Neural Network Gauge Field Transformation for 4D SU(3) gauge fields

Xiao-Yong Jin[a,*]

[a]*Computational Science Division, Argonne National Laboratory,*
Lemont, IL 60439, USA

E-mail: xjin@anl.gov

We construct neural networks that work for any Lie group and maintain gauge covariance, enabling smooth, invertible gauge field transformations. We implement these transformations for 4D SU(3) lattice gauge fields and explore their use in HMC. We focus on developing loss functions and optimizing the transformations. We show the effects on HMC's molecular dynamics and discuss the scalability of the approach.

---

[*]Speaker

1. Introduction

Neural networks can approximate arbitrary functions [1–4]. When combined with gauge symmetries [5, 6] and adapted to lattice fields [7–12], they offer new tools for developing algorithms [13–20] in lattice QCD. One key application is improving the efficiency of the Hybrid Monte Carlo (HMC) algorithm [21].

HMC uses Hamiltonian dynamics to generate proposal configurations for a Markov Chain, sampling the configuration space according to a known unnormalized probability density function. The Hamiltonian dynamics, however, can struggle with potential barriers due to limited energy or may linger in lengthy potential valleys. Lattice QCD systems, with their high degrees of freedom and separation of long and short distance modes, exhibit critical slowing down [22] near the continuum limit, with the cost of generating independent gauge field configurations also increasing as the force in the Hamiltonian dynamics requires finer numerical integration steps. This conference offers two reviews [23, 24] on improvements to HMC algorithms.

This work focuses on creating smooth and invertible gauge field transformations with tunable parameters, incorporating neural networks while ensuring gauge covariance. Generalizing prior work on the perturbative Wilson flow [25] and extending a study in 2D U(1) gauge field [10], we apply tuned transformations in HMC with 4D SU(3) gauge fields.

We first describe the gauge field transformation, then introduce the neural networks, and present results in optimizing and applying the transformation in molecular dynamics. We discuss the scalability and transferability of our approach across different lattice volumes and gauge couplings.

2. Gauge field transformation

We introduce a continuously differentiable bijective map $\mathcal{F}$ with $U = \mathcal{F}(V)$ on the gauge fields $V \mapsto U$, and apply a change of variables in the path integral for an observable O,

$$\langle O \rangle = \frac{1}{Z} \int \mathrm{D}[U] O(U) e^{-S(U)} = \frac{1}{Z} \int \mathrm{D}[V] O\big(\mathcal{F}(V)\big) e^{-S_{\mathrm{FT}}(V)}, \tag{1}$$

with the effective action after the field transformation,

$$S_{\mathrm{FT}}(V) = S\big(\mathcal{F}(V)\big) - \ln \big|\mathcal{F}_*(V)\big|, \tag{2}$$

and the Jacobian of the transformation,

$$\mathcal{F}_*(V) = \frac{\partial \mathcal{F}(V)}{\partial V}. \tag{3}$$

We construct the gauge field transformation with

$$U_{x,\mu} = \exp\left[\sum_l \epsilon_{x,\mu,l} \partial_{x,\mu} W_{x,\mu,l}\right] V_{x,\mu}, \tag{4}$$

where $V_{x,\mu}$ denotes the gauge link from lattice site x to $x + \hat{\mu}$, in the direction μ. The Wilson loop, $W_{x,\mu,l}$, is an ordered product of gauge links along the loop labeled l that goes through the link at x, μ. The group derivative, $\partial_{x,\mu} W_{x,\mu,l}$, is with respect to the gauge link, $V_{x,\mu}$.
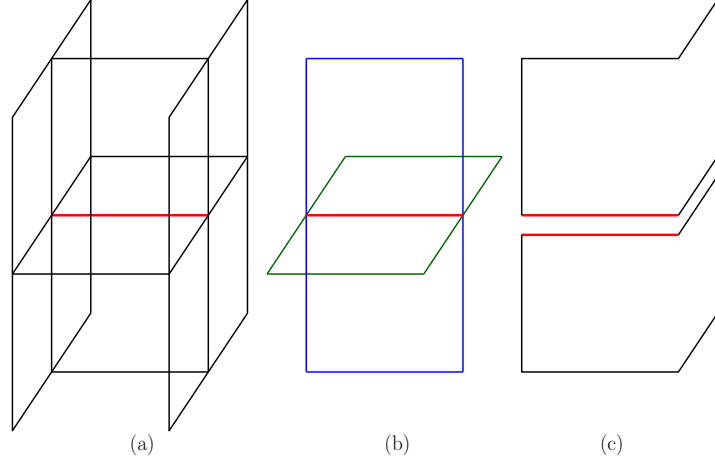
Figure 1: Wilson loops used in smearing update to the red link in a 3D lattice. From left to right, (a) the links in black used to update the red link, (b) four plaquette loops, (c) two 6-link chair shaped loops with one side perpendicular to, and at the right side of, the red link.

Generalizing the stout smear [26] or the Wilson flow [25], we optionally make the coefficients depend on Wilson loops $(X, Y, \ldots)$ independent of $V_{x,\mu}$,

$$\epsilon_{x,\mu,l} = c_l \frac{2}{\pi} \tan^{-1} \left[\mathcal{N}_l(X, Y, \ldots) \right]. \tag{5}$$

The additional application of $\tan^{-1}$ and the multiplication of scalar coefficients c_l serve to constrain the possible values of ϵ, ensuring the resulting Jacobian determinant is positive definite. The independence of the Wilson loops $(X, Y, \ldots)$ from $V_{x,\mu}$ simplifies the computation of the Jacobian determinant and results in the constraint [25]

$$c_l = \frac{3}{4N_l}, \tag{6}$$

for N_l the number of distinct Wilson loops l. Naturally, $\mathcal{N}_l$ can be any neural network.

In practice, we update only a subset of the gauge links at a time, a single direction out of four space-time directions, and on either even or odd lattice sites lexicographically. For each subset, we pick the derivatives of the Wilson loops, $\partial_{x,\mu} W_l$, such that the only link to be updated in W_l is x, μ. This ensures that the Jacobian matrix of the transformation is diagonal. Updating the whole lattice requires eight separate subset updates.

Figure 1 shows the Wilson loops used for taking derivatives, in equation (4), when updating a single gauge link, in our implementation. While the figure depicts a 3D lattice, in 4D space-time, there are 6 plaquette loops and 48 chair loops, which go through the updating link marked red in the figure. The derivatives of those loops with respect to the red link generate the basis, the linear combination of which forms the $su(3)$ algebra space in the exponential map in equation (4).

3. Neural network architectures

Stacking layers of individual transformations as in equation (4) forms a deep residual neural network [9] generalized for gauge fields. Further generalizing it with standard neural networks,
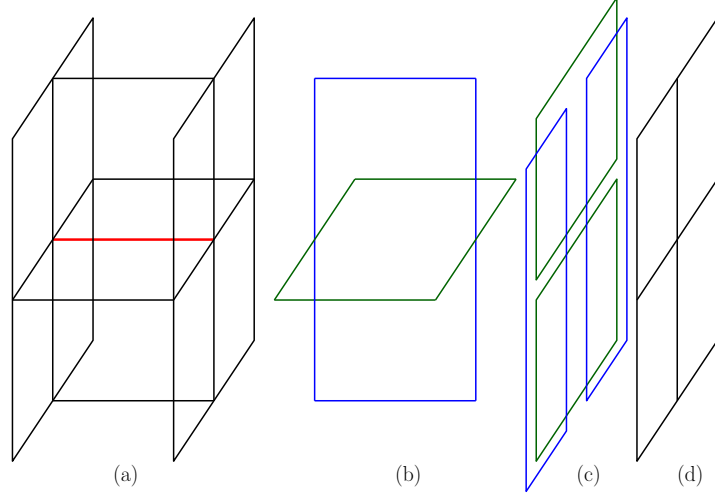
Figure 2: Input to the neural network that computes smearing coefficients for updating the red link in a 3D lattice. From left to right, (a) the links in black used to compute Wilson loops as input to a neural network, (b) two 6-link rectangle loops parallel to the red link, (c) four 6-link rectangle loops perpendicular to the red link on one side, (d) four plaquette perpendicular to the red link on one side.

we employ basic neural network layers that sequentially apply to traced Wilson loops and produce localized smearing coefficients $\epsilon_{x,\mu,l}$ as introduced in equation (5). As mentioned, to simplify the computation of the Jacobian matrix, the traced Wilson loops used at each layer are independent of the links to be updated at that layer with equation (4).

To limit implementation complexity, we use only plaquette and 2×1 rectangular traced Wilson loops for computing local smearing coefficients, as shown in figure 2 for updating the red link (showing a 3D lattice as an example). In a 4D space-time lattice gauge field, for each link to be updated, we use four plaquettes in each of the three planes parallel to and on each side of the updating link, and the same amount of the 2×1 rectangular loops in those planes, and three 2×1 rectangular loops that are parallel to the updating link. These amount to 51 distinct loops that are locally close to the updating link. To directly use available neural network implementations from TensorFlow [27], we compute the traces of the ordered product, and the traces of their squared and cubic powers, of the gauge links along the loops, and then use the real and imaginary parts of the complex numbers as the input to the traditional neural networks, resulting in six real floating point numbers per Wilson loop, and a total of 306 real numbers per lattice site.

$$Y_{x,\mu,l} = \{\operatorname{Re} y_{x,\mu,l,i}, \operatorname{Im} y_{x,\mu,l,i}\}_{i=1,2,3} \tag{7}$$

$$y_i = \operatorname{tr} W^i_{x,\mu,l} \tag{8}$$

Instead of directly calling 4D convolutional neural networks, we implement symmetrical shift, which shifts all the traces of Wilson loops of a fixed distance to the lattice link to be updated, and then call dense neural networks,

$$Y_{x,\mathrm{out}} = \sigma(WY_{x,\mathrm{in}} + b), \tag{9}$$

where the weight matrix W and bias vector b are shared among the lattice sites inside a particular layer. We choose Swish [28] as the activation function, σ, for its continuous derivative. We omit the

index μ here, since we always update links in a single direction per neural network layer.

We also introduce a normalization layer similar to typical layer-normalization in computer vision, but we use site-local scaling coefficients (γ_i) and shift (β_i), which have the same dimension as the vector at each site and have the same values across the lattice.

$$m = \frac{1}{DV/2} \sum_{x,i} Y_{x,i,\text{in}}, \tag{10}$$

$$\tilde{Y}_{x,i} = Y_{x,i,\text{in}} - m \tag{11}$$

$$\sigma_m^2 = \frac{1}{DV/2} \tilde{Y}_{x,i}^2 \tag{12}$$

$$Y_{x,i,\text{out}} = \gamma_i \tilde{Y}_{x,i}/\sigma_m + \beta_i \tag{13}$$

where D is the vector dimension per site, V is the lattice volume, and the summation goes through the lattice (single parity) and the vector index.

Additionally, we implemented self-attention, feedforward, and residue networks, which operate on numerical vectors on a 4D lattice. However, neural network layers using these operations exceed the available compute and memory for reasonably large lattice volumes.

As we update the gauge links in a single direction on either even or odd lattice sites in a layer, this computation creates a vector of real floating-point numbers on each lattice site of a given parity, even or odd. Each operation—symmetric shift, site-local dense network, normalization, and etc.—applies to the lattice field of vectors on a single parity and returns a vector field of the same lattice size with optionally different degrees of freedom per site. Thus, we can compose many layers of sequential operations that create the local coefficients used for gauge field transformation in the smearing update as in equation (4). We sequentially apply transformation layers with different weights to gauge link subsets in a single direction and parity. This transforms the whole lattice gauge field, yielding a continuously differentiable, bijective deep neural network map of the fields.

Our code for HMC with neural network field transformation is available online [29].

4. Test result

We use the DBW2 action with $c_1 = -1.4088$, $\beta = 0.7796$, corresponding to $a/r_0 \simeq 0.4$ and $a = 0.2000(20)$ as a baseline following Ref [30, 31]. We optimize the map so that the gauge force of the effective action matches an action with a stronger gauge coupling,

$$S'_{\text{FT}}(V; \beta = 0.7796) = \partial_{x,\mu} \left[S(\mathcal{F}(V; \beta = 0.7796)) - \ln |\mathcal{F}_*(V)| \right] \tag{14}$$

$$\sim S'(V; \beta = \beta_{\text{T}}). \tag{15}$$

As β is a multiplicative constant in the gauge action, matching forces effectively seeks a transformation that reduces the gauge force per link by a constant factor.

Denote the difference in force computed from the transformed action and the original action with different couplings as,

$$\Delta_{x,\mu,c} = \partial_{x,\mu,c} S_{\text{FT}}(V; \beta) - \partial_{x,\mu,c} S(V; \beta_{\text{T}}), \tag{16}$$

with the subscript c denote the degree of freedom of the gauge Lie algebra, and per link (x, μ) l2-norm as,

$$\Delta_{x,\mu} = \left(\sum_c \left(\partial_{x,\mu,c} S_{\text{FT}}(V; \beta) - \partial_{x,\mu,c} S(V; \beta_{\text{T}}) \right)^2 \right)^{1/2} . \tag{17}$$

We choose the exponential of this l2-norm per link as the loss function for optimization:

$$L_{\text{LMEN}} = \log \sum_{x,\mu} \exp \left(\Delta_{x,\mu} \right) - \log(4\text{Vol}). \tag{18}$$

For training the models, we generate configurations at $\beta = 0.7796$, with a lattice size of $8^3 \times 16$, using 4 HMC streams, and a trajectory length of 4, saving configurations every 16 trajectories, or 64 molecular dynamic time units, which is larger than the integrated autocorrelation lengths reported in reference [31]. The computed topological charges show negligible autocorrelation. During training, we use the target $\beta_{\text{T}} = 0.7099$, corresponding to $a/r_0 \simeq 0.6$, from an extrapolation formula in Ref [30].

In a four-dimensional lattice, our code currently implement the transformation, using a stack of smearing layers, each using the derivatives of 6 plaquette and 48 chair terms for the smearing kernel per updating gauge link. The coefficients can be globally tuned numbers ($6 + 48 = 54$ parameters for a single link) or the output of a constructed neural network as described in the previous section that depends on nearby traced Wilson loops. In the following, we show two different models of gauge field transformation: one with global coefficients (GC) and one with a Dense-Normalization-Dense neural network (NN), each consisting of a stack of layers.

The order of layers in the GC model in Python's list comprehension is:

```
transform=transform.TransformChain(
    [StoutSmearSlice(coeff=CoefficientVariable(p0, chair=c0, rng=rng), dir=dir, is_odd=eo)
        for _ in range(3) for dir in range(4) for _ in range(3) for eo in {False,True}]),
```

The layers update the gauge link in a single direction for the even and then odd subset 3 times. These are repeated for 4 different directions, covering the whole lattice gauge links. Overall, the whole lattice is updated 3 times. With 54 real parameters per layer, this transformation model uses $54 \times 2 \times 3 \times 4 \times 3 = 3888$ parameters. The number of parameters is independent of the lattice size.

The order of layers in the NN model is:

```
transform=transform.TransformChain(
    [StoutSmearSlice(coeff=CoefficientNets([Dense(units=8, activation='swish'),
                        Normalization(),
                        Dense(units=54, activation=None)]), dir=dir, is_odd=eo)
        for _ in range(4) for dir in range(4) for _ in range(1) for eo in {False,True}]
```

with the coefficients generated by a Dense network, Normalization, and another Dense network. The units in the Dense network initialization denotes the dimension of the output vector. The last Dense network does not use a non-linear activation function, as its 54 output numbers directly feed into the input of the $\tan^{-1}$ function in equation (5).

We train our models using a single Nvidia A100 GPU with 40 GB memory. Training the GC model uses 64 gauge configurations for 8 epochs, with each epoch going through the 64 configurations in a random order. It takes 3 hours and uses a maximum of 30 GB GPU memory. The NN model, on
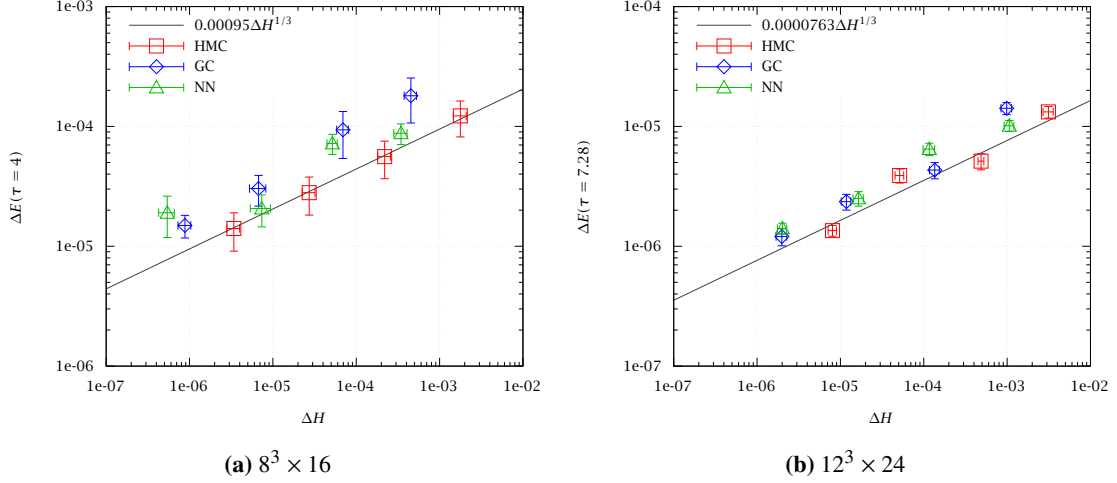
(a) $8^3 \times 16$ **(b)** $12^3 \times 24$

Figure 3: The difference in energy density versus the difference in Hamiltonian after a single Omelyan step. The energy density is at Wilson flow time, $\tau = 4$ for $8^3 \times 16$ with $\beta = 0.7796$, and $\tau = 7.28$ for $12^3 \times 24$ with $\beta = 0.8895$. The step sizes for the Omelyan 2MN integrator are $\delta t = 0.0025, 0.005, 0.01$, and 0.02, for symbols from left to right.

the other hand, has more parameters and uses 256 configurations for training over 16 epochs. That takes 11 hours and uses a maximum of 20 GB GPU memory.

During evaluation on a single Nvidia A100 GPU with 40 GB memory, we use gauge configurations generated with HMC at $\beta = 0.7796$ for $8^3 \times 16$ lattices, and $\beta = 0.8895$ for $12^3 \times 24$ lattices, using the DBW2 gauge action, from separate HMC streams that are different from those used for training. With a lattice size of $8^3 \times 16$, a single force evaluation of the effective action, equation (2), with the GC model takes 1.6 seconds and uses a maximum of 10 GB GPU memory. A single force evaluation with the NN model takes 1.1 seconds and uses a maximum of 8 GB GPU memory. To compare, using the same framework, a single force evaluation of the DBW2 lattice action without transformations takes less than 0.005 seconds and uses a maximum of 128 MB GPU memory. The evaluation of the same models on the $12^3 \times 24$ lattices requires more memory than the 40 GB provided by the single Nvidia A100 we use, and thus runs on CPUs.

Since our machine learning framework (TensorFlow) has limited ability to distribute lattice gauge configurations across GPUs, and our models have no lattice size dependencies, we focus on how the machine-learned model scales with lattice size.

To study the effects of field transformation on the molecular dynamics in HMC, we start from thermalized configurations and perform a single step of the Omelyan 2MN integrator [32]. We measure the energy density using the lattice field tensor computed at 1-loop order with four plaquette loops [33] at Wilson flow time, $\tau = 4$, and examine how the integration step changes the energy density and the Hamiltonian. Figure 3 shows the difference in energy density versus the difference in Hamiltonian, before and after the integration step. The three kinds of symbols are from the lattice action without transformation (labeled ''HMC''), the lattice effective action with the field transformation with global coefficients (labeled ''GC''), and the lattice effective action with the field transformation with a Dense-Normalization-Dense neural network (labeled ''NN''). The same kind

of symbols from left to right correspond to four different step sizes, $\delta t = 0.0025, 0.005, 0.01$, and 0.02. With a single integration step, ΔH scales with δt^3, and ΔE scales with δt, thus $\Delta E \propto \Delta H^{1/3}$. We fit the line through the HMC data and show it in the figure.

The left panel in figure 3 shows the model performance on $8^3 \times 16$ at $\beta = 0.7796$, while the right shows performance on $12^3 \times 24$ at $\beta = 0.8895$, which has $a/r_0 \simeq 0.265$ and $a = 0.1326(13)$ following Ref [30, 31], roughly keeping the same physical volume. The trained model achieves smaller ΔE and ΔH with the same step size. For $8^3 \times 16$ lattices, the models manage to gain larger ΔE at much smaller ΔH, indicating a potential factor of a few better than HMC without field transformations, keeping ΔH constant while inducing a larger change in Wilson-flowed energy density. In addition, the same models trained on $8^3 \times 16$ work well with the $12^3 \times 24$ lattices, maintaining a decent speedup with the larger lattice volume.

5. Conclusion

We implement and optimize gauge field transformation models for reduced effective gauge forces. Applying the machine-optimized models as a change of variables in HMC reduces the Hamiltonian violation, ΔH, and increases the difference in Wilson-flowed energy, ΔE, in our preliminary results.

The improvements remain when we directly apply the models to lattice configurations larger than the lattice volume used for training, indicating the transferability of the learned transformations. With further improvements in evaluation metrics, frameworks, and transformation models, this approach remains a valuable research direction for enhancing sampling efficiency in lattice QCD simulations.

Acknowledgments

References

[1] G. Cybenko, *Approximation by superpositions of a sigmoidal function*, *Math. Control Signals Syst.* **2** (1989) 303.

[2] K. Hornik, M. Stinchcombe and H. White, *Multilayer feedforward networks are universal approximators*, *Neural Networks* **2** (1989) 359.

[3] K. Hornik, *Approximation capabilities of multilayer feedforward networks*, *Neural Networks* **4** (1991) 251.

[4] M. Leshno, V.Y. Lin, A. Pinkus and S. Schocken, *Multilayer feedforward networks with a nonpolynomial activation function can approximate any function*, *Neural Networks* **6** (1993) 861.

[5] T. Cohen, M. Weiler, B. Kicanaoglu and M. Welling, *Gauge equivariant convolutional networks and the icosahedral CNN*, in *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, eds., vol. 97 of *Proceedings of Machine Learning Research*, pp. 1321–1330, PMLR, 09–15 Jun, 2019, https://proceedings.mlr.press/v97/cohen19d.html [1902.04615].

[6] M. Finzi, S. Stanton, P. Izmailov and A.G. Wilson, *Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data*, 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. 3165–3176, PMLR, 13–18 Jul, 2020, https://proceedings.mlr.press/v119/finzi20a.html [2002.12880].

[7] D. Luo, G. Carleo, B.K. Clark and J. Stokes, *Gauge Equivariant Neural Networks for Quantum Lattice Gauge Theories*, *Phys. Rev. Lett.* **127** (2021) 276402 [2012.05232].

[8] M. Favoni, A. Ipp, D.I. Müller and D. Schuh, *Lattice Gauge Equivariant Convolutional Neural Networks*, *Phys. Rev. Lett.* **128** (2022) 032003 [2012.12901].

[9] Y. Nagai and A. Tomiya, *Gauge covariant neural network for 4 dimensional non-abelian gauge theory*, 2103.11965.

[10] X.-Y. Jin, *Neural Network Field Transformation and Its Application in HMC*, *PoS* **LATTICE2021** (2022) 600 [2201.01862].

[11] J. Aronsson, D.I. Müller and D. Schuh, *Geometrical aspects of lattice gauge equivariant convolutional neural networks*, 2303.11448.

[12] A. Tomiya and Y. Nagai, *Equivariant transformer is all you need*, *PoS* **LATTICE2023** (2024) 001 [2310.13222].

[13] 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].

[14] 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].

[15] S. Foreman, X.-Y. Jin and J.C. Osborn, *Deep Learning Hamiltonian Monte Carlo*, *The Ninth International Conference on Learning Representations, SimDL Workshop* (2021) [2105.03418].

[16] D. Boyda et al., *Applications of Machine Learning to Lattice Quantum Field Theory*, in *Snowmass 2021*, 2, 2022 [2202.05838].

[17] R. Abbott et al., *Gauge-equivariant flow models for sampling in lattice field theories with pseudofermions*, *Phys. Rev. D* **106** (2022) 074506 [2207.08945].

[18] C. Lehner and T. Wettig, *Gauge-equivariant neural networks as preconditioners in lattice QCD*, *Phys. Rev. D* **108** (2023) 034503 [2302.05419].

[19] C. Lehner and T. Wettig, *Gauge-equivariant pooling layers for preconditioners in lattice QCD*, 2304.10438.

[20] Y. Nagai and A. Tomiya, *Self-learning Monte Carlo with equivariant Transformer*, 2306.11527.

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

[22] ALPHA collaboration, *Critical slowing down and error analysis in lattice QCD simulations*, *Nucl. Phys. B* **845** (2011) 93 [1009.5228].

[23] G. Kanwar, *Flow-based sampling for lattice field theories*, in *40th International Symposium on Lattice Field Theory*, 1, 2024 [2401.01297].

[24] P.A. Boyle, *Advances in algorithms for solvers and gauge generation*, 2401.16620.

[25] M. Luscher, *Trivializing maps, the Wilson flow and the HMC algorithm*, *Commun. Math. Phys.* **293** (2010) 899 [0907.5491].

[26] C. Morningstar and M.J. Peardon, *Analytic smearing of SU(3) link variables in lattice QCD*, *Phys. Rev. D* **69** (2004) 054501 [hep-lat/0311018].

[27] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro et al., *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015.

[28] P. Ramachandran, B. Zoph and Q.V. Le, *Searching for activation functions*, 2017.

[29] X.-Y. Jin, ''Field transformation HMC with neural networks.'' https://github.com/nftqcd/nthmc.

[30] S. Necco, *Universality and scaling behavior of RG gauge actions*, *Nuclear Physics B* **683** (2004) 137.

[31] G. McGlynn and R.D. Mawhinney, *Diffusion of topological charge in lattice QCD simulations*, *Phys. Rev. D* **90** (2014) 074502 [1406.4551].

[32] I. Omelyan, I. Mryglod and R. Folk, *Symplectic analytically integrable decomposition algorithms: classification, derivation, and application to molecular dynamics, quantum and celestial mechanics simulations*, *Computer Physics Communications* **151** (2003) 272 .

[33] M. Lüscher, *Properties and uses of the Wilson flow in lattice QCD*, *JHEP* **08** (2010) 071 [1006.4518].