sampling from bimodal distributions, including $\phi^4$ in its broken phase.

Further progress has mostly been on the side of developing the necessary machinery to apply these ideas to lattice *gauge* theories—specifically, those that are invariant under local U($N$) or SU($N$) transformations. Rezende *et al.* [59] explored several possible approaches to using normalizing flows in cases where the field variables are defined on an $n$-sphere or $n$-torus. A procedure for constructing normalizing flows that are equivariant under gauge transformations was initially developed by Kanwar *et al.* [31] for the U(1) case and then extended to SU(2) and SU(3) by Boyda *et al.* [33]. By definition, a gauge-equivariant flow is one that commutes with the action of the gauge group, which implies that gauge *invariance* is preserved by the flow. Hence, representative samples of gauge fields can be generated using latent variables drawn from the uniform (Haar) measure for the gauge group and passing them through a gauge-equivariant flow. More recently yet, Albergo *et al.* [77] developed the flow-based approach to sampling from theories with dynamical fermions. A code-based introduction to these methods has been provided by Albergo *et al.* [34], which we made use of when implementing convolution-based flow models.

Another recent and highly relevant contribution was made by Lawrence and Yamauchi [35] who showed that, in certain cases at least, it is possible to use a normalizing flow to sample from a theory possessing a "sign problem," which is to say the action is complex and $\exp(-S)$ cannot be interpreted as a measure of probability.

Several alternative ideas that involve training parametric models to perform collective updates predate the use of normalizing flows. For example, in the "self-learning Monte Carlo" method [78] the parametric model describes an effective action for a spin system with $n$th nearest neighbor interactions whose couplings have been inferred from pregenerated training data, which can then be used to generate Wolff cluster updates [16]. Restricted Boltzmann machines (RBMs) have been embedded in traditional MCMC algorithms [25,26], though training the RBM requires pregenerated configurations and its sampling procedure (Gibbs sampling) introduces its own autocorrelation. Generative adversarial networks (GANs), which can be more flexible than normalizing flows but for which $\tilde{p}(\phi)$ is defined implicitly and cannot be directly computed, have also been used to generate candidate field configurations but require a lot of additional machinery on top of the GAN itself to ensure that the distribution being sampled from is close to the correct one [27,29] or otherwise estimate the discrepancy [30].

While there have been substantial advances in the use of machine learning to extract physical information for lattice field theories, the generation of samples from some approximation of the true path integral has generally come as an add-on when the tool being used is a generative model. In contrast, the key strength of normalizing flows is the explicit and tractable density $\tilde{p}(\phi)$ which makes exact sampling possible using the Metropolis test.

## V. EXPERIMENTAL SETUP

### A. Field theory and observables

For the main part of our study we used the following action:

$$S(\phi) = \sum_{x \in \Lambda} \left[ -\beta \sum_{\mu=1}^{2} \phi_{x+e_\mu} \phi_x + \phi_x^2 + \lambda(\phi_x^2 - 1)^2 \right], \quad (40)$$

which describes a discretized analog of two-dimensional scalar $\phi^4$ theory with dimensionless couplings $\beta$ and $\lambda$, defined on a periodic lattice $\Lambda$, using $e_\mu$ to denote a unit lattice vector in the $\mu$th dimension. Experiments with the noninteracting theory used the "standard" action described in Appendix A, which is given by Eq. (A6) with $g_0 = 0$. We focus on isotropic lattices with $6^2 \leq |\Lambda| \leq 20^2$ sites.

A nice feature of the parameterization given above is that the limit $\lambda \to \infty$, $\phi^2 \to 1$ is very clearly identified as the Ising model at temperature $T \equiv \beta^{-1}$. Indeed, in the continuum limit $\phi^4$ theory belongs to the Ising universality class, with spontaneous breaking of the $\phi \mapsto -\phi$ symmetry occurring along a critical line $(\lambda, T_c(\lambda))$ in the space of couplings. Defining the reduced temperature $t = \frac{T - T_c(\lambda)}{T_c(\lambda)}$, the asymptotic behavior of observables as the system approaches criticality is described by power-law dependence on $t$. For example, the magnetic susceptibility diverges as $\chi \sim t^{-\gamma}$, and the correlation length diverges with a different critical exponent, $\xi \sim t^{-\nu}$. Eliminating $t$, we see that $\chi \sim \xi^{\gamma/\nu}$.

In a finite volume observables depend on both the couplings and the system size in a nontrivial manner, and their behavior in the critical region is described by finite-size scaling. For example, the susceptibility in a volume of linear extent $L$ can be written in the following manner:

$$\chi = \xi^{\gamma/\nu} g_\chi(L/\xi), \quad (41)$$

in which finite-volume effects have been bundled into a dimensionless scaling function $g_\chi(L/\xi)$, which we notice must tend toward a constant value as $L \to \infty$ and approach $(L/\xi)^{\gamma/\nu}$ for $L \ll \xi$ so as to act as a cutoff.

We now need to specify how we actually measure observables on the lattice. The basic building blocks are the two-point correlation function

$$G(y) = \frac{1}{|\Lambda|} \sum_{x \in \Lambda} \langle (\phi_{x+y} - \langle \phi \rangle)(\phi_x - \langle \phi \rangle) \rangle \quad (42)$$

and its Fourier transform

$$\tilde{G}(q) = \sum_{y \in \Lambda} e^{iq \cdot y} G(y). \quad (43)$$

We have used the translation invariance of Eq. (40) to take a volume average in Eq. (42) for the simple reason that it improves the statistics.

The susceptibility is identified with $\tilde{G}(0)$, but in the classical spin setting it is often expressed in terms of the magnetization $M(\phi) = \sum_{x \in \Lambda} \phi_x$:

$$\tilde{G}(0) \equiv \chi = \frac{1}{|\Lambda|} \langle (M - \langle M \rangle)^2 \rangle. \quad (44)$$

Estimators for these observables are easily obtained by exchanging $\langle . \rangle$ for a sample mean and uncertainties estimated using the bootstrap method [79,80]. However, without explicitly breaking the $\mathbb{Z}_2$ symmetry one will always measure $\langle \phi \rangle = |\Lambda|^{-1} \langle M \rangle = 0$, so if one is interested in the phase transition one can compute separate sample averages for configurations with positive and negative magnetization, to properly account for the fact that the field variable distribution is bimodal.

The correlation length requires a little more work to measure. It is the longest mode in the spectrum of $\sum_{x_1=0}^{L-1} G(x_1, x_2)$, the correlation function in time-momentum representation, at momentum $q_1 = 0$. For sufficiently large separations $x_2$, this takes the form of a pure exponential (a cosh due to lattice periodicity):

$$\sum_{x_1=0}^{L-1} G(x_1, x_2) \equiv \hat{G}(x_2) \propto \cosh\left(\frac{x_2 - L/2}{\xi}\right), \quad (45)$$

from which the correlation length can be extracted through a fit[12] or by computing

$$\xi^{-1}(x_2) = \text{arcosh}\left(\frac{\hat{G}(x_2 + 1) + \hat{G}(x_2 - 1)}{2\hat{G}(x_2)}\right). \quad (46)$$

In general, this can be challenging due to low signal-to-noise ratio at large separations, but with $L \leq 20$ we also suffer from having very few data points to fit. To slightly improve the situation, we average over the two dimensions when computing Eq. (46).

Another option exploits the fact that the lattice propagator takes the form $\tilde{G}(q) \propto (\sum_\mu 4 \sin^2(q_\mu/2) + \xi^{-2})^{-1}$ in the low-momentum limit, which lets us write [82]

---

[12] One might hope to fit a sum of exponentials and hence refrain from discarding short separations, but unfortunately this is an ill-conditioned problem [81].

$$\xi^2 = \frac{1}{2}\sum_{\mu=1}^{2}\frac{1}{4\sin^2(\pi/L)}\left(\frac{\tilde{G}(0)}{\mathrm{Re}\tilde{G}(\hat{q}_\mu)}-1\right). \quad (47)$$

Here, $\hat{q}_1 = (2\pi/L, 0)$ and $\hat{q}_2 = (0, 2\pi/L)$ are the smallest possible nonzero momenta, and we have used $\tilde{G}(q) + \tilde{G}(-q) = 2\,\mathrm{Re}\tilde{G}(q)$.

Our intention will be to tune the couplings so as to obtain systems with correlation length $\xi = L/4$, meaning that as we increase the lattice size we are studying essentially the same theory with an increasingly fine resolution. The purpose of doing this is so that we only see the effect of the number of degrees of freedom on algorithmic efficiency, as we keep the physical size in units of the correlation length constant. The choice proportionality constant (four) is a reasonable trade-off between the rate at which criticality is approached as we increase $L$ and the size of finite-volume effects contained within scaling functions. Fixing $\lambda = 0.5$ and allowing $\beta$ to vary, we obtained three separate predictions for the value of $\beta$ that corresponded to $\xi = L/4$ on the symmetric side of the phase transition. These values are provided in Table I.

### B. Model details

When investigating the scaling of training costs (Sec. VI D), we used normalizing flows that are a specific hybrid of affine coupling layers and rational quadratic splines, with the parameters of the transformations generated by fully connected feed-forward neural networks containing a single hidden layer of size $H = |\Lambda|$. In Sec. VI C we report on the observations that led us to converge on this particular design.

The metric we use to measure the quality of trained models is the average rate at which configurations generated by the model are accepted when used as proposals for a Metropolis-Hastings simulation. In Sec. VI A we verify that this acceptance rate entirely governs the integrated autocorrelation times of the resulting Markov chains, as claimed in Sec. II B and specifically Eq. (13).

We used the ADAM optimization algorithm [60] to update the parameters of our models. The step size or "learning rate" was annealed during training according to a cosine schedule:

$$\eta_t = \frac{\eta_0}{2}\left[1 + \cos\left(\frac{t}{T}\pi\right)\right], \quad (48)$$

where $T$ is the total number of training iterations. Note that this learning schedule requires that we specify $T$ before training begins. There are no additional "stopping criteria." The ADAMW variant [83] along with "warm restarts" [84] (which amount to resetting $t = 0$) is a useful generalization which permits us to continue training (perhaps with a larger batch size) if we are not happy with the outcome after $T$ iterations. After some experimentation with faster initial learning rates, which typically resulted in lower acceptances if the number of training iterations was large, we generally opted for $\eta_0 = 0.001$.

The batch size, i.e., the number of configurations used to estimate the objective function at each training iteration, varied from 250 to 32000 configurations. In the vast majority of cases the difference between the batch size and the number of training iterations was a factor of 1, 2 or 4. Note that these batch sizes are much larger than those conventionally used in stochastic optimisation. In fact, it is quite typical to intentionally aim for a highly stochastic trajectory through the space of parameters, by using a very small number of training inputs (as low as two in Ref. [85]) for each update of the model's parameters. This may seem surprising, particularly since the graphical processing units (GPUs) on which these models are run are entirely optimized for highly parallel computations, so a small batch size is an underutilization of these capabilities. The motivations behind this choice are that the stochasticity reduces the tendency of the model to overfit the training inputs or otherwise get stuck in local optima [85,86] and tends to find "better" global optima [87]. However, we have no reason to prefer small batch sizes *a priori*; as explained in Sec. III A, the problem of overfitting training inputs does not apply to us, and we expect the issue of local optima to be alleviated, to some extent, thanks to stochasticity inherited from the random number generator that produces our training inputs.

Unless stated otherwise, one can assume the following for all models presented in the remainder of this paper.
   (i) The $\phi^4$ couplings are given by Table I.
  (ii) The flow comprises a number of affine coupling blocks followed by a single rational quadratic spline coupling block.

TABLE I. $\phi^4$ couplings and correlation length measurements for the main part of our study. The inverse temperature $\beta$ was tuned such that $\xi \approx L/4$ for each lattice size.

| $L$ | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 |
|---|---|---|---|---|---|---|---|---|
| $\lambda$ | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| $\beta$ | 0.537 | 0.576 | 0.601 | 0.616 | 0.626 | 0.634 | 0.641 | 0.645 |
| $\xi$ (fit) | 1.57(2) | 2.05(5) | 2.53(2) | 3.10(8) | 3.40(4) | 4.03(9) | 4.56(5) | 5.1(2) |
| $\xi$ (46) | 1.525(3) | 2.005(2) | 2.529(2) | 3.013(3) | 3.471(5) | 3.940(3) | 4.502(6) | 4.903(9) |
| $\xi$ (47) | 1.501(3) | 1.990(2) | 2.524(3) | 3.010(5) | 3.487(8) | 3.970(4) | 4.555(9) | 4.96(1) |

(iii) $\mathbb{Z}_2$ equivariance is enforced in the affine and additive coupling layers, as described in Sec. III F.

(iv) The splines have eight segments and do not have $\mathbb{Z}_2$ equivariance enforced.

(v) Neural networks are of the fully connected kind with a single hidden layer containing exactly $|\Lambda|$ (i.e., $L^2$) elements, as defined in Eq. (35) with $H = |\Lambda|$.

(vi) We do not apply an activation function to the output layer of the **s** and **t** networks in the affine (or additive) layers.

(vii) Metropolis-Hastings simulations ran for $10^5$ steps.

(viii) In figures, data points and error bars are an average and range taken over three identical models with different random initializations.

Our code, ANVIL [88], is publicly available. It uses the PyTorch library [89] for constructing and training models, and Reportengine [90], a declarative framework for performing scientific analysis.

## C. Summary of the procedure

A training iteration consists of the following steps.

(i) Sample from Eq. (17) to generate a batch of $N$ "latent configurations"—$\{z^{(1)}, z^{(2)}, \ldots, z^{(N)}\}$, where $z^{(n)} \sim r(z)$—with each configuration $z^{(n)}$ comprising $|\Lambda|$ uncorrelated Gaussian numbers.

(ii) Pass these variables through the layers of the model, calculating the logarithm of the Jacobian determinant, $\log |\partial g_i/\partial v_i|$, for each layer $g_i$ as it transforms one of the two partitions. This results in $N$ candidate field configurations and $N$ Jacobian determinants $\log |\partial f_\theta/\partial z|$ corresponding to the full transformation $\phi = f_\theta(z)$.

(iii) Compute the action [Eq. (40)] for the batch of candidate field configurations.

(iv) Average the action and Jacobian over the batch, to provide an estimate of the reverse Kullbach-Leibler divergence [Eq. (20)].[13]

(v) Update the parameters of the model by a small increment in the direction of steepest gradient using the ADAM or ADAMW optimization algorithms.

Once we have a trained model, we move onto the sampling. We generate a large sample of candidate configurations from the model, along with their Jacobian terms, and immediately calculate the quantity $\log w(\phi) = -\log \tilde{p}(\phi) - S(\phi)$ for each candidate configuration. We are now fully equipped to run a Metropolis-Hastings simulation as described in Sec. II; for the Metropolis test we simply exponentiate $\log w(\phi') - \log w(\phi)$ to obtain the

---

[13]Strictly speaking we require the *gradient* of Eq. (20) with respect to the parameters of the model. This is performed automatically by PyTorch's "autograd" machinery [89]. Gradients are propagated through neural networks using the backpropagation algorithm [91].
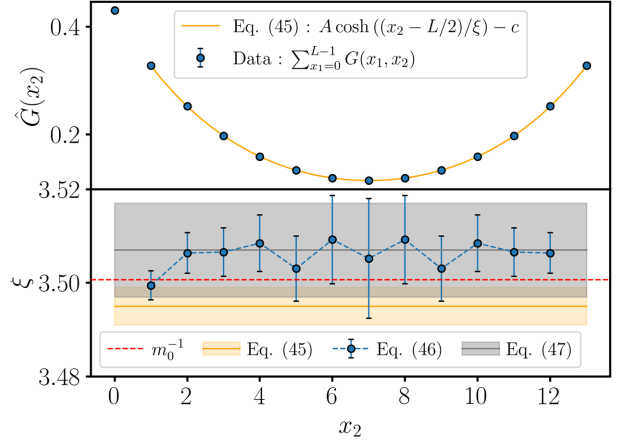


FIG. 3. Estimates of the correlation length extracted from a sample of $10^6$ field configurations generated by a model trained against Eq. (A6), the action from a free theory with $L = 14$ and $m_0 L = 4$. When this model was used as a generator of proposals for a Metropolis-Hastings simulation, the acceptance rate was 93%. However, the sample used here is the "raw" output from the model, *without applying the Metropolis test*. Error bars are estimated using the bootstrap method.

acceptance probability $A(\phi \to \phi')$ [see Eq. (9) with $q(\phi|\phi') = \tilde{p}(\phi)$ and $p(\phi) \propto \exp(-S(\phi))$].

## VI. RESULTS

### A. Proof of principle

As a basic check that the types of models described in Sec. III have the capacity to encode the information necessary to trivialize field theories, we trained a set of models to generate free fields. For this, we found that a sequence of 2–4 blocks of additive transformations performed on equal par with the more flexible affine and spline layers. Figure 3 demonstrates that the candidate field configurations generated by models with very high acceptance rates are indeed representative of the desired field theory.

In this case we know exactly what is required of $f_\theta$; it must perform a rescaling of the latent degrees of freedom followed by a Fourier transform to real space. However, it would be wrong to suppose that this is a trivial exercise, because the map is built out of a peculiar set of transformations that individually transform half of the degrees of freedom, conditioned on the other half, and the actual transformation learned by the model does not, and *cannot*, decompose into the simple steps described above.[14]

Moving onto the $\phi^4$ theory, we fixed $\lambda = 0.5$ and trained hybrid affine-spline models at various values of the inverse temperature $\beta$ so as to cross the phase transition. In this

---

[14]We are interested in gaining a deeper understanding of the transformation learned by the model but leave this matter to further investigation.
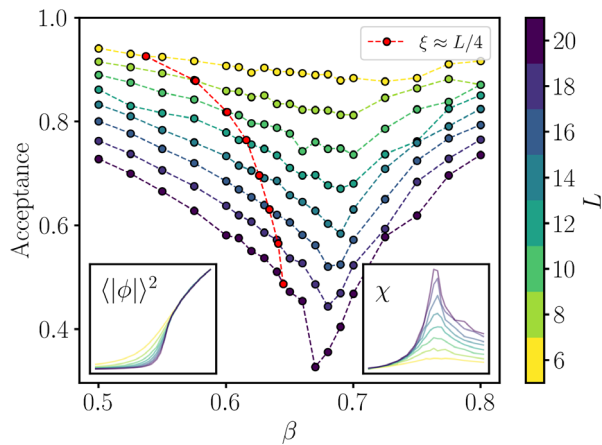
FIG. 4. Metropolis-Hastings acceptance rates for a set of models trained at different values of the inverse temperature $\beta$, crossing the critical temperature ($\beta \approx 0.67$ for largest lattice). The red points correspond to temperatures used in our main study (values provided in Table I). The inset figures show the magnetization and susceptibility over the same range of $\beta$. Models consisted of a block of affine layers followed by a spline block. The affine layers had $\mathbb{Z}_2$ equivariance enforced as described in Sec. III F, but this was only true of the spline layers in the low-temperature phase (see Fig. 12 for explanation). For each lattice size, models were identical (as outlined in Sec. V B) and were trained in an identical fashion (16000 iterations with a batch size of 16000).

study, emphasis was placed on like-for-like comparison of models trained against different targets rather than maximizing the acceptance rate. Figure 4 shows that high acceptance rates are possible in both the symmetric and the broken phase of $\phi^4$ when using a unimodal Gaussian prior. We also see that, as should be expected, trivializing the theory becomes increasingly challenging as the phase transition is approached. Though this is still interesting, one should be cautious when interpreting Fig. 4. In reality the problem is probably much easier for short correlation lengths than implied by these results, in part because the fully connected neural networks will contain a high level of redundancy since many degrees are effectively decoupled.

As an aside, we found that this sort of "parameter scan" can be performed efficiently using a single model that is initially trained at a high temperature, by adjusting the temperature over a sequence of training phases; in other words, a model trained at temperature $\beta_0$ can be retrained at temperature $\beta_0 + \delta\beta$ with relatively little effort. A thorough investigation into the potential of this feature was recently provided in Ref. [61] (see "adiabatic retraining").

## B. Acceptance rates and autocorrelation times

In Fig. 5 we used the traditional approach to estimating integrated autocorrelation time, based on autocorrelations in the magnetization of each configuration in the Markov chain, which is described in Appendix B. We now compare
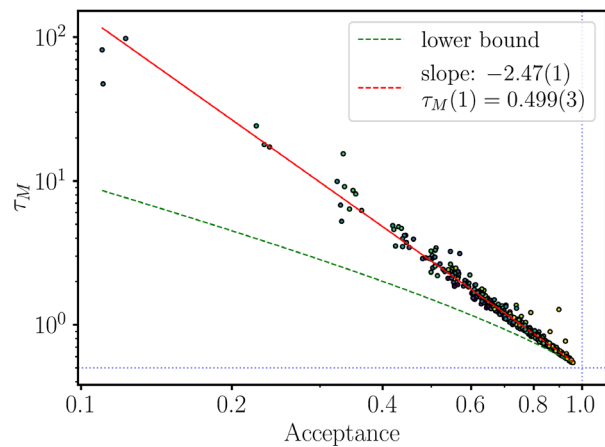


FIG. 5. Relationship between Metropolis-Hastings acceptance fraction and integrated autocorrelation time. Both the gradient and the intercept were left unconstrained in the least-squares fit. The theoretical lower bound from Eq. (14) is also plotted in green, for comparison. The data were obtained by sampling from models trained on a broad range of systems, including those specified in Table I, those depicted in Fig. 4, and others trained during various experiments.

this with the observable-independent estimator $\tau_{\mathrm{rej}}$, defined by Eqs. (13) and (4). The results, shown in Fig. 6, show generally good agreement between the two, though where significant discrepancies exist they are always such that the rejection-based estimator returns a larger integrated autocorrelation time than that calculated using the magnetization.

In Sec. II B it was claimed that, when the proposal distribution $q(\phi'|\phi)$ in the Metropolis step [Eq. (9)] is *not* conditioned on the current state of the Markov chain and is instead given by $\tilde{p}(\phi)$, the integrated autocorrelation time is
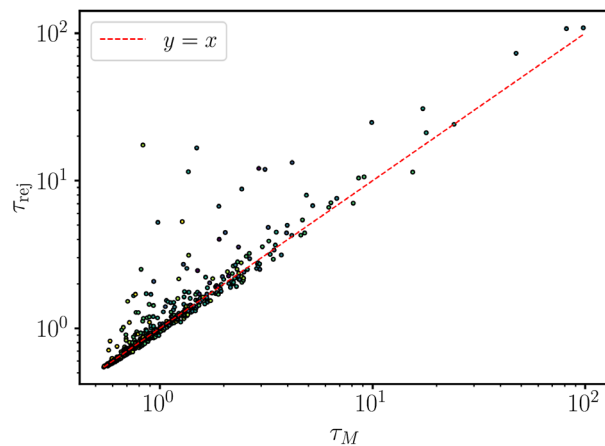


FIG. 6. Comparison of alternative estimates of the integrated autocorrelation time. On the $x$ axis, $\tau_M$ is calculated in the traditional way, by measuring the autocorrelation function of the magnetization (see Appendix B). On the $y$ axis, the estimator is based on the Metropolis-Hastings rejection rate, using Eqs. (4) and (13).

TABLE II.   Based on the power-law fit from Fig. 5, $2\tau_{\mathcal{O}}$ is the ratio between the effective sample size (that controls the statistical error on observables) and the total length of the Markov chain.

| $2\tau_{\mathcal{O}}$ | 1 | 2 | 5 | 10 | 100 | 1000 |
|---|---|---|---|---|---|---|
| $\mathbb{E}_{\phi\sim p}\mathbb{E}_{\phi'\sim\tilde{p}}[A(\phi\to\phi')]$ | 1 | 0.76 | 0.52 | 0.40 | 0.16 | 0.06 |

determined entirely by the rate at which proposals are rejected or, more precisely, by Eq. (13). Since this has nothing to do with the specifics of the flow model, the system size or the values of the couplings, we simply combined results from a large number of previously trained models (539, to be precise) to verify this property.

Figure 5 provides the necessary empirical evidence that we have indeed nullified any dependence of $\tau_{\mathcal{O}}$ on the correlation length of the system and have therefore eliminated critical slowing down in the sampling phase. The geometric lower bound on the integrated autocorrelation time from Eq. (14) is also plotted, but we find that the relationship between acceptance rate and integrated autocorrelation time is fit rather well by a power law:

$$\tau_{\mathcal{O}} \approx \frac{1}{2}\mathbb{E}_{\phi\sim p,\phi'\sim\tilde{p}}[A(\phi\to\phi')]^{-2.48(1)}. \qquad (49)$$

Table II contains no new information but rephrases this power-law relation in terms of the acceptance rate required for the effective sample size to be a particular fraction of the Markov chain length. However, for reasons discussed shortly, we take this scaling relation with a pinch of salt and advise against extrapolating to lower acceptances and larger autocorrelation times.

As shown in Fig. 7, the discrepancy between $\tau_M$ and $\tau_{\text{rej}}$ is directly related to the presence of long phases in the Metropolis-Hastings simulation in which every proposed

configuration was rejected. These "rare events," arising from the tails of the distributions depicted in Fig. 8, are problematic when it comes to estimating the integrated autocorrelation time. The rejection-based estimator is highly sensitive to them and hence picks up a large statistical error (the situation resembles the slow convergence of estimators when local-update sampling algorithms are required to traverse large energy barriers). Conversely, the traditional estimator is relatively insensitive to a small number of uncharacteristically long periods of consecutive rejections, which may result in an underestimate of the true integrated autocorrelation time.

In Fig. 8 we see the distribution describing the length of runs of consecutive rejections become less long-tailed as the acceptance rate increases. Although this trend is somewhat obvious in a qualitative sense, the circumstances by which these long-tailed distributions arise are an interesting facet of the scheme used to train these models. We expand on this in Sec. VII B.

## C. Finding efficient representations

The size of models, i.e., the number of trainable parameters $|\theta|$, will obviously play an important role in determining the scaling of training costs. Hence, it is not simply a question of building normalizing flows out of highly expressive transformations, but also one of minimizing the number of redundant parameters in the model. More specifically, our ambition must be to find architectures which are able to learn the most *efficient* representations of trivializing maps, for which $|\theta|$ grows slowly as we
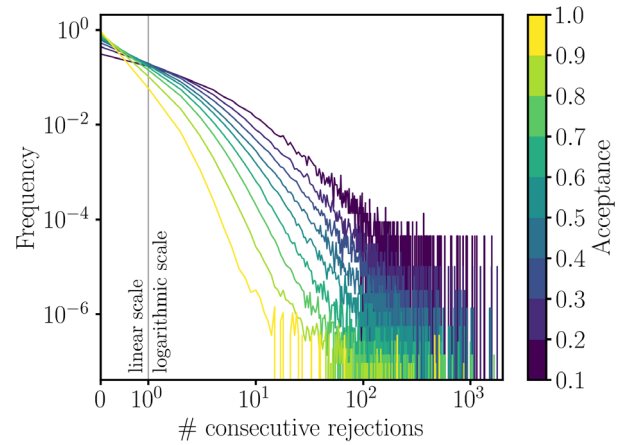
FIG. 8.   Empirical distributions describing the length of periods of consecutive rejections in Metropolis-Hastings simulations involving models with different average acceptance rates. Each empirical distribution combines accept-reject statistics from models with various combinations of layers, trained against different target theories (in terms of lattice size and couplings). The scale on the $x$ axis is logarithmic other than between zero and one (with zero corresponding to instances of two consecutive successful updates).
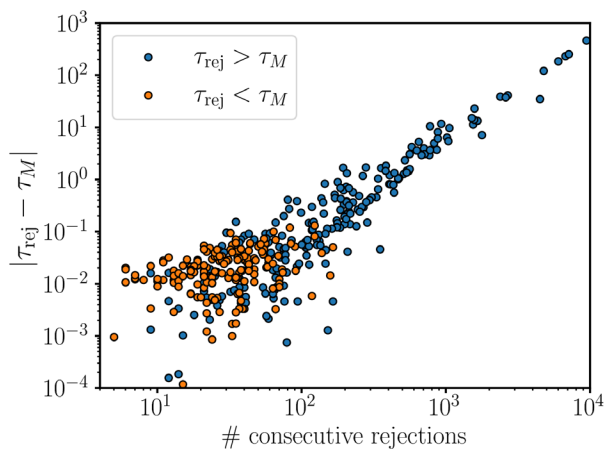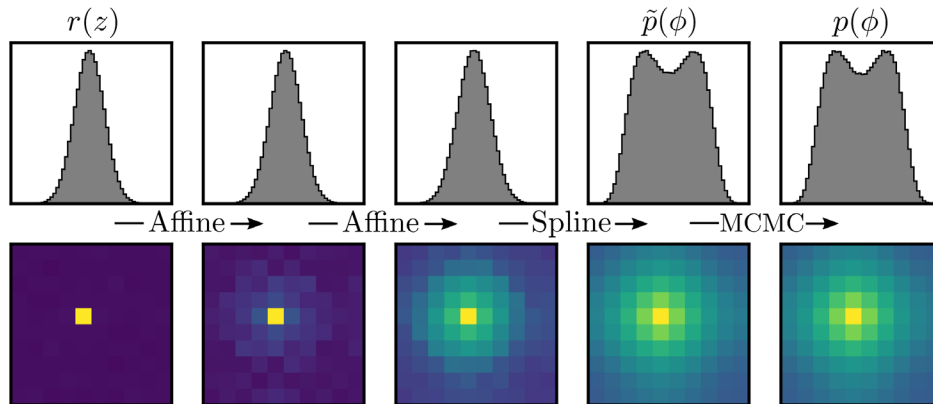
FIG. 7.   Discrepancy between the alternative estimates of the integrated autocorrelation time shown in Fig. 6, with $x$ coordinates corresponding to the length of the longest consecutive run of rejections occurring in the sampling phase.

FIG. 9. Samples of field configurations drawn from each step in the process of generating a representative sample of $\phi^4$ configurations with $\{L, \beta, \lambda\} = \{10, 0.601, 0.5\}$, using a flow model consisting of two affine coupling blocks, followed by a spline. The process starts with uncorrelated latent variables, $z \equiv v_1$, and we then sample from the model after each coupling block (layers $i = 3, 5, 7$) and finally take the output of the Metropolis-Hastings phase (keeping one in every $2\tau_{\mathrm{rej}}$ configurations). The top row contains histograms of the field variables and the bottom is the two-point correlation function from Eq. (42). The normalization used in the color mapping uses a combination of linear and logarithmic scaling (a symmetric log scaling with a linear threshold of 0.1) to make it easier to see the gradual emergence of correlations.

increase the number of degrees of freedom in the target density.

As stated previously, we found that the flexibility of affine and rational quadratic spline transformations was not put to good use when the target density corresponded to a free theory, with models built from additive layers performing equally well using fewer trainable parameters in total. We now turn to our main point of focus, which is finding efficiency representations of trivializing maps for strongly interacting theories. It is a tremendously useful feature of normalizing flows that we are able to sample from the intermediate layers, which define probability densities in their own right—see Fig. 9. This gives us insight into the role played by each individual layer.

The transition from disorder to long-range order in lattice $\phi^4$ theory is characterized by a gradual separation of the initial unimodal probability density into two distinct peaks, corresponding to a positive and a negative net magnetization. Therefore, in the regime where the density is bimodal, a normalizing flow transforming Gaussian latent variables must at some point enact a $\mathbb{Z}_2$-symmetric bulk shifting of probability density to $\pm \langle |\phi| \rangle$. For flows using solely affine layers, we observed that the task of transforming a unimodal density into a bimodal one was almost always designated to the final coupling block, after earlier layers had resolved the general structure of correlations in the unimodal setting. We suspect that this is because the functions being modeled by the neural networks simplify when their inputs are distributed unimodally around zero, whereas they must become more strongly nonlinear and hence harder for the neural networks to approximate, when their inputs are distributed bimodally. When an RQS block was introduced, this always took on the unimodal-bimodal transformation regardless of its position in the flow.

However, Fig. 10, which compares various orderings of affine and spline blocks, shows that the strongest design starts with affine layers and finally applies a single block of spline transformations.

Our expectation was that the additional flexibility offered by RQS transformations might help to perform the challenging splitting and shifting of density in a way that is more sensitive to subtle differences in the inputs than is possible with affine layers. This is confirmed emphatically by the results in Fig. 11, for which we took a seven-block affine
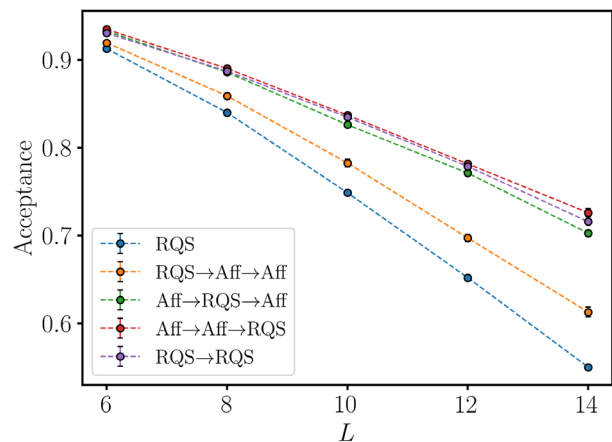


FIG. 10. Comparison of flow models with different arrangements of affine (Aff) and rational quadratic spline (RQS) coupling blocks. For example, the blue model contains a single RQS block, i.e., two RQS coupling layers, whereas the orange model passes the latent variables through an RQS block and then two affine blocks. *All* neural networks contained a single hidden layer of size $H = |\Lambda|$. All models were trained for 16000 iterations with a batch size of 16000. The physical parameters used are those listed in Table I.

flow and substituted six blocks of affine coupling layers for a single RQS block in order to compare models with an approximately equal number of trainable parameters. However, the substitution of many inexpressive affine layers for a single, highly flexible RQS block is a one-off trick; the results in Fig. 10 show that adding a second spline block fails to improve the acceptance rate any more than adding another affine block. It appears that, with the unimodal-bimodal transformation taken care of in the final layer, the remainder of the trivializing map can be modeled more efficiently using the simpler transformations.

From Fig. 11 we also see that enforcing $\mathbb{Z}_2$ equivariance in the affine layers, as described in Sec. III F, leads to higher acceptances that reduce less steeply as the lattice size increases. Despite the theoretical flaw in the design of the $\mathbb{Z}_2$-equivariant spline layers, we still report on a brief investigation into their performance. Figure 12 shows that in the symmetric phase (which includes the couplings in Table I) we did not find that using these layers improved acceptance rates, though they appear to do so in the broken phase.

We also report on some preliminary experiments with convolutional networks, summarized in Fig. 13. The convolution-based models are highly parameter efficient in comparison with the fully connected ones, since the number of parameters is decoupled from the lattice size (with the caveat that deeper models are required when the correlation length grows proportionally to the lattice size). We expect this parameter efficiency to translate to improved scaling of training costs with respect to models



FIG. 12. Changes in the Metropolis-Hastings acceptance rate (denoted by $\bar{A}$) of spline-based models, due to enforcing $\mathbb{Z}_2$ equivariance in the spline layers, as described in Sec. III F. Couplings and training hyperparameters were identical to those in Fig. 4. We verified (by inspection of histograms) that each nonequivariant model retained an approximate symmetry, such that both positive and negative magnetizations were sampled with similar frequency.

using fully connected networks. However, on these small lattices we were able to reach significantly higher acceptance rates using fully connected networks, in less time than it took to train even the simplest convolution-based model. We are currently working on a more systematic and large-scale comparison of models based on convolutions versus
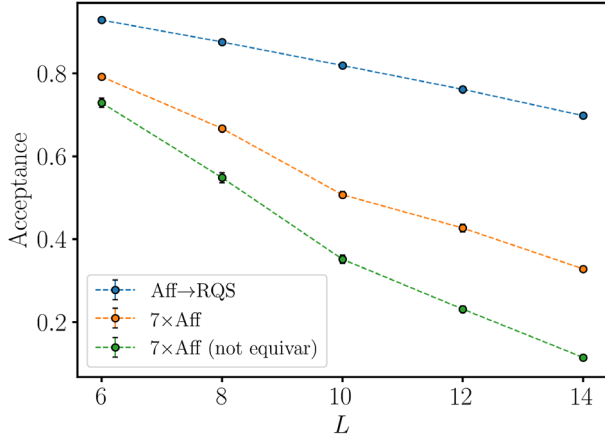


FIG. 11. Comparison between three groups of flow models using different coupling layers. One RQS block contains approximately the same number of parameters as six affine blocks. "Equivar" refers to the $\mathbb{Z}_2$-equivariant affine layers described in Sec. III F. The green data points correspond to flows in which the affine layers were similar to those used in Ref. [22] (though ours used neural networks that were both narrower and shallower). Models were trained in an identical fashion, for 16000 iterations with a batch size of 16000. The physical parameters are given in Table I.
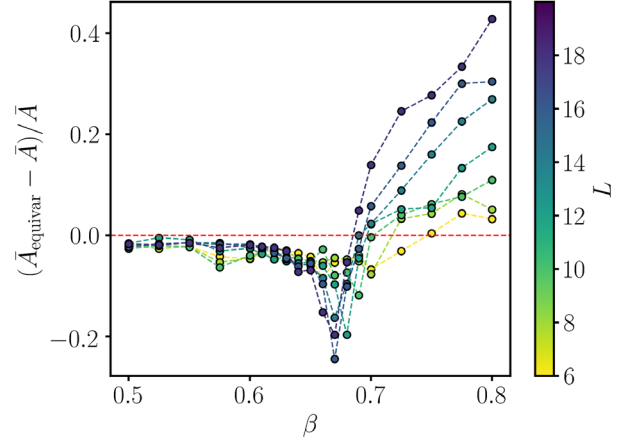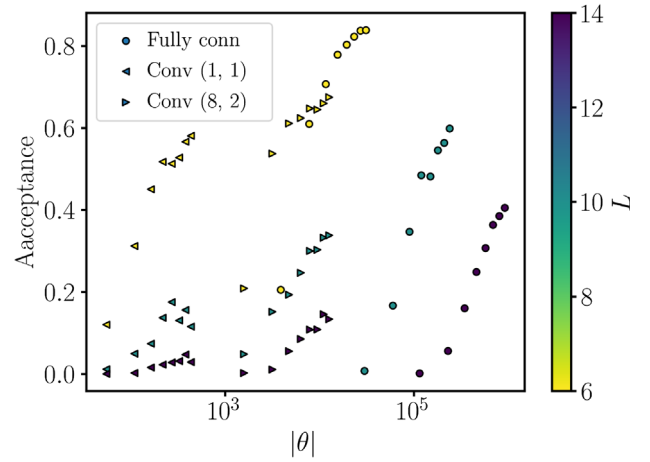


FIG. 13. Results from preliminary experiments using *affine* flows with convolutional networks, compared to the fully connected networks used throughout the rest of this study (single hidden layer of size $|\Lambda|$). The number of parameters $|\theta|$ varies through the number of affine coupling blocks (1–12). Convolutional networks had square kernels of size $3 \times 3$. The brackets in the legend denote the number of channels per hidden convolutional layer and the number of hidden layers, respectively. Models were trained for 16000 iterations with a batch size of 16000. Physical parameters are as in Table I for $6^2$, $10^2$ and $14^2$ lattices.

fully connected networks, focusing particularly on the scaling toward the continuum limit.

Having converged on a general recipe for $f_\theta$—some $\mathbb{Z}_2$-equivariant affine layers followed by a single RQS block, using fully connected neural networks—we have a number of possible ways to make the map more flexible: adding more affine layers; increasing the size of neural networks; and increasing the number of segments in the splines. Our goal is to figure out the extent to which each of these improve the ability of the model to fit $p(\phi)$ while minimizing the amount of redundancy in the model's parameters.

Firstly, our experiments, shown in Fig. 14, indicate that prepending more affine layers to the flow leads to modest improvements in the acceptance rate for a fixed training length, becoming increasingly worthwhile as the system size increases. However, there is certainly a law of diminishing returns, and on the lattices that we explored for this work the acceptance seems to plateau by the time we reach five affine layers. The law of diminishing returns also applies to adding more segments to a spline transformation; we found that using eight segments was reasonable and resulted in RQS layers that were substantially more flexible than affine layers without being too slow to train.

During numerous experiments with deep fully connected networks (only some of which are reported in Fig. 15) we failed to observe any consistent improvements, in terms of the Kullbach-Leibler divergence or the acceptance rate, over an equivalent model using neural networks with a single hidden layer, i.e., exactly as defined in Eq. (35). We checked that the networks did not suffer from the vanishing gradient problem by training them for at least as many epochs as were required for the gradients from all layers to
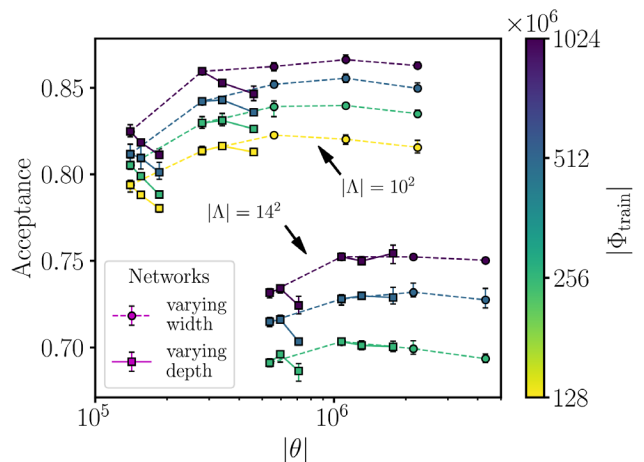


FIG. 15. Comparison of Metropolis-Hastings acceptance rates for flow models with a different number of trainable parameters $|\theta|$ within their neural networks. The color axis $|\Phi_{\mathrm{train}}|$ denotes the total number of configurations used in the optimization (the batch size multiplied by the number of training iterations). The dashed lines connect models whose neural networks had a single hidden layer, of varying width, whereas following the solid lines equates to increasing the number of hidden layers without changing the widths. Models comprised one affine block followed by one RQS block and were trained for 32000 training iterations with a range of batch sizes. Physical parameters are provided in Table I.

reach the same order of magnitude. We also substituted the tanh activation function for a ReLU which only saturates in one direction, although this meant we were no longer able to enforce $\mathbb{Z}_2$ equivariance within the affine layers, which proved to be a poor exchange.

As shown by Fig. 15, we also found that the benefits of increasing the width of neural networks quickly diminished once the hidden layer contained more than $H = |\Lambda|$ elements (i.e., twice the size of the input layer, which is the passive partition only). Since a doubling of the neural network widths in spline layers increases the number of parameters by a factor of approximately $3K|\Lambda|$, incurring a considerable increase in training costs, it was almost never a better investment of resources than increasing the batch size or number of training iterations.

Figure 15 exemplifies one of the key conclusions of our work—that the acceptance rate of our models is strongly dependent on the amount of effort we put into training while being relatively oblivious to the act of adding to the total number of trainable parameters in the model. In the following section we explore this in a more quantitative fashion.
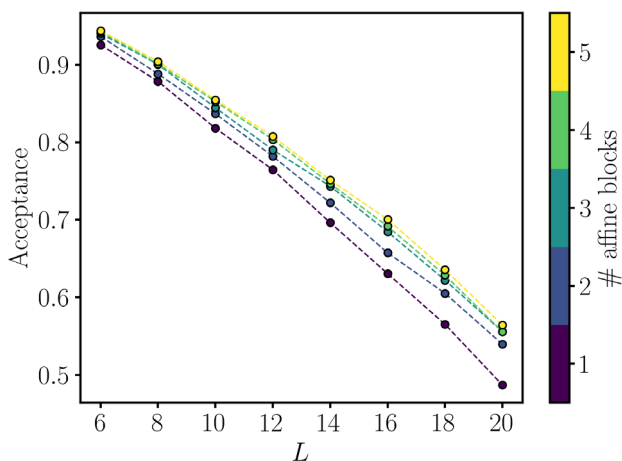


FIG. 14. Improvements in the Metropolis-Hastings acceptance rate due to adding more affine coupling blocks, before a rational quadratic spline block performs the final transformation. Models were trained for 16000 iterations with a batch size of 16000, and the physical parameters are those found in Table I.

## D. Scaling of training costs

Neglecting algorithmic or hardware-related factors that contribute to scalability, the cost of training a model up to a given value in some performance metric (e.g., acceptance rate) can be measured in terms of the number of training

iterations and the batch size. We will combine these two factors into a single number, $|\Phi_{\text{train}}|$, that is the total number of configurations that the model has been exposed to during training. This happens to also be rather convenient for making comparisons with critical slowing down in traditional algorithms, where the integrated autocorrelation time is proportional to the total number of configurations that must be generated to achieve a target error on expectation values. See Appendix C for a set of indicative training times, measured in seconds.

For this part of the study, we trained a large number of models, as per the recipe of the previous subsection (1–5 affine blocks followed by a single spline), using the $\phi^4$ couplings given in Table I. For training we covered a range of batch sizes (250–32000) and training lengths (500–64000 iterations), both of which were incremented in factors of 2. In Fig. 16 we plot the mean and standard deviation of the acceptance, taken over each set of models trained with the same $|\Phi_{\text{train}}|$. The error bars are typically much smaller than the difference between adjacent data points, indicating that the overall quality of optimization is relatively stable under mixing of the batch size and the number of training iterations, provided the total number of configurations from which the model can learn remains fixed. It is striking that the acceptance rate does not appear to plateau, even for the smallest lattice. This is a sign that the acceptance is limited not by the expressivity of the model but by the amount of training (whereas Fig. 11 shows that pure affine flows are genuinely limited by their expressivity).

When attempting to quantify the scaling of $|\Phi_{\text{train}}|$ in a way that can be contrasted with critical slowing down, we



FIG. 17. Scaling of the "training cost," measured by the total number of configurations used for optimization, for models to reach a certain target sampling efficiency. Models were sorted into "bins" by the integrated autocorrelation time measured during the sampling phase, and data points represent the best model (lowest $\tau_M$) from each bin. As usual, the correlation length was fixed at $\xi \approx L/4$ (see Table I). Hence, the values in the legend can in some sense be compared to the critical exponent $z_{\mathcal{O}}$ which determines the critical slowing down in traditional simulations.

found that, in order to get sets of points that could be fit reasonably well using a power law, we needed to group the models by sorting them into bins according to their integrated autocorrelation time and select the "best" model from each group. These fits are shown in Fig. 17. The results are sobering; despite the seemingly large improvements compared to the original formulation of Ref. [22], the amount of effort required to train these models is growing at an astonishing rate. While it is true that $|\Phi_{\text{train}}|$ is a one-off overhead cost, unlike the number of configurations required in a traditional sampling simulation, scaling that goes with the ninth power of the correlation length makes it impossible to avoid the conclusion that this prescription remains far away indeed from a solution to critical slowing down. In Sec. VII D we discuss why it is that models with relatively few trainable parameters appear to require such a colossal effort to train.

## VII. DISCUSSION

### A. The most efficient representations are shallow

With few exceptions, machine learning has followed a trend toward increasingly deep architectures, in spite of the fact that many of these architectures, including the fully connected feed-forward neural networks defined used in this work, require only a single hidden layer to act as universal approximators. The preference for deep architectures stems from two key principles [92]. Firstly, deep architectures are less inclined to overfit training data and hence generalize better than shallow ones. However, as explained in Sec. III A, overfitting is not a problem we will
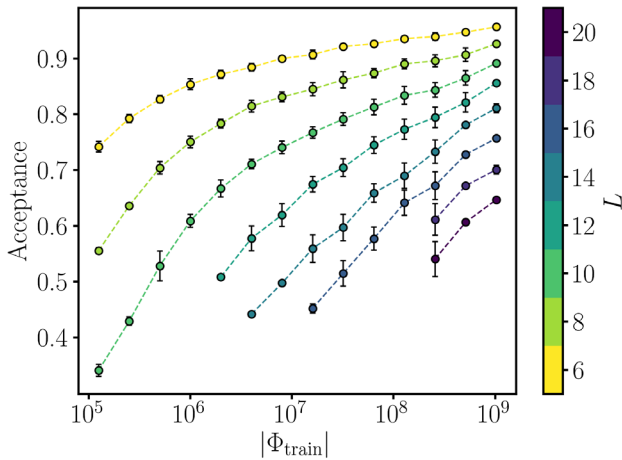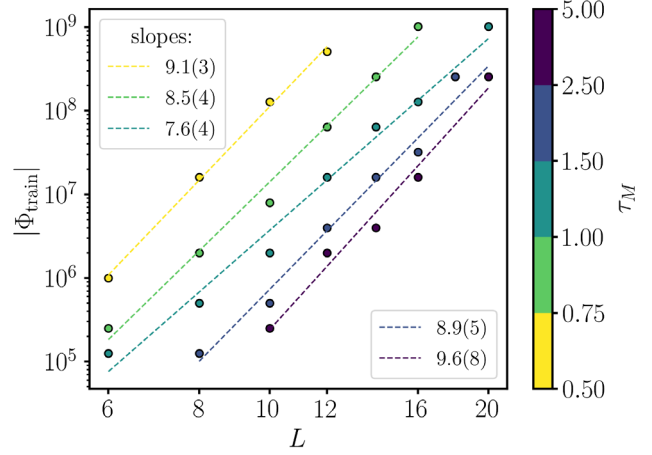


FIG. 16. Relationship between the average acceptance rate from fully trained models and the total number of configurations exposed to the models during training—i.e., the product of the batch size and the number of training iterations. The $\phi^4$ couplings are given in Table I. Error bars are standard deviations over a number of different models trained using different batch sizes ($\leq$32000) and training lengths ($\leq$32000), as well as a variable number of affine layers ($\leq$5).

face since each training iteration exposes the model to a set of previously unseen training inputs. Secondly, deep architectures can represent *some* highly nonlinear functions more efficiently than shallow architectures. This is connected to their ability to represent functions through hierarchical dependencies between abstract "features" (Ref. [93] argues, convincingly, that this is essentially a consequence of the laws of physics). The truth of this can be rigorously demonstrated for a number of architectures comprising specific functions [93–95], and empirical evidence is so overwhelming that "deep and cheap" has become somewhat of a mantra in machine learning.

Nevertheless, the question of whether deep architectures are more efficient is task-specific and should be answered through experimentation. In our case, experiments indicate that increasing the depth of neural networks does not increase the expressive capabilities of the model in a way that translates to a better fit to the target density. It is possible that this result is not merely a peculiar outcome of building the map out of coupling layers, whose structure is quite unusual; we draw attention to Ref. [28], in which the conclusion was that a single-layered restricted Boltzmann machine provides a more efficient representation of an Ising system near criticality than any of its deep generalizations. Furthermore, given a fixed number of trainable parameters, we found that a shallow flow with a more flexible spline layer dramatically outperformed a deeper flow using affine layers, for the interacting theory at least.

### B. A closer look at the training algorithm

Conventional training, via optimizing Eq. (18) using a fixed training set, penalizes the model for underestimating the density at any point at which there is a training input. This tends to result in models that are smoothed approximations of $p(\phi)$, spanning the space of training inputs. If we were to use such a model as the basis of our sampling algorithm, we would expect to see a steady flux of configurations originating from regions of configuration space "between the peaks" in $p(\phi)$, which would typically be rejected by the Metropolis test. The situation for our models is quite different. Instead, the characteristic behavior, which is known as "zero forcing," is for the optimization to quickly purge the density from any region of configuration space in which $p(\phi)$ is very small, resulting in models that fit the modes of the target well but frequently underestimate the low-density tails [96]. When these underestimated regions are eventually sampled from, the probability of transitioning away is suppressed by a factor of $\tilde{p}(\phi)/p(\phi)$ due to the Metropolis test. Thus, Markov chain histories in which the acceptance rate is typically quite high, but with occasional, "surprisingly" long periods of consecutive rejections, are a generic feature of models trained in this way.

It is enlightening to look more closely at the precise way in which optimization based on Eq. (20) acts to fit a model

to a target density. For this purpose it will be more convenient to write the Kullbach-Leibler divergence in the following form:

$$\hat{D}_{\mathrm{KL}}(\tilde{p}\|p) = \mathbb{E}_{\phi\sim\tilde{p}(\phi)}[\log \tilde{p}(\phi) - \log p(\phi)], \quad (50)$$

i.e., with the "irrelevant terms" that are neglected in Eq. (20) put back in. The origin of zero forcing is clear; if $\log \tilde{p}(\phi)$ is large in some region of configuration space in which $p(\phi)$ is small, then the optimization will receive a strong gradient signal, because configurations from this region are generated with high frequency and each contributes a large positive term to the objective function.

Using the chain rule, the gradient of Eq. (50) can be written

$$\nabla_\theta \hat{D}_{\mathrm{KL}}(\tilde{p}\|p) = \nabla_\phi \hat{D}_{\mathrm{KL}}(\tilde{p}\|p)\nabla_\theta f_\theta(z). \quad (51)$$

The gradient with respect to generated configurations contains two terms whose roles can be understood intuitively. One of these acts to reduce the action, $\mathbb{E}_{\phi\sim\tilde{p}(\phi)}[-\log p(\phi)]$, driving the optimization toward the mode(s) of the target density. This process is "boosted" by the zero-forcing property described previously. The second term acts to increase the entropy, $\mathbb{E}_{\phi\sim\tilde{p}(\phi)}[-\log \tilde{p}(\phi)]$, driving density away from the mode(s), which is necessary to fit the tails of the target. Figure 18 shows a typical training profile, in which the rapid fitting of the modes, boosted by zero forcing, yields a fairly high acceptance rate early on in the training, after which a more gradual process of expanding around the modes to fit the low-density tails takes over. Also note that, as the model expands to fill the tails of the target and and the acceptance rate goes up, we
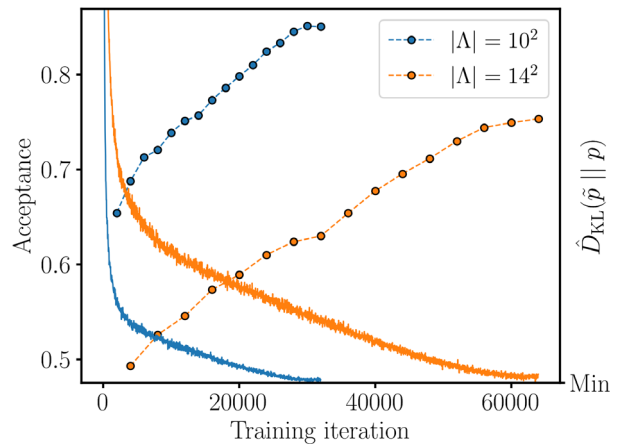


FIG. 18. Examples of typical training profiles for hybrid affine-spline flow models being trained against interacting theories with correlation length $\xi \approx L/4$. The objective function (right axis) has been shifted for the purpose of fitting both profiles on one figure. The learning rate is being annealed down from $\eta_0$ to zero according to Eq. (48).

also see that the distribution of consecutive rejections becomes less long-tailed, as demonstrated by Fig. 8.

### C. A small warning regarding ergodicity

It is worth considering the expected outcome of training a model that is seriously deficient in its inherent capacity to approximate the target density. In such a situation Eq. (50) can be expected to drive the model toward prioritizing a certain subset of features in $p(\phi)$ at the expense of almost completely ignoring others; the objective function cannot penalize the model for making such a terrible error if the model never generates a configuration from that region. This is well documented in the literature [97,98]. Recall that the flow-based approach to sampling is guaranteed to be ergodic; by construction, $\tilde{p}(\phi)$ has support on $\mathbb{R}^{|\Lambda|}$ (since it is bijectively related to a Gaussian distribution), which means there is a nonzero probability of *any* flow model proposing *any* configuration. However, when the probability of sampling from important regions of configuration space is suppressed to the extent that they are almost never sampled from on the timescales of practical simulations, we might speak of an *effective* breaking of ergodicity.

This appears to be worrying and might remind the reader of the "mode collapse" problem faced when training generative adversarial networks. Fortunately, in our case the problem is much less severe, because underestimates of $p(\phi)$ must necessarily be compensated by overestimates elsewhere in configuration space, which are then more likely to be penalized (this is a key strength of likelihood-based training). This is not to say that the sort of dramatic error just described cannot occur, but in practice we found that it is far more likely to be caused by instability in the training due to using a large learning rate and small batch size and not because the objective function is genuinely minimized by learning a density that is qualitatively different from the target due to model inflexibility.

In the top subfigure of Fig. 19 we show an example of a model that has erroneously broken the $\mathbb{Z}_2$ symmetry by being trained too aggressively, the result being that it generates samples of field configurations whose magnetizations all possess the same sign. The other two subfigures demonstrate that the problem is easily resolved by decreasing the learning or increasing the batch size, thereby reducing the likelihood of making a sequence of optimization steps which collapse one of the two modes. Note that the "wrong" model attained a higher acceptance rate than both of the "correct" models. However, this is misleading; the nonsymmetric model is a highly *inefficient* generator of proposals for the Metropolis-Hastings algorithm. If the sampling was run for long enough, eventually a configuration from the collapsed mode would be generated, at which point the Markov chain would freeze due to the enormously suppressed probability of transitioning away.
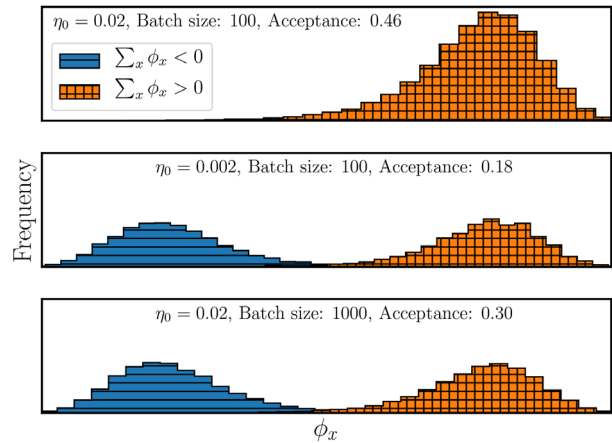


FIG. 19. Histograms of field variables taken from three samples of $10^5$ configurations, generated by three different models. The color labels the sign of the magnetization. The $\phi^4$ parameters are $\{L, \beta, \lambda\} = \{6, 0.8, 0.5\}$. The models had two blocks of affine layers for which $\mathbb{Z}_2$ equivariance was not enforced. In the top subfigure, the large learning rate and small batch size result in the breaking of the $\mathbb{Z}_2$ symmetry during optimization. This is easily avoided by using more sensible learning rates and batch sizes. However, note that the acceptance rate for the top model is larger.

Of course, this is exactly the level of inefficiency we should expect from a process that is attempting to perform a reweighting to a distribution with which there is very little overlap.

Although this example is rather contrived and very easily avoided by simply choosing sensible hyperparameters, it serves as a warning not to rely on the acceptance rate as the sole indicator of model quality. This conclusion was also reached by the authors of Ref. [61]. Looking ahead to more complicated field theories, it will be sensible to check for violations of the symmetries which one knows to be present but which may be broken by the model. In the $\phi^4$ case, we can easily see if the $\mathbb{Z}_2$ symmetry has been broken by looking at histograms of the field variables, and broken translational or rotational symmetries leave clear imprints in the correlation function.

### D. Critical slowing down of the training

Our results strongly suggest that the quality of the overall fit of our models to the target theory is limited by how extensively they have been optimized, not the inherent expressivity of the models. For the $\phi^4$ couplings given in Table I and our hybrid affine-spline models, Fig. 17 shows that these training costs scale in proportion to the number of degrees of freedom in the target density raised to a fairly high power—in the region of 7–10. An explanation for this disappointing result would be nice.

Of course, by moving to larger lattices we also increased the number of trainable parameters. Since we fixed the neural networks to have a hidden layer of size $H = |\Lambda|$, the

size of models grows as $|\theta| \propto |\Lambda|^2 = L^4$.[15] The results of Fig. 17 then state that, given a fixed target acceptance rate, the number of configurations $|\Phi_{\text{train}}|$ needed to train models scales with at least the square of the number of parameters requiring optimization. The numbers here are less important than the fact that we are *not* seeing exploding training costs because the dimensionality of the optimization problem is exploding (this was the point of looking for *efficient* representations).

A convincing explanation for this apparent reduction in efficiency arises from a thorough analysis of the training procedure, specifically, the two terms in the gradient of the Kullbach-Leibler divergence described above. Huang *et al.* [96] showed both theoretically and empirically[16] that, in several typical cases, the "expansion signal" due to the entropic term is feeble in comparison to the strength of the signal that drives the optimization toward the mode(s) of the target, due to the term driving action minimization. Put differently, the probability of generating a configuration which produces a gradient, $\nabla_\theta \hat{D}_{\text{KL}}(\tilde{p} \| p)$, that points away from the local mode of $p(\phi)$ is related to the spectrum of eigenvalues in the target's covariance matrix; the more *ill-conditioned* this matrix (i.e., the more eigenvalues that are close to being zero, relative to the principal eigenvalue), the lower the probability. The consequence of this is that the later stages of the training can be extremely inefficient, since a potentially tiny fraction of training inputs contribute to the process of "expanding" around the mode(s) of the target, through which the model learns to match the low-density tails.

We refer the reader to the original reference [96] for the details of this argument and empirical results, although the argument is quite intuitive: an ill-conditioned covariance matrix arises when the probability density effectively resides on a low-dimensional submanifold—this is sometimes referred to as having a low *intrinsic dimension* with respect to the actual number of degrees of freedom in the distribution—and in such situations we would expect the majority of training inputs generated by the model (which after all started life as an isotropic Gaussian) to fall outside this low-dimensional submanifold, producing a signal for the model to contract further toward the mode. Meanwhile, the rate at which configurations are generated in a direction along which the model needs to expand is very low. Our suspicion is, therefore, that the later stages of the training are predominantly spent compressing the model density onto a low-dimensional submanifold and only very slowly expanding on this submanifold to better approximate the tails of the target.

The observations of Ref. [96] are highly relevant for data-driven applications because realistic data distributions tend to
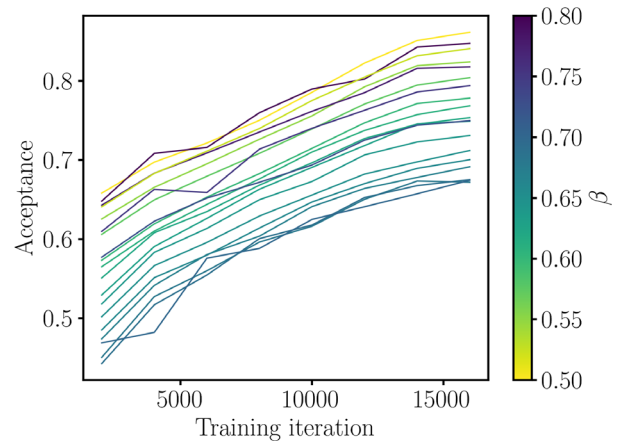


FIG. 20. Metropolis-Hastings acceptance rate measured at intervals of 2000 iterations during the course of training the $L = 12$ models from Fig. 4, in which $\lambda$ was held constant and $\beta$ varied so as to cross the phase transition.

have low intrinsic dimension. They are also relevant here; the action becomes increasingly ill-conditioned as the correlation length increases (i.e., effective mass tends to zero). If further convincing is required, Ref. [99] recently provided empirical evidence that, for a number of lattice models, critical points exist at minima of the intrinsic dimension. Figure 20 shows how the acceptance of the $L = 12$ models from Fig. 4, with varying correlation length but a constant number of degrees of freedom, improved over the course of training. Intriguingly, the profiles look very similar, other than being shifted with respect to each other. Although interpreting this plot in light of the previous discussion is an exercise in speculation, it is possible that a consequence of increasing the correlation length is that a greater proportion of the target density is fit in the slow phase as it becomes more concentrated on a manifold of low dimension.

To be clear, it is unsurprising that model optimization should becoming increasingly challenging as a critical point is approached. What is perhaps more interesting is the way in which a long correlation length afflicts the particular training scheme used here, although it is very possible that effect just described is subdominant on the small lattices studied in this work (in which case we need to identify the dominant source of inefficiency). Much more work will be needed to disentangle the various factors contributing to the overall scaling of training costs and paint a more quantitative picture, but if this alternate manifestation of critical slowing down turns out to be a key bottleneck, then attention should turn to establishing whether it can be tamed more readily than the familiar version that hampers traditional MCMC. In Sec. VIII B we discuss some possible avenues for further investigation.

## VIII. CONCLUSIONS AND OUTLOOK

We have verified the key results of Ref. [22]: firstly, that there exist approximately trivializing maps in

---

[15]The weights of the connections between two neural network layers of widths $n$ and $m$ can be represented by an $n \times m$ matrix.

[16]Albeit in a slightly different context, in which the target density is given by a second generative model.

two-dimensional lattice $\phi^4$ theory that are accessible *in practice* to normalizing flow models with tractable Jacobian determinant; and secondly, that using such models as generators of proposals for a Metropolis-Hastings simulation represents a complete transfer of the computational costs normally associated with critical slowing down to the cost of training the flow model. As is often the case in machine learning, the efficiency of the procedure depends on the tuning of hyperparameters, which we have investigated in this work. We have shown that fairly modest modifications of the original prescription—inserting a more expressive transformation at the final layer of the flow and drastically reducing the size of the neural networks—lead to much more efficient representations of approximately trivializing maps for this system. The extent of the associated reduction in training costs is such that the systems studied here and in Ref. [22] are accessible with the sorts of computing resources typically found on personal computers. However, our main finding is that the rate at which training costs scales as we move toward the continuum limit is extremely large, increasing far more quickly than the size of the models. Our work demonstrates a rather urgent need to understand and mitigate inefficiencies in the training algorithm itself when the target of optimization is a lattice field theory in the critical regime, and this must be done in parallel with efforts toward building more sophisticated flow models.

Below, we outline our immediate intentions for further research and describe several potential options for improving on the scaling of training costs.

### A. Nonlinear $\sigma$ models

Further work on our part will focus on the $O(N)$ and $CP^{N-1}$ nonlinear $\sigma$ models. The latter class of models are a good milestone on the journey to QCD since they exhibit many of the interesting nonperturbative features observed in QCD while being much more amenable to numerical study, being two-dimensional theories without even a fundamental gauge field. There is precedent for expecting $CP^{N-1}$ to challenge this method; previous studies uncovered pathological critical slowing down of the topological charge $Q$, consistent with $\tau_Q \sim e^{c\xi}$ [9], and it was precisely this situation which the original formulation of trivializing maps failed to resolve [53].

Nonlinear $\sigma$ models also have a nontrivial field space, though one can parameterize any $O(N)$ element or $CP^{N-1}$ representative using the unit $n$-spheres. Fortunately for us, several recent efforts have augmented the set of tools at our disposal with several that are dedicated to problems involving angular variables [58,59,100,101]. The generalization is straightforward in principle; as well as being invertible and continuously differentiable the flow is required to respect the topological properties of the field space. However, one should be aware that in most implementations of normalizing flows, the present work

included, a Euclidean setting is implicit in the change-of-variables formula defined by Eq. (16) as well as, more fundamentally, in the neural networks, which naturally act on Euclidean vectors.

### B. Improving on the scalability with physics

More work is required to determine just how large a role this analog of critical slowing down (see Sec. VII D) plays in the exploding cost of training. Our suspicion is that the correlation length of the target theory will contribute significantly as one moves to larger lattices. If our understanding is correct, it may be more efficient to learn a trivializing map to the corresponding free theory, implying that (with a suitably chosen bare mass) the latent density $r(z)$ resides on a similarly low-dimensional manifold as the target. We can easily generate real-space configurations of noninteracting fields by simply rescaling and performing a Fourier transform, and the log-density term is simply given by the action of the free theory. A cursory look at some models trained with free fields acting as latent variables indicates that starting from free fields does not dramatically alter the overall quality of optimisation if the target theory is strongly interacting. However, this could be because any structure in the latent density is obfuscated during the fast zero-forcing stage, which may fail to keep the intrinsic dimension low. A more robust strategy might involve gradually flowing through the space of couplings from the free theory to the theory of interest, more akin to the original trivializing maps of Ref. [52]. This could be achieved by updating the couplings during training[17] or by a layerwise approach to training that forces the trivializing map to follow a constrained path through the space of couplings.

Notwithstanding issues related to specific training schemes, there are reasons to be a touch more optimistic about the scalability of this method. Most actions or Hamiltonians that are of interest in physics contain exclusively local interaction terms, possess symmetries under certain transformations, and are often approximately or exactly self-similar over multiple scales. Each of these qualities might be exploited to build highly constrained models that yield efficient representations of trivializing maps.

Locality is partly responsible for the success of convolutional architectures, which pass localized filters (convolution kernels) over the data. "Features" can be extracted at multiple scales by downsampling the data between convolutional layers, a procedure which shares much in common with Kadanoff's block decimation [102], the precursor to renormalization group transformations

---

[17]There is no reason why the couplings of the theory have to remain constant during training. Rather, they can be thought of as another tunable parameter of the optimization, alongside the batch size and learning rate [61].

[103]. One of the implications of renormalizability is that coarse-grained representations of the fields contain useful information encoded as relevant variables, providing a physical motivation for the use of multiscale architectures which progressively resolve features at finer scales. Multiscale flows form the basis of the information-preserving renormalization group algorithm of Ref. [45], and similar architectures have performed well in the classic application of image synthesis [51], where the same concepts of locality and coarse-grained descriptions hold true, albeit less formally. We also note that convolutional networks that are equivariant or covariant with respect to other types of symmetry transformations have been introduced in Refs. [104–108].

### C. Alternatives for improving scalability

One might describe the ideas suggested above as "physics based"; the ambition is to somehow incorporate known physics to construct models and training schemes that are "better informed" about the nature of the target density. However, there are a number of "technical" options which might (at least partially) circumvent the issue with training. One of these is to devote some effort to generating a training set of configurations via traditional MCMC (potentially as a bootstrapping of the generative training update) and include these in the optimization. Even a relatively small number of training examples sampled from the low-density tails may help to improve the rate at which the model expands to fit them, though we note that Ref. [22] did not report any significant improvements by using a pregenerated training set. There are also ways forward in which we just accept that sampling from regions of low density is a weakness of the generative approach. For example, it would be trivial to incorporate regular local updates alongside generative proposals in the sampling phase, which may get the best of both worlds; local moves could offset biases in regions of low target density more efficiently than the Metropolis test alone, which as we have seen results in long periods of consecutive rejections. Another possibility, more appealing in terms of scalability, is to embed generative proposals within a multilevel sampling algorithm such as those developed in Refs. [109,110]. In effect, this means training models to trivialize subvolumes of the lattice in which the correlation length is cut off by $L_{\mathrm{subvol}}$, trading improved efficiency in the training stage for the additional cost of having to stitch together these subvolumes.

Even in very optimistic scenarios where improvements to the training scheme alleviate the worst effects of critical slowing down, we would still expect larger batches and longer training runs to universally lead to improvements (up to a point where the expressivity of the model prevents it from being able to fit the target any better), since there is no overfitting to speak of. Distributing batches over multiple processing units, also known as "data-parallel" training, is therefore likely be an essential ingredient of

future attempts to scale this method up to larger systems. The efficiency of the data-parallel approach is reduced as node memory limits are exceeded and the batch size per node must be decreased. Hence, it bodes well that we found shallow neural networks to outperform their deep counterparts, suggesting that memory costs will increase relatively slowly. To contrast this point, several state-of-the-art models based on residual networks with hundreds or thousands of layers have such high memory requirements that *the layers themselves* must be distributed over multiple nodes. In fact the situation as regards memory is even better since normalizing flows lend themselves to low memory requirements by their very design; backpropagation through reversible layers (such as coupling layers) can be performed without storing the intermediate vectors [111], meaning that memory requirements do not increase in proportion to the number of coupling layers.

### D. Wider context

On a final note, we find it thought provoking that several generic features of "real-world" data, such as locality, symmetries and scale invariance, arise in lattice field theories but in a manner that is more precisely defined, e.g., by terms in the action, renormalization group transformations, etc. Meteorological data, for want of an example, contains correlations over multiple scales and emergent phenomena arising from purely local interactions. We also draw attention to an intriguing study in which machine learning techniques were used to *measure* the presence of various symmetries [e.g., $\mathbb{Z}_2$, O(2)] in pieces of artwork [112]. The framework of lattice field theory provides low-level control over these quasiuniversal properties through our ability to simply write down an action. Hence, lattice field theory is in many ways an ideal test bed for improving our understanding of how these properties can be efficiently encoded into statistical models, even when the technique or model under study is destined for completely unrelated applications. A further convenience of working with lattice field theories is that there are several options for generating reproducible data for training or validation in a way that can be done on demand (with the caveat that state-of-the-art ensembles are extremely expensive to generate) instead of requiring permanent storage. Perhaps it is not too outrageous to imagine that a set of lattice field theories may, in future, become a standard suite for testing and benchmarking new models and algorithms.

resources provided by the Edinburgh Compute and Data Facility (ECDF) [113].

## APPENDIX A: $\phi^4$ THEORY ON THE LATTICE

The standard $\phi^4$ action in two-dimensional Euclidean space is

$$S[\varphi] = \int d^2x \left[ \frac{1}{2}(\partial_\mu \varphi(x))(\partial_\mu \varphi(x)) + \frac{1}{2}m_0^2 \varphi(x)^2 \right. \\ \left. + \frac{1}{4!}g_0 \varphi(x)^4 \right], \tag{A1}$$

where $m_0$ is the bare mass and $g_0$ is the bare coupling for the quartic interaction term.

We can define a discretized analog of this theory on a periodic lattice $\Lambda$ with lattice spacing $a$ and spatial extent $L_\mu = a N_\mu$ using the following steps.

(i) Use the vanishing boundary term (due to periodicity) to replace the derivative term with the Laplacian:

$$(\partial_\mu \varphi(x))(\partial_\mu \varphi(x)) \to -\varphi(x)\partial^2 \varphi(x). \tag{A2}$$

(ii) Adopt the following discretized version of the Laplacian:

$$\partial^2 \varphi_x \to \delta^2 \varphi_x \\ = \frac{1}{a^2} \sum_{\mu=1}^{2} (\varphi_{x+ae_\mu} + \varphi_{x-ae_\mu} - 2\varphi_x), \tag{A3}$$

where $e_\mu$ represents a unit vector in the $\mu$th dimension.

(iii) Replace the integral with a sum:

$$\int d^2x \to a^2 \sum_{x\in\Lambda}. \tag{A4}$$

(iv) For convenience, define dimensionless couplings $m_0^2 \to m_0^2 a^2$ and $g_0 \to g_0 a^2$.

This leads to the following lattice action:

$$S(\varphi) = \sum_{x\in\Lambda} \left[ \frac{1}{2}\varphi_x(-\delta^2 + m_0^2)\varphi_x + \frac{g_0}{4!}\varphi_x^4 \right]. \tag{A5}$$

Using (A3) and the translational invariance of the action yields

$$S(\varphi) = \sum_{x\in\Lambda} \left[ -\sum_{\mu=1}^{2} \varphi_x \varphi_{x+e_\mu} + \left(2 + \frac{m_0^2}{2}\right)\varphi_x^2 + \frac{g_0}{4!}\varphi_x^4 \right]. \tag{A6}$$

Equation (40) is related to (A6) through

$$\varphi = \sqrt{\beta}\phi, \quad 2 + \frac{m_0^2}{2} = \frac{1-2\lambda}{\beta}, \quad \frac{g_0}{4!} = \frac{\lambda}{\beta^2}. \tag{A7}$$

## APPENDIX B: ESTIMATION OF INTEGRATED AUTOCORRELATION TIME

In practice, the integrated autocorrelation time defined by Eq. (4) must be estimated from a Markov chain of finite length $N$. However, the statistical error on the autocovariance estimator,

$$\hat{\Gamma}_\mathcal{O}(t) = \frac{N}{N-1}\left[ \frac{1}{N}\sum_{n=1}^{N-t} \mathcal{O}(\phi^{(n+t)})\mathcal{O}(\phi^{(n)}) - \bar{\mathcal{O}}^2 \right], \tag{B1}$$

increases with $t$, so it is preferable to truncate the sum at some separation $W < N$.

We thus have the estimator

$$\hat{\tau}_\mathcal{O}(W) = \frac{1}{2} + \sum_{t=1}^{W} \frac{\hat{\Gamma}_\mathcal{O}(t)}{\hat{\Gamma}_\mathcal{O}(0)} \tag{B2}$$

and must attempt to find the value of $W$ which minimizes the sum of

(i) the bias due to truncating the sum,

$$\varepsilon_{\text{trunc}}(W) \equiv \text{bias}[\hat{\tau}_\mathcal{O}(W)] \\ = -\sum_{t=W+1}^{\infty} \frac{\Gamma_\mathcal{O}(t)}{\Gamma_\mathcal{O}(0)} \approx -\tau_\mathcal{O} e^{-W/T_\mathcal{O}}, \tag{B3}$$

where we have assumed that $W$ is sufficiently large that the autocorrelation takes a pure exponential form, with $T_\mathcal{O}$ being the characteristic relaxation time of the slowest mode of $\mathcal{O}$;

(ii) the statistical error approximated by the Madras-Sokal formula [1,114],

$$\varepsilon_{\text{stat}}^2(W) \equiv \text{var}[\hat{\tau}_\mathcal{O}(W)] \approx \frac{2(2W+1)}{N}\tau_\mathcal{O}^2, \tag{B4}$$

which uses the approximation $T_\mathcal{O} \ll W \ll N$.

We would therefore like to find the minimum of $\Delta(W) = |\varepsilon_{\text{trunc}}(W)| + |\varepsilon_{\text{stat}}(W)|$. To do so we follow the "automatic windowing" procedure detailed in Ref. [115] (Sec. 3.3).

First, note that we can recast the integrated autocorrelation time in terms of the equivalent pure exponential decay:

$$\tau_\mathcal{O}^* = -\left[ \log\left(\frac{2\tau_\mathcal{O} - 1}{2\tau_\mathcal{O} + 1}\right) \right]^{-1}, \tag{B5}$$

which is equal to zero, rather than $1/2$, for uncorrelated data and so offers improved precision in situations where decorrelation occurs very quickly.

We furthermore assume that it is valid to substitute the slowest mode $T_\mathcal{O}$ for $\lambda\tau_\mathcal{O}^*$ with $\lambda$ being a small constant factor, to be tuned such that smallest value of $W$ for which

TABLE III. Measurements of the real time taken to train our models to reach an acceptance rate of 70% for the systems studied in Ref. [22]. The models were trained on a desktop PC with an Intel i7-7700K quad-core CPU and 16 GB RAM. Since we were aiming for speed of training rather than reaching the highest possible acceptance rates, we increased the initial learning rate $\eta_0$ with respect to our main study, although we do not recommend doing this in general.

| $L$ | # Hidden elements | # Affine blocks | # Spline segments | Batch size | # Training iterations | $\eta_0$ | Time | Time (s) |
|---|---|---|---|---|---|---|---|---|
| 6 | 36 | 2 | 8 | 300 | 350 | 0.02 | 9 s | $9 \times 10^0$ |
| 8 | 64 | 2 | 8 | 500 | 750 | 0.01 | 37 s | $3.7 \times 10^1$ |
| 10 | 100 | 3 | 8 | 1250 | 2000 | 0.005 | 5.0 min | $3.0 \times 10^2$ |
| 12 | 144 | 3 | 8 | 4000 | 5000 | 0.003 | 43 min | $2.6 \times 10^3$ |
| 14 | 196 | 3 | 8 | 7000 | 14000 | 0.001 | 4.5 h | $1.6 \times 10^4$ |

$$\lambda \frac{\partial \hat{\Delta}(W)}{\partial W} = -e^{-W/(\lambda \hat{\tau}_{\mathcal{O}}^*)} + \frac{\lambda \hat{\tau}_{\mathcal{O}}^*}{\sqrt{WN}} \qquad \text{(B6)}$$

drops below zero occurs, generally, at a point at which $\hat{\tau}_{\mathcal{O}}(W)$ levels off to a plateau.

Since we generally encountered very small integrated autocorrelation times, the approximation in Eq. (B3) is probably a poor one. By the same token, however, statistical errors were minimal. Ultimately, a little tuning of $\lambda$ by visual inspection of $\hat{\tau}_{\mathcal{O}}(W)$ for a small number of experiments was sufficient.

## APPENDIX C: COMPARISON WITH LITERATURE RESULTS

Albergo, Kanwar, and Shanahan [22] used flows comprising solely affine coupling layers parameterized by fully connected networks to generate $\phi^4$ configurations on lattices ranging from $6^2$ to $14^2$ sites. They reported training times of 1–2 GPU-weeks in order to reach an average acceptance rate of 70% in the Metropolis-Hastings phase.

Although theirs was a proof-of-principle study, we were nonetheless curious to check how quickly our hybrid affine and spline models could be trained to reach a 70% acceptance rate, using the same action and couplings provided in Ref. [22] (though our previous results for $|\Lambda| = 6^2 - 14^2$, $\lambda = 1/2$ and $L/\xi = 4$ correspond to essentially the same systems). We trained these models on a desktop PC. The real time taken to train models to reach at least 70% acceptance are given in Table III.

Presumably, the main reason that the times reported in the original study are so much larger than what we find is the use of much larger neural networks; for the $14^2$ lattice the original study used networks with width 1024 and depth 6 whereas for the same system our models used comparatively minuscule networks of width 196 and depth 2 (i.e., a single hidden layer). Still, given the near order-of-magnitude increase in real train times for each lattice size increase, with this setup and a target of 70% acceptance we would likely encounter times measured in weeks as soon as $L = 18$ or 20. Despite the modest hardware, it is abundantly clear that scaling this technique up to system sizes at which critical slowing down becomes a serious problem will require significantly more than fine-tuning the current approach.

[1] A. D. Sokal, Monte Carlo methods in statistical mechanics: Foundations and new algorithms, in *Functional Integration* (Springer, New York, 1997), pp. 131–192.

[2] U. Wolff, Critical slowing down, Nucl. Phys. B, Proc. Suppl. **17**, 93 (1990).

[3] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, Equation of state calculations by fast computing machines, J. Chem. Phys. **21**, 1087 (1953).

[4] W. K. Hastings, Monte Carlo sampling methods using Markov Chains and their applications, Biometrika **57**, 97 (1970).

[5] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, Hybrid Monte Carlo, Phys. Lett. B **195**, 216 (1987).

[6] A. D. Kennedy and B. J. Pendleton, Acceptances and autocorrelations in hybrid Monte Carlo, Nucl. Phys. **B20**, 118 (1991).

[7] M. Campostrini, P. Rossi, and E. Vicari, Monte Carlo simulation of CP$^{N-1}$ models, Phys. Rev. D **46**, 2647 (1992).

[8] L. Del Debbio, H. Panagopoulos, P. Rossi, and E. Vicari, Spectrum of confining strings in SU(N) gauge theories, J. High Energy Phys. 01 (**2002**) 009.

[9] L. Del Debbio, G. M. Manca, and E. Vicari, Critical slowing down of topological modes, Phys. Lett. B **594**, 315 (2004).

[10] J. Flynn, A. Jüttner, A. Lawson, and F. Sanfilippo, Precision study of critical slowing down in lattice simulations of the $\mathbb{CP}^{N-1}$ model, arXiv:1504.06292.

[11] C. Bonati and M. D'Elia, Topological critical slowing down: Variations on a toy model, Phys. Rev. E **98**, 013308 (2018).

[12] B. Allés, G. Boyd, M. D'Elia, A. Di Giacomo, and E. Vicari, Hybrid Monte Carlo and topological modes of full QCD, Phys. Lett. B **389**, 107 (1996).

[13] S. Schaefer, R. Sommer, and F. Virotta, Critical slowing down and error analysis in lattice QCD simulations, Nucl. Phys. **B845**, 93 (2011).

[14] E. Vicari, Monte Carlo simulation of lattice $\mathbb{CP}^{N-1}$ models at large $N$, Phys. Lett. B **309**, 139 (1993).

[15] R. H. Swendsen and J. S. Wang, Nonuniversal Critical Dynamics in Monte Carlo Simulations, Phys. Rev. Lett. **58**, 86 (1987).

[16] U. Wolff, Collective Monte Carlo Updating for Spin Systems, Phys. Rev. Lett. **62**, 361 (1989).

[17] D. Kusnezov and J. Sloan, Global demons in field theory. Critical slowing down in the XY model, Nucl. Phys. **B409**, 635 (1993).

[18] H. G. Evertz, G. Lana, and M. Marcu, Cluster Algorithm for Vertex Models, Phys. Rev. Lett. **70**, 875 (1993).

[19] N. Prokof'ev, I. S. Tupitsyn, and B. V. Svistunov, Exact, complete, and universal continuous-time worldline Monte Carlo approach to the statistics of discrete quantum systems, J. Exp. Theor. Phys. **87**, 310 (1998).

[20] J. Xu, H. Li, and S. Zhou, An overview of deep generative models, IETE Technical Review **32**, 131 (2014).

[21] S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks, Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models, arXiv:2103.04922.

[22] M. S. Albergo, G. Kanwar, and P. E. Shanahan, Flow-based generative models for Markov chain Monte Carlo in lattice field theory, Phys. Rev. D **100**, 034515 (2019).

[23] G. Torlai and R. G. Melko, Learning thermodynamics with Boltzmann machines, Phys. Rev. B **94**, 165134 (2016).

[24] L. Wang, Can Boltzmann machines discover cluster updates? Phys. Rev. E **96**, 051301 (2017).

[25] L. Huang and L. Wang, Accelerated Monte Carlo simulations with restricted Boltzmann machines, Phys. Rev. B **95**, 035105 (2017).

[26] A. Tanaka and A. Tomiya, Towards reduction of autocorrelation in HMC by machine learning, arXiv:1712.03893.

[27] Z. Liu, S. P. Rodrigues, and W. Cai, Simulating the Ising model with a deep convolutional generative adversarial network, arXiv:1710.04987.

[28] A. Morningstar and R. G. Melko, Deep learning the Ising model near criticality, J. Mach. Learn. Res. **18**, 5975 (2018).

[29] J. M. Urban and J. M. Pawlowski, Reducing autocorrelation times in lattice simulations with generative adversarial networks, Mach. Learn. Sci. Tech. **1**, 045011 (2020).

[30] J. Singh, V. Arora, V. Gupta, and M. S. Scheurer, Generative models for sampling and phase transition indication in spin systems, SciPost Phys. **11**, 043 (2021).

[31] G. Kanwar, M. S. Albergo, D. Boyda, K. Cranmer, D. C. Hackett, S. Racanière, D. J. Rezende, and P. E. Shanahan, Equivariant Flow-Based Sampling for Lattice Gauge Theory, Phys. Rev. Lett. **125**, 121601 (2020).

[32] K. A. Nicoli, C. J. Anders, L. Funcke, T. Hartung, K. Jansen, P. Kessel, S. Nakajima, and P. Stornati, On Estimation of Thermodynamic Observables in Lattice Field Theories with Deep Generative Models, Phys. Rev. Lett. **126**, 032001 (2021).

[33] D. Boyda, G. Kanwar, Racanière, D. J. Rezende, M. S. Albergo, K. Cranmer, D. C. Hackett, and P. E. Shanahan, Sampling using SU(N) gauge equivariant flows, Phys. Rev. D **103**, 074504 (2021).

[34] M. S. Albergo, D. Bodya, D. C. Hackett, G. Kanwar, K. Cranmer, S. Racanière, D. J. Rezende, and P. E. Shanahan, Introduction to normalizing flows for lattice field theory, arXiv:2101.08176.

[35] S. Lawrence and Y. Yamauchi, Normalizing flows and the real-time sign problem, Phys. Rev. D **103**, 114509 (2021).

[36] S. Foreman, X.-Y. Jin, and J. C. Osborn, Deep learning Hamiltonian Monte Carlo, arXiv:2105.03418.

[37] D. Wu, R. Rossi, and G. Carleo, Unbiased Monte Carlo cluster updates with autoregressive neural networks, arXiv:2105.05650.

[38] M. Cristoforetti, G. Jurman, A. I. Nardelli, and C. Furlanello, Towards meaningful physics from generative models, arXiv:1705.09524.

[39] K. Zhou, G. Endrődi, L.-G. Pang, and H. Stöcker, Regressive and generative neural networks for scalar field theory, Phys. Rev. D **100**, 011501 (2019).

[40] L. Wang, Y. Jiang, L. He, and K. Zhou, Recognizing the topological phase transition by variational autoregressive networks, arXiv:2005.04857.

[41] D. Bachtis, G. Aarts, and B. Lucini, Adding machine learning within hamiltonians: Renormalization group transformations, symmetry breaking and restoration, Phys. Rev. Research **3**, 013134 (2021).

[42] P. Mehta and D. J. Schwab, An exact mapping between the variational renormalization group and deep learning, arXiv:1410.3831.

[43] M. Koch-Janusz and Z. Ringel, Mutual information, neural networks and the renormalization group, Nature (London) **14**, 578 (2018).

[44] P. M. Lenggenhager, D. E. Gökmen, Z. Ringel, S. D. Huber, and M. Koch-Janusz, Optimal Renormalization Group Transformation from Information Theory, Phys. Rev. X **10**, 011037 (2020).

[45] S. H. Li and L. Wang, Neural Network Renormalization Group, Phys. Rev. Lett. **121**, 260601 (2018).

[46] S. Efthymiou, M. J. S. Beach, and R. G. Melko, Super-resolving the Ising model with convolutional neural networks, Phys. Rev. B **99**, 075113 (2019).

[47] E. G. Tabak and E. Vanden-Eijnden, Density estimation by dual ascent of the log-likelihood, Commun. Math. Sci. **8**, 217 (2010).

[48] E. G. Tabak and C. V. Turner, A family of nonparametric density estimation algorithms, Commun. Pure Appl. Math. **66**, 145 (2013).

[49] D. J. Rezende and S. Mohamed, Variational inference with normalizing flows, arXiv:1505.05770.

[50] L. Dinh, D. Krueger, and Y. Bengio, NICE: Non-linear independent components estimation, arXiv:1410.8516.

[51] L. Dinh, J. Sohl-Dickstein, and S. Bengio, Density estimation using Real NVP, arXiv:1605.08803.

[52] M. Lüscher, Trivializing maps, the Wilson flow and the HMC algorithm, Commun. Math. Phys. **293**, 899 (2010).

[53] G. P. Engel and S. Schaefer, Testing trivializing maps in the Hybrid Monte Carlo algorithm, Comput. Phys. Commun. **182**, 2107 (2011).

[54] M. Lüscher, Instantaneous stochastic perturbation theory, J. High Energy Phys. 04 (2015) 142.

[55] K. L. Chung, *Markov Chains with Stationary Transition Probabilities*, 2nd ed. (Springer-Verlag, Berlin, 1967).

[56] L. Tierney, Markov Chains for exploring posterior distributions, Ann. Statist. **22**, 1701 (1994).

[57] S. Kullbach and R. A. Leibler, On information and sufficiency, Ann. Math. Stat. **22**, 79 (1951).

[58] M. C. Gemici, D. J. Rezende, and S. Mohamed, Normalizing flows on Riemannian manifolds, arXiv:1611.02304.

[59] D. J. Rezende, G. Papamakarios, S. Racanière, M. S. Albergo, G. Kanwar, P. E. Shanahan, and K. Cranmer, Normalizing flows on tori and spheres, arXiv:2002.02428.

[60] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980.

[61] D. C. Hackett, C.-C. Hsieh, M. S. Albergo, D. Boyda, J.-W. Chen, K.-F. Chen, K. Cranmer, G. Kanwar, and P. Shanahan, Flow-based sampling for multimodal distributions in lattice field theory, arXiv:2107.00734.

[62] D. P. Kingma and P. Dhariwal, Glow: Generative flow with invertible $1 \times 1$ convolutions, arXiv:1807.03039.

[63] G. Papamakarios, T. Pavlakou, and I. Murray, Masked autoregressive flow for density estimation, arXiv:1705.07057.

[64] C. W. Huang, D. Krueger, A. Lacoste, and A. Courville, Neural autoregressive flows, arXiv:1804.00779.

[65] W. Grathwohl, R. T. Q. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud, FFJORD: Free-form continuous dynamics for scalable reversible generative models, arXiv:1810.01367.

[66] T. Müller, B. Mcwilliams, F. Rousselle, M. Gross, and J. Novák, Neural importance sampling, arXiv:1808.03856.

[67] E. Hoogeboom, R. van den Berg, and M. Welling, Emerging convolutions for generative normalizing flows, arXiv:1901.11137.

[68] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios, Cubic spline flows, arXiv:1906.02145.

[69] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios, Neural spline flows, arXiv:1906.04032.

[70] C. Meng, Y. Song, J. Song, and S. Ermon, Gaussianization flows, arXiv:2003.01941.

[71] J. A. Gregory and R. Delbourgo, C2 rational quadratic spline interpolation to monotonic data, IMA J. Numer. Anal. **3**, 141 (1983).

[72] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Netw. **4**, 251 (1991).

[73] D. Yarotsky, Universal approximations of invariant maps by neural networks, arXiv:1804.10306.

[74] D.-X. Zhou, Universality of deep convolutional neural networks, arXiv:1805.10769.

[75] A. Heinecke, J. Ho, and W.-L. Hwang, Refinement and universal approximation via sparsely connected relu convolution nets, IEEE Signal Process. Lett. **27**, 1175 (2020).

[76] J. Köhler, L. Klein, and F. Noé, Equivariant flows: Exact likelihood generative learning for symmetric densities, arXiv:2006.02425.

[77] M. S. Albergo, G. Kanwar, S. Racanière, D. J. Rezende, J. M. Urban, D. Boyda, K. Cranmer, D. C. Hackett, and P. E. Shanahan, Flow-based sampling for fermionic lattice field theories, arXiv:2106.05934.

[78] J. Liu, Y. Qi, Z. Y. Meng, and L. Fu, Self-learning Monte Carlo method, Phys. Rev. B **95**, 041101 (2017).

[79] B. Efron, Bootstrap methods: Another look at the jackknife, Ann. Statist. **7**, 1 (1979).

[80] B. Efron and R. Tibshirani, Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy, Stat. Sci. **1**, 54 (1986).

[81] B. Kaufmann, Fitting a sum of exponentials to numerical data, arXiv:physics/0305019.

[82] S. Caracciolo and A. Pelissetto, Corrections to finite-size scaling in the lattice $N$-vector model for $N = \infty$, Phys. Rev. D **58**, 105007 (1998).

[83] I. Loshchilov and F. Hutter, Decoupled weight decay regularization, arXiv:1711.05101.

[84] I. Loshchilov and F. Hutter, SGDR: Stochastic gradient descent with warm restarts, arXiv:1608.03983.

[85] D. Masters and C. Luschi, Revisiting small batch training for deep networks, arXiv:1804.07612.

[86] R. Ge, F. Huang, C. Jin, and Y. Yuan, Escaping from saddle points—online stochastic gradient descent for tensor decomposition, arXiv:1503.02101.

[87] C. Zhang, Q. Liao, A. Rakhlin, B. Miranda, N. Golowich, and T. Poggio, Theory of deep learning III: Generalization properties of SGD, in *CBMM Memo No. 067, 2017*, https://cbmm.mit.edu/sites/default/files/publications/CBMM-Memo-067.pdf.

[88] M. R. Wilson, J. Marsh Rossney, and L. Del Debbio, ANVIL (version 0.9), https://doi.org/10.5281/zenodo.4792249 (2021).

[89] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein *et al.*, PyTorch: An imperative style, high-performance deep learning library, in *Advances in Neural Information Processing Systems 32*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc., 2019), pp. 8024–8035.

[90] Z. Kassabov, Reportengine: A framework for declarative data analysis, https://doi.org/10.5281/zenodo.2571601 (2019).

[91] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning representations by back-propagating errors, Nature (London) **323**, 533 (1986).

[92] Y. Bengio, Learning deep architectures for AI, Found. Trends Mach. Learn. **2**, 1 (2009).

[93] H. W. Lin, M. Tegmark, and D. Rolnick, Why does deep and cheap learning work so well?, J. Stat. Phys. **168**, 1223 (2017).

[94] Y. Bengio and Y. LeCun, Scaling learning algorithms towards AI, in *Large-scale Kernel Machines* (MIT Press, Cambridge, MA, 2007).

[95] H. Mhaskar, Q. Liao, and T. Poggio, When and why are deep networks bettter than shallow ones, in *AAAI'17: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence* (AAAI Press, San Francisco, 2017).

[96] C.-W. Huang, F. Ahman, K. Kumar, A. Lacoste, and A. Courville, Probability distillation: A caveat and alternatives in *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference* (2020), http://proceedings.mlr.press/v115/huang20c/huang20c.pdf.

[97] K. Murphy, *Machine Learning: A Probabilistic Perspective* (MIT Press, Cambridge, 2012), Vol. 58.

[98] T. Minka, Divergence measures and message passing, technical report, 2005.

[99] T. Mendes-Santos, A. Angelone, A. Rodriguez, R. Fazio, and M. Dalmonte, Intrinsic dimension of path integrals: data mining quantum criticality and emergent simplicity, PRX Quantum **2**, 030332 (2021).

[100] J. Sohl-Dickstein, Two equalities expressing the determinant of a matrix in terms of expectations over matrix-vector products, arXiv:2005.06553.

[101] S. Passenheim and E. Hoogeboom, Variational determinant estimation with spherical normalizing flows, arXiv:2012.13311.

[102] L. P. Kadanoff, Scaling laws for Ising models near $T_c$, Physics **2**, 263 (1966).

[103] K. G. Wilson and J. Kogut, The renormalization group and the $\epsilon$ expansion, Phys. Rep. **12**, 75 (1973).

[104] T. S. Cohen and M. Welling, Group equivariant convolutional networks, arXiv:1602.07576.

[105] T. S. Cohen, M. Geiger, J. Koehler, and M. Welling, Spherical CNNs, arXiv:1801.10130.

[106] T. S. Cohen, M. Weiler, B. Kicanaoglu, and M. Welling, Gauge equivariant convolutional networks and the icosahedral CNN, arXiv:1902.04615.

[107] E. J. Bekkers, B-spline CNNs on Lie groups, arXiv:1909.12057.

[108] A. Tomiya and Y. Nagai, Gauge covariant neural network for 4 dimensional non-abelian gauge theory, arXiv:2103.11965.

[109] M. Lüscher, Locality and exponential error reduction in numerical lattice gauge theory, J. High Energy Phys. 09 (2001) 010.

[110] H. B. Meyer, Locality and statistical error reduction on correlation functions, J. High Energy Phys. 01 (2003) 048.

[111] A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse, The reversible residual network: Backpropagation without storing activations, arXiv:1707.04585.

[112] G. Barenboim, J. Hirn, and V. Sanz, Symmetry meets ai, SciPost Phys. **11**, 014 (2021).

[113] http://www.ecdf.ed.ac.uk/.

[114] N. Madras and A. D. Sokal, The pivot algorithm: A highly efficient Monte Carlo method for the self-avoiding walk, J. Stat. Phys. **50**, 109 (1988).

[115] U. Wolff, Monte Carlo errors with less errors, Comput. Phys. Commun. **156**, 143 (2004).