NeuLat: a toolbox for neural samplers in lattice field theories

Kim A. Nicoli,[a,b,c,d,*] **Christopher J. Anders,**[d,e] **Lena Funcke,**[a,b,c] **Karl Jansen,**[f] **Shinichi Nakajima**[d,e,g] **and Pan Kessel**[h]

[a] *Transdisciplinary Research Area "Building Blocks of Matter and Fundamental Interactions" (TRA Matter), University of Bonn, Germany*
[b] *Helmholtz Institute for Radiation and Nuclear Physics (HISKP), University of Bonn, Nussallee 14-16, 53115 Bonn, Germany*
[c] *Bethe Center for Theoretical Physics (BCTP), University of Bonn, Nussallee 12, 53115 Bonn, Germany*
[d] *Berlin Institute for the Foundation of Learning and Data (BIFOLD), Technische Universität Berlin, Berlin, Germany*
[e] *Technische Universität Berlin, Machine Learning Group, Marchstrasse 23, Berlin,10587, Germany*
[f] *Deutsches Elektronen-Synchrotron DESY, Platanenallee 6, 15738 Zeuthen, Germany*
[g] *RIKEN Center for AIP, 1-4-1 Nihonbashi, Chuo-ku, Tokyo, Japan*
[h] *Prescient Design, gRED, Roche, Basel, Switzerland*

E-mail: knicoli@uni-bonn.de, anders@tu-berlin.de, lfuncke@uni-bonn.de, karl.jansen@desy.de, nakajima@tu-berlin.de, pan.kessel@gmail.com

The application of normalizing flows for sampling in lattice field theory has garnered considerable attention in recent years. Despite the growing community at the intersection of machine learning (ML) and lattice field theory, there is currently a lack of a software package that facilitates efficient software development for new ideas in this field. In these proceedings, we present NeuLat, a fully customizable software package that unifies recent advances in the fast-growing field of deep generative models for lattice field theory in a single software library. NeuLat is designed to be modular, supports a variety of lattice field theories as well as normalizing flow architectures, and is easily extensible. We believe that NeuLat has the potential to considerably simplify the application and benchmarking of ML methods for lattice quantum field theories and beyond.

The 40th International Symposium on Lattice Field Theory, LATTICE2023
31th July – 1st August 2023
Fermilab, Batavia, Illinois, USA

*Speaker

1. Introduction

In recent years, there has been increasing interest in utilizing normalizing flows for sampling in lattice field theory (see e.g. Refs. [1–26] and Ref. [27] for a review). Despite the rapid progress, there is currently no software package that streamlines efficient software development for novel concepts in this domain.

In these proceedings, we introduce NeuLat, a fully adaptable software package consolidating recent advances in deep generative models for lattice field theory into a unified software library. The paper is structured as follows: first, we review the required theoretical background in Sec. 2 and summarize the advantages of using generative models, in particular normalizing flows, for sampling lattice field configurations. Second, in Sec. 3, we explain the idea behind NeuLat and comment on related work, the software structure, and the key features of the package. Lastly, in Sec. 4, we conclude the paper highlighting the aim of NeuLat to become a community-wide effort.

2. Theoretical Background

Normalizing flows are a widely used type of generative models, which were first introduced at the beginning of the last decade [28, 29] and started to gain in popularity few years later [30, 31]. Towards the end of the last decade, Ref. [32] introduced normalizing flows for sampling in quantum chemistry and called them *Boltzmann generators* for their capability of sampling from unnormalized Boltzmann densities. Concurrently, related works using autoregressive neural networks for sampling statistical mechanical systems also appeared [33, 34]. A few months later, Ref. [1] used normalizing flows to sample configurations of scalar field theories and thereby paved the way for the development of Boltzmann generators in lattice field theory. In the following years, numerous follow-up studies have been conducted with applications in scalar [1–16], pure gauge [17–22] and fermionic [23, 24] field theories and beyond [25]. Notably, the number of works in the field started to substantially increase upon the release of a jupyter notebook [26], where the code from three important papers [1, 17, 35] was partly released. This release had a great impact on research and education, as many researchers have benefited from reusing part of the code to develop and boost novel research ideas. This phenomenon represents the main motivation for NeuLat, the first Python library which implements flow-based samplers for different types of lattice quantum field theories.

2.1 Sampling Lattice Configurations Using Generative Models

In lattice field theory, one aims to estimate physical observables, e.g., the spectrum, the topological susceptibility, the free energy, the magnetization, and so forth. This is achieved using the path integral formalism, where the expectation value of an observable is computed with respect to a distribution of the Gibbs-type, $p(\phi) = e^{-S(\phi)}/Z$. Here, ϕ denotes the lattice fields, such as gluon, quark, or scalar fields, S represents the action of the system, and Z is the normalization constant, i.e., the partition function. The expectation value for a generic observable O reads

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

where $\mathcal{D}[\phi] = \prod_{i=0}^{|\Lambda|} \mathrm{d}\phi_i$ is the measure of the high-dimensional path integral and $|\Lambda|$ denotes the number of lattice points of the lattice Λ. Exactly solving the path integral (1) is generically intractable; thus, one has to resort to numerical methods such as Hamiltonian Monte Carlo (HMC) [36]. Despite being a powerful method, HMC faces many drawbacks, such as critical slowing down and sequential sampling. Moreover, estimating thermodynamic observables, such as the free energy, is challenging with HMC, because it involves the generation of Markov chains for each point along a (disrectized) trajectory through parameter space [2].

In light of this, sampling via deep generative models, such as normalizing flows, holds the promise to provide a new alternative route for addressing these challenges in lattice field theory. The advantages of flow-based sampling over standard HMC sampling can be summarized as follows:

- **Non-correlated sampling**: Samples can be drawn independently and identically distributed (i.i.d.) from the trained generative density, without the need of running sequential algorithms which suffer from strongly correlated samples.

- **Embarrassingly parallelizable**: Once a flow model is trained, its neural network parameters θ can be loaded onto different devices, and the sampling can be embarrassingly parallelized. This is crucial as it allows to drastically increase the numbers of samples which can be drawn simultaneously. The training can also be efficiently parallelized over multiple GPUs.

- **Partition function estimation**: Generative models such as normalizing flows, see e.g., the reviews [37, 38], or autoregressive neural networks [39] give access to the *normalized* sampling density $q_\theta(\phi)$ of the flow model. In Refs. [2, 34], the authors have shown that this can be used to directly estimate thermodynamic observables, such as the entropy and the free energy, which cannot be directly computed with standard HMC approaches.

- **Inductive biases**: Prior knowledge of some properties of the physical system, such as its symmetries, can be encoded into the generative model in order to eliminate unnecessary degrees of freedom for efficient training and sampling.

Despite their promise, we note that flow models currently struggle to scale to larger volumes, such as those pertinent to high-precision Lattice QCD calculations. So far, successful simulations have been limited to toy problems in limited volumes. Consequently, the research field is actively evolving, and the development of a robust software package is crucial for advancing further.

2.2 Normalizing Flows for Lattice Field Theory

When using normalizing flows [40] for estimating lattice field theory observables, one first has to train a flow and then use it to obtain a Monte Carlo estimate of Eq. (1). In the following, we briefly summarize the mathematical concepts behind this flow method. For the sake of brevity, we omit a detailed discussion of normalizing flows and refer the reader to the review papers [37, 38].

The core part of a normalizing flow is a parametric bijective map between a base space $\mathcal{Z}$ and a target space Φ. Formally, one can define

$$f_\theta : \mathcal{Z} \to \Phi, \tag{2}$$

where $\mathcal{Z}$ and Φ are equipped with probability densities q_Z and q_θ, respectively, with q_Z being the base density, which is often a Gaussian or uniform. Leveraging the properties of normalizing flows, one can obtain the likelihood of the transformed base density as

$$q_\theta(\phi) = q_z(f_\theta^{-1}(\phi)) \left| \frac{\mathrm{d}f_\theta^{-1}(\phi)}{\mathrm{d}\phi} \right| = q_z(\mathbf{z}) \left| \frac{\mathrm{d}f_\theta}{\mathrm{d}\mathbf{z}} \right|^{-1} . \tag{3}$$

For the rest of the paper, we assume that q_θ represents our variational density, which is trained to match the target density $p(\phi) = \exp\{-S[\phi]\}/Z$ of our lattice field theory.

For training a normalizing flow, one often minimizes the Kullback-Leibler (KL) divergence [41], which is a type of similarity measure between two densities. In our case, these are the target density $p(\phi)$ and the variational density $q_\theta(\phi)$. The KL divergence reads

$$\mathrm{KL}(q_\theta||p) = \int \mathcal{D}[\phi]\, q_\theta(\phi) \ln \frac{q_\theta(\phi)}{p(\phi)} . \tag{4}$$

This expression is known as the *reverse*-KL and can be understood as the expectation value with respect to the variational density q_θ of the log-ratio of the two densities. From its definition (4), it follows that KL ≥ 0, and the equality holds if and only if $q_\theta = p$. As shown in several works (e.g., Refs. [1, 2, 32]), this objective is particularly suited for learning a target density of Boltzmann type which is known in closed-form, up to the normalization constant Z. Specifically, training can be performed by *self-training*, as samples are drawn from the model q_θ that is being optimized during training. While being very effective, the reverse-KL is not the ideal objective when the target density is multimodal [3, 15], because of its mode-seeking property. A valid alternative[1] is the *forward*-KL

$$\mathrm{KL}(p||q_\theta) = \int \mathcal{D}[\phi]\, p(\phi) \ln \frac{p(\phi)}{q_\theta(\phi)} . \tag{5}$$

In this case, the similarity measure between the two densities can be expressed as the expectation value of the inverse log-ratio with samples drawn from the target density p. In other words, the forward-KL requires samples from the target (Boltzmann) density $\phi \sim p$ for training. The forward-KL is often used for training normalizing flows in other ML applications where training samples, e.g., images, audio, or text, are available in large amounts. However, for our application in lattice field theory, one would need to collect such training samples by using, e.g., HMC [15].

Once the model is trained and the flow q_θ becomes a good approximation of the target density p, one can use samples drawn from the flow for computing observables via Eq. (1). However, additional care is required. Indeed, $q_\theta = p$ never holds in practice, and one needs to take this into account when sampling. To this end, two approaches can be taken:

- **Neural Monte Carlo Markov Chain (NMCMC) (or Metropolization)** uses the flow q_θ as the proposal density $q_\theta(\phi|\phi') = q_\theta(\phi)$ for a Markov Chain. The samples are then accepted or rejected with the Metropolis-Hastings accept-reject probability [1, 34].
- **Neural Importance Sampling (NIS)** provides a Monte Carlo estimator for a given observable by sampling from q_θ and using a reweighting technique. The importance weights $w(\phi) =$

[1]We note that the forward- and reverse-KL divergences are not symmetric, hence $\mathrm{KL}(p||q) \neq \mathrm{KL}(q||p)$.

$p(\phi)/q_\theta(\phi)$ are used to reweight the Monte Carlo samples, thus making the estimator asymptotically unbiased [2, 34, 42] under certain conditions [15].

Using Neural MCMC or NIS allows to correct for an imperfect approximation of the target p with q_θ or, more precisely, to obtain statistically consistent estimates of expectation values. When the variational density closely approximates the target, those approaches allow to avoid important drawbacks of HMC, such as critical slowing down.

3. NeuLat: Neural Samplers for Lattice Field Theories

In the previous section, we provided our view on normalizing flows as a promising new approach for sampling lattice field configurations and for ultimately estimating physical observables. However, it is nontrivial to implement and train such models. Most of the material discussed in Sec. 2 relied on the implementation of architectures which were inherited from prior works within the same groups. Moreover, even if the implementation of such architectures is feasible, it can still take weeks or months to fine-tune the hyperparameters to reproduce results of previously published papers, which substantially slows down the research progress. As a consequence, it is often challenging for early practitioners with little background in software development for machine learning to enter this interdisciplinary field when lacking fundamental reference code. Indeed, the number of publications within this domain started to increase dramatically since the partial code release [26] from the seminal papers [1, 17]. Because of this, we decided to develop NeuLat, a software package for neural samplers in lattice field theory. We foresee this software framework to serve as reference code for everyone interested in the development of flow-based models for custom physical systems, or for researchers interested in taking their first steps with ML for lattice field theory, building on various techniques described in many recent works.

3.1 Related Work

A few first attempts have been made to develop software packages for the flow-based simulation of lattice field theories. The closest attempt in a similar direction of NeuLat is a package called GomalizingFlow.jl [43]. Written in Julia with GPU acceleration support, this software package mainly focuses on scalar field theories. In contrast, NeuLat is implemented in Python. While Julia is recently enjoying an up-rise in popularity in the field of ML, it does not feature the vast amount of ML frameworks implemented in Python. On the other hand, the Jupyter Notebook released with Ref. [26] is implemented in Python and provides examples for training flow-based models for both scalar and gauge field theories. While it represents a great source of inspiration for NeuLat, it is not structured as a software framework, and can thus not easily be built upon or extended. Another library combining flow-based sampling and HMC has been developed in Ref. [44]. While this implementation provides a well-designed and extendable software framework, it is optimized for a different physical problem: using normalizing flows within the HMC Leapfrog integration. Lastly, another software package written in JAX is called FlowMC [45]. This framework mainly implements the techniques proposed in Refs. [46, 47] and is not specifically tailored to sampling in lattice field theory, as opposed to NeuLat. Therefore, to the best of our knowledge, while many standalone repositories and few partially related software packages exist, NeuLat is the first attempt

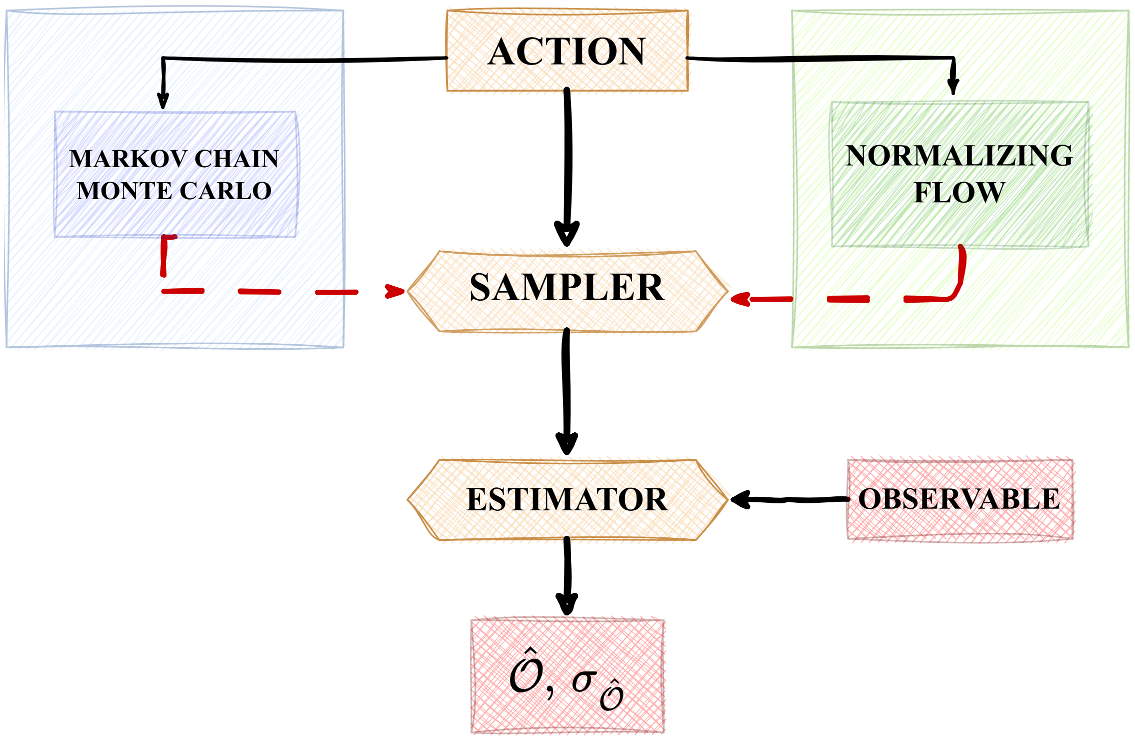

Figure 1: High-level structure of NeuLat. The orange blocks represent the core of the workflow: the action, the sampler, and the estimator. The action, defined by the user, is fed into a sampler, which can be a normalizing flow (green) or a Markov Chain Monte Carlo (blue) sampler. The estimator uses the sampler to draw configurations for estimating expectation values of physical observables (red).

to combine five years of research results in the field of flow-based sampling for lattice field theories into one single software package that is flexible, easy to extend, and intuitively customizable.

3.2 The Software in a Nutshell

The main structure of NeuLat is sketched in Fig. 1, and a code example is provided in Fig. 2. Within its basic workflow, NeuLat expects two main inputs from the user: the physical theory to sample from and the type of normalizing flow. The former is specified by the action[2] and some associated parameters, e.g., the hopping parameter κ and the bare coupling λ as shown in line 2 of Fig. 2. The normalizing flow is specified using the class `Flow` and requires a set of hyper-parameters, such as the base density and the type of coupling layers. For the implementation of the normalizing flows, NeuLat inherits its structure from the nflows package [48]. In this way, many types of normalizing flows are readily available, and one can naturally extend them by implementing new architectures following the same structure. Once the action has been instantiated, NeuLat offers two possibilities: performing standard HMC sampling (see the blue box on the left-hand-side of Fig. 1 corresponding to line 4 in Fig. 2) or training a normalizing flow (see the green box in Fig. 1 and line 5 to 17 in Fig. 2). The former allows the user to obtain an HMC baseline to compare

[2]We note that the example code uses the same parametrization for the action of the ϕ^4 theory as in Ref. [2].

```
 1 n_samples, train_steps, batch = 1000, 1000, 500
 2 action = Phi4Action(kappa=0.3, lamb=0.022)
 3
 4 hmc_chain = HMCMarkovChain(action, lat_shape=[16, 16], burn_in=100)
 5 model = Flow(
 6     lat_shape=[16, 16],
 7     base_dist=NormalDistribution(),
 8     coupling=NiceCoupling(),
 9     n_couplings=6
10 )
11
12 optim = torch.optim.Adam(model.parameters(), lr=learning_rate)
13 loss_fn = ReverseKLLoss()
14 for _ in range(train_steps):
15     configs, log_probs = model.sample_and_log_prob(batch)
16     loss = loss_fn(action(configs), log_probs)
17     optim.zero_grad(); loss.backward(); optim.step()
18
19 obs = (Magnetization(), AbsMagnetization(), action)
20
21 flow_obs = IidEst(obs).evaluate(model.sample(n_samples))
22 hmc_obs = CorrelatedEst(obs).evaluate(hmc_chain.sample(n_samples))
```

Figure 2: Code example showing how NeuLat works in practice. Once the action object is instantiated, this code can be used for a) sampling using HMC (line 4) or b) training a normalizing flow (lines 5-17). Samples can be drawn from the respective models for an estimation of observables of interest (lines 19-22). **Note that this structure may be subject to change**.

against the flow results. Once a flow is trained, one can draw samples and estimate observables using Neural MCMC or NIS. Both approaches are provided in NeuLat within the `Sampler` class. For estimating physical observables, NeuLat implements different estimators, e.g., `IidEst()` and `CorrelatedEst()`. Any observables can be instantiated (see line 19) and passed individually or as a collection of multiple observables (here tuple) into an appropriate estimator. Corresponding values of the observables are computed by calling the `evaluate` method of the respective estimator on a collection of samples, either drawn from the normalizing flow or via standard HMC sampling. One crucial aspect of Monte Carlo simulations is the accurate assessment of statistical and systematic errors. Specifically, the `CorrelatedEst` estimator accounts for correlated samples (see line 22). This estimator is internally implemented using the unew [49] library, a Python package which implements the Γ-method [50].

NeuLat is currently under active development and a first release will soon be available under https://github.com/neulat/neulat. We remark that a preliminary version of NeuLat was used to produce the results published in previous works [2, 4, 14, 15, 51].

3.3 Key Features

In the following, we would like to summarize the key features of NeuLat and point out several benefits of using this software framework instead of individual re-implementations. The indisputable advantage of well-designed software frameworks is their considerable simplification of implementations that require a common basis. Their common goal is to provide an intuitive solution to swiftly implement existing methods, and easily extend and customize. For NeuLat specifically, this means that a user can effortlessly integrate a new custom coupling layer or an entirely new type of normalizing flow, as well as a custom action, without re-implementing already well-established ground work. For instance, sampling from new theories which are not part of NeuLat only requires the implementation of a custom action, which in extreme cases might be possible in only three lines of code. Moreover, to boost the whole field of research on specific new theories or types of normalizing flows, the custom implementations may be shared with the community through a simple pull request[3], initiating the permanent incorporation in NeuLat. In this way, the project aims to become a central implementation hub of common actions and flows in the field, serving as a starting point for future implementations and benchmarking.

To summarize its capabilities, NeuLat can be utilized to accomplish various tasks, including but not limited to:

- Density estimation, e.g., learning target densities as discussed in Sec. 2.2.
- Sampling using HMC [52], Neural MCMC [1, 53] and NIS [1, 2].
- Asymptotically unbiased estimation of general physical observables [34] and estimation of partition functions and thermodynamic observables [4].
- Assessing the severity of mode collapse [15], with which the bias can be bounded.

We note that only some of these features will be part of the first software release, and many additional features will be added in future releases, or by the community with the help of pull requests.

Furthermore, NeuLat is planned to contain many additional recent ML tools proposed in several different works, thus combining and simplifying access to software which is currently split into many independent code bases. At the current state, these works include: stochastic normalizing flows [6], conditional normalizing flows [5, 8], path gradients [14, 51], mode-dropping estimators [15], and many more in the future. We believe that the presence of a unified framework would be an extremely beneficial contribution to the community. Similar efforts within the context of ML for quantum chemistry, including Refs. [54, 55], proved to be highly valuable resources for the research community, used by many different groups. Furthermore, employing a common software framework shared across multiple research groups can significantly streamline reproducibility for new studies, thus simplifying the comparison of newly developed methods with baselines, thus further accelerating the research field.

[3]Following the provided guidelines will ensure smooth integration of new contributions.

4. Conclusion

These proceedings introduced NeuLat, a Python-based software package for flow-based sampling of lattice field theories. The software will be designed to be accessible, modular, and easy to extend and maintain. In this way, NeuLat aims to eliminate the overhead of re-implementing existing code between different formats and languages. Most of the relevant tools for flow-based simulations in lattice field theories, will be or can be implemented within this software package. The first release of the software will soon be available at the GitHub page https://github.com/neulat/neulat.

NeuLat is aimed to be a community-wide effort. Therefore, we strongly encourage researchers to contribute to its growth by integrating their own ideas and architectures, reporting potential bugs, and making suggestions for improvements. We also encourage readers to reach out to us if they are interested in contributing to the development of NeuLat. We foresee NeuLat as a highly valuable contribution to the research field and to anyone wishing to learn about ML-enhanced sampling techniques. We believe that a modular and resilient software package will significantly help in successfully addressing and eventually overcoming the many challenges this field has to face.

Acknowledgements

C.A. and S.N. are funded by the German Ministry for Education and Research as BIFOLD - Berlin Institute for the Foundations of Learning and Data (ref. BIFOLD23B). The authors thank Niccolò Luboz for helping design the logo of the software package.

References

[1] 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.

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

[3] 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*, *ArXiv e-prints* (2021) [2107.00734].

[4] K. Nicoli, C. Anders, L. Funcke, T. Hartung, K. Jansen, P. Kessel et al., *Machine Learning of Thermodynamic Observables in the Presence of Mode Collapse*, *PoS* **LATTICE2021** (2022) 338.

[5] M. Gerdes, P. de Haan, C. Rainone, R. Bondesan and M.C. Cheng, *Learning lattice quantum field theories with equivariant continuous flows*, *ArXiv e-prints* (2022) [2207.00283].

[6] M. Caselle, E. Cellini, A. Nada and M. Panero, *Stochastic normalizing flows as non-equilibrium transformations*, *JHEP* **2022** (2022) 15.

[7] M. Caselle, E. Cellini, A. Nada and M. Panero, *Stochastic normalizing flows for lattice field theory*, *PoS* **LATTICE2022** (2022) 005.

[8] A. Singha, D. Chakrabarti and V. Arora, *Conditional normalizing flow for markov chain monte carlo sampling in the critical region of lattice field theory*, *Phys. Rev. D* **107** (2023) 014512.

[9] D. Albandea, L. Del Debbio, P. Hernandez, R. Kenway, J.M. Rossney and A. Ramos Martinez, *Learning trivializing flows*, *PoS* **LATTICE2022** (2023) 001.

[10] J.M. Pawlowski and J.M. Urban, *Flow-based density of states for complex actions*, *Phys. Rev. D* **108** (2023) 054511.

[11] L. Del Debbio, J.M. Rossney and M. Wilson, *Machine Learning Trivializing Maps: A First Step Towards Understanding How Flow-Based Samplers Scale Up*, *PoS* **LATTICE2021** (2022) 059.

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

[13] P. de Haan, C. Rainone, M. Cheng and R. Bondesan, *Scaling up machine learning for quantum field theory with equivariant continuous flows*, *ArXiv e-prints* (2021) [2110.02673].

[14] L. Vaitl, K.A. Nicoli, S. Nakajima and P. Kessel, *Path-gradient estimators for continuous normalizing flows*, in *Proceedings of the 39th International Conference on Machine Learning*, vol. 162, pp. 21945–21959, PMLR, 2022, https://proceedings.mlr.press/v162/vaitl22a.html.

[15] K.A. Nicoli, C.J. Anders, T. Hartung, K. Jansen, P. Kessel and S. Nakajima, *Detecting and mitigating mode-collapse for flow-based sampling of lattice field theories*, *Phys. Rev. D* **108** (2023) 114501.

[16] A. Faraz, A. Singha, D. Chakrabarti and V. Arora, *Scalable Lattice Sampling using Factorized Generative Models*, *ArXiv e-prints* (2023) [2308.08615].

[17] 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.

[18] J. Finkenrath, *Tackling critical slowing down using global correction steps with equivariant flows: the case of the schwinger model*, *ArXiv e-prints* (2022) [2201.02216].

[19] D. Boyda, G. Kanwar, S. 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.

[20] M. Favoni, A. Ipp, D.I. Müller and D. Schuh, *Lattice gauge equivariant convolutional neural networks*, *Phys. Rev. Lett.* **128** (2022) 032003.

[21] S. Bacchio, P. Kessel, S. Schaefer and L. Vaitl, *Learning trivializing gradient flows for lattice gauge theories*, *Phys. Rev. D* **107** (2023) L051504.

[22] A. Singha, D. Chakrabarti and V. Arora, *Sampling U(1) gauge theory using a retrainable conditional flow-based model*, *Phys. Rev. D* **108** (2023) 074518 [2306.00581].

[23] 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*, *Phys. Rev. D* **104** (2021) 114507.

[24] R. Abbott, M.S. Albergo, D. Boyda, K. Cranmer, D.C. Hackett, G. Kanwar et al., *Gauge-equivariant flow models for sampling in lattice field theories with pseudofermions*, *Phys. Rev. D* **106** (2022) 074506.

[25] M. Caselle, E. Cellini and A. Nada, *Sampling the lattice nambu-goto string using continuous normalizing flows*, *ArXiv e-prints* (2023) [2307.01107].

[26] M.S. Albergo, D. Boyda, D.C. Hackett, G. Kanwar, K. Cranmer, S. Racaniere et al., *Introduction to normalizing flows for lattice field theory*, *ArXiv e-prints* (2021) [2101.08176].

[27] K. Cranmer, G. Kanwar, S. Racanière, D.J. Rezende and P.E. Shanahan, *Advances in machine-learning-based sampling motivated by lattice quantum chromodynamics*, *Nature Reviews Physics* **5** (2023) 526.

[28] E.G. Tabak and C.V. Turner, *A family of nonparametric density estimation algorithms*, *Communications on Pure and Applied Mathematics* **66** (2013) 145.

[29] E.G. Tabak and E. Vanden-Eijnden, *Density estimation by dual ascent of the log-likelihood*, *Communications in Mathematical Sciences* **8** (2010) 217 .

[30] D. Rezende and S. Mohamed, *Variational inference with normalizing flows*, in *Proceedings of the 32nd International Conference on Machine Learning*, vol. 37, pp. 1530–1538, PMLR, 2015, https://proceedings.mlr.press/v37/rezende15.html.

[31] L. Dinh, D. Krueger and Y. Bengio, *NICE: non-linear independent components estimation*, in *3rd International Conference on Learning Representations (Workshop Track Proceedings)*, 2015 [1410.8516].

[32] F. Noé, S. Olsson, J. Köhler and H. Wu, *Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning*, *Science* **365** (2019) eaaw1147.

[33] D. Wu, L. Wang and P. Zhang, *Solving statistical mechanics using variational autoregressive networks*, *Phys. Rev. Lett.* **122** (2019) 080602.

[34] K.A. Nicoli, S. Nakajima, N. Strodthoff, W. Samek, K.-R. Müller and P. Kessel, *Asymptotically unbiased estimation of physical observables with neural samplers*, *Phys. Rev. E* **101** (2020) 023304.

[35] D.J. Rezende, G. Papamakarios, S. Racaniere, M. Albergo, G. Kanwar, P. Shanahan et al., *Normalizing flows on tori and spheres*, in *Proceedings of the 37th International Conference on Machine Learning*, H.D. III and A. Singh, eds., vol. 119 of *Proceedings of Machine Learning Research*, pp. 8083–8092, PMLR, 13–18 Jul, 2020, https://proceedings.mlr.press/v119/rezende20a.html.

[36] C. Gattringer and C. Lang, *Quantum chromodynamics on the lattice: an introductory presentation*, vol. 788, Springer Science (2010), 10.1007/978-3-642-01850-3.

[37] G. Papamakarios, E. Nalisnick, D.J. Rezende, S. Mohamed and B. Lakshminarayanan, *Normalizing flows for probabilistic modeling and inference*, *J. Mach. Learn. Res.* **22** (2021) .

[38] I. Kobyzev, S.J. Prince and M.A. Brubaker, *Normalizing flows: An introduction and review of current methods*, *IEEE transactions on pattern analysis and machine intelligence* **43** (2020) 3964.

[39] A. van den Oord, N. Kalchbrenner, L. Espeholt, k. kavukcuoglu, O. Vinyals and A. Graves, *Conditional image generation with pixelcnn decoders*, in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon and R. Garnett, eds., vol. 29, Curran Associates, Inc., 2016.

[40] D.J. Rezende, S. Mohamed and D. Wierstra, *Stochastic backpropagation and approximate inference in deep generative models*, in *Proceedings of the 31st International Conference on Machine Learning*, vol. 32, pp. 1278–1286, PMLR, 2014, https://proceedings.mlr.press/v32/rezende14.html.

[41] S. Kullback and R.A. Leibler, *On Information and Sufficiency*, *The Annals of Mathematical Statistics* **22** (1951) 79 .

[42] T. Müller, B. McWilliams, F. Rousselle, M. Gross and J. Novák, *Neural importance sampling*, *ACM Trans. Graph.* **38** (2019) 145:1.

[43] A. Tomiya and S. Terasaki, *Gomalizingflow. jl: A julia package for flow-based sampling algorithm for lattice field theory*, *ArXiv e-prints* (2022) [2208.08903].

[44] S. Foreman, T. Izubuchi, L. Jin, X.-Y. Jin, J.C. Osborn and A. Tomiya, *HMC with Normalizing Flows*, *ArXiv e-prints* (2021) [2112.01586].

[45] K. Wong and M. Gabrié, "Flowmc: Normalizing-flow enhanced sampling package for probabilistic inference." `https://github.com/kazewong/flowMC`, 2022.

[46] M. Gabrié, G.M. Rotskoff and E. Vanden-Eijnden, *Efficient bayesian sampling using normalizing flows to assist markov chain monte carlo methods*, in *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*, 2021, https://openreview.net/forum?id=mvtooHbjOwx.

[47] M. Gabrié, G.M. Rotskoff and E. Vanden-Eijnden, *Adaptive monte carlo augmented with normalizing flows*, *Proceedings of the National Academy of Sciences* **119** (2022) e2109420119 [`https://www.pnas.org/doi/pdf/10.1073/pnas.2109420119`].

[48] C. Durkan, A. Bekasov, I. Murray and G. Papamakarios, *nflows: normalizing flows in PyTorch*, Nov., 2020. 10.5281/zenodo.4296287.

[49] B. De Palma, M. Erba, L. Mantovani and N. Mosco, *A python program for the implementation of the γ-method for monte carlo simulations*, Computer Physics Communications **234** (2019) 294.

[50] U. Wolff, *Erratum to "monte carlo errors with less errors" [comput. phys. comm. 156 (2004) 143–153]*, Computer Physics Communications **176** (2007) 383.

[51] L. Vaitl, K.A. Nicoli, S. Nakajima and P. Kessel, *Gradients should stay on path: Better estimators of the reverse- and forward kl divergence for normalizing flows*, Mach. Learn.: Sci. Tech. **3** (2022) 045006.

[52] M. Betancourt, *A conceptual introduction to hamiltonian monte carlo*, *ArXiv e-prints* (2017) [1701.02434].

[53] K. Nicoli, P. Kessel, N. Strodthoff, W. Samek, K.-R. Müller and S. Nakajima, *Comment on "solving statistical mechanics using vans": Introducing savant-vans enhanced by importance and mcmc sampling*, *ArXiv e-prints* (2019) [1903.11048].

[54] K.T. Schütt, P. Kessel, M. Gastegger, K.A. Nicoli, A. Tkatchenko and K.-R. Müller, *Schnetpack: A deep learning toolbox for atomistic systems*, Journal of Chemical Theory and Computation **15** (2019) 448.

[55] F. Noé, S. Olsson, J. Köhler and H. Wu, "Bgflow." https://github.com/noegroup/bgflow, 2022.