PROCEEDINGS OF SCIENCE

Progress in Normalizing Flows for 4d Gauge Theories

Ryan Abbott,[a,b,*] **Denis Boyda,**[a,b] **Daniel C. Hackett,**[c,a,b] **Gurtej Kanwar,**[d,e] **Fernando Romero-López,**[e,a,b] **Phiala E. Shanahan**[a,b] **and Julian M. Urban**[a,b]

[a] *Center for Theoretical Physics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA*
[b] *The NSF AI Institute for Artificial Intelligence and Fundamental Interactions*
[c] *Fermi National Accelerator Laboratory, Batavia, IL 60510, U.S.A.*
[d] *Higgs Centre for Theoretical Physics, University of Edinburgh, Edinburgh EH9 3FD, UK*
[e] *Albert Einstein Center, Institute for Theoretical Physics, University of Bern, 3012 Bern, Switzerland*

Normalizing flows have arisen as a tool to accelerate Monte Carlo sampling for lattice field theories. This work reviews recent progress in applying normalizing flows to 4-dimensional nonabelian gauge theories, focusing on two advancements: an architectural improvement referred to as learned active loops, and the application of correlated ensemble methods to QCD with $N_f = 2$ dynamical fermions.

The 41st International Symposium on Lattice Field Theory (LATTICE2024)
28 July - 3 August 2024
Liverpool, UK

[*] Speaker

PoS(LATTICE2024)066

1. Introduction

Monte Carlo importance sampling forms the basis of all lattice field theory calculations. Many practical developments over the past decades—Hybrid Monte Carlo, multigrid preconditioners, and efficient numerical codes—have enabled precision studies of dynamical lattice gauge theories, including lattice QCD. Despite these advances, critical slowing down and topological freezing limit the lattice spacings accessible in state-of-the-art calculations, which has lead to recent interest in new algorithms, such as those provided by machine learning. One such approach has been the application of normalizing flows to lattice field theory, which has been demonstrated to eliminate critical slowing down in certain lattice field theories [1–3] and, for example, may provide near-term practical advantages through correlated sampling to access specific observables [4, 5]. Normalizing flow methods have also been extended to Monte Carlo sampling for dynamical lattice QCD at a coarse lattice spacing [6]. These promising early results have motivated recent efforts to improve the quality of models and find new practical applications of flows.

The present work is focused on two recent developments in applying normalizing flows to 4d gauge theories, particularly QCD. The first is an architectural improvement called "learned active loops". This method replaces what has typically been a static choice in previous models with a learned component, expanding the ability for each layer in the flow to modify the physical degrees of freedom and substantively improving model quality. The second development is the application of the correlated ensemble methods of Ref. [4] to a system with dynamical fermions, specifically to calculate the gluon momentum fraction of the pion in $N_f = 2$ QCD with twisted mass fermions at $M_\pi \simeq 540$ MeV. This is the first application of this method to a theory with dynamical fermions. For this particular study, a computational advantage of the approach that incorporates flows is shown, even when training costs are taken into account.

2. Learned Active Loops

This work introduces learned active loops as an architectural improvement for the coupling-based spectral flow architectures described in Ref. [7]. Coupling-based flows are constructed by stacking several layers, each of which transforms some subset $\{(x, \mu)\}_A$ of the gauge links $U_\mu(x)$, referred to as the active links. Transformations of these active links are defined by first constructing untraced Wilson loops containing each active link, then transforming these loops, then "pushing back" the transformation onto the active links. The constructed loops are referred to as *active loops*. Spectral flows in particular use an eigenvalue decomposition of the active loops to isolate and act on gauge-invariant information [2, 7, 8].

Previous flow architectures have typically used a fixed active loop geometry per layer, often using a particular orientation of a small loop, such as an untraced plaquette or 2×1 loop, alternating the direction across successive layers. The central idea behind learned active loops is to replace this fixed choice with a learned component, adding additional parameters that specify a linear combination of loops to transform in each layer. This is a strict generalization of the previous approach, and expands the degrees of freedom acted on by each layer of the flow. Intuitively, the benefit of this approach is that it allows the transformation to more directly access and correlate

both small and large active loops in an individual layer, which combinatorially increases the space of transformations.

The implementation of learned active loops in this work has two components: first, the frozen degrees of freedom are passed through a "staple network" to produce an "active staple", denoted by $S_\mu^{\text{act}}(x)$. Here a staple $S_\mu(x)$ refers to any set of complex matrices derived from gauge links that transforms like the gauge link $U_\mu(x)$ with the same (x, μ) under gauge transformations (notably a staple is *not* required to be an element of $\mathrm{SU}(N)$). Once the active staple has been computed, the active loop is then defined by

$$L_\mu(x) = \mathcal{P}_{\mathrm{SU}(N)}(U_\mu(x) S_\mu^{\text{act}}(x)^\dagger) \tag{1}$$

where $\mathcal{P}_{\mathrm{SU}(N)}$ denotes a conjugation equivariant projection from general complex matrices onto $\mathrm{SU}(N)$. Here, conjugation equivariant means that the projection must satisfy

$$\mathcal{P}_{\mathrm{SU}(N)}(gMg^\dagger) = g\mathcal{P}_{\mathrm{SU}(N)}(M)g^\dagger \tag{2}$$

for any $g \in \mathrm{SU}(N)$.

In the architectures of this work, each coupling layer contains its own independent staple network as a subcomponent. Each such staple network is constructed out of a series of N_ℓ layers, which successively transforms the input staples as

$$S_{\mu k}^{(0)} \to S_{\mu k}^{(1)} \to \cdots \to S_{\mu k}^{(N_\ell)}. \tag{3}$$

The index k is an additional channel index that ranges from 1 to $N_h^{(i)}$. All staple networks used in this work fix $N_h^{(0)} = N_h^{(N_\ell)} = 1$ and $N_h^{(i)} = N_h$; the constant N_h is referred to as the width of the network. The layers involved in this work are all linear transformations on the space of all staples, though any gauge-equivariant architecture could be used here. For example, the L-CNN architectures from Ref. [9] could be explored as an alternative architecture for the staple network.

The inputs to the network are initialized by setting the staples equal to the frozen links, zeroing out any active links by setting

$$S_{\mu 0}^{(0)}(x) = (1 - m_\mu(x))U_\mu(x), \tag{4}$$

where $m_\mu(x)$ denotes the active mask, defined by

$$m_\mu(x) = \begin{cases} 1 & U_\mu(x) \text{ is active} \\ 0 & U_\mu(x) \text{ is frozen.} \end{cases} \tag{5}$$

Schematically, each successive layer acts by parallel transporting same-direction staples from neighboring sites and then taking a linear combination. In detail, to apply successive layers in the staple network, first left and right staples are constructed out of the input staples via

$$\begin{aligned} S_{\mu\nu k}^{(i)L} &= S_\nu^{(i)\dagger}(x + \hat{\mu} - \hat{\nu}) S_\mu^{(i)\dagger}(x - \hat{\nu}) S_\nu^{(i)}(x - \hat{\nu}), \\ S_{\mu\nu k}^{(i)R} &= S_\nu^{(i)}(x + \hat{\mu}) S_\mu^{(i)\dagger}(x + \hat{\nu}) S_\nu^{(i)\dagger}(x). \end{aligned} \tag{6}$$

These new staples are then concatenated and flattened along with the original staples $S_{\mu k}^{(i)}$, forming an augmented vector $S_{\mu k}^{(i),\text{aug}}$—i.e., the ν and k indices of the three original objects $S_{\mu\nu k}^{(i)}$ and $S_{\mu\nu k}^{(i)L/R}$

are subsumed into the k index of $S_{\mu k}^{(i),\text{aug}}$. Finally, the different elements k of the augmented vector are linearly combined to form the outputs of the layer via

$$S_{\mu k}^{(i+1)}(x) = \sum_{\ell} c_{k\ell}^{(i)} S_{\mu \ell}^{(i),\text{aug}}(x), \tag{7}$$

where $c_{k\ell}^{(i)}$ are learned coefficients. After iterating through the layers of the staple net, the active staple is taken to be the sole element of the final set of staples:

$$S_{\mu}^{\text{act}}(x) = S_{\mu 0}^{(N_\ell)}(x). \tag{8}$$

After the active staple is chosen, what remains is to define the conjugation-equivariant projection $\mathcal{P}_{\text{SU}(N)}$ used in Eq. (1). In order to satisfy Eq. (2), the flow models in this work utilize a two-step projection. First the matrix is projected onto $U(N)$ via polar projection, defined by

$$\mathcal{P}_{U(N)}(M) = (M^\dagger M)^{-1/2} M. \tag{9}$$

In practice this projection can be implemented by performing a singular value decomposition $M = U\Sigma V^\dagger$, and returning $\mathcal{P}_{U(N)}(M) = UV^\dagger$, which is equivalent to, but more numerically stable than, Eq. (9). Once the polar projection is computed, the matrix can be projected onto $\text{SU}(N)$ by dividing out the determinant:

$$\mathcal{P}_{\text{SU}(N)}(M) = \frac{\mathcal{P}_{U(N)}(M)}{(\det[\mathcal{P}_{U(N)}(M)])^{1/3}}. \tag{10}$$

It is straightforward to check via Eqs. (9) and (10) that this projection satisfies Eq. (2), and hence is conjugation-equivariant.

To test the effectiveness of learned active loops, we perform a simple benchmark using two models, with and without learned active loops. Both models are spectral flow models constructed similarly to the models of Ref. [7], combining direction and location updates, repeated a total of 3 times for $3 \times (48 + 16) = 192$ layers. The only difference between the two models is that one model uses learned active loops in its direction updates while the other uses a fixed active loop, namely the untraced plaquette. Both models are trained using the Adam optimizer [10] with path gradients [11]. For the model with learned active loops, the staple networks are constructed using 2 hidden layers with $N_h = 8$, and the coefficients of the staple network are initialized using the PyTorch `xavier_normal_` initialization scheme [12] with a gain of 0.5. In addition, the weights $c_{k\ell}^{(i)}$ of the staple network are initialized near the identity, meaning that the weights are set so that initialization with a gain of 0 would result in each layer of the staple network producing the identity function.

The results of training both models to target a pure gauge theory with a plaquette action at coupling $\beta = 2$ on a 4^4 lattice is shown in Fig. 1. The model quality here is demonstrated by the effective sample size (ESS), defined by [13, 14]

$$\text{ESS} = \frac{\mathbb{E}_{U\sim q(U)}[w(U)]^2}{\mathbb{E}_{U\sim q(U)}[w(U)^2]}, \tag{11}$$

where $w(U) = e^{-S(U)}/q(U)$ is the (unnormalized) reweighting factor, $S(U)$ is the target action defining the target density $p(U) \propto e^{-S(U)}$, and $q(U)$ is the model density. The ESS can be thought

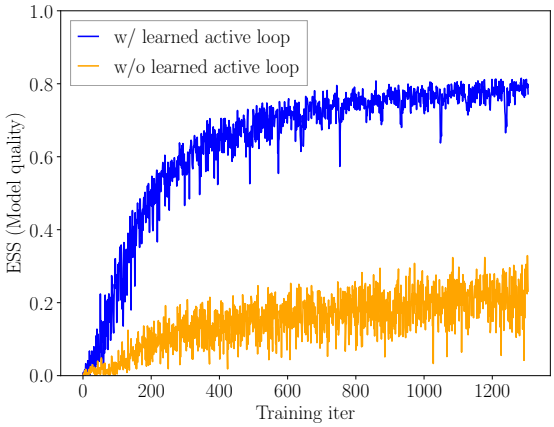

Figure 1: Example training curves with and without learned active loops. The models are otherwise identical, and the target theory is a pure-gauge plaquette action with $\beta = 2$ on a 4^4 lattice, optimized as described in the text.

of as the effective number of independent samples per model sample, with an ESS of 1 indicating a perfect model. The model that uses learned active loops shows a clear advantage over the model without, achieving almost 80% ESS in comparison to 20%. Similar results have also been seen across a wide variety of different tests, indicating that learned active loops appear to be a strict improvement over previous spectral flow models.

3. Correlated ensembles for hadron structure in $N_f = 2$ QCD

One promising application of flow models is as a map between two nearby distributions with a small difference in action parameters, $\Delta\lambda$. Such an application of flow models, extensively discussed in Ref. [4], can be used to compute derivatives with respect to action parameters:

$$\frac{d\langle O\rangle}{d\lambda} \simeq \frac{\langle O\rangle_{\lambda_1} - \langle O\rangle_{\lambda_2}}{\lambda_1 - \lambda_2}, \tag{12}$$

where the right-hand side is a finite-difference approximation of the derivative, and $\langle O\rangle_\lambda$ indicates the expectation value in the theory with action parameter λ. Using a flow model, this derivative can be computed as

$$\frac{d\langle O\rangle}{d\lambda} \simeq \Big\langle O(U) - w(f(U))\, O(f(U))\Big\rangle_{\lambda_1}, \tag{13}$$

where $f(U)$ is the flow acting on the gauge fields, and $w(f(U)) = p_{\lambda_2}(f(U))/q(f(U))$ is the corresponding reweighting factor for the flow from λ_1 to λ_2. The same derivatives can be computed also by other means, such as ϵ-reweighting, or using independent ensembles generated at λ_1 and λ_2. The methods that incorporate flows were shown in Ref. [4] to offer reduced uncertainties at fixed number of configurations, due to the possibility of using larger $\Delta\lambda$ while benefiting from correlated cancellations.

A particularly promising application of this idea is to the computation of hadronic matrix elements using Feynman-Hellmann techniques. The main idea is to add a term to the QCD action:

$$S = S_{\text{QCD}} + \lambda O, \tag{14}$$

where O is some operator of interest. Then, the matrix element of that operator between hadronic states can be computed as a derivative of the hadron mass, M_h:

$$\langle h|O|h\rangle = \frac{1}{2M_h}\frac{dM_h}{d\lambda}\bigg|_{\lambda\to 0}. \tag{15}$$

One example is an operator related to the gluon momentum fraction of the pion:

$$O = -\frac{\beta}{N_c}\,\mathrm{Tr}\,\mathrm{Re}\left(\sum_i U_{i0} - \sum_{i<j} U_{ij}\right). \tag{16}$$

This operator can be added as an asymmetry between the coupling of spatial and temporal plaquettes, U_{ij} and U_{i0}, respectively. The gluon momentum fraction can then be computed as

$$\frac{dM_h}{d\lambda}\bigg|_{\lambda\to 0} = -\frac{3M_h}{2}\langle x\rangle_g^{\mathrm{latt}} \simeq \frac{M_h(\lambda) - M_h(0)}{\lambda}. \tag{17}$$

In Ref. [4], we presented a proof-of-concept application of this approach in quenched QCD with a small lattice volume and quark masses corresponding to a larger-than-physical pion mass, and observed a >10× reduction of variance with a fixed number of configurations. Given this promising result, the next step is to perform the same calculation in a more physical setting with dynamical fermions, a larger lattice volume, and at lower pion mass. Below, we present our update on that front. In particular, we use $N_f = 2$ twisted mass fermions using the Lüscher-Weisz gauge action. Specifically, the action parameters and geometry are

$$\kappa = 0.164111, \quad \beta = 3.8, \quad \mu = 0.012, \quad L^3 \times T = 12^3 \times 24. \tag{18}$$

This follows the same setup as the "A" $N_f = 2$ ensembles by the ETMC collaboration, see Ref. [15]. The pion mass is $aM_\pi = 0.276(2)$. Using the lattice spacing $a = 0.1$ fm quoted in Ref. [15], this corresponds to $M_\pi \simeq 540$ MeV. The volume is $M_\pi L \sim 3.31$. We use 10k configurations generated with Chroma [16]. For the Feynman-Hellmann calculation, we use $\lambda = 0.005$.

Since this is dynamical QCD, the flow approach has to be adapted to the presence of fermions. For this, we utilize the pseudofermion approach of Ref. [17]. Specifically, we have a marginal model and a conditional model, which are trained separately. The marginal model acts on the gauge fields, while the conditional model is used to reduce the variance in the stochastic estimator of the fermion determinant ratio. For the marginal model, we use the same architecture presented in Ref. [4]. For the conditional model we follow the approach of Ref. [17], with a significant modification: since both the base and target distributions are nontrivial, we allow for parallel transports with the flowed and unflowed fields. Further details will be presented in an upcoming publication.

The marginal model is trained using the action with an explicit fermion determinant at a small volume, $V = 4^4$. It has 48 layers alternating the M2, M4, M2 and M4 masking pattern described in Ref. [4]. It achieves 99.7% ESS, which is an increase over the 94% ESS achieved with an untrained identity flow. The conditional model has 16 layers with an alternating masking pattern and is also trained at $V = 4^4$. It uses even-odd preconditioning, and it achieves a 99.6% ESS, higher than the baseline 99.4% ESS when using the identity pseudofermion flow. Both models have been trained with path gradients. At the evaluation volume, $12^3 \times 24$, the ESS of the marginal

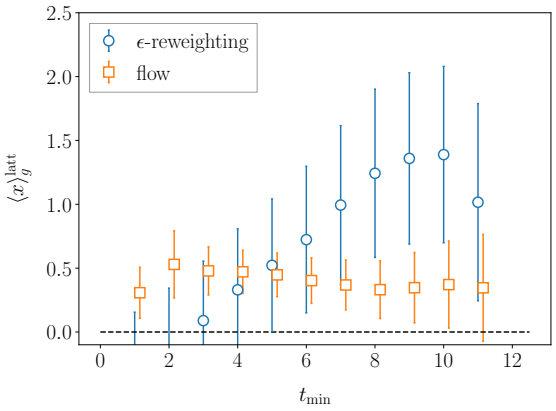

Figure 2: Gluon momentum fraction of the pion computed using the Feynman-Hellmann approach as function of the upper end of the fit range, t_{min}. Orange squares correspond to using flows, while blue circles are the baseline of ϵ reweighting.

model with no pseudofermion flow and four pseudofermion noise shots is 53%. Including the trained conditional model, it raises to 58%. Without a trained marginal model, i.e. simply using unimproved reweighting, the ESS would be $< 10^{-3}$.

The results are shown in Fig. 2. Specifically, we show the extracted gluon momentum fractions of the pion extracted from the fit results for the pion mass [1] as a function of the fit range, defined as $[t_{\text{min}}, T/2]$. We compare the flow approach to ϵ reweighting. We find that statistical errors are 2-3 times smaller when using flows. This means that for a fixed target statistical uncertainty, 5-10 times fewer configurations are needed.

A final comment about computational costs is due, made with the caveat that the numbers necessarily depend on the implementation and hardware. We present here the costs of this particular demonstration on A100 NVIDIA GPUs. A breakdown of the costs for each approach is:

$$\begin{aligned} \text{COST}(\epsilon\text{−rew}) &= \text{generation} + \text{measurement}, \\ \text{COST}(\text{flows}) &= \text{generation} + \text{training} + \text{flow eval.} + 2 \times \text{measurement} + \text{reweighting}. \end{aligned} \tag{19}$$

The difference for the flow approach is thus that 1) flows need to be trained and evaluated, 2) the reweighting factor involves fermions in the flow case and thus requires inversions to stochastically estimate the determinant ratio, and 3) an additional measurement is needed on the flowed configurations. We find that both measurement and generation costs per configuration are 0.17 GPU · hour. In contrast flow evaluation costs barely accounts for 0.01 GPU · hour, while the reweighting cost per flowed configuration is 0.04 GPU · hour. Thus, per configuration, the flow approach costs around 2× times more. As for training, this model was trained for 8 days on 16 A100s, and thus 4608 GPU · hour. Given the $5 - 10\times$ reduction of variance demonstrated in Fig. 2, a computational advantage is achieved after 4k configurations if training costs are taken into account. Note that this demonstration used 10k, and thus the results are well into the regime of computational advantage when flows are used.

[1] 16 stochastic sources at random timeslices per configuration are used.

4. Conclusion

Normalizing flows are a promising avenue for accelerating lattice field theory calculations. This work has reviewed two recent advances in applying normalizing flows to 4d gauge theories. The first, learned active loops, provides a substantive improvement to spectral flow models and shows that there are still many possibilities for improving model performance. In the future this work could be extended, for instance by increasing the expressivity of the staple networks using L-CNNs [9], or by finding other methods to increase the ability of the flow to modify different degrees of freedom. The second advancement presented here is the application of correlated ensemble methods to a theory with dynamical fermions, yielding a true computational advantage in a scenario not far from practical applicability. Future directions include applying these ideas to larger volumes and lower pion masses, approaching the scale of modern calculations. Separately, the method may be applied for other operators to obtain e.g. quark mass derivatives and sigma terms. These two advances are among many improvements in the application of normalizing flows to lattice field theory, bringing this method closer to practical applicability at the state of the art.

Acknowledgements

We thank Michael Albergo, Aleksandar Botev, Kyle Cranmer, Jacob Finkenrath, Alexander G. D. G. Matthews, Sébastien Racanière, Ali Razavi, and Danilo J. Rezende for useful discussions and valuable contributions to the early stages of this work. 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 U.S. DOE Early Career Award DE-SC0021006, by a NEC research award, and by the Carl G and Shirley Sontheimer Research Fund. FRL acknowledges support by the Mauricio and Carlota Botton Fellowship. RA was also partially supported by the High Energy Physics Computing Traineeship for Lattice Gauge Theory (DE-SC0024053). GK was supported by the Swiss National Science Foundation (SNSF) under grant 200020_200424. This manuscript has been authored by Fermi Forward Discovery Group, LLC under Contract No. 89243024CSC000002 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics. This work is supported 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/) and 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 [18] for providing HPC resources that have contributed to the research results reported within this paper. Numerical experiments and data analysis used PyTorch [19], JAX [20], Haiku [21], Horovod [22], NumPy [23], and SciPy [24]. Figures were produced using matplotlib [25].

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 [1904.12072].

[2] 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].

[3] M.S. Albergo, D. Boyda, K. Cranmer, D.C. Hackett, G. Kanwar, S. Racanière et al., *Flow-based sampling in the lattice Schwinger model at criticality*, 2202.11712.

[4] R. Abbott, A. Botev, D. Boyda, D.C. Hackett, G. Kanwar, S. Racanière et al., *Applications of flow models to the generation of correlated lattice QCD ensembles*, *Phys. Rev. D* **109** (2024) 094514 [2401.10874].

[5] S. Bacchio, *A novel approach for computing gradients of physical observables*, 2305.07932.

[6] R. Abbott, M.S. Albergo, A. Botev, D. Boyda, K. Cranmer, D.C. Hackett et al., *Sampling QCD field configurations with gauge-equivariant flow models*, in *39th International Symposium on Lattice Field Theory*, 8, 2022 [2208.03832].

[7] R. Abbott, M.S. Albergo, A. Botev, D. Boyda, K. Cranmer, D.C. Hackett et al., *Normalizing flows for lattice gauge theory in arbitrary space-time dimension*, 2305.02402.

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

[9] M. Favoni, A. Ipp, D.I. Müller and D. Schuh, *Lattice Gauge Symmetry in Neural Networks*, *PoS* **LATTICE2021** (2022) 185 [2111.04389].

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

[11] 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*, *Machine Learning: Science and Technology* **3** (2022) 045006 [2207.08219].

[12] X. Glorot and Y. Bengio, *Understanding the difficulty of training deep feedforward neural networks*, in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, JMLR Workshop and Conference Proceedings, 2010.

[13] A. Doucet, N. De Freitas, N.J. Gordon et al., *Sequential Monte Carlo methods in practice*, vol. 1, Springer (2001).

[14] J.S. Liu and J.S. Liu, *Monte Carlo strategies in scientific computing*, vol. 10, Springer (2001).

[15] European Twisted Mass collaboration, *Lattice QCD with two light Wilson quarks and maximally twisted mass*, *PoS* **LATTICE2007** (2007) 022 [0710.1517].

[16] SciDAC, LHPC, UKQCD collaboration, *The Chroma software system for lattice QCD*, *Nucl. Phys. B Proc. Suppl.* **140** (2005) 832 [hep-lat/0409003].

[17] 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 [2207.08945].

[18] A. Reuther, J. Kepner, C. Byun, S. Samsi, W. Arcand, D. Bestor et al., *Interactive supercomputing on 40,000 cores for machine learning and data analysis*, *2018 IEEE High Performance extreme Computing Conference (HPEC)* (2018) 1 [1807.07814].

[19] J. Ansel, E. Yang, H. He, N. Gimelshein, A. Jain, M. Voznesensky et al., *PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation*, in *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*, ACM, Apr., 2024, DOI.

[20] J. Bradbury, R. Frostig, P. Hawkins, M.J. Johnson, C. Leary, D. Maclaurin et al., *JAX: composable transformations of Python+NumPy programs*, 2018.

[21] T. Hennigan, T. Cai, T. Norman and I. Babuschkin, *Haiku: Sonnet for JAX*, 2020.

[22] A. Sergeev and M. Del Balso, *Horovod: fast and easy distributed deep learning in TensorFlow*, 1802.05799.

[23] C.R. Harris, K.J. Millman, S.J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau et al., *Array programming with NumPy*, *Nature* **585** (2020) 357.

[24] P. Virtanen, R. Gommers, T.E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau et al., *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*, *Nature Methods* **17** (2020) 261 [1907.10121].

[25] J.D. Hunter, *Matplotlib: A 2d graphics environment*, *Comput. Sci. Eng.* **9** (2007) 90.