

# NEURAL TRANSFORMATIONS FOR EFFICIENT TOPOLOGICAL MIXING

**Anonymous authors**

Paper under double-blind review

## ABSTRACT

We propose a generalized version of the L2HMC algorithm (Lévy et al., 2018), and evaluate its ability to sample from different topologies in a two-dimensional lattice gauge theory. In particular, we demonstrate that our model is able to successfully mix between modes of different topology, significantly reducing the computational cost required to generate independent gauge configurations.

## 1 INTRODUCTION

TODO: Complete introduction

## 2 MAIN CONTRIBUTIONS

1. We propose a generalized version of the L2HMC algorithm that allows for different networks with different step sizes for each distinct leapfrog step. We represent this generalization by carrying a discrete index  $k$  through the augmented leapfrog equations, indicating that these functions are allowed to vary.
2. We introduce a topological charge metric  $\delta_{\mathcal{Q}_{\mathbb{R}}}(\xi', \xi)$  defined as the squared difference between the initial and proposed configurations  $\delta_{\mathcal{Q}_{\mathbb{R}}}(\xi', \xi) = (\mathcal{Q}'_{\mathbb{R}} - \mathcal{Q}_{\mathbb{R}})^2$
3. We introduce an *annealing schedule*  $\{\beta\}_{t=0}^N = \{\beta_0, \beta_1, \dots, \beta_N\}$ , with  $\beta_0 < \beta_1 < \dots < \beta_N$  and  $\beta_{t+1} - \beta_t \ll 1$ .
4. We use the *integrated autocorrelation time* of the topological charge  $\tau_{\text{int}}^{\mathcal{Q}}$  as a metric for determining the efficiency of our trained sampler.
5. We compare our results to traditional HMC across a variety of trajectory lengths and inverse coupling constants, and show that our trained model consistently outperforms traditional HMC.

## 3 RELATED WORK:

Related work: (Albergo et al., 2021; Favoni et al., 2020; Medvidovic et al., 2020; Neklyudov & Welling, 2020; Li et al., 2020; Boyda et al., 2020; Kanwar et al., 2020; Toth et al., 2019; Hoffman et al., 2019; Albergo et al., 2019; Neklyudov et al., 2020; Wehenkel & Louppe, 2020; Pasarica & Gelman, 2010; Dinh et al., 2016; Wilson, 1974; Tanaka & Tomiya, 2017; Schaefer et al., 2009; Cossu et al., 2018; Rezende et al., 2020)

## 4 BACKGROUND

### 4.1 HAMILTONIAN MONTE CARLO

The Hamiltonian Monte Carlo (HMC) algorithm is a widely used technique that allows us to sample from an analytically known target distribution  $p(x)$  by constructing a chain of states  $\{x^{(0)}, x^{(1)}, \dots, x^{(n)}\}$ , such that  $x^{(n)} \sim p(x)$  in the limit  $n \rightarrow \infty$ . For our purposes, we assume that our target distribution can be expressed as a Boltzmann distribution,  $p(x) = \frac{1}{Z} e^{-S(x)} \propto e^{-S(x)}$ ,

where  $S(x)$  is the *action* of our theory, and  $\mathcal{Z}$  is often referred to as the *partition function*, which ensures our target distribution is correctly normalized to unity. In this case, HMC begins by augmenting the state space with a fictitious momentum variable  $v$ , normally distributed independently of  $x$ , i.e.  $v \sim \mathcal{N}(0, \mathbb{I})$ . Our joint distribution can then be written as

$$p(x, v) = p(x) \cdot p(v) \propto e^{-S(x)} \cdot e^{-\frac{1}{2}v^T v} = e^{-\mathcal{H}(x, v)} \quad (1)$$

where  $\mathcal{H}(x, v)$  is the Hamiltonian of the joint  $(x, v)$  system. Notably, this system obeys Hamilton's equations

$$\dot{x} = \frac{\partial \mathcal{H}}{\partial v}, \quad \dot{v} = -\frac{\partial \mathcal{H}}{\partial x} \quad (2)$$

which can be integrated using the *leapfrog integrator* along iso-probability contours defined by  $\mathcal{H} = \text{const}$ . Explicitly, for a step size  $\varepsilon$  and initial state  $\xi = (x, v)$ , the leapfrog integrator generates a proposal configuration  $\xi' \equiv (x', v')$  by performing the following series of updates:

1. Half-step momentum update:  $v^{1/2} \equiv v(t + \frac{\varepsilon}{2}) = v - \frac{\varepsilon}{2} \partial_x S(x)$
2. Full-step position update:  $x' \equiv x(t + \varepsilon) = x + \varepsilon v^{1/2}$
3. Half-step momentum update:  $v' \equiv v(t + \varepsilon) = v^{1/2} - \frac{\varepsilon}{2} \partial_x S(x')$

We can then construct a complete *trajectory* of length  $\lambda = \varepsilon \cdot N_{\text{LF}}$  by performing  $N_{\text{LF}}$  leapfrog steps in sequence. At the end of our trajectory, we either accept or reject the proposal configuration according to the Metropolis-Hastings acceptance criteria,

$$x_{i+1} = \begin{cases} x' & \text{with probability } A(\xi'|\xi) \\ x & \text{with probability } (1 - A(\xi'|\xi)), \end{cases} \quad \text{where } A(\xi'|\xi) = \min \left\{ 1, \frac{p(\xi')}{p(\xi)} \left| \frac{\partial \xi'}{\partial \xi^T} \right| \right\}. \quad (3)$$

The generic leapfrog integrator is known to be symplectic (conserves energy), so the Jacobian factor reduces to  $\left| \frac{\partial \xi'}{\partial \xi^T} \right| = 1$ .

## 4.2 GENERALIZING THE LEAPFROG INTEGRATOR: L2HMC

### Notable changes:

1. Compared to the original implementation, we carry throughout the updates a discrete index  $k = 0, 1, \dots, N_{\text{LF}}$ , parameterizing the current leapfrog step.  
In doing so, we are free to consider the case where we use different update functions (Equation 5, Equation 6), with completely independent step sizes  $\varepsilon_j^k$ , for each of the  $j = x, v$  updates.
2. We propose a modified loss function, defined in terms of the topological charge metric,

$$\mathcal{L}_\theta(\xi', \xi, A(\xi'|\xi)) = \frac{-A(\xi'|\xi) \cdot \delta_{\mathcal{Q}_{\mathbb{R}}}(\xi', \xi)}{a^2} \quad (4)$$

where  $\delta_{\mathcal{Q}_{\mathbb{R}}}(\xi', \xi) \equiv (\mathcal{Q}'_{\mathbb{R}} - \mathcal{Q}_{\mathbb{R}})^2$  where  $\mathcal{Q}_{\mathbb{R}}$  is the *topological charge*, defined in Section 6

In (Lévy et al., 2018), the authors propose the L2HMC (“Learning to Hamiltonian Monte Carlo”) algorithm, and demonstrate its ability to outperform traditional Hamiltonian Monte Carlo (HMC) on a variety of two-dimensional target distributions. For example, the trained L2HMC sampler is shown to be capable of exploring regions of phase space which are typically inaccessible with traditional HMC. Additionally, they show that the trained sampler is efficient at mixing between modes of a multi-modal target distribution, a feature which is highly desirable for MCMC simulations of lattice gauge theory.

We denote a complete state by  $\xi = (x, v, d)$  with target distribution  $p(\xi) = p(x, v, d) = p(x) \cdot p(v) \cdot p(d)$ . Here we’ve introduced a (uniformly drawn) binary direction variable  $d \in \{-, +\}$  that determines the “direction” of our update, and is distributed independently of both  $x$  and  $v$ . The key modification of the L2HMC algorithm is the introduction of six auxiliary functions  $s_i, t_i, q_i$  for  $i = x, v$  into the leapfrog updates, which are parameterized by weights  $\theta$  in a neural network.

For simplicity, we consider the forward  $d = +1$  direction, and introduce the notation:

$$v'_k \equiv \Gamma_k^+(v_k; \zeta_{v_k}) = v_k \odot \exp\left(\frac{\varepsilon_v^k}{2} s_v^k(\zeta_{v_k})\right) - \frac{\varepsilon_v^k}{2} [\partial_x S(x_k) \odot \exp(\varepsilon_v^k q_v^k(\zeta_{v_k})) + t_v^k(\zeta_{v_k})], \quad (5)$$

$$x'_k \equiv \Lambda^+(x_k; \zeta_{x_k}) = x_k \odot \exp(\varepsilon_x^k s_x^k(\zeta_{x_k})) + \varepsilon_x^k [v'_k \odot \exp(\varepsilon_x^k q_x^k(\zeta_{x_k})) + t_x^k(\zeta_{x_k})] \quad (6)$$

where (1.)  $\zeta_{v_k} = (x_k, \partial_x S(x_k), \tau(k))$ , and  $\zeta_{x_k} = (x_k, v_k, \tau(k))$  are subsets of the augmented space independent of the variable being updated ( $v$ ,  $x$  respectively), (2.)  $\tau(k) = \left[\cos \frac{2\pi k}{N_{\text{LF}}}, \sin \frac{2\pi k}{N_{\text{LF}}}\right]$ ,  $k = 0, 1, \dots, N_{\text{LF}}$ , is a discrete time variable parameterizing our trajectory, and (3.) we indicate the forward  $d = +1$  direction by the  $+$  superscript on  $\Gamma^+$ , and  $\Lambda^+$ . Using this notation, we can write the complete leapfrog update (in the forward  $d = +1$  direction) as:

1. Half-step momentum update:  $v'_k = \Gamma_k^+(v_k; \zeta_{v_k})$
2. Full-step half-position update<sup>1</sup>:  $x'_k = \bar{m}^t \odot x_k + m^t \odot \Lambda_k^+(x_k; \zeta_{x_k})$
3. Full-step half-position update:  $x''_k = \bar{m}^t \odot \Lambda_k^+(x'_k; \zeta_{x'_k}) + m^t \odot x'_k$
4. Half-step momentum update:  $v''_k = \Gamma_k^+(v'_k; \zeta_{v'_k})$

Note that in order to keep our leapfrog update reversible, we've split the  $x$  update into two sub-updates by introducing a binary mask  $m^t = m^t \odot \mathbb{1} + \bar{m}^t \odot \mathbb{1}$  that updates half of the components of  $x$  sequentially, as shown (with the general transformation  $\Lambda_k^\pm$ ) in Figure 1.

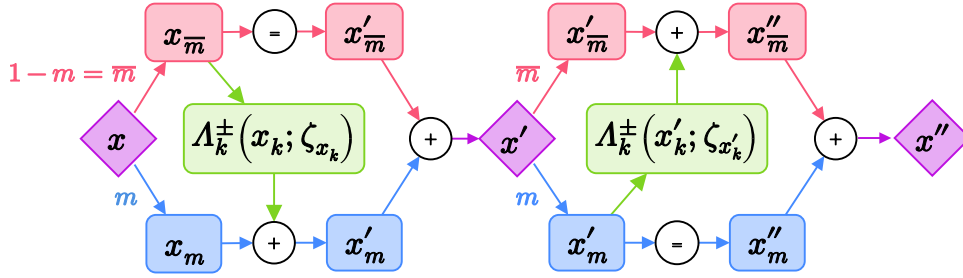


Figure 1: Illustration of the split  $x$  update. Here we include the general form of the transformation  $\Lambda_k^\pm$ ,

As in HMC, we form a complete trajectory by performing  $N_{\text{LF}}$  leapfrog steps in sequence, followed by a Metropolis-Hastings accept/reject step as described in Equation 3. However, unlike in the expression for HMC, we must take into account the overall Jacobian factor from the update  $\xi \rightarrow \xi'$ , which can be easily computed as

$$\left| \frac{\partial v'_k}{\partial v_k} \right| = \exp\left(\frac{\varepsilon_v^k}{2} s_v^k(\zeta_{v_k})\right), \quad \left| \frac{\partial x'_k}{\partial x_k} \right| = \exp\left(\varepsilon_x^k s_x^k(\zeta_{x_k})\right). \quad (7)$$

In order to perform the updates in the generalized leapfrog integrator, we need to evaluate each of the functions  $s, t, q$ . Without loss of generality<sup>2</sup>, we temporarily ignore the discrete leapfrog index  $k$ , and restrict our attention to the  $s_x, t_x, q_x$  functions used in the  $x$  update, Equation 6.

Each of the  $s_x, t_x, q_x$  functions takes as input  $\zeta_x = (x, v, \tau)$ , with  $x \in \mathbb{R}^n$ ,  $v \in \mathbb{R}^n$ , and  $\tau \in \mathbb{R}^2$ . The network splits these inputs and constructs the following intermediate variable (where  $\sigma$  denotes an arbitrary activation function)

$$z_1 = \sigma(w_x^T x + w_v^T v + w_\tau^T \tau + b). \quad (8)$$

This intermediate variable  $z_1$  is then passed through another series of fully-connected layers,

$$z_n = \sigma(w_n^T z_{n-1} + b_n), \quad z_{n-1} = \sigma(w_{n-2}^T z_{n-2} + b_{n-2}), \quad \dots, \quad z_2 = \sigma(w_2^T z_1 + b_2). \quad (9)$$

<sup>1</sup>By this we mean we are performing a complete update step that only updates half of the components of  $x$  determined by the mask  $m^t$  and its complement  $\bar{m}^t$ .

<sup>2</sup>Because we maintain a separate network with identical architecture for evaluating the  $s, t, q$  functions in the momentum update Equation 5, the procedure is identical

The network outputs  $s_x, t_x, q_x$  are then defined in terms of this final hidden variable  $z_n$  as

$$s_x(\zeta_x) = \alpha_s \tanh(w_s z_n + b_s), \quad t_x(\zeta_x) = w_t^T z_n + b_t, \quad q_x(\zeta_x) = \alpha_q \tanh(w_q z_n + b_q) \quad (10)$$

where  $\alpha_s$ , and  $\alpha_q$  are trainable scaling factors. The only requirement on the details of the network is that the dimensionality of the outputs  $s_x, t_x, q_x$  match the dimensionality of our physical variables  $x, v \in \mathbb{R}^n$ .

Next, we introduce a loss function

$$\mathcal{L}_\theta(\xi, \xi', A(\xi'|\xi)) = -\frac{A(\xi'|\xi) \cdot \delta(\xi, \xi')}{a^2} \quad (11)$$

where  $\delta(\xi, \xi')$  is a suitably chosen *metric function*, and  $a$  is a scaling factor.

## 5 ANNEALING SCHEDULE

To help our sampler overcome the large energy barriers between isolated modes, we introduce an *annealing schedule*, during the training phase

$$\{\beta_t\}_{t=0}^N = \{\beta_0, \beta_1, \dots, \beta_{N-1}, \beta_N\}, \quad (12)$$

where  $\beta_0 < \beta_1 < \dots < \beta_N \equiv 1$ ,  $\beta_{t+1} - \beta_t \ll 1$ ,  $N$  denotes the total number of training steps to be performed. Note that we are free to vary  $\beta$  during the initial training phase as long as we recover the true distribution with  $\beta \equiv 1$  at the end of training and evaluate our trained model without this factor. Explicitly, for  $\beta_t < 1$  this rescaling factor helps to reduce the height of the energy barriers, making it easier for our sampler to explore previously inaccessible regions of the phase space. In terms of this additional annealing schedule, our target distribution picks up an additional index  $t$  to represent our progress through the training phase, which can be written explicitly as

$$p_t(x) \propto e^{-\beta_t S(x)} \quad (13)$$

for  $t = 0, 1, \dots, N$ .

## 6 LATTICE GAUGE THEORY

**Note:** We define the lattice gauge theory in terms of the link variables  $-\pi \leq \varphi_\mu(x) < \pi$ , which are identified as being the *position* variable  $x$  in Section 4.2, i.e.  $\varphi_\mu(x) \iff x$ .

We consider a two-dimensional  $U(1)$  lattice gauge theory defined on an  $N_x \times N_t$  lattice with periodic boundary conditions. Our target distribution  $p(\varphi)$  is defined in terms of the Wilson action as

$$p(\varphi) \propto e^{\beta_t S(\varphi)}, \quad \text{where} \quad S(\varphi) = \sum_P 1 - \cos(\varphi_P(x)). \quad (14)$$

Here,  $\beta = 2/g_0^2$  is the inverse coupling constant of the theory, and  $\beta \rightarrow \infty$  gives the continuum limit. Here,  $\varphi_P$  denotes the sum of the link variables around the elementary plaquette,

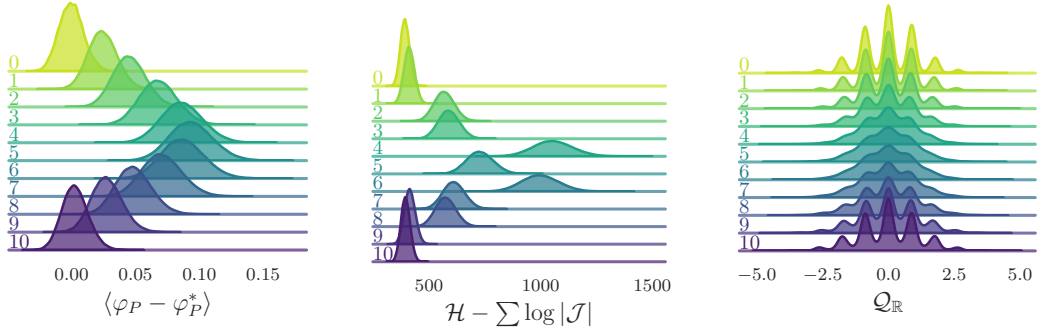
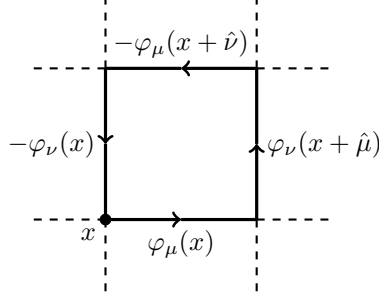
$$\varphi_P = \varphi_\mu(x) + \varphi_\nu(x + \hat{\mu}) - \varphi_\mu(x + \hat{\nu}) - \varphi_\nu(x) \in [0, 4\pi] \quad (15)$$

as shown in Figure 2. Note that we have written the Wilson action (Equation 14) in terms of the link variables  $\varphi_\mu(x) \in [-\pi, \pi]$

$$U_\mu(x) = e^{i\varphi_\mu(x)}, \quad \varphi_\mu(x) \in [-\pi, \pi] \quad (16)$$

## ACKNOWLEDGMENTS

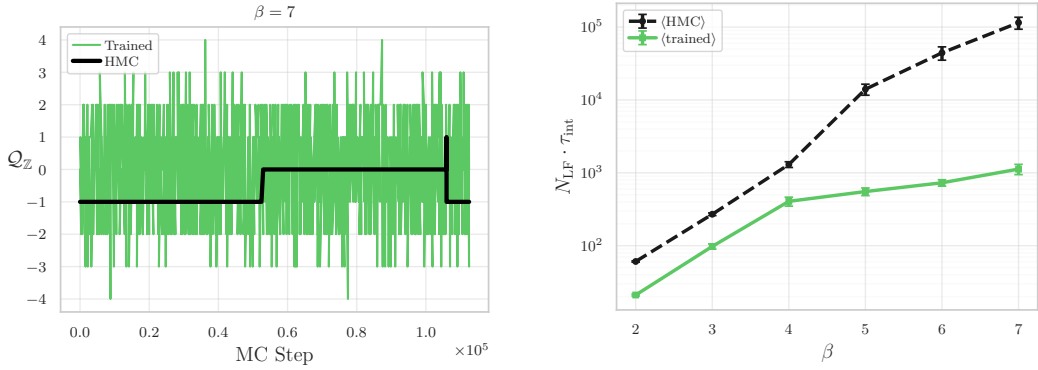
This research used resources of the argonne leadership computing facility, which is a doe office of science user facility supported under contract DE-AC02-06CH11357. This work describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the work do not necessarily represent the views of the u.s. doe or the united states government.

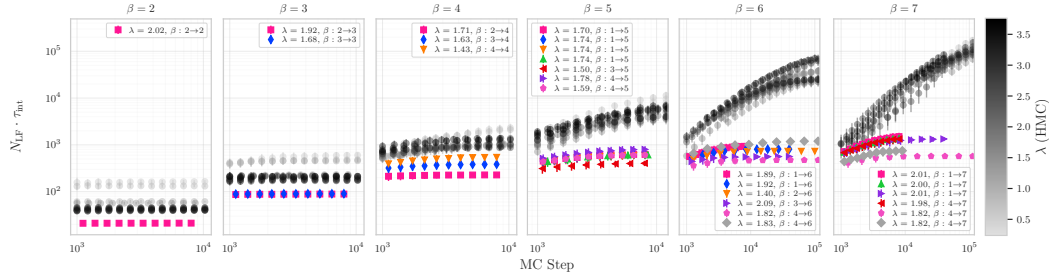
Figure 2: Elementary plaquette,  $P$  on the lattice.(a) Illustration of the error in the average plaquette,  $\langle \varphi_P - \varphi_P^* \rangle$  vs leapfrog step.

(b) Illustration of the error in the rescaled energy, vs leapfrog step.

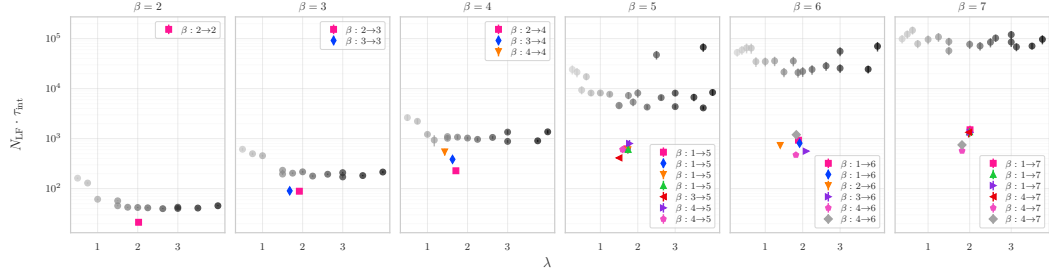
(c) Illustration of the error in the rescaled energy,  $Q_{\mathbb{R}}$  vs leapfrog step.

Figure 3: Illustration of how various observables vary over a single trajectory.

(a) Plot of the topological charge history  $Q_{\mathbb{R}}$  vs MC step for both generic HMC (black line), and the trained model (green line)(b) Plot of the estimated integrated autocorrelation time  $\tau_{\text{int}}^{Q_{\mathbb{R}}}$  vs  $\beta$  for both generic HMC (black dashed line), and the trained model (solid green line).



(a) Plot of the integrated autocorrelation time  $\tau_{\text{int}}^{\text{Q}_R}$  vs MC step for both generic HMC and the trained model.



(b) Plot of the integrated autocorrelation time  $\tau_{\text{int}}^{\text{Q}_R}$  vs  $\lambda$  for both generic HMC and the trained model.

## REFERENCES

- Michael S Albergo, Denis Boyda, Daniel C Hackett, Gurtej Kanwar, Kyle Cranmer, Sébastien Racanière, Danilo Jimenez Rezende, and Phiala E Shanahan. Introduction to normalizing flows for lattice field theory. *arXiv preprint arXiv:2101.08176*, 2021.
- MS Albergo, G Kanwar, and PE Shanahan. Flow-based generative models for markov chain monte carlo in lattice field theory. *Physical Review D*, 100(3):034515, 2019.
- Denis Boyda, Gurtej Kanwar, Sébastien Racanière, Danilo Jimenez Rezende, Michael S Albergo, Kyle Cranmer, Daniel C Hackett, and Phiala E Shanahan. Sampling using  $su(n)$  gauge equivariant flows. *arXiv preprint arXiv:2008.05456*, 2020.
- Guido Cossu, Peter Boyle, Norman Christ, Chulwoo Jung, Andreas Jüttner, and Francesco Sanfilippo. Testing algorithms for critical slowing down. In *EPJ Web of Conferences*, volume 175, pp. 02008. EDP Sciences, 2018.
- Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv:1605.08803 [cs, stat]*, May 2016. URL <http://arxiv.org/abs/1605.08803>. arXiv: 1605.08803.
- Matteo Favoni, Andreas Ipp, David I Müller, and Daniel Schuh. Lattice gauge equivariant convolutional neural networks. *arXiv preprint arXiv:2012.12901*, 2020.
- Matthew Hoffman, Pavel Sountsov, Joshua V Dillon, Ian Langmore, Dustin Tran, and Srinivas Vasudevan. Neutra-lizing bad geometry in hamiltonian monte carlo using neural transport. *arXiv preprint arXiv:1903.03704*, 2019.
- Gurtej Kanwar, Michael S Albergo, Denis Boyda, Kyle Cranmer, Daniel C Hackett, Sébastien Racanière, Danilo Jimenez Rezende, and Phiala E Shanahan. Equivariant flow-based sampling for lattice gauge theory. *Physical Review Letters*, 125(12):121601, 2020.
- Daniel Lévy, M. Hoffman, and Jascha Sohl-Dickstein. Generalizing hamiltonian monte carlo with neural networks. *ArXiv*, abs/1711.09268, 2018.
- Zengyi Li, Yubei Chen, and Friedrich T Sommer. A neural network mcmc sampler that maximizes proposal entropy. *arXiv preprint arXiv:2010.03587*, 2020.

- Matija Medvidovic, Juan Carrasquilla, Lauren E Hayward, and Bohdan Kulchytskyy. Generative models for sampling of lattice field theories. *arXiv preprint arXiv:2012.01442*, 2020.
- Kirill Neklyudov and Max Welling. Orbital mcmc. *arXiv preprint arXiv:2010.08047*, 2020.
- Kirill Neklyudov, Max Welling, Evgenii Egorov, and Dmitry Vetrov. Involutive mcmc: a unifying framework. In *International Conference on Machine Learning*, pp. 7273–7282. PMLR, 2020.
- Cristian Pasarica and Andrew Gelman. Adaptively scaling the metropolis algorithm using expected squared jumped distance. *Statistica Sinica*, pp. 343–364, 2010.
- Danilo Jimenez Rezende, George Papamakarios, Sébastien Racaniere, Michael Albergo, Gurtej Kanwar, Phiala Shanahan, and Kyle Cranmer. Normalizing flows on tori and spheres. In *International Conference on Machine Learning*, pp. 8083–8092. PMLR, 2020.
- Stefan Schaefer, Rainer Sommer, and Francesco Virota. Investigating the critical slowing down of qcd simulations. *arXiv preprint arXiv:0910.1465*, 2009.
- Akinori Tanaka and Akio Tomiya. Towards reduction of autocorrelation in hmc by machine learning. *arXiv preprint arXiv:1712.03893*, 2017.
- Peter Toth, Danilo Jimenez Rezende, Andrew Jaegle, Sébastien Racanière, Aleksandar Botev, and Irina Higgins. Hamiltonian generative networks. *arXiv preprint arXiv:1909.13789*, 2019.
- Antoine Wehenkel and Gilles Louppe. You say normalizing flows i see bayesian networks. *arXiv preprint arXiv:2006.00866*, 2020.
- Kenneth G. Wilson. Confinement of quarks. *Phys. Rev. D*, 10:2445–2459, 1974.

## A APPENDIX

## B TRAINING ALGORITHM

**Algorithm 1:** Training procedure

---

```

input : [1]
    1. Target distribution,  $p_t(x) \propto e^{-\beta_t U(x)}$ 
    2. Loss function,  $\mathcal{L}_\theta(\xi', \xi, A(\xi'|\xi))$ 
    3. Learning rate schedule,  $\{\alpha_t\}_{t=0}^{N_{\text{train}}}$ 
    4. Annealing schedule,  $\{\beta_t\}_{t=0}^{N_{\text{train}}}$ 
    5. Batch of initial states,  $x$ 
;Initialize weights  $\theta$  [2]
for  $0 \leq t < N_{\text{train}}$  : [3]
    [4]
    [5]
    update:  $p_t(x) \propto e^{-\beta_t U(x)}$  [6]
    resample:  $v \sim \mathcal{N}(0, \mathbb{1})$  [7]
    resample:  $d \sim \mathcal{U}(+, -)$  [8]
    construct:  $\xi \equiv (x, v, d)$  [9]
    [10]
    for  $0 \leq \ell < N_{\text{LF}}$  : [11]
        | propose:  $\xi'_\ell \leftarrow \mathbf{FL}_\ell^\pm \xi_\ell$  [12]
        [13]
        compute:  $A(\xi'|\xi) = \min \left\{ 1, \frac{p(\xi')}{p(\xi)} \left| \frac{\partial \xi'}{\partial \xi^T} \right| \right\}$  [14]
        [15]
        update:  $\mathcal{L} \leftarrow \mathcal{L}_\theta(\xi', \xi, A(\xi'|\xi))$  [16]
        [17]
        backprop:  $\theta \leftarrow \theta - \alpha_t \nabla_\theta \mathcal{L}$  [18]
        [19]
        assign:  $x_{t+1} \leftarrow \begin{cases} x' & \text{with probability } A(\xi'|\xi) \\ x & \text{with probability } (1 - A(\xi'|\xi)). \end{cases}$  [20]
    [21]

```

---