PROCEEDINGS OF SCIENCE

# Machine Learning Trivializing Maps: A First Step Towards Understanding How Flow-Based Samplers Scale Up

**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, UK*

*E-mail:* Luigi.Del.Debbio@ed.ac.uk, Joe.Marsh-Rossney@ed.ac.uk

A trivializing map is a field transformation whose Jacobian determinant exactly cancels the interaction terms in the action, providing a representation of the theory in terms of a deterministic transformation of a distribution from which sampling is trivial. Recently, a proof-of-principle study by Albergo, Kanwar and Shanahan [Phys. Rev. D **100** 034515 (2019)] demonstrated that approximations of trivializing maps can be 'machine-learned' by a class of invertible, differentiable neural models called *normalizing flows*. By ensuring that the Jacobian determinant can be computed efficiently, asymptotically exact sampling from the theory of interest can be performed by drawing samples from a simple distribution and passing them through the network. From a theoretical perspective, this approach has the potential to become more efficient than traditional Markov Chain Monte Carlo sampling techniques, where autocorrelations severely diminish the sampling efficiency as one approaches the continuum limit. A major caveat is that it is not yet understood how the size of models and the cost of training them is expected to scale. As a first step, we have conducted an exploratory scaling study using two-dimensional $\phi^4$ with up to $20^2$ lattice sites. Although the scope of our study is limited to a particular model architecture and training algorithm, initial results paint an interesting picture in which training costs grow very quickly indeed. We describe a candidate explanation for the poor scaling, and outline our intentions to clarify the situation in future work.

*The 38th International Symposium on Lattice Field Theory, LATTICE2021 26th-30th July, 2021 Zoom/Gather@Massachusetts Institute of Technology*

[*] Speaker

## 1. Sampling and Critical Slowing Down in Lattice Field Theory

Numerical lattice field theory refers to a collection of strategies for approximating expectation values of the form

$$\langle O \rangle = \frac{1}{Z} \int \mathcal{D}\phi e^{-S(\phi)} O(\phi) \,, \qquad \mathcal{D}\phi = \prod_{x\in\Lambda} \mathrm{d}\phi_x \,, \tag{1}$$

by simulating field theories, defined on a space-time lattice $\Lambda \equiv a\mathbb{N}^d$ and described by an action $S(\phi)$, on a computer. By simultaneously reducing the lattice spacing $a$ and keeping the physical volume fixed, one can obtain increasingly accurate extrapolations to the continuum limit with (in principle) quantifiable systematic errors.

This 'simulation' is almost always a form of Markov Chain Monte Carlo (MCMC), whose function is to generate a representative sample of field configurations, i.e. one in which any configuration, $\phi$, may appear with a probability proportional to $e^{-S(\phi)}$. Given such a sample of $N_\phi$ configurations, $\{\phi\}$, an unbiased estimate of $\langle O \rangle$ is given by the sample mean,

$$\overline{O} = \frac{1}{N_\phi} \sum_{\{\phi\}} O(\phi) \,, \qquad \mathrm{SE}_{\overline{O}} = \sigma_O \sqrt{\frac{2\tau_O}{N_\phi}} \,. \tag{2}$$

The statistical error on $\overline{O}$, denoted $\mathrm{SE}_{\overline{O}}$ above, depends on the process through which the sample was generated through the *integrated autocorrelation time*,

$$\tau_O = \frac{1}{2} \sum_{t=-\infty}^{\infty} \frac{\Gamma_O(t)}{\Gamma_O(0)} \geq \frac{1}{2} \,, \tag{3}$$

which is defined for a particular observable in terms of correlations between evaluations of that observable separated by $t$ steps in a thermalised Markov chain, $\Gamma_O(t) = \left\langle O(\phi^{(t)}) O(\phi^{(0)}) \right\rangle - \langle O \rangle^2$ (note that $\Gamma_O(0) \equiv \sigma_O^2$). The integrated autocorrelation time is a measure of the statistical efficiency of an MCMC simulation; it takes approximately $2\tau_O$ steps for the simulation to generate an effectively independent evaluation of $O$. An algorithm is said to suffer from *critical slowing down* if $\tau_O$ varies in proportion to the correlation length of the system (in units of $a$), $\xi$, raised to some power, $z_O > 1$. Unfortunately critical slowing down remains for the most part an intractable encumbrance in lattice QCD, where one would ideally like to perform continuum limit extrapolations with physical quark masses, implying large correlation lengths.

Lüscher suggested a modification of the Hybrid Monte Carlo (HMC) algorithm [1] based on the idea of an invertible *trivializing map* from the theory of interest to a limit in which the field variables decouple [2]. This original semi-analytical approach has yet to prove advantageous in practice. However, the underlying idea — that one can in principle construct a representation of the theory in which non-trivial aspects are encoded into a deterministic transformation of a distribution from which sampling is easy — remains worthy of serious consideration.

Recently, this thread was picked up in a seminal study by Albergo, Kanwar and Shanahan [3], and followed up in Refs. [4–7]. The key idea is that field transformations can be performed using an invertible neural network with a large number of adjustable parameters. The task of constructing a trivializing map is cast as an optimisation problem; if the network is flexible enough, a careful tuning of the parameters (i.e. 'training') can yield a good approximation of a trivializing map.

## 2. A Short Introduction to Normalizing Flows for Scalar Field Theory

Consider the following action which describes a single-component scalar field with a quartic interaction term,

$$S(\phi) = \frac{1}{2} \sum_{x \in \Lambda} \left[ -\beta \sum_{\mu=1}^{d} \phi_x \phi_{x+\hat{\mu}} + \phi_x^2 + \lambda (\phi_x^2 - 1)^2 \right] . \tag{4}$$

Let $\mathcal{F} : \mathbb{R}^{|\Lambda|} \longrightarrow \mathbb{R}^{|\Lambda|}$ be a continuously differentiable bijective mapping whose effect is equivalent to taking $\beta, \lambda \longrightarrow 0$, such that the probability density factorises into a product of univariate Gaussians,[1]

$$p\left(\mathcal{F}(\phi)\right) = p(\phi) \left| \frac{\partial \mathcal{F}(\phi)}{\partial \phi} \right|^{-1} = \mathcal{N} \prod_{x \in \Lambda} \exp\left( -\frac{\mathcal{F}(\phi)_x^2}{2} \right) , \tag{5}$$

where $p(\phi) \equiv Z^{-1} e^{-S(\phi)}$ and $|\partial \mathcal{F}(\phi) / \partial \phi|$ denotes the Jacobian determinant. In this representation the degrees of freedom are decoupled Gaussian variables, which may be trivially sampled, while the correlated structure of $p(\phi)$ has been transferred to the Jacobian determinant of $\mathcal{F}$. For convenience we will use $z \equiv \mathcal{F}(\phi)$ to label field configurations in this limit.

Our goal is to use neural networks to parametrise a transformation that, given sufficient training, inverts the trivializing map.[2] Let this neural model, the *normalizing flow*, be denoted $f_\theta$ where $\theta$ labels its adjustable parameters. The training strategy proposed by Albergo, Kanwar and Shanahan [3] is to first sample from $p(z)$, pass these Gaussian variates through $f_\theta$, and then finally adjust the parameters of the networks such that the following 'loss function' decreases:

$$L_\theta(\{z\}) = \frac{1}{N_z} \sum_{\{z\}} \left( S(f_\theta(z)) - \log \left| \frac{\partial f_\theta(z)}{\partial z} \right| \right) . \tag{6}$$

Note that this involves computing the (logarithm of the) Jacobian determinant. One hopes that iterating this training step many times will converge to an optimal set of parameters,

$$\theta^\star = \arg\min_\theta L_\theta(\{z\}) \;\Rightarrow\; f_{\theta^\star} \approx \mathcal{F}^{-1} . \tag{7}$$

Why should we expect this to work? Minimising the Eq. (6) with respect to $\theta$ is equivalent to minimising a stochastic estimator of the *Kullback-Leibler divergence*,

$$D_{\mathrm{KL}}(\tilde{p} \| p) = \int \mathcal{D}\phi \tilde{p}(\phi) \log \frac{\tilde{p}(\phi)}{p(\phi)} , \tag{8}$$

which is a measure of distance[3] between $p(\phi)$ and the transformed product of Gaussians,

$$\tilde{p}(f_\theta(z)) = p(z) \left| \frac{\partial f_\theta(z)}{\partial z} \right|^{-1} . \tag{9}$$

Using Eq. 5, $D_{\mathrm{KL}}(\tilde{p} \| p) = 0$ precisely when the Jacobian determinants of $f_\theta$ and $\mathcal{F}$ cancel.

[1] In fact this definition may be overkill; the $\beta \longrightarrow 0$ limit is not necessary to trivialize the theory.

[2] If we consider $\mathcal{F}$ to be an 'encoder', we want to build the corresponding 'decoder' that restores the information (correlations) we are interested in.

[3] With a caveat being that it is asymmetric: $D_{\mathrm{KL}}(\tilde{p}; p) \neq D_{\mathrm{KL}}(p; \tilde{p})$.

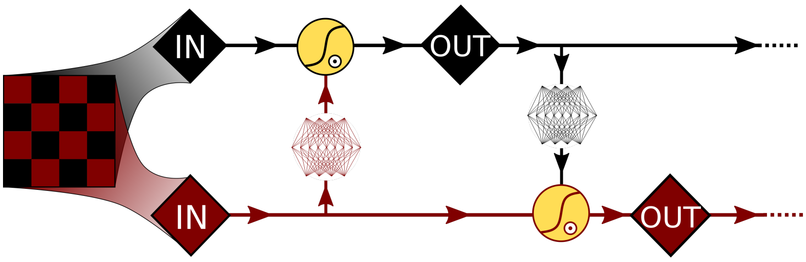


**Figure 1:** A visual representation of a pair of coupling layers using an even-odd ‘checkerboard’ partitioning of the lattice, as described by Eq. (10). Let the inputs (‘IN’) represent a field configuration, $v_i$. Then the outputs (‘OUT’) represent $g_{i+1} \circ g_i \circ v_i = v_{i+2}$, where each element of $v_i$ has been transformed (exactly once) by a transformation that is parametrised by a complicated function of the elements from the opposite colour on the checkerboard. These functions are what the neural networks are trained to approximate.

In general, $f_\theta$ must be a very flexible or ‘expressive’ in order to invert $\mathcal{F}$, which we can safely assume is highly non-linear in the interesting case of a strongly interacting theory. However, there is an important and opposing constraint which is that the Jacobian determinant must remain efficient to compute so that training does not become prohibitively expensive.

Though not the only approach, a popular compromise is to build the normalizing flow out of a sequence of transformations now widely referred to as *coupling layers* [8]. Coupling layers are essentially a template for building flexible, pointwise 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 non-trivial transformation that is parametrised ‘on the fly’ using information derived from the remaining, non-transformed inputs. Considering a theory with a single degree of freedom at each lattice site, this implies a partitioning of the lattice into two disjoint subsets, $\Lambda^A$ and $\Lambda^P$, which we refer to as the ‘active’ and ‘passive’ partitions, respectively. Motivated by locality, it is common to split the lattice into ‘even’ and ‘odd’ sites, and alternate between even-transforming layers and odd-transforming layers, as suggested in Figure 1.

Defining a depth-$D$ normalizing flow as the composition of $D$ coupling layers, $f_\theta \equiv g_D \circ g_{D-1} \circ \ldots \circ g_2 \circ g_1$, and writing $v_1 \equiv z$ and $v_{D+1} \equiv \phi$, one can express the action of the $i$-th coupling layer as $g_i : v_i \mapsto v_{i+1} = g_i(v_i)$, where

$$v_{i+1,x} = \begin{cases} v_{i,x}\,, & x \in \Lambda_i^P \\ C_{i,x}\left(v_{i,x}\,; \mathbf{N}_{i,x}(v_i^P; \theta_i)\right)\,, & x \in \Lambda_i^A\,. \end{cases} \tag{10}$$

In the above, $v_i^P$ is short-hand for $\{v_{i,x} \mid x \in \Lambda_i^P\}$. A set of invertible transformations $C_i$ (which are as yet unspecified) act on the active partition and depend on a set of parameters $\mathbf{N}_i(v_i^P; \theta_i)$ that are themselves functions of the passive variables. The choice of labelling is suggestive of the fact that these functions are to be modelled by neural networks, with $\theta_i$ denoting the weights and biases.

The payoff for building $f_\theta$ out of these curious layers is that the logarithm of the Jacobian determinant reduces to nothing more than a sum over the log-gradients of every individual transformation in the normalizing flow, which is potentially extremely efficient to compute.

$$\log \left| \frac{\partial f_\theta}{\partial z} \right| = \sum_{i=1}^{D} \sum_{x \in \Lambda_i^A} \log \left| \frac{\partial C_{i,x}}{\partial v_{i,x}} \right|\,. \tag{11}$$

In fact this is the same Jacobian factor that would arise were we to replace the neural networks with parameters that had no dependence on the passive partition. This is very nice because we can insert arbitrarily complex neural networks into the coupling layer at insignificant cost in the computation of $\log \tilde{p}(\phi)$.

So far, we have described how one might construct a model that is capable of generating samples of *independent* field configurations with probability proportional to $\tilde{p}(\phi)$, and how such a model may be trained so that $\tilde{p}(\phi) \approx p(\phi)$. In practice, however, the quality of the approximation will almost certainly be insufficient to justify neglecting the biases that would arise by taking expectation values over $\tilde{p}(\phi)$ rather than $p(\phi)$. Some form of reweighting is required to correct for these discrepancies. Two approaches that have been used to date are, firstly, to weight the configurations generated by the model by running a Metropolis-Hastings simulation by using the model outputs as proposals that are accepted with a probability $A(\phi \to \phi')$ according to the 'Metropolis test' [3],

$$A(\phi \to \phi') = \min\left(1, \frac{\tilde{p}(\phi)}{\tilde{p}(\phi')} \frac{p(\phi')}{p(\phi)}\right) , \tag{12}$$

and, secondly, to insert a reweighting factor $w(\phi) = p(\phi)/\tilde{p}(\phi)$ directly into Eq. (2) [9]. Both of these approaches result in asymptotically exact sampling from $p(\phi)$, but crucially they rely on our ability to compute $\tilde{p}(\phi)$ exactly (up to a normalization) and efficiently. In general, although one can construct training schemes that do not rely on an exact log-probability density for the model, it is essential for guaranteeing correct sampling.

The fraction of generated configurations that are accepted by the Metropolis test is a convenient metric for comparing models.[4] After verifying a monotonic correspondence with the integrated autocorrelation time, we generally used the acceptance fraction (or just 'acceptance') in place of an estimate of $\tau_O$ to quantify the sampling efficiency of trained models.

## 3. Experiments and Scaling Study

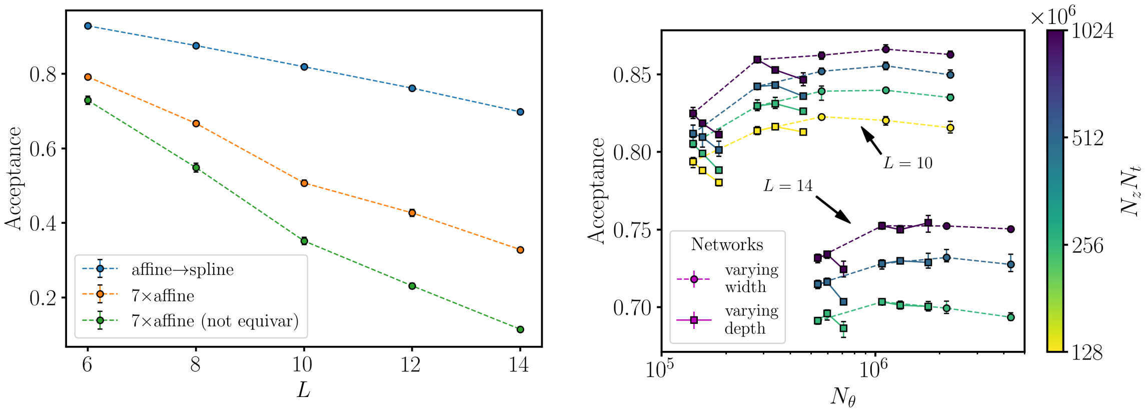
We have been experimenting with various recipes for building normalizing flows out of coupling layers, primarily with the aim of sampling from the action in Eq. (4). For greater detail on the transformations used, the experimental setup, and additional observations, please refer to Ref. [10].

The original proof-of-principle study [3] used normalizing flows built out of *affine* coupling layers [8] parametrised by deep fully-connected neural networks. As we were experimenting, we noticed that three key modifications led to substantial improvements in acceptances and autocorrelations (see also Figure 2a):

1. Upgrade the final pair of affine layers to more flexible *rational quadratic spline* layers [11].
2. Enforce *equivariance* with respect to the $\mathbb{Z}_2$ symmetry in the preceding affine layers.
3. Replace the deep neural networks with shallow ones — a single hidden layer suffices.

By computing the two-point correlation function, and hence estimating the correlation length, at various stages in the transformation (i.e. on the intermediate states, $v_i$), we concluded that the

[4] However, the acceptance rate is poor at diagnosing the issue of 'mode-collapse': see Ref. [10] Sec.VII/C and also Ref. [7] Sec.VII/C.



**(a)** Comparison between three groups of models, each using approximately the same number of parameters in total; one pair of spline layers contains approximately the same number as six pairs of affine layers, given that the networks have one hidden layer of a common size. In the legend, 'affine' and 'spline' refer to blocks of two coupling layers that together transform all of the field variables once.

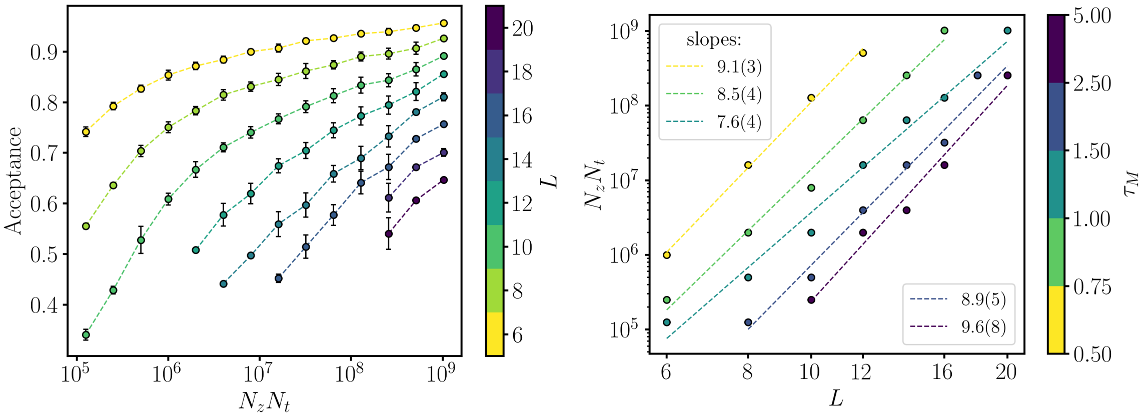
**(b)** Acceptance rates resulting from varying the width (follow dotted lines) and depth (follow filled lines) of the neural networks within coupling layers. The $x$-axis represents the number of trainable parameters, $N_\theta$, in the entire flow, and the colour axis measures the extent of the training, $N_t$ being the number of steps involving batches of $N_z$ training examples.

**Figure 2**

initial affine layers mainly resolved strong correlations at short separations, whereas the final pair of spline layers split the unimodal distribution into a bimodal one (with modes at $\pm\langle\phi\rangle$), and resolved weaker correlations. We also found it beneficial to forcefully anneal the learning rate from an initial maximum value down to zero at the end of training, allowing progressively weaker correlations to be learned via increasingly delicate optimisation steps. Like Ref. [7], we found that annealing the temperature, $\beta^{-1}$, during training also slightly improved results, particularly in the strongly bimodal phase of $\phi^4$. Presumably, both of these strategies result in 'learning' being more uniformly distributed among the training iterations.

The key motivation behind the flow-based approach is that that once a normalizing flow model has been trained sampling is extremely efficient. Given what appeared to be significant improvements in acceptance rates with respect to the proof-of-principle study [3], we were interested to determine how the cost of training these models could be expected to scale towards the continuum limit, since this 'overhead' cost is ultimately the crucial factor that will determine whether this approach is more efficient in practice.

Before concentrating on the scaling we took some time to establish which hyper-parameters had the greatest influence over the quality of the trained model. The key conclusions from these tests, exemplified by Figure 2b, were that increasing the size of models (bigger neural networks, more layers, etc.) had a small and sometimes adverse effect on the sampling efficiency of the trained model, whereas acceptances were strongly dependent on two training hyper-parameters: the batch size ($N_z$ from earlier) and the number of training iterations, $N_t$. We also did a post-hoc check that our results were not unduly poor due to the use of fully-connected neural networks (**N**) in the coupling layers rather than convolutional neural networks. In fact what we observed was that models using fully-connected networks reached higher acceptance rates than those using convolutional networks,



**(a)** Average acceptance rates from fully-trained models, grouped by the number $N_z N_t$ of training examples (field configurations) from which they were able to learn. Data points and error bars are means and standard deviations taken over multiple, independently trained models, with different ratios $N_t / N_z = 1, 2, 4$, and different numbers of affine layers.

**(b)** Scaling of 'training costs', defined as the number $N_z N_t$ of training examples that were required to be generated. Models were grouped by integrated autocorrelation time, and the model with the lowest $N_z N_t$ selected from each group. This particular architecture appears to be very far from being competitive with traditional local-updates MCMC, where costs typically scale as $\sim L^{d+2}$ ($z_O \approx 2$).

**Figure 3**

with far less time spent training. There are, however, several reasons to expect this hierarchy to invert as lattices and flow models increase in size, and we intend to investigate this in a future study.

We conducted a first investigation into the scaling; we restricted ourselves to effectively one, carefully chosen architecture involving up to five pairs of affine coupling layers followed by a final pair of spline coupling layers, each parametrised by a depth-two fully-connected neural network with $L^2$ neurons in the hidden layer. We fixed the correlation length at $\xi \approx L/4$ and increased the lattice size from $L^2 = 6^2, 8^2, \ldots, 20^2$. The largest models had $N_\theta < 10^7$ trainable parameters, and the most extensive training schedule involved $N_t = 32,000$ training updates, each involving a batch of $N_z = 32,000$ configurations.

More so than $N_z$ and $N_t$ individually, results were sensitive to the product $N_z N_t$, i.e. the number of 'training examples' seen by the model. Surprisingly, we did not observe any plateauing of acceptances (on a log scale) as $N_z N_t$ increased over 3 orders of magnitude, even for the smallest lattice size — see Figure 3a. This suggests that the limiting factor is not the inherent capacity of the network to approximate a trivializing map (i.e. expressivity) but begs the question: what is the origin of this slow convergence, and of the astonishingly poor scaling of training costs depicted in Figure 3b?

A first, uncontroversial interpretation of slow convergence might be that the latter stages of optimisation require a large number (large $N_t$) of very precise (large $N_z$) steps. However, this statement does not have any explanatory power. Nor does it provide any insight into why the error bars in Figure 3a are so small; i.e. why the acceptance rate depends strongly on the product $N_z N_t$, but far less so on $N_z$ or $N_t$ individually, or on the number of layers/parameters in the model.

Drawing on observations made in Ref. [12], we suggest a candidate explanation, that is, after an initial training phase where the acceptance grows from effectively zero to the order of $10^{-1}$ rather quickly, the rate at which *useful* training examples are generated is extremely low. By 'useful' we mean that the training example produces a contribution to the training update (a gradient) that causes the density $\tilde{p}(\phi)$ to expand along a specific direction in which it is currently underestimating $p(\phi)$. To justify this we begin by stating two well-established facts. Firstly, the typical behaviour of a training algorithm based on a loss function derived from Eq. (8) is to quickly pick out the mode(s) of $p(\phi)$.[5] Secondly, as we increase the number of degrees of freedom in the system or increase the correlation length, $p(\phi)$ becomes increasingly concentrated on a low-dimensional manifold embedded in $\mathbb{R}^{L^2}$. Recalling that $\tilde{p}(\phi)$ is an isotropic Gaussian at $t = 0$, this suggests an initial, fast 'mode-seeking' phase, after which $\tilde{p}(\phi)$ is likely to overestimate $p(\phi)$ in almost every direction. Since training examples are generated with a probability proportional to $\tilde{p}(\phi)$, it follows that a large proportion of a training update will be devoted to attempting to compress $\tilde{p}(\phi)$ ever closer to the nearest mode of $p(\phi)$, but many of these gradients will oppose each other because total probability mass is conserved. Consequently, the optimisation becomes heavily dependent on those relatively few training examples which produce gradients that guide the expansion of $\tilde{p}(\phi)$ along the manifold, away from the nearest mode. Assuming this expansion phase is slow, the total number of these useful configurations that is generated during training grows in proportion with $N_z N_t$, and is of course oblivious to any efforts to improve results by changing the model itself.

In more general terms, given a highly correlated $p(\phi)$, there are theoretical grounds to expect a training scheme based on minimising Eq. (6) to encounter this very inefficient expansion phase [12]. However, while this provides a strong motivation to consider this as a potential contributing factor to the terrible scaling of our models, we wish to be clear that this is a *candidate* explanation that needs verification. For example, it is possible that this issue would become severe on large lattices, but is actually relatively mild on the small lattices we studied. What is required is a far more systematic investigation aimed at disentangling the various factors contributing to the scaling of training costs. As a next step, it would be instructive to separately quantify the effect of increasing the number of degrees of freedom and increasing the correlation length on training costs. Once the picture has become clearer, if our description of a slow expansion phase turns out to be accurate, we can begin to try to alleviate the problem by, for example, modifying the training scheme.

## 4. Conclusions

We conducted a short study of the scaling of training costs based on an improved formulation of the normalizing flow models presented in Ref. [3]. Our results show that the particular combination of model architecture and training scheme used in several studies including our own [3, 9, 10] does not scale up in a way that would make it competitive with traditional MCMC sampling techniques. It would be very unwise, however, to extrapolate from these conclusions; it is not yet clear what the root cause of the rapidly increasing training costs is, therefore neither is it clear whether this is a problem that can be alleviated or circumvented. Our intention is to scale up this study to much larger lattices and systematically quantify the factors influencing the scaling of training costs by varying the model architecture (including using convolutional networks), training algorithm, and hyper-parameters. We also aim to incorporate other lattice field theories, such as the topologically non-trivial $\mathrm{CP}^{N-1}$ models, into this study.

[5] With the major caveat that it is possible to completely miss a subset of the modes.

## References

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

[2] M. Lüscher, *Trivializing maps, the Wilson flow and the HMC algorithm*, *Commun Math Phys* **293** (2009) [0907.5491].

[3] 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** (2019) 034515 [1904.12072].

[4] G. Kanwar, M.S. Albergo, D. Boyda, K. Cranmer, D.C. Hackett, S. Racanière et al., *Equivariant flow-based sampling for lattice gauge theory*, *Phys. Rev. Lett.* **125** (2020) 121601 [2003.06413].

[5] D. Boyda, G. Kanwar, Racanière, D.J. Rezende, M.S. Albergo, K. Cranmer et al., *Sampling using* $\mathrm{SU}(N)$ *gauge equivariant flows*, *Phys. Rev. D* **103** (2021) 074504 [2008.05456].

[6] M.S. Albergo, G. Kanwar, S. Racanière, D.J. Rezende, J.M. Urban, D. Boyda et al., *Flow-based sampling for fermionic lattice field theories*, 2106.05934.

[7] D.C. Hackett, C.-C. Hsieh, M.S. Albergo, D. Boyda, J.-W. Chen, K.-F. Chen et al., *Flow-based sampling for multimodal distributions in lattice field theory*, 2107.00734.

[8] L. Dinh, J. Sohl-Dickstein and S. Bengio, *Density estimation using Real NVP*, 1605.08803.

[9] K.A. Nicoli, C.J. Anders, L. Funcke, T. Hartung, K. Jansen, P. Kessel et al., *On estimation of thermodynamic observables in lattice field theories with deep generative models*, *Phys. Rev. Lett.* **126** (2021) 032001 [2007.07115].

[10] L. Del Debbio, J. Marsh Rossney and M. Wilson, *Efficient modelling of trivializing maps for lattice* $\phi^4$ *theory using normalizing flows: A first look at scalability*, *Phys. Rev. D* **104** (2021) 094507 [2105.12481].

[11] C. Durkan, A. Bekasov, I. Murray and G. Papamakarios, *Neural spline flows*, *Advances in Neural Information Processing Systems* **32** (2019) [1906.04032].

[12] C.-W. Huang, F. Ahmed, K. Kumar, A. Lacoste and A. Courville, *Probability distillation: A caveat and alternatives*, *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference* **115** (2020) 1212.

## Acknowledgments

LDD is supported by an STFC Consolidated Grant, ST/P0000630/1, and a Royal Society Wolfson Research Merit Award, WM140078. JMR is supported by STFC, grant ST/T506060/1. MW is supported by STFC, grant ST/R504737/1. This work has made use of the resources provided by the Edinburgh Compute and Data Facility (ECDF).