Aspects of scaling and scalability for flow-based sampling of lattice QCD

Ryan Abbott\textsuperscript{1,2}, Michael S. Albergo\textsuperscript{3}, Aleksandar Botev\textsuperscript{6}, Denis Boyda\textsuperscript{1,2,4}, Kyle Cranmer\textsuperscript{3,5},
Daniel C. Hackett\textsuperscript{1,2,4}, Alexander G. D. G. Matthews\textsuperscript{6}, Sébastien Racanière\textsuperscript{6}, Ali Razavi\textsuperscript{6}, Danilo J. Rezende\textsuperscript{6},
Fernando Romero-López\textsuperscript{1,2}, Phiala E. Shanahan\textsuperscript{1,2}, Julian M. Urban\textsuperscript{1,2}

\textsuperscript{1} Center for Theoretical Physics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
\textsuperscript{2} The NSF AI Institute for Artificial Intelligence and Fundamental Interactions, Cambridge, USA
\textsuperscript{3} Center for Cosmology and Particle Physics, New York University, New York, NY 10003, USA
\textsuperscript{4} Argonne Leadership Computing Facility, Argonne National Laboratory, Lemont, IL 60439, USA
\textsuperscript{5} Physics Department, University of Wisconsin-Madison, Madison, WI 53706, USA
\textsuperscript{6} DeepMind, London, UK

Published online: 04 November 2023

Abstract

Recent applications of machine-learned normalizing flows to sampling in lattice field theory suggest that such methods may be able to mitigate critical slowing down and topological freezing. However, these demonstrations have been at the scale of toy models, and it remains to be determined whether they can be applied to state-of-the-art lattice quantum chromodynamics calculations. Assessing the viability of sampling algorithms for lattice field theory at scale has traditionally been accomplished using simple cost scaling laws, but as we discuss in this work, their utility is limited for flow-based approaches. We conclude that flow-based approaches to sampling are better thought of as a broad family of algorithms with different scaling properties, and that scalability must be assessed experimentally.

1 Introduction

Lattice quantum field theory (LQFT) is the only known systematically improvable approach to calculating physical observables in quantum field theories that exhibit non-perturbative dynamics. LQFT has been applied to first-principles studies of quantum chromodynamics (QCD) at low energy scales [1–7], to test proposed models for physics beyond the Standard Model [8–11], and to investigate various condensed matter systems [12,13]. In this framework, discretized path integrals are evaluated numerically using stochastic Monte Carlo estimators,

\begin{equation}
\langle \mathcal{O} \rangle \equiv \int d\phi \; p(\phi) \mathcal{O}(\phi) \approx \frac{1}{N} \sum_{i=1}^{N} \mathcal{O}(\phi_i),
\end{equation}

where \( \phi_i \) are samples of the lattice field degrees of freedom drawn from the probability distribution defined by the Euclidean lattice action \( S \),

\begin{equation}
p(\phi) = \frac{1}{Z} e^{-S(\phi)}.
\end{equation}

The partition function \( Z \) is typically unknown, but this is not an obstacle to sampling with Markov chain Monte Carlo (MCMC) algorithms. At present, Hybrid/Hamiltonian Monte Carlo (HMC) [14–16] is the state-of-the-art MCMC algorithm for generating QCD field configurations, but its computational cost grows rapidly as the continuum limit is approached [17–19].

Recent work has explored whether improved configuration generation algorithms can be achieved using machine learning (ML) [20–62]. For example, promising proof-of-principle results using normalizing flows [63–65] have been obtained in a range of theories with different properties including gauge symmetries [27–35], fermionic degrees of freedom [32–36], and the existence of distinct topological sectors or multiple modes of the probability distribution [23–27,29–35]. This includes a first demonstration for lattice QCD [35].

These results raise the question of whether a flow-based approach can be applied to lattice QCD calculations at state-of-the-art scale. The practical question is whether flow models can provide more cost-effective sampling than HMC for
QCD at parameters and volumes of interest. Discussions of this question often conflate two separate concerns,

- **Scalability**: whether an approach can be practically applied to some target theory, for which the ultimate question is the cost to generate $N$ samples at a given set of parameters;
- **Cost scaling**: how the cost of an approach changes as certain parameters are varied—in this case, the parameters of the target lattice QCD theory, including the parameters of the action and the lattice geometry.

When we ask whether an approach to gauge field sampling for QCD is “scalable,” we are asking not about its precise scaling properties, but whether it will work for QCD on state-of-the-art volumes at state-of-the-art parameters, and whether it can be used to push the state of the art further. For the sampling of gauge field configurations using HMC, cost scaling relations often directly and usefully predict scalability. However, the range of applicability, and hence utility, of any practical cost-scaling relations for flow-based algorithms is much more limited. The rest of this manuscript is devoted to elucidating this statement.

To that end, Sect. 2 first reviews HMC and flow-based sampling methods for lattice field theory, establishing definitions and notation for the rest of the discussion. Section 3 then compares flow-based approaches with HMC, emphasizing counterintuitive differences to the more familiar HMC paradigm. Section 4 presents numerical illustrations of different aspects of the scaling of flow-based approaches, demonstrating the limitations of scaling laws in assessing scalability. Finally, Sect. 5 speculates on potential paradigms for flow-based sampling at scale and discusses the outlook for these methods.

## 2 Preliminaries

### 2.1 HMC

In the HMC approach, gauge field configurations are generated in a Markov chain, where approximate molecular dynamics evolution in the fictitious “Monte Carlo time” direction is used to propose updates to the chain. Each step of the HMC algorithm proceeds by drawing fictitious momentum variables conjugate to each lattice field degree of freedom, integrating the Hamiltonian equations of motion, then accepting or rejecting the resulting new field configuration with a Metropolis test to correct for integrator errors and guarantee exactness of sampling [14–16]. For gauge theories with fermion content, such as QCD, the gauge fields are evolved in a fixed background of auxiliary “pseudofermion” fields, which encode fermionic effects stochastically [66,67].

In this case, each integrator step requires solving large sparse systems of linear equations using implicit methods like conjugate gradient. For QCD, these solves typically dominate the computational cost. See e.g. Refs. [68–70] for more detailed reviews.

### 2.2 Normalizing flows and flow-based sampling

Normalizing flows are a framework for building exact, numerically tractable, machine-learned maps between probability distributions. In this construction, a diffeomorphic flow function $f$ is applied to transform samples $z$ drawn from a base distribution $r$ to obtain samples $\phi = f(z)$ distributed according to a model distribution $q$. The flow $f$ is parametrized by neural networks and can be optimized to some objective by the minimization of a “loss function”. Conservation of probability gives the density of transformed samples,

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

Various frameworks have been developed to construct expressive, trainable flow transformations for which this expression can be evaluated tractably [27,28,34,36,64,71–77].

Flow transformations are a general tool which can be applied as components of different sampling approaches (and in non-sampling applications). In particular, in the “direct sampling” approach, which is the primary concern of this work, flows relate a simple, easy-to-sample base distribution to a learned approximation $q \approx p$ of the target lattice field distribution, $p = \exp[-S]/Z$. Evaluating the reweighting factors $w(\phi) = p(\phi)/q(\phi)$ between the model $q$ and target $p$ provides sufficient information to compute expectations under $p$. This may be accomplished e.g. by computing reweighted expectations as $\langle O \rangle_p = \langle w O \rangle_q$. Alternatively, samples from $p$ can be generated by using flow model samples as proposals for the Independence Metropolis algorithm [78–80] with acceptance probability $p_{\text{acc}}(\phi \rightarrow \phi') = \min[1, w(\phi')/w(\phi)]$. Powerfully, this approach allows composition with other MCMC algorithms with complementary properties [25]. As discussed further below, these straightforward applications are only a few examples of how flow-based sampling algorithms may be constructed.

Even within a particular sampling framework, there is no unique way to construct a flow model. Each concrete realization – a model architecture – is defined by many different choices. From high- to low-level, these are broadly:

- domain of the variables $z, \phi$ (e.g. $\mathbb{R}$, $\text{SU}(N)$, multiple fields);
• the choice of base distribution \( r(z) \) (e.g. Gaussian, Haar uniform, free theory);
• strategy for constructing flow functions (e.g. convex potential flows [74–76], neural ODEs [77], or coupling layers [64,71] constructed using affine transformations [64], non-compact projections [73], neural splines [72], or gauge-equivariant transformations [27, 28,34]);
• structure of neural networks parametrizing the flow transformation (e.g. fully connected vs. convolutional networks, choice of activation functions);
• and various hyperparameters (e.g. number of coupling layers, neural network depths and widths).

While flow-based sampling approaches in principle guarantee unbiased results by construction even for models with \( q \) arbitrarily different from \( p \), performant sampling requires training the models. Just as for model architectures, many choices define a particular training scheme, including:

• scheme for initializing model parameters (e.g. random distribution, retraining);
• approach to training data (e.g. self-training, training on existing configurations);
• choice of loss function to minimize (e.g. KL divergence [81], Stein discrepancies [82,83], or gradient-based divergences like score matching or Fisher divergence [84,85]);
• optimization algorithm (e.g. SGD, Adam [86], second-order optimizers);
• all hyperparameters of training (e.g. optimizer parameters, batch size);
• and the schedule for training, which may vary any or all of these choices over time.

Besides direct sampling, other approaches to sampling lattice field distributions have been explored which use normalizing flows for different statistical modeling tasks. For example, Ref. [33] explored the use of flows which model a localized patch of a lattice field, conditioned on its environment. References [29,31] employed flows to generalize the proposal distribution in HMC. Reference [36] used flows to model conditional distributions for Gibbs sampling. Stochastic normalizing flows [87,88] and related approaches [44,89,90] use flows to relate a sequence of distributions interpolating between the base and target distributions. Reparametrization-based methods apply flows to the target distribution to obtain a “latent-space” distribution which may be more efficiently sampled with HMC [29,30,40, 91–97]. Flows play a different role in each case, requiring different architectures and training schemes.

2.3 Cost decomposition

For a generic sampling algorithm, the cost to generate a dataset equivalent to \( N_{\text{indep}} \) independent samples from a target distribution can be decomposed as

\[
C_{\text{total}}(N_{\text{indep}}) = C_{\text{setup}} + C_{\text{samp}}(N_{\text{indep}}),
\]

where \( C_{\text{setup}} \) is any up-front cost that must be paid before beginning data generation (e.g. the cost of training for flows, or of equilibration or decorrelating a forked stream for HMC), and \( C_{\text{samp}} \) is the cost of sampling thereafter. Generically, sampling costs may be further decomposed as

\[
C_{\text{samp}}(N_{\text{indep}}) = \Delta C_{\text{samp}} N_{\text{raw}}(N_{\text{indep}})
= \Delta C_{\text{samp}} N_{\text{indep}}^{\frac{1}{\text{ESS}}},
\]

where \( \Delta C_{\text{samp}} \) is the cost to generate a single configuration (e.g. the cost of a single HMC trajectory, or of drawing a sample from a flow model) and \( N_{\text{raw}} \) is the total number of configurations output by the procedure. The effective sample size per configuration \( \text{ESS} \equiv N_{\text{indep}}/N_{\text{raw}} \in [0, 1] \) quantifies the loss of statistical power due to e.g. MCMC autocorrelations or the increase in variance due to reweighting. For HMC, \( \text{ESS} = 1/\tau_{\text{int}} \), where \( \tau_{\text{int}} \) is the integrated autocorrelation time. For flow-based direct sampling, the reweighting-inspired metric \( \text{ESS} = 1/\langle w^2 \rangle_q \) [98,99] is commonly employed (see Appendix A for further discussion). Note however that no single scalar metric may fully quantify the performance of any sampling algorithm, as the true ESS is always observable-dependent: for HMC, observables are sensitive to different autocorrelation times, whereas for flows, each observable has different correlations with the reweighting factors. Both \( C_{\text{setup}} \) and \( \Delta C_{\text{samp}} \) carry units of compute time (e.g. GPU hours), and depend strongly on the precise details of the algorithm implementation and hardware.

3 Costs and cost scaling: HMC versus flow-based approaches

In this section, we first analyze the role and limitations of cost scaling relations in the familiar context of HMC for QCD. Subsequently, we discuss different general aspects of flow-based approaches and their cost scaling properties, emphasizing differences with HMC. Features better demonstrated with numerical examples are deferred to Sect. 4.
3.1 Expectations for scaling laws from HMC

The cost to sample QCD field configurations using HMC is often parametrized as a function of the physical parameters of the target theory as [69,100,101]

\[ C(N_{\text{indep}}) \propto N_{\text{indep}} \left( \frac{L}{a} \right)^{z_L} M_\pi^{-z_M} a^{-z_a} \]  

(6)

where \( a \) is the lattice spacing, \( L/a \) is the extent of the lattice in units of sites, and \( M_\pi \) is the pion mass. The exponents \( z_L, z_M, \) and \( z_a \) define the cost scaling relation. A key aspect of the difficulty in assessing scalability is that their values depend on many factors, in particular both algorithmic choices and the targeted regime of physical parameters, i.e.,

1. Scaling laws depend on how algorithm parameters are varied, and
2. Scaling laws have limited regimes of applicability.

Naturally, not only the parameters of an algorithm are relevant, but also the structure of the algorithm itself, with the important consequence that

3. Algorithmic developments can improve scaling properties.

Each of these key aspects are elaborated on below.

1. **Scaling laws depend on how algorithm parameters are varied.** Defining and quantifying a scaling relation requires choosing some scheme to vary algorithm parameters as the parameters of the target theory are varied. Different choices will yield different scaling relations. For example, discussions of HMC scaling typically quote a value of \( z_L = 5 \) for the exponent of \( L/a \) in Eq. (6). Part of this, \( (L/a)^4 \), is due to simple operation counting on a four-dimensional lattice. However, the remaining factor of \( L/a \) follows from choosing a particular scheme to vary the algorithm parameters with the volume, specifically increasing the number of integrator steps per trajectory to keep the acceptance rate fixed [102]. A different choice would yield a different scaling relation: for example, keeping the step size fixed will result in a rapid decline of the acceptance rate, inducing long autocorrelation times and poorer scaling.

2. **Scaling laws have limited regimes of applicability.** Although universal behavior may be expected in some regimes of target physical parameters, in practice a wide range of values of the exponent of the lattice spacing \( z_a \) have been quoted in the literature [69,103]. For example, many early studies [100,104–106] of HMC cost scaling found \( z_a \) near its conjectured lower bound of 2 [107]. However, approaching the continuum limit, increasingly large potential barriers prevent HMC from tunneling between disjoint topological sectors. This effect, known as “topological freezing”, induces very long autocorrelation times and thus large \( z_a \). In fact, the \( a \)-dependence may instead be consistent with exponential [19,108]. This large variation demonstrates the importance of assessing the limitations of any scaling law: extrapolating using an incorrect value of \( z_a \) for the parameter regime under consideration will not be predictive.

3. **Algorithmic developments can improve scaling properties.** Historically, new algorithmic developments have led to qualitative improvements in the capabilities and reach of HMC. For example, the scaling relation Eq. (6) was famously used to diagnose the “Berlin wall”: at the time (ca. 2001), large measured values of \( z_M \) implied that calculations at physical pion masses would be practically impossible with near-future hardware [100,101]. However, Hasenbusch preconditioning [109] was developed soon afterwards, reducing \( z_M \) and opening access to the physical pion mass regime. The development of multigrid preconditioners has provided additional improvements [110,111].

3.2 Costs and cost scaling for flow-based approaches

History has demonstrated that algorithmic advances can redefine the limits of LQFT methods and enable new physics results. To that end, flow-based approaches provide an unprecedented new space of algorithms to explore. However, their costs and cost scaling properties can be counterintuitive from the HMC perspective. To understand these methods and their scaling properties, several key features of flow models must be appreciated, in particular that:

1. Model weights are algorithm parameters;
2. Flow evaluation costs do not typically vary with physical parameters, but model quality does.

In addition, the practical requirement for model training introduces conceptual complications. Specifically:

3. Training and sampling are different dynamical processes, but cannot be considered independently;
4. Training costs may be significant.

Finally, certain low-level computational concerns must be considered:

5. Operation counting can predict raw computational costs; and
6. Flow-based approaches may parallelize more effectively than serial samplers.

Each of these points is discussed below, emphasizing the conceptual differences between flow-based sampling approaches and the HMC paradigm.
1. Model weights are algorithm parameters. A fixed flow model architecture represents a parametric family of probability distributions. Within this family, a set of values for the neural network parameters $\theta$ ("model weights") specifies a particular model distribution, $q_{\theta}$. The ESS to sample some target theory $p$ will vary as a function of every model weight; their number thus defines the dimension of the space of algorithms to be explored. Relaxing the restriction to fixed architectures presents even further possibilities.

Conceptually, this is a drastically different situation from HMC, where the space of algorithm parameters is tractable to explore; flow models may have potentially billions of parameters. Of course, in practice, hand-tuning is impossible and model weights must instead be set implicitly by training e.g. with stochastic gradient descent methods. The practical analog to the set of HMC's algorithm parameters are thus the set of all of the choices involved in defining a training scheme and architecture described in Sect. 2.2. Different scaling behavior results from how these choices are varied with target parameters. Note that although training schemes are often described in terms of relatively few hyperparameters, this apparent reduction in complexity is artificial. As discussed further below, the need for training only adds complexity, rather than providing any simplification.

2. Flow evaluation costs do not typically vary with physical parameters, but model quality does. For HMC, the cost to generate each sample, $\Delta C_{\text{samp}}$, varies strongly with the parameters of the target theory due to the changing problem difficulty for the linear solvers. In contrast, applying a flow transformation typically involves only explicit algebraic operations. In this case, $\Delta C_{\text{samp}}$ is independent of the precise values of the weights of the model, and thus the target parameters they implicitly encode. It follows that, for a fixed architecture, the dominant cost scaling with target parameters is due to the variation of the ESS (note that this may not hold if $\Delta C_{\text{samp}}$ is dominated by linear solves to evaluate $p$, or e.g. for hybrid algorithms incorporating HMC updates). This severely limits the applicability of empirically measured cost scaling relations: as explored below, model quality is an unpredictable consequence of the complicated interplay between architecture, training scheme, and target theory.

3. Training and sampling are different dynamical processes, but cannot be considered independently. For HMC, the same process – evolution by Hamiltonian dynamics – is used for both setup and sampling. However, the same is not true for flows. Training is a search process in the high-dimensional space of model parameters, typically involving stochastic gradient descent, with dynamics arising from all the choices described in Sect. 2.2. Sampling dynamics depend on the interaction of fine-grained details of model quality with the role played by the model in the sampling algorithm (giving rise to e.g. the distribution of rejection run lengths in direct sampling with Metropolis). Unlike for HMC sampling, there is no reason to expect any common parametrization to apply to both cost components.

However, this does not mean that scaling behaviors of training and sampling costs can be studied in isolation. For flows, sampling efficiency is a function of expenditure on training, i.e., $\text{ESS} = \text{ESS} (C_{\text{train}})$. Among other consequences, this implies that the scaling behavior of sampling and training costs are not defined without precisely specifying the stopping condition for training. Different choices of stopping condition result in qualitatively different scaling relations. For example, if training always uses a fixed amount of computation, then $C_{\text{train}}$ scales trivially with the target parameters by construction, and all cost scaling is pushed on to $C_{\text{samp}}$, as explored in Sect. 4.2. Instead, one may choose to train until some target ESS is achieved. In this case, the ESS scales trivially while $C_{\text{train}}$ alone varies, as explored in Sect. 4.1. As demonstrated in Sects. 4.1 and 4.2, even the precise choice of fixed $C_{\text{train}}$ or target ESS can strongly affect scaling properties. Other choices of stopping condition between these extremes are possible, each inducing different scaling properties. This sensitivity implies that scaling behaviors for flows are highly non-generic, thus the generalizability of empirically assessed scaling relations is severely limited.

4. Training costs may be significant. Most existing applications of flows to sampling in LQFT have employed training methods which apply the flow (or its inverse) to batches of field samples. For such a training approach, training costs can be decomposed similarly to the decomposition of sampling costs in Eq. (5):

$$C_{\text{setup}} = \Delta C_{\text{train}} N_{\text{train}}$$

where $N_{\text{train}}$ is the total number of samples flowed during training and $\Delta C_{\text{train}}$ is the combined cost of flowing a single sample and then backpropagating gradients. Typically, $\Delta C_{\text{train}}$ is a few times larger than $\Delta C_{\text{samp}}$. Training from a random initialization often involves $\sim 1000s$ of optimizer steps using batches of $\sim 100s$ to $1000s$ of configurations, meaning $N_{\text{train}}$ can be much larger than typical QCD ensemble sizes. This potentially large scale for training costs is an important factor in determining what will be computationally viable for QCD. However, efficient parallelization (as discussed below) may help mitigate this cost in practice. Moreover, importantly, as explored in Sect. 4, these costs are highly non-generic and can be optimized significantly.

5. Operation counting can predict raw computational costs. In some cases, simple operation counting can yield relations describing the dependence of $\Delta C_{\text{samp}}$ and $\Delta C_{\text{train}}$ ($\Delta C$, collectively) on model hyperparameter choices and certain target theory parameters, such as the number of lattice sites $\Omega$. For example, for flow models built from composed sub-transformations (e.g., coupling layers), the cost of apply-
ing a flow is linear in the number of sub-transformations. Similarly, for flow transformations parametrized by neural networks which interconnect all lattice sites, operation counting predicts that \( \Delta C \propto \Omega^2 \). More favorably, transformations parametrized by convolutions over the lattice geometry may have \( \Delta C \propto \Omega \). (Note that these relations are often broken by e.g. cache effects or adaptive algorithm swapping, and may not apply for implementations on hardware such as GPUs and TPUs if their parallelism is not fully utilized.) Operation counting thus provides a useful but limited tool for assessing scaling properties and scalability: if the architecture itself is varied with the target, operation counting can predict scaling of \( \Delta C \), but cannot account for the (equally important) effect of varying model quality. Real-world resource constraints may induce further complications, e.g. if limited memory necessitates a trade-off between model size and the batch size for training.

6. Flow-based approaches may parallelize more effectively than serial samplers. Sampling from a flow model is embarrassingly parallelizable, and thus flow-based sampling admits new strategies for parallelism unavailable in the context of serial MCMC algorithms like HMC. For example, multiple compute nodes might independently and locally generate and store configurations \( \phi_i \), and pass only \( q(\phi_i) \) and \( p(\phi_i) \) to a central coordinator which updates the state of a global, distributed Markov chain with Independence Metropolis. Potential new strategies include not only the running of multiple independent samplers (without incurring additional setup costs), but also new schemes for problem division such as pipeline parallelism. Separately, the high arithmetic density of neural network operations suggests that flow-based sampling may not be communications-bound. Taken together, these properties suggest potential for flow-based methods to make more efficient use of computational resources than serial sampling algorithms, given real-world walltime and hardware constraints.

4 Limitations of scaling relations for flow-based approaches

The previous section discussed properties of flow-based sampling approaches for LQFT that are apparent from the definition of the approach. However, research and experimentation in this area has identified other properties of flows relevant to assessing scalability. This section presents simple numerical demonstrations of these features and discusses their implications for the utility of scaling relations. Note that these examples are applications to two-dimensional scalar field theory and the two-flavor Schwinger model, which are physically very different from QCD. However, the conclusions drawn from these experiments are not specific to the target theory, as

![Image](image_url)

**Fig. 1** Model quality as a function of training cost for two approaches to training, learning rate (LR) scheduling and batch size (BS) scheduling, for a flow model targeting real scalar field theory for \( m^2 = -1 \) and \( \lambda = 1 \) on a \( 10 \times 10 \) lattice. Each training scheme is applied to the same architecture. Cost is measured in units of the number of samples generated for training, \( N_{\text{train}} \). Dashed black lines indicate potential target ESSes of 0.2 and 0.6, as referenced in the main text. The batch size for “Schedule LR” is 16,384 throughout, while for “Schedule BS” it begins at 128 and doubles every 5000 steps until it reaches 16384. For “Schedule LR”, the learning rate is halved every 5000 steps. The ESS is measured using the training batch size every step, and smoothed over a rolling window of width 250 steps. Results vary by \( \approx 5\% \) under repeated experiments with different pseudorandom seeds. Further details are provided in Appendix A.

The necessary conclusion is that the scaling of training costs is entirely specific to the approach employed.

1. Training costs may be optimized by orders of magnitude,
2. Transfer learning can mitigate training costs, and
3. Training cost scaling depends on training protocol.

The precise improvement factor depends on the criterion used to stop training. For example, in this demonstration, batch size scheduling is \( \approx 110 \) times less expensive
than learning rate scheduling for training to a target ESS of 0.2, while for a target ESS of 0.6 the improvement is only a factor of ~12. Note that the intent of this demonstration is not to emphasize the utility of this batch size reduction method, but rather to show the degree to which training costs may vary with approach. The practical consequence for assessing scalability is that training costs, and their scaling, are extremely sensitive to even minor perturbations in training protocol.

**2. Transfer learning can mitigate training costs.** Flow models can be retrained between theories with the same lattice geometry but different action parameters. Figure 2 illustrates the utility of this technique – “parameter transfer” – in training models for three different sets of Schwinger model target parameters. The large cost of training a model from a random initialization must always be paid once. However, as demonstrated, models targeting further parameters may be retrained from this initial model at significantly reduced cost compared with from-scratch training. Sequentially retraining along a trajectory through parameter space provides further improvement [25], at the cost of serializing training for different parameters.

Certain flow architectures, such as those parametrized by convolutional neural networks, additionally permit “volume transfer”: a model trained to generate configurations with one geometry may be used to generate configurations with another [28]. This makes it possible to perform the bulk of training for smaller volumes than the final target, and enables methods that are intractable for larger volumes, e.g., training with on-the-fly HMC generation [25] or using exact fermion determinants [32,36]. In practice, we often find that retraining on the larger volume is unnecessary. For models trained on smaller volumes and applied to larger ones with no retraining, there is a generic cost scaling relationship with changing volume, detailed in Sect. 4.2.

The consequence of these observations is that, for the purposes of assessing scalability, the cost scaling properties of training from a random initialization are irrelevant if transfer learning is employed. Instead, the relevant scaling properties are those of the cost of retraining each new model, which will depend intimately on the set of physical parameters that are of interest in a particular study, and the order in which models are retrained between them.

**3. Training cost scaling depends on training protocol.** Training costs and their scaling properties are determined by various choices made in training. As discussed in Sect. 3.2, this includes the stopping condition for training, without which cost scaling behavior is not well-defined. For example, training for a fixed number of iterations implies a fixed training cost but varying model quality. Alternatively, training to a target ESS results in fixed sampling costs while the training costs vary with the target theory. The scaling of those training costs with the parameters of the target theory depends sensitively on the precise choice of target ESS, as illustrated in Fig. 3. As already discussed in Sect. 3.2, this demonstration that scaling relations are highly specific to the training approach implies that, even if scaling behaviour can be determined empirically for some particular approach, it is unlikely to be generic across different training approaches, let alone across different theories or flow models.

### 4.2 Sampling costs

In this section, we demonstrate various limitations of scaling laws in describing sampling costs in flow-based approaches. In particular, as model quality determines sampling efficiency, the scaling of model quality with the parameters of the target theory is considered as a proxy for the scaling of sampling costs. We provide arguments for and numerical illustrations of several key properties. First, defining the maximum achievable model quality is a subtle question, since

1. Achievable model quality is not a function of architecture alone.

Moreover, the scaling of model quality with parameters of the theory is complex, since

2. Model quality scaling depends on training protocol, and
3. Different architectures scale differently.

One scaling, however, can be derived generically:

4. For fixed models, quality scales exponentially in volume.

Nevertheless, assessing scalability – i.e., utility at scale – from scaling laws is difficult, since

5. Best scaling does not imply best performance, and
6. Architecture dependence implies theory dependence.

Finally, we discuss the most relevant concern for scalability at present:

7. Improving performance requires physics-informed algorithm design.

While the provided numerical demonstrations exclusively use the direct sampling approach, the conclusions apply more generally to sampling algorithms including flow models or other machine-learned components.

1. **Achievable model quality is not a function of architecture alone.** It is typical in gradient-descent-based optimization that after a period of rapid initial learning, optimization will enter a regime where model quality plateaus or improves...
The practical consequences are that cost scaling laws are not direct retrained from a single model; sequentially retraining parameters for a fixed architecture, although we emphasize that these functional forms are necessarily specific to how the model is retrained from a single model, and green for sequentially retraining from $\kappa = 0 \rightarrow 0.25 \rightarrow 0.263$. Training is stopped when the target ESS is achieved in average over 100 epochs. Horizontal dotted lines mark the target ESS = 0.9, 0.7, 0.45 for $\kappa = 0, 0.25, 0.263$, respectively. Further details are provided in Appendix A.

Figure 3 Demonstration in real scalar field theory, with $m^2 = -2$ on a $10 \times 10$ lattice, of the differences in training cost scaling behavior in $1/\lambda$ due to different choices of the stopping condition for training. Each model is retrained from $\lambda = 4$ to the target $\lambda$, halting training on the first step where the target ESS is achieved. The lack of smoothness is an inherent complication of quantifying scaling laws when using stopping conditions based on model quality, here due to the stochastic evaluation of the ESS as well as the inherent noise of training. Further details are provided in Appendix A.

2. Model quality scaling depends on training protocol. The scaling behavior of model quality is necessarily determined by the choice of training scheme, which governs how model weights vary with the parameters of the target theory. As for training costs, this includes the stopping condition for training. Figure 5 demonstrates how different stopping criteria – specifically, different choices of fixed expenditure on training – induce different functional forms in model quality as a function of target theory parameters when retraining between target parameters. In this figure, models for all parameters are directly retrained from a single model; sequentially retraining instead would result in different functional forms again. The practical consequences are that cost scaling laws are not unique even for a particular architecture and training method.

3. Different architectures scale differently. Figure 6 provides a simple demonstration of how the model quality of even very similar architectures may exhibit different scaling behaviors with the parameters of the target theory. In this example, larger models (with more free parameters) exhibit better overall ESS and better scaling properties towards criticality – although the changing trade-off between ESS and $\Delta C_{\text{samp}}$ results in crossovers of the sampling performance, as discussed further below. The smooth dependence of ESS (and consequently performance) on target parameters implies useful extrapolation can be possible over a small range of target parameters for a fixed architecture, although we emphasize that these functional forms are necessarily specific to how the models are trained.
4. For fixed models, quality scales exponentially in volume. As demonstrated throughout this work, generic cost scaling laws for flow models are difficult to obtain due to the complexity of the method. However, a particular restriction allows one to be derived analytically. Specifically, a straightforward argument implies that the quality of a fixed model constructed from a local flow transformation degrades exponentially under volume transfer: if both $p$ and $q$ are defined for arbitrary volumes $V$ and may be characterized by correlation lengths $\xi_p$ and $\xi_q$, then when $L \gg \xi_p, \xi_q$ the integral defining the ESS factorizes over decorrelated subvolumes and scales as

$$\text{ESS}(V) = \text{ESS}(V_0)^{V/V_0}. \quad (8)$$

Figure 7 demonstrates the onset of this effect in the large-volume regime for the Schwinger model, but we observe it to hold for other target theories as well. This effect was first described in Ref. [33].

It is important to emphasize the limited applicability of this scaling relation. Specifically, it applies only when the physical extent $L/\xi_p$ is large, compared with the extent in lattice sites $L/a$. Thus, while this relation may present an obstruction to flow-based sampling in the thermodynamic limit $L/\xi_p \rightarrow \infty$, it does not obstruct sampling in the continuum limit $\xi_p/a \rightarrow \infty$ with $L/\xi_p$ fixed. Away from the thermodynamic limit, the breakdown of this relation can have counterintuitive effects: as shown in Fig. 7, we have observed better-than-exponential scaling in the Schwinger model when $L/\xi_p$ is small. The exponential scaling relation furthermore does not apply between theories in different dimensions or with different numbers of internal degrees of freedom. Most importantly, however, it does not relate different approaches or otherwise constrain what overall model quality (and thus quality of volume scaling properties) is achievable. Finally, the argument for this scaling relation holds only for fixed models transferred without retraining.

This effect does not present any principled obstacle to flow-based sampling at state-of-the-art QCD parameters where typically $L/\xi_p = M_p L \sim 4–10$. In practice, we are not interested in the thermodynamic limit, only control over finite-volume effects. However, it does emphasize the importance of developing high-quality models to reach volumes of interest. Sampling approaches which require modeling only subvolumes [33] may control the effect directly.

5. Best scaling does not imply best performance. Of course, scaling of model quality is not the sole factor determining sampling efficiency. In fact, in the demonstration of Fig. 6, an architecture with one of the least favorable scaling behaviors...
Fig. 6 Scaling behavior of model quality with respect to theory parameters, for different model architectures applied to real scalar field theory with \( m^2 = -4 \) on a 16 \( \times \) 16 lattice. Each line denotes the ESS (top panel) and sampling performance (bottom panel) of a different affine coupling architecture, all trained using the same protocol. Performance is measured in effective samples per second, quantified for each model as its ESS divided by the computational cost of generating a configuration in units of RTX2080 Ti GPU-seconds. Note that these timings are specific to our implementation. Models are trained as described in Appendix A. In the legend, \( K \) denotes the convolutional kernel size and \( n_L \) denotes the number of coupling layers. Further details are provided in Appendix A.

provides the best performance in our implementation. For this small model, the lower ESS is more than compensated for by its low cost of evaluation, \( \Delta C_{\text{samp}} \). This emphasizes the important point that, ultimately, only sampling performance at parameters of interest matters; scaling laws are only important insofar as they can be used to predict the performance of an algorithm at one target set of parameters from its performance elsewhere.

6. Architecture dependence implies theory dependence. As demonstrated in Fig. 6, even different choices of hyperparameters within an architecture class can lead to significantly different scaling of model quality with the parameters of the target theory. As discussed in Sect. 2.2, much more significant differences in architecture are possible, from which we should expect an even greater variety in scaling behaviors. Importantly, treating different physical theories requires structurally different architectures. For example, SU(\( N \)) variables cannot be flowed using the same transformations applicable to real scalar fields. Further theory-specific engineering is necessary to encode symmetries and other physical features. Given the dissimilarity between architectures required for different theories, it should not be assumed that features of model quality obtained for one theory apply for any other theory of interest. This is further complicated by the interaction of model quality with different varieties of critical behavior. We conclude that studying the scaling of flow model sampling for toy models such as \( \phi^4 \)-theory or even the Schwinger model provides little information about

Fig. 7 Volume scaling of the ESS for models trained to sample the Schwinger model at \( \beta = 2 \) and three different values of the fermion mass, including the pure-gauge limit, \( \kappa = 0 \). The top and bottom panels illustrate the volume dependence of the ESS and pseudoscalar mass \( aM_\pi \), respectively. For each target \( \kappa \), models are trained from scratch at \( L = 8 \) and volume transferred without retraining. In the top panel, the dotted lines are extrapolations backwards from the largest volume \( L = 24 \), where the ESS takes values \( \approx 0.58, 0.44, \) and \( 0.24 \), using Eq. (8). Deviations of the markers below these lines correspond to better-than-exponential scaling at smaller volumes, where the lower panel shows significant finite-size effects in \( aM_\pi \). The ESS is evaluated on \( 10^4 \) samples at each volume. Error bars are smaller than the markers. Further details are provided in Appendix A.

Fig. 8 Dependence of ESS on number of coupling layers \( n_L \) for two different architectures, with all other architecture and training hyperparameters fixed, for real scalar field theory on a 16 \( \times \) 16 lattice with \( m^2 = -4 \) and \( \lambda = 1.25 \). The architectures differ by only the base distribution: independent normal on each site (“Normal”), and the free-field distribution with a learned pole mass (“Free field”; as described in Appendix A). Each model is trained from a random initialization for 150 \( k \) steps, which is sufficient to reach a slowly improving regime in all cases. Training uses batch size 1024, and the learning rate is decayed by a factor 2 every 10,000 steps. Uncertainties for each point include the spread between two repeated experiments with different pseudorandom seeds. Further details are provided in Appendix A.
scaling properties for any flow-based approach to sampling gauge field configurations for QCD.

7. Improving performance requires physics-informed algorithm design. As seen in Figs. 1, 2, and 4, training with a fixed protocol eventually enters a slowly improving regime, after which point the expense of additional training will not reliably provide a practical increase in sampling efficiency. Similarly, as illustrated in Fig. 8, increasing the model size within fixed architecture classes often provides diminishing improvements in the model quality achieved by a fixed training protocol. Although further training or increase in model size may provide sudden improvements (e.g., second-wind training dynamics as in Fig. 4), these saturation-like behaviors present a practical obstacle to increasing sampling efficiency using “brute force” application of additional computational power.

Instead, improving sampler efficiency requires more qualitative algorithmic improvements. Among the infinite possible variations of model architecture, it is natural to expect that choices which incorporate a priori physics understanding will lead to better model quality. This is illustrated in Fig. 8, which compares two closely related architectures for scalar field theory. Here it is clearly visible that the physics-informed modification is far more successful at improving model quality than simply increasing the model size. Along the same lines, developing efficient flow-based sampling algorithms for QCD will require developing and testing new architectures, training schemes, and sampling approaches which incorporate a priori physics knowledge.

5 Outlook

For flow-based approaches to the sampling of lattice field configurations, even small differences in the ML approach—architecture, training scheme, and how they are varied with the parameters of the target theory—result in not only different overall costs, but also different cost scalings with the parameters of the target theory. In effect, each ML approach defines a different sampling algorithm with different cost scaling properties. Further, because theory-specific modeling is required, scaling properties assessed in one theory should not be expected to generalize to others. Taken together, this implies a very different paradigm for assessing algorithm scalability than has applied for QCD algorithms thus far, where scaling properties assessed in toy theories often translate directly to QCD. For flow-based methods, assessing scalability will require direct, experimental investigation of applications to QCD itself, which has only just begun [35].

It is not yet clear what at-scale applications of flow-based sampling methods to QCD will look like, but we can speculate, and some key aspects are already clear. For example, transfer learning and retraining will play a central role in mitigating training costs. Thus, architectures well-suited for transfer learning across wide ranges of parameters and volumes will be important in order to exploit this technique. Exponential volume scaling suggests that high-quality models will be necessary to achieve efficient sampling at scale. Coupled with the large overall scale of training costs, this suggests a paradigm of use very different from HMC. As seen for large ML models in industry applications, such as GPT [116] and MT-NLG [117], the typical paradigm is that significant computational resources and human time are invested over years in exploration and training. The resulting models can then be shared as a community resource, amortizing the bulk of training and development costs across the community as a whole.

Direct sampling with flow models has significant natural advantages over serial algorithms such as HMC, but requires high-quality models for efficient sampling. However, there are many other sampling approaches where flows may be employed. “Hybrid” approaches involving both flow components and more traditional sampling algorithms can exploit lower-quality models while leveraging the decades of engineering invested in algorithms like HMC. It is likely that the first at-scale application will involve such a hybrid approach. Over the longer term, improving flow model technology will enable increasingly efficient sampling approaches.

While the growing body of work on flow-based sampling methods continues to provide promising early results, we have only just begun to explore the space of what is possible. The broader field of machine learning is advancing rapidly, and experience dictates that we cannot anticipate what capabilities will be enabled by new developments. Creativity guided by physical intuition remains our most effective means of making progress.

Acknowledgements The authors thank Heiko Strathmann for detailed comments on a draft of this manuscript, and Gurtej Kanwar for useful discussions. RA, DCH, FRL, PES, and JMU are supported in part by the U.S. Department of Energy, Office of Science, Office of Nuclear Physics, under grant Contract Number DE-SC0011090. PES is additionally supported by the National Science Foundation under EAGER grant 2035015, by the U.S. DOE Early Career Award DE-SC0021006, by a NEC research award, and by the Carl G and Shirley Sontheimer Research Fund. KC and MSA are supported by the National Science Foundation under the award PHY-2141336. MSA thanks the Flatiron Institute for their hospitality. DB is supported by the Argonne Leadership Computing Facility, which is a U.S. Department of Energy Office of Science User Facility operated under contract DE-AC02-06CH11357. This work is funded by the U.S. National Science Foundation under Cooperative Agreement PHY-2019786 (The NSF AI Institute for Artificial Intelligence and Fundamental Interactions, http://iaifi.org/). This work is associated with an ALCF Aurora Early Science Program project, and used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DEAC02-06CH11357. The authors acknowledge the MIT SuperCloud and Lincoln Laboratory Supercomputing Center [118] for providing HPC resources that have contributed to the research results reported within this paper. Numerical experiments and data...
analysis used PyTorch [119], JAX [120], Haiku [121], Horovod [122], NumPy [123], and SciPy [124]. Figures were produced using matplotlib [125].

Funding Open Access funding provided by the MIT Libraries.

Data availability This manuscript has no associated data or the data will not be deposited. [Authors’ comment: No associated data.]

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

Appendix A: Details of numerical examples

Numerical illustrations of flow models trained to sample scalar field configurations are optimized to model the action

\[ S(\phi) = \sum_x \left[ -\sum_{\mu=1}^d \phi(x) \bar{\phi}(x + \hat{\mu}) \right. \]
\[ \left. + \frac{1}{2} (m^2 + 2d) \phi(x)^2 + \lambda \phi(x)^4 \right] \tag{A1} \]

where \( d = 2 \). Architectures are stacks of \( n_L \) affine coupling transformations with checkerboard variable partitioning, parametrized by convolutional neural networks, as implemented in Ref. [22]. All architectures have \( n_L = 24 \) coupling layers, convolutional kernel size \( K = 5 \), and two layers of hidden channels of width 12 within each neural network, except for the model used to generate the results shown in Figs. 6 and 8 where \( n_L \) and \( K \) differ as noted. Reverse KL self-training protocols are also similar to Ref. [22], but with the addition of learning rate or batch size scheduling where indicated. Gradient clipping has also been applied, specifically rescaling all gradients by a common factor as necessary to prevent the norm over all gradients from exceeding 100. Training and ESS evaluation both use batch size 16,384, except as noted in Fig. 1. Results shown in Fig. 6 are initially trained for 150,000 steps at \( \lambda = 1.25 \), then for 500 steps at each subsequent \( \lambda \), one after the other. This is sufficient to reach a slowly improving regime for all \( \lambda \), and produces comparable final results as from-scratch training or starting instead from \( \lambda = 4 \). In all cases, from-scratch training begins with learning rate \( 10^{-3} \). For the results shown in Figs. 3, 5, and 6, when retraining, the learning rate is set to \( 10^{-4} \) and the optimizer state is never reset (i.e. it is carried over from training for the previous parameters). Throughout, optimizer hyperparameters are the Pytorch 1.10 defaults unless otherwise specified.

Sampling from the “Free field” base distribution used in the demonstration of Fig. 8 is accomplished using a layer which transforms the scalar field variables in momentum space. We first perform a change of basis into Fourier space via \( \phi(k) = \sum_x \phi(x) e^{-ik} \). We then transform the resulting momentum-space variables as \( \phi'(k) = \sigma_k \phi(k) \), with

\[ \sigma_k^2 = \left( \mu + 2d - 2 \sum_{i=1}^4 \cos(k_i) \right)^{-1} \tag{A2} \]

where \( \mu \) is a learned parameter. Finally, the updated variables are transformed back to position space via \( \phi'(x) = \sum_k \phi'(k) e^{ik} / \Omega \). Comparing with the free-field (i.e. \( \lambda = 0 \)) action in momentum space,

\[ S = \frac{1}{2} \sum_k \left( m^2 + 2d - 2 \sum_v \cos(k_v) \right) |\phi(k)|^2, \tag{A3} \]

we see that if \( \phi(k) \) are independent Gaussians for each \( k \), then the transformation amounts to sampling from the free theory with learnable pole mass \( \mu \). This holds when \( \phi(x) \) are independent Gaussians on each site, i.e. if the layer is applied directly after the draw from the typical base distribution, as employed above.

Results shown in Figs. 2 and 7 are obtained for models trained to sample from the theory defined by the two-flavor Wilson fermion action as employed in Ref. [32], specifically

\[ S_E(U) = -\beta \sum_x \text{Re} P(x) - \log \det D[U] D[U] \tag{A4} \]

where \( \beta = 2/g_0^2 \) encodes the bare gauge coupling \( g_0 \), and

\[ P(x) = U_0(x) U_1(x + \hat{0}) U_0^\dagger(x) U_1^\dagger(x) \tag{A5} \]

is the plaquette. The Wilson discretization of the Dirac operator is

\[ D[U](y, x) = \delta(y - x) \]
\[ - \kappa \sum_{\mu=0,1} \left[ (1 - \sigma_{\mu}) U_{\mu}(y) \delta(y - x + \hat{\mu}) \right. \]
\[ \left. + (1 + \sigma_{\mu}) U_{\mu}^\dagger(y - \hat{\mu}) \delta(y - x - \hat{\mu}) \right], \tag{A6} \]

where \( \sigma_0, \sigma_1 = \sigma_x, \sigma_y \) are the Pauli matrices and \( \kappa = 1/(4 + 2m_0) \) encodes the bare fermion mass \( m_0 \). Architec-
tures are similar to those of Ref. [32], with three differences: (1) models are built of 24 gauge-equivariant coupling layers rather than 48; (2) neural networks use standard convolutions with kernel size 3 rather than dilated convolutions; and (3) intermediate layers have 32 channels rather than 64. All models used to create the results in Figs. 2 and 7 have the same architecture. Note that training and sampling uses exact computation of the fermion determinant, without any stochastic pseudofermion estimators like in Ref. [34].

To generate the results shown in Fig. 7, models are self-trained with batch size 6144, for ~ 17 k, 43 k, and 85 k steps for κ = 0.25, and 0.263, respectively, at which points training has entered a slowly improving regime. The initial learning rate 3 \times 10^{-5} is decayed by a factor 0.5 every 10 k steps. The pseudoscalar mass \( M_p \) is obtained as the effective mass at the center of the lattice, as measured on configurations generated using flow-model proposals with independence Metropolis.

Throughout this work, the ESS metric used to evaluate model quality is estimated as

\[
\text{ESS} = \frac{1}{\langle w^2 \rangle_q} \approx \frac{1}{N} \left( \sum_i \bar{w}_i \right)^2 \quad (A7)
\]

where \( \bar{w}_i = \exp[-S(\phi_i)]/q(\phi_i) \). This estimator is known to be positively biased at finite sample size, especially in the presence of mode collapse, where undersampling of mismodeled regions may result in an apparently large ESS when the true value may be near zero [25]. This can be diagnosed using validation data drawn from the target distribution, using the reweighted estimator [25]

\[
\text{ESS} = \frac{1}{\langle w \rangle_p} \approx \frac{1}{N} \left( \frac{1}{\sum_i \bar{w}_i} \right) \left( \frac{1}{N} \sum_i \bar{w}_i \right). \quad (A8)
\]

Using the target-sample estimator to spot-check the modelsample ESS estimates presented throughout this work reveals no significant discrepancies.

References

1. C. Morningstar, The Monte Carlo method in quantum field theory (2007). arXiv:hep-lat/0702020
2. C. Lehner et al., Opportunities for lattice QCD in quark and lepton flavor physics. Eur. Phys. J. A 55(11), 195 (2019). https://doi.org/10.1140/epja/i2019-12891-2. arXiv:1904.09479 [hep-lat]
3. A.S. Kronfeld, D.G. Richards, W. Detmold, R. Gupta, H.W. Lin, K.F. Liu, A.S. Meyer, R. Sufian, S. Syritsyn, Lattice QCD and neutrino-nucleus scattering. Eur. Phys. J. A 55(11), 196 (2019). https://doi.org/10.1140/epja/i2019-12916-x. arXiv:1904.09931 [hep-lat]
4. V. Cirigliano, Z. Davoudi, T. Bhattacharya, T. Izubuchi, P.E. Shanahan, S. Syritsyn, M.L. Wagman, The role of lattice QCD in searches for violations of fundamental symmetries and signals for new physics. Eur. Phys. J. A 55(11), 197 (2019). https://doi.org/10.1140/epja/i2019-12899-8. arXiv:1904.09704 [hep-lat]
5. W. Detmold, R.G. Edwards, J.J. Dudek, M. Engelhardt, H.W. Lin, S. Meinel, K. Orginos, P. Shanahan, Hadrons and nuclei. Eur. Phys. J. A 55(11), 193 (2019). https://doi.org/10.1140/epja/i2019-12902-4. arXiv:1904.09512 [hep-lat]
6. A. Bazavov, F. Karsch, S. Mukherjee, P. Petreczky, Hot-dense lattice QCD: USQCD white paper 2018. Eur. Phys. J. A 55(11), 194 (2019). https://doi.org/10.1140/epja/i2019-12922-0. arXiv:1904.09951 [hep-lat]
7. B. Joó, C. Jung, N.H. Christ, W. Detmold, R. Edwards, M. Savage, P. Shanahan, Status and future perspectives for lattice gauge theory calculations to the exascale and beyond. Eur. Phys. J. A 55(11), 199 (2019). https://doi.org/10.1140/epja/i2019-12919-7. arXiv:1904.09725 [hep-lat]
8. R.C. Brower, A. Hasenfratz, E.T. Neil, S. Catterall, G. Fleming, J. Giedt, E. Rinaldi, D. Schau, E. Weinberg, O. Witzel, Lattice gauge theory for physics beyond the standard model. Eur. Phys. J. A 55(11), 198 (2019). https://doi.org/10.1140/epja/i2019-12901-5. arXiv:1904.09964 [hep-lat]
9. T. DeGrand, Lattice tests of beyond Standard Model dynamics. Rev. Mod. Phys. 88, 015001 (2016). https://doi.org/10.1103/RevModPhys.88.015001. arXiv:1510.05018 [hep-ph]
10. B. Svetitsky, Looking behind the Standard Model with lattice gauge theory. EPL Conf. 175, 01,017 (2018). https://doi.org/10.1051/epiconf/201817501017. arXiv:1708.04840 [hep-lat]
11. G.D. Kirb, E.T. Neil, Review of strongly-coupled composite dark matter models and lattice simulations. Int. J. Mod. Phys. A 31(22), 1643004 (2016). https://doi.org/10.1142/S0217751X16430041. arXiv:1604.04627 [hep-ph]
12. I. Ichinose, T. Matsui, Lattice gauge theory for condensed matter physics: ferromagnetic superconductivity as its example. Mod. Phys. Lett. B 28, 1430012 (2014). https://doi.org/10.1142/S0217751X14300121. arXiv:1408.0089 [cond-mat.str-el]
13. M. Mathur, T.P. Sreeaj, Lattice gauge theories and spin models. Phys. Rev. D 94, 085029 (2016). https://doi.org/10.1103/PhysRevD.94.085029. arXiv:1604.00315 [hep-lat]
14. S. Duane, A.D. Kennedy, B.J. Pendleton, D. Roweth, Hybrid Monte Carlo. Phys. Lett. B 195, 216–222 (1987). https://doi.org/10.1016/0370-2693(87)91197-X
15. R.M. Neal, Probabilistic inference using Markov chain Monte Carlo methods (Department of Computer Science, University of Toronto Toronto, ON, Canada, 1993), chap. 5
16. R.M. Neal, Bayesian Learning for Neural Networks. Lecture Notes in Statistics, vol. 118 (1996)
17. U. Wolff, Critical slowing down. Nucl. Phys. Proc. Suppl. 17, 93–102 (1990). https://doi.org/10.1016/0920-5632(90)90224-I
18. S. Racanière, D.J. Rezende, P.E. Shanahan, Introduction to normalizing flows for lattice field theory (2021). arXiv:2101.08176 [hep-ph]
19. S. Racanière, D.J. Rezende, P.E. Shanahan, Normalizing flows for lattice field theory calculations to the exascale and beyond. Eur. Phys. J. A 55(11), 197 (2019). https://doi.org/10.1140/epja/i2019-12899-8. arXiv:1904.09704 [hep-lat]
62. S. Chen, O. Savchuk, S. Zheng, B. Chen, H. Stocek, L. Wang, K. Zhou, Fourier-flow model generating Feynman paths (2022). arXiv:2211.03470 [hep-lat]
63. D.J. Rezende, S. Mohamed, Variational inference with normalizing flows (2016). arXiv:1505.05770 [stat.ML]
64. L. Dinh, J. Sohl-Dickstein, S. Bengio, Density estimation using Real NVP (2017). arXiv:1605.08803 [cs.LG]
65. G. Papamakarios, E. Nalisnick, D.J. Rezende, S. Mohamed, B. Lakshminarayanan, Normalizing flows for probabilistic modeling and inference (2019). arXiv:1912.02762 [stat.ML]
66. D.H. Weingarten, D.N. Petcher, Monte Carlo integration for lattice fermionic systems. Nucl. Phys. B 180, 369 (1981). https://doi.org/10.1016/0550-3213(81)90055-9
67. F. Fucito, E. Marinari, G. Parisi, C. Rebbi, A proposal for Monte Carlo simulations of fermionic systems. Nucl. Phys. B 180, 369 (1981). https://doi.org/10.1016/0550-3213(81)90055-9
68. R.M. Neal et al., MCMC using Hamiltonian dynamics. Handbook of Markov chain Monte Carlo vol. 2(11), p. 2 (2011)
69. C. Gattringer, C.B. Lang, Quantum chromodynamics on the lattice, vol. 788 (Springer, Berlin, 2010). https://doi.org/10.1007/978-3-642-01850-3
70. T. DeGrand, C.E. Detar, Lattice methods for quantum chromodynamics (2006)
71. L. Dinh, D. Krueger, Y. Bengio, Nice: non-linear independent components estimation (2014). arXiv:1410.8516
72. D. Kingma, J. Ba, Adam: a method for stochastic optimization (2017). arXiv:1412.6980 [cs.LG]
73. D.J. Rezende, G. Papamakarios, S. Mohamed, Variational inference with normalizing flows on Tori and spheres (2020). arXiv:2002.02428 [stat.ML]
74. D.P. Kingma, J. Ba, Adam: a method for stochastic optimization (2022). arXiv:2002.06707 [stat.ML]
75. S. Schaefer, Status and challenges of simulations with dynamical fermions. PoS LATTICE2012, 001 (2012). https://doi.org/10.22323/1.164.0001. arXiv:1211.5069 [hep-lat]
76. S. Jaffard, D.J. Fradin, O. Lague, H. Sász, A. Vourdas, VAE flows: surjections to bridge the gap between VAEs and flows. Adv. Neural Inf. Process. Syst. 33, 12685–12696 (2020)
77. D. Nielsen, P. Jaini, E. Hoogeboom, O. Winther, M. Welling, Sur-VAE flows: surjections to bridge the gap between VAEs and flows. Adv. Neural Inf. Process. Syst. 33, 5933–5944 (2020).
78. M. Arbel, A. Matthews, A. Doucet, Annealed flow transport Monte Carlo. In: International Conference on Machine Learning, pp. 318–330 (2021)
79. T. Lippert, Cost of QCD simulations with n(f) = 2 dynamical Wilson fermions. Nucl. Phys. B Proc. Suppl. 106, 193–
107. M. Luscher, S. Schaefer, Non-renormalizability of the HMC algorithm. JHEP 04, 104 (2011). https://doi.org/10.1007/JHEP04(2011)104. arXiv:1103.1810 [hep-lat]

108. L. Del Debbio, G.M. Manca, E. Vicari, Critical slowing down of topological modes. Phys. Lett. B 594, 315–323 (2004). https://doi.org/10.1016/j.physletb.2004.05.038. arXiv:hep-lat/0403001

109. M. Hasenbusch, Speeding up the hybrid Monte Carlo algorithm for dynamical fermions. Phys. Lett. B 519, 177–182 (2001). https://doi.org/10.1016/S0370-2693(01)01102-9. arXiv:hep-lat/0107019

110. J. Brannick, R.C. Brower, M.A. Clark, J.C. Osborn, C. Rebbi, Adaptive multigrid algorithm for lattice QCD. Phys. Rev. Lett. 100, 041–601 (2008). https://doi.org/10.1103/PhysRevLett.100.041601. arXiv:0707.4018 [hep-lat]

111. R. Babich, J. Brannick, R.C. Brower, M.A. Clark, T.A. Manteuffel, S.F. McCormick, J.C. Osborn, C. Rebbi, Adaptive multigrid algorithm for the lattice Wilson–Dirac operator. Phys. Rev. Lett. 105, 201–602 (2010). https://doi.org/10.1103/PhysRevLett.105.201602. arXiv:1005.3043 [hep-lat]

112. S.L. Smith, P.J. Kindermans, C. Ying, Q.V. Le, Don’t decay the learning rate, increase the batch size (2017). arXiv:1711.00489

113. M.D. Zeiler, Adadelta: an adaptive learning rate method (2012). arXiv:1212.5701

114. K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: surpassing human-level performance on ImageNet classification (2015). arXiv:1502.01852 [cs.CV]

115. X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, pp. 249–256 (2010)

116. T. Brown, B. Mann, N. Ryder, M. Subbiah, J.D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell et al., Language models are few-shot learners. Adv. Neural. Inf. Process. Syst. 33, 1877–1901 (2020)

117. S. Smith, M. Patwary, B. Norick, P. LeGresley, S. Rajbhandari, J. Casper, Z. Liu, S. Prabhakaran, G. Zervas, V. Korthikanti, et al., Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model (2022). arXiv:2201.11990

118. A. Reuther, J. Kepner, C. Byun, S. Samsi, W. Arcand, D. Bestor, B. Bergeron, V. Gadeppally, M. Houle, M. Hubbell, et al., Interactive supercomputing on 40,000 cores for machine learning and data analysis, in 2018 IEEE High Performance extreme Computing Conference (HPEC), pp. 1–6 (2018). https://doi.org/10.1109/hpec.2018.8547629. arXiv:1807.07814 [cs.DC]

119. A. Paszke, et al., in Advances in Neural Information Processing Systems 32, ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’ Alché-Buc, E. Fox, R. Garnett (Curran Associates, Inc., 2019), pp. 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

120. J. Bradbury, R. Frostig, P. Hawkins, M.J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, Q. Zhang, JAX: composable transformations of Python+NumPy programs (2018). http://github.com/google/jax

121. T. Hennigan, T. Cai, T. Norman, I. Babuschkin, Haiku: Sonnet for JAX (2020). http://github.com/deepmind/dm-haiku

122. A. Sergeev, M. Del Balso, Horovod: fast and easy distributed deep learning in TensorFlow (2018). arXiv:1802.05799 [cs.LG]

123. C.R. Harris, K.J. Millman, S.J. Van Der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N.J. Smith et al., Array programming with numpy. Nature 585(7825), 357–362 (2020)

124. P. Virtanen, R. Gommers, T.E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright et al., SciPy 1.0: fundamental algorithms for scientific computing in python. Nat. Methods 17(3), 261–272 (2020)

125. J.D. Hunter, Matplotlib: a 2d graphics environment. Comput. Sci. Eng. 9(3), 90–95 (2007). https://doi.org/10.1109/MCSE.2007.55