Accelerating Flow-Based Sampling for Large-N Gauge Theories

by

Michael S. Zhang

S.B. in Computer Science and Engineering and Mathematics
Massachusetts Institute of Technology (2023)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2024

Authored by:	Michael S. Zhang Department of Electrical Engineering and Computer Science January 19, 2024
Certified by:	Phiala E. Shanahan Associate Professor Thesis Supervisor
Accepted by:	Katrina LaCurts Chair, Master of Engineering Thesis Committee

Accelerating Flow-Based Sampling for Large-N Gauge Theories

by

Michael S. Zhang

Submitted to the Department of Electrical Engineering and Computer Science
on January 19, 2024, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Due to its consistency with numerous experimental observations, the Standard Model of particle physics is widely accepted as the best known formulation of elementary particles and their interactions. However, making experimental predictions using the Standard Model involves mathematical and computational challenges due to its complexity. Quantum chromodynamics (QCD), which can be described as an SU(3) gauge theory due to the 3 quark colors and 8 gluon types, is one sector of the Standard Model for which computing solutions is especially challenging. A natural theoretical generalization of QCD is the class of all SU(N) gauge theories; these theories also provide a method for some QCD computations in the $N \to \infty$ limit. To study these theories numerically, approximations are calculated from configuration samples due to the mathematical complexity and lack of analytical solutions.

In this thesis, we explore asymptotically efficient flow-based sampling algorithms for the twisted Eguchi-Kawai (TEK) model, a method for analyzing large-N QCD numerically. We introduce an original architecture based on SU(2) matrix multiplication that allows for efficient Jacobian computation. In addition, we explore the possibility of transfer learning with respect to the number of colors N and demonstrate that a model trained quickly on the SU(N) setting also provides useful information in SU(N'), $N' > N$ cases.

Thesis Supervisor: Phiala E. Shanahan
Title: Associate Professor

Acknowledgments

I would first like to thank my mentor, Fernando Romero-López for all his help and support throughout my thesis process. He has provided me with numerous ideas whenever I felt like I was at an impasse, and his guidance has helped me grow tremendously as a researcher. I am also very grateful for his patience and understanding when explaining difficult concepts to me, especially early in the research process.

I would also like to express my gratitude to Phiala Shanahan, my faculty supervisor, for letting me join her lab and conduct interesting research despite my initially rudimentary physics knowledge. In addition, I would also like to thank other members of her lab – Ryan Abbott, Denis Boyda, Dan Hackett, and Julian Urban – for their contributions to my research progress as well.

Finally, I would like to thank my friends for all the great memories at MIT and my parents for their unconditional support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Standard Model of particle physics is the widely accepted formulation of elementary particles and their interactions through the electromagnetic, weak, and strong forces [12, 25, 22, 24]. Nevertheless, making real-world predictions using the Standard Model presents numerous mathematical and computational challenges due to its complexity. To remedy these issues, perturbative approaches have been used to compute approximate solutions for the electroweak sector, such as in [17]. Similar approaches can also be applied to quantum chromodynamics (QCD), the theory governing the strong interaction [18], but they only yield accurate results under high-energy conditions, such as those found in the Large Hadron Collider. As a result, alternative methods are necessary to achieve accurate predictions for QCD computations regarding typical hadron physics.

One such alternative approach is lattice QCD, a non-perturbative approach where the gluon fields of QCD are discretized onto a lattice [26]. In particular, the expected value of observables can be expressed as path integrals in QCD, and the lattice discretization provides a way to compute these integrals. As the lattice spacing decreases, the continuous-spacetime limit of QCD is recovered, so studies are conducted by performing calculations on increasingly fine lattices and extrapolating to the continuous zero-spacing limit. To perform these calculations, it is necessary to sample configurations ϕ of quantum fields defined on a spacetime lattice according to a specified density

$$p(\phi) = \frac{1}{Z} e^{-S(\phi)}, \tag{1.1}$$

where S is the action – a function dictated by the equations of QCD. However, because the partition function Z is intractable, directly sampling from this distribution is not possible. Instead, approaches involving Markov chain Monte Carlo (MCMC) are often used [19]. Simply applying MCMC faces other challenges still; for example, critical slowing-down and topological freezing may cause MCMC autocorrelation times to be large [23]. In order to remedy these issues, one approach that has been developed involves using flow-based generative models in addition to MCMC to improve sampling efficiency at the cost of some model-training time [3, 1].

While QCD is an SU(3) gauge theory due to the 3 colors of quarks and 8 independent types of gluons, a natural theoretical extension is the class of SU(N) gauge theories for large N. In fact, this actually presents a method for QCD, since computations can be done in the $N \to \infty$ limit and subsequently adapted to the $N = 3$ real-world setting through $O(\frac{1}{N})$ corrections, such as in [7]. The $N \to \infty$ limit also allows for a reduction in the spacetime degrees of freedom; in particular, a model with a single spacetime site is sufficient in this limit [10]. In response to this discovery, the twisted Eguchi-Kawai (TEK) model was proposed [13], and it now stands as a primary method for analyzing the large-N single-site formulation of QCD numerically.

In this thesis, we explore the application of flow-based sampling algorithms for the TEK model. While flow models developed for lattice QCD may be adapted to the TEK model as well, these models are not designed with computational complexity with respect to N in mind, so computations are infeasible when N is large. The work of this thesis will primarily focus on remedying this issue: we contribute alternative flow-model architectures and training methods that scale efficiently with N, making them more applicable to large-N gauge theories.

Chapter 2

Background

In this chapter, we provide further background information on QCD and the large-N reduction with the TEK model. Furthermore, we introduce MCMC and flow-based generative models with an emphasis on their applications for computational QCD approaches.

2.1 Quantum chromodynamics (QCD)

QCD is the currently accepted theory describing the strong interactions between quarks mediated by gluons. It is a gauge theory with symmetry group SU(3) defined by the Lagrangian

$$\mathcal{L}_{\mathrm{QCD}} = \bar{\psi}_i(i\gamma^\mu(D_\mu)_{ij} - m\delta_{ij})\psi_j - \frac{1}{4}F^a_{\mu\nu}F^{\mu\nu}_a, \tag{2.1}$$

where $\psi_i(x)$ represents the quark fields, and

$$F^a_{\mu\nu} = \partial_\mu \mathcal{A}^a_\nu - \partial_\nu \mathcal{A}^a_\mu + g f^{abc}\mathcal{A}^b_\mu \mathcal{A}^c_\nu \tag{2.2}$$

is the gluon field strength tensor defined in terms of the gluon fields $\mathcal{A}^\mu_a(x)$ [14]. The quark fields are indexed $i = 1, 2, 3$ and together they transform according to the fundamental representation of SU(3). Additionally, the gluon fields are indexed $a = 1, 2, \ldots, 8$ with one field for each generator t_a of the Lie algebra $\mathfrak{su}(3)$, and

$\mu = 1, 2, 3, 4$ for one component in each spacetime dimension. In Equation (2.2), g denotes the coupling constant, and f^{abc} denotes the stucture constants of SU(3).

In particular, the Lagrangian is invariant under the local gauge transformations

$$\psi \to G\psi, \tag{2.3}$$

$$\mathcal{A}_\mu \to G\mathcal{A}_\mu G^\dagger + \frac{i}{g}G(\partial_\mu G^\dagger), \tag{2.4}$$

with $G(x) = e^{i\varepsilon_a(x)t_a}$ for real-valued $\varepsilon_a(x)$.

The QCD action is then simply given by

$$S_{\text{QCD}} = \int \mathrm{d}^4x\mathcal{L}_{\text{QCD}}, \tag{2.5}$$

where the integral is over the 4 spacetime dimensions.

Computations in QCD, using lattice QCD for instance, have been able to accurately describe many phenomena and particle properties consistent with experimental observation. For instance, the masses of hadrons such as the proton and pion have been accurately measured [16, 8], and these measurements align with values obtained from lattice QCD.

However, computational techniques for QCD is still an active area of research. There remains discrepancies between experimental measurements and theoretical predictions of key quantities in the Standard Model, and more accurate QCD calculations may provide clues toward resolving these discrepancies. For instance, the anomalous magnetic moment of the muon is one value for which theoretical predictions appear to disagree with experimental data [5]. As a result, improvement in computational techniques will lead to a more accurate characterization of the difference between theoretical and observed values, potentially paving the way for new discoveries regarding the muon.

2.2 Large-N reduction

As introduced in Chapter 1, the large-N reduction provides a simplification to large-N gauge theories by reducing spacetime to a single lattice site. Additionally, the TEK model provides a numerical framework for large-N, single-site computations through the Lagrangian

$$S_{\text{TEK}} = -\beta N \sum_{\mu \neq \nu} Z_{\mu\nu} \operatorname{Tr}(U_\mu U_\nu U_\mu^\dagger U_\nu^\dagger), \tag{2.6}$$

where β is the coupling constant and U_μ is the fundamental representation of the μ component of the gluon field $\mathcal{A}_\mu$ at the single site [13]. Conceptually, these U_μ are cyclic links with both endpoints at the single site, analogous to the lattice QCD links that connect two adjacent lattice sites. Furthermore, the "twist tensor"

$$Z_{\mu\nu} = \begin{cases} e^{\frac{2\pi i}{N}} & \mu < \nu \\ e^{\frac{-2\pi i}{N}} & \mu > \nu \end{cases} \tag{2.7}$$

ensures that $U(1)^4$ symmetry of the action is not broken at weak coupling, as would be the case if $Z_{\mu\nu} = 1$.

The correspondence between the QCD Lagrangian and TEK action lies in the fact that the last term of Equation (2.1) yields the right-hand side of Equation (2.6). Contributions from the first term of the QCD Lagrangian can be ignored in this formulation since the second term dominates as $N \to \infty$. Note that this also leads to neglecting the quark fields altogether, since they do not appear in $F^a_{\mu\nu}$.

Under this formulation, gauge transformations of the gluon field components are given by

$$U_\mu^{[G]} \leftarrow G U_\mu G^\dagger \quad \text{for all } \mu \tag{2.8}$$

for arbitrary $G \in \mathrm{SU}(3)$. Gauge invariance of the action through these transformations can also be easily verified:

$$\begin{aligned} S^{[G]}_{\text{TEK}} &= -\beta N \sum_{\mu\neq\nu} Z_{\mu\nu} \operatorname{Tr}(U^{[G]}_{\mu} U^{[G]}_{\nu} (U^{[G]}_{\mu})^{\dagger} (U^{[G]}_{\nu})^{\dagger}) && (2.9) \\ &= -\beta N \sum_{\mu\neq\nu} Z_{\mu\nu} \operatorname{Tr}((GU_{\mu}G^{\dagger})(GU_{\nu}G^{\dagger})(GU_{\mu}G^{\dagger})^{\dagger}(GU_{\nu}G^{\dagger})^{\dagger}) && (2.10) \\ &= -\beta N \sum_{\mu\neq\nu} Z_{\mu\nu} \operatorname{Tr}(GU_{\mu}U_{\nu}U^{\dagger}_{\mu}U^{\dagger}_{\nu}G^{\dagger}) && (2.11) \\ &= -\beta N \sum_{\mu\neq\nu} Z_{\mu\nu} \operatorname{Tr}(U_{\mu}U_{\nu}U^{\dagger}_{\mu}U^{\dagger}_{\nu}) && (2.12) \\ &= S_{\text{TEK}}. && (2.13) \end{aligned}$$

2.3 Numerical methods for QCD

This section will present background on numerical methods for computational QCD approaches. We are specifically interested in methods for the TEK model for the work presented in this thesis, but some methods developed for lattice QCD can be easily adapted to the TEK model as well.

2.3.1 Markov chain Monte Carlo (MCMC)

A key component of computational QCD methods is sampling from the action. However, it is not possible to directly sample from the probability distribution defined by the TEK action in Equation (2.6)

$$p(U) = \frac{1}{Z_{\text{TEK}}} e^{-S_{\text{TEK}}(U)} \qquad (2.14)$$

since the partition function

$$Z_{\text{TEK}} = \int \mathrm{d}U p(U) = \int \mathrm{d}U e^{-S_{\text{TEK}}(U)} \qquad (2.15)$$

is unknown and intractable.

As a result, alternative sampling methods such as MCMC are used. In MCMC, an irreducible and aperiodic Markov chain with stationary distribution p is defined, and samples are generated through Markov chain state updates.

One MCMC algorithm is the Metropolis-Hastings algorithm, which defines a procedure that converts any arbitrary irreducible and aperiodic Markov chain into one with stationary distribution p [15]. It does so by assigning an acceptance probability

$$\rho(s_i \to s_j) = \min\left\{\frac{p(s_j)}{p(s_i)} \cdot \frac{q(s_i|s_j)}{q(s_j|s_i)}, 1\right\} \tag{2.16}$$

between every two states s_i and s_j. Here, $q(\cdot|s_i)$ and $q(\cdot|s_j)$ denote the transition distributions at states s_i and s_j, respectively, of the original Markov chain. The algorithm then generates samples according to the rule below.

Algorithm 1 Metropolis-Hastings

```
generate initial state $x_0$
for $t$ in $1, 2, \dots, T$ do
    sample $\tilde{x}_{t+1} \sim q(\cdot|x_t)$                ▷ propose transition to $\tilde{x}_{t+1}$
    sample $u \sim \mathcal{U}(0, 1)$
    if $u \leq \rho(x_t \to \tilde{x}_{t+1})$ then    ▷ accept proposal with probability $\rho(x_t \to x_{t+1})$
        $x_{t+1} \leftarrow \tilde{x}_{t+1}$
    else
        $x_{t+1} \leftarrow x_t$
    end if
end for
```

As a result, the empirical distribution of samples $x_0, x_1, \dots, x_{T-1}$ generated by Metropolis-Hastings approaches the goal distribution p asymptotically. However, while the algorithm produces correct results theoretically, many approaches face challenges due to correlations between $\tilde{x}_{t+1}$ and x_t resulting from the sampling methodologies used. These issues include critical slowing down and topological freezing, both of which occur when the algorithm transitions to configurations that result in extremely high autocorrelations.

Current state-of-the-art MCMC sampling algorithms for both lattice QCD and the TEK model involve hybrid Monte Carlo, a specific type of Metropolis-Hastings that introduces a Hamiltonian and uses Hamiltonian dynamics to specify the evolution of Markov chain states [9]. However, this thesis will not explore methods involving hybrid Monte Carlo, although approaches involving both hybrid Monte Carlo and flow-based generative models have been explored [11].

2.3.2 Flow-based generative models

One idea to potentially decrease MCMC autocorrelations is to propose transitions from a single distribution q that is independent of the current state of the Markov chain. This way, we eliminate correlations that occur due to differing proposal distributions entirely. However, if $\rho(x_t \to \tilde{x}_{t+1}) \ll 1$ with high probability at some step t, x_t and x_{t+1} will be highly correlated since the probability that $x_{t+1} = x_t$ is large. To remedy this issue, we should select distributions q that are similar to p so that $\rho(x_t \to \tilde{x}_{t+1})$ is close to 1.

Directly selecting such a distribution q that can be sampled from is also challenging because the distribution p is quite complicated, especially for large values of N and β. Rather, we may define q to be the result of an invertible transformation Φ on an easy-to-sample distribution such as the SU(N) Haar measure μ_N. That is, instead of expressing the distribution q directly, we can define Φ and assert that q is the distribution such that $x = \Phi(y) \sim q$ when $y \sim \mu_N$.

Flow-based generative models, a class of machine learning models that specializes in transforming probability distributions, provides a flexible way to define Φ [21]. As a result, they provide a way to define such a distribution q for lattice QCD applications [3, 1]. Like in [3, 1], we will take $\Phi = f(\cdot; \Theta)$, where f is a flow-based generative model whose neural networks are parameterized by Θ. Furthermore, if

$$f = f_k \circ f_{k-1} \circ \cdots \circ f_1 \tag{2.17}$$

is composed entirely of invertible layers $f_1, f_2, \ldots, f_k$ parameterized by $\Theta_1, \Theta_2, \ldots, \Theta_k$, then we can sample from q and perform density evaluation by computing

$$x = f(y; \Theta), \tag{2.18}$$

$$q(x) = q(f(y; \Theta)) \tag{2.19}$$

$$= \mu_N(y) \left| \det \left[\frac{\mathrm{d}f^{-1}(y;\Theta)}{\mathrm{d}y} \right] \right| \tag{2.20}$$

$$= \mu_N(y) \left| \det \left[\frac{\mathrm{d}f(y;\Theta)}{\mathrm{d}y} \right] \right|^{-1} \tag{2.21}$$

$$= \mu_N(y) \prod_{i=1}^{k} \left| \det \left[\frac{\mathrm{d}f_i(y_{i-1};\Theta_i)}{\mathrm{d}y_{i-1}} \right] \right|^{-1}, \tag{2.22}$$

where $y_i = (f_i \circ f_{i-1} \circ \cdots \circ f_1)(y)$ and $y = f^{-1}(x;\Theta)$. For numerical stability reasons, the log-likelihood

$$\log q(x) = \log \mu_N(y) - \sum_{i=1}^{k} \log \left| \det \left[\frac{\mathrm{d}f_i(y_{i-1};\Theta_i)}{\mathrm{d}y_{i-1}} \right] \right| \tag{2.23}$$

is typically evaluated instead.

Note that such a model is sufficient to run the Metropolis-Hastings algorithm with a state-independent proposal distribution q because the values

$$\rho(x_t \to \tilde{x}_{t+1}) = \min \left\{ \frac{p(\tilde{x}_{t+1})}{p(x_t)} \cdot \frac{q(x_t|\tilde{x}_{t+1})}{q(\tilde{x}_{t+1}|x_t)}, 1 \right\} \tag{2.24}$$

$$= \min \left\{ \frac{e^{-S_{\mathrm{TEK}}(\tilde{x}_{t+1})}}{e^{-S_{\mathrm{TEK}}(x_t)}} \cdot \frac{q(x_t)}{q(\tilde{x}_{t+1})}, 1 \right\} \tag{2.25}$$

can be computed.

To train such a model, an empirical estimate of the Kullback-Leibler divergence (KL-divergence), a measure of distance between probability distributions, is typically used as the loss function [21]. Additionally, due to scarcity of data sampled from p for QCD applications, existing computational methods choose to use the backwards KL-divergence

$$D_{\mathrm{KL}}(q||p) = \mathbb{E}_q \left[\frac{\log q(x)}{\log p(x)} \right] \tag{2.26}$$

$$= \mathbb{E}_q[\log q(x)] - \mathbb{E}_q[\log p(x)] \tag{2.27}$$

$$= \mathbb{E}_q[\log q(x)] + \mathbb{E}_q[S_{\mathrm{TEK}}(x)] + \log Z_{\mathrm{TEK}} \tag{2.28}$$

instead [3, 1]. This allows for an easy self-training scheme where training samples are trivially generated from the prior distribution μ_N. While the final $\log Z_{\text{TEK}}$ term is intractable, it only represents a constant shift, so dropping this term does not affect model training gradients. As a result, given a training set $\mathbf{y} = \{y^{(1)}, y^{(2)}, \ldots, y^{(n)}\}$ sampled from μ_N, the loss can be defined as

$$\mathcal{L}(\Theta; \mathbf{y}) = \frac{1}{n} \sum_{i=1}^{n} q(x^{(i)}; \Theta)(\log q(x^{(i)}; \Theta) + S_{\text{TEK}}(x^{(i)})), \tag{2.29}$$

where $x^{(i)} = f(y^{(i)}, \Theta)$.

Furthermore, flow-based generative models, and machine-learning methods in general, have the advantage that model evaluation is relatively cheap computationally even though training may be difficult. Evaluation of large language models such as GPT-3 has already been proven to efficient and effective despite the large training cost [6]. Similarly, while the training of flow-based generative models for the TEK model may be much more expensive than existing hybrid Monte Carlo methods, they can provide greater sampling efficiency once training has been completed.

Equation (2.23) also provides insight on desirable properties for the model layers $f_1, f_2, \ldots, f_k$. Notably, model efficiency necessitates efficient computation of both the updated sample

$$y_i = f_i(y_{i-1}; \Theta_i) \tag{2.30}$$

and the log-Jacobian

$$\log |\mathbf{J}_i(y_{i-1})| = \log \left| \det \left[\frac{\mathrm{d} f_i(y_{i-1}; \Theta_i)}{\mathrm{d} y_{i-1}} \right] \right|. \tag{2.31}$$

In this thesis, we focus on designing model layers that meet these criteria.

Chapter 3

Methods

We now present flow-based generative model architectures for learning distributions mirroring the TEK action. In particular, we consider the formulation where the single-site model is represented by an element of $\mathrm{SU}(N)^d$, where there is one link represented by an $\mathrm{SU}(N)$ matrix in each of the d dimensions. While typically $d = 4$ spacetime dimensions, the architectures presented in the section allow for generalization to arbitrary d as well.

The first two sections detail existing QCD methods with downsides when applied to the large-N setting, but nevertheless provide inspiration for architectures of interest. The final two sections will present original work on designing efficient large-N layers for the TEK model.

3.1 Potential-based layers

One flow-based generative model architecture for lattice QCD involves using the gradient of a potential function $\phi(\cdot;\theta)$ computed by a neural network to specify the flow layers f_i in Equation (2.17) [2]. Specifically, each layer defines a function that updates one link U_μ using the rule

$$U'_\mu \leftarrow f(U;\theta)_\mu = e^{iT_a \partial_{a,\mu}\phi(U;\theta)} U_\mu, \tag{3.1}$$

$$U'_\nu \leftarrow f(U;\theta)_\nu = U_\nu, \quad \nu \neq \mu. \tag{3.2}$$

Here,

$$\mathbf{T} = (T_1, T_2, \dots, T_{N^2-1}) \tag{3.3}$$

is the set of Hermitian generators for the (N^2-1)-dimensional lie algebra $\mathfrak{su}(N)$ normalized such that

$$\mathrm{Tr}(T_k T_l) = \mathrm{Tr}(T_k^\dagger T_l) = \delta_{kl}, \tag{3.4}$$

where δ_{kl} is the Kronecker delta. In addition, $\partial_{a,\mu}\phi(U;\theta)$ refers to the partial covariant derivative of $\phi(U,\theta)$ along T_a for the link U_μ.

This model also takes advantage of the gauge invariance of the TEK action under the transformations given in Equation (2.8). In particular, if we select a gauge-invariant potential function $\phi(\cdot,\theta)$, then the resulting transformation is gauge equivariant; that is,

$$\begin{aligned}(U_\mu^{[G]})' &= e^{iT_a\partial_{a,\mu}\phi(U^{[G]};\theta)}(GU_\mu G^\dagger) &(3.5)\\ &= e^{G(iT_a\partial_{a,\mu}\phi(U,\theta))G^\dagger}(GU_\mu G^\dagger) &(3.6)\\ &= G(e^{iT_a\partial_{a,\mu}\phi(U;\theta)}U_\mu)G^\dagger &(3.7)\\ &= (U'_\mu)^{[G]}. &(3.8)\end{aligned}$$

As a result, we impose the gauge symmetry $p((U')^{[G]}) = p(U')$ because for $U' = f(U;\theta) \sim q$ when $U \sim \mu_N$,

$$\frac{q((U')^{[G]})}{q(U')} = \frac{\mu_N(U^{[G]})}{\mu_N(U)} = 1. \tag{3.9}$$

However, due to the $\Omega(N^4)$ gradient computations necessary in evaluating the covariant derivative, calculating the Jacobian of the transformation, and performing parameter updates when training, these potential-based layers do not scale efficiently with N. So, while this architecture is expressive and accurate if N is small, it is not appropriate for the TEK model.

3.2 Spectral-flow layers

To remedy the computational complexity of the potential-based layers, spectral-flow layers provide a more efficient but still expressive architecture for QCD [3, 2]. More specifically, these layers update the eigenvalues of one link U_μ in a way such that the Jacobian matrix is almost triangular, allowing for quick determinant computation.

Concretely, each layer maintains a neural network $\phi(\cdot;\theta)$ and computes updates as follows:

1. Diagonalize $U_\mu = V_\mu L_\mu V_\mu^\dagger$, enforcing a canonical ordering on the eigenvalues (i.e. enforce a unique value of of L_μ),
2. Compute $D_\nu = (V_\mu^\dagger U_\nu V_\mu) \odot (V_\mu^\dagger U_\nu^\dagger V_\mu)^T$ for all $\nu \neq \mu$,
3. Compute $t_\nu^{(k)} = \mathrm{Tr}(U_\nu^k)$ for $k = 1, 2, \ldots, N$ for all $\nu \neq \mu$,
4. Consolidate either the odd-indexed or the even-indexed eigenvalues in L_μ; that is, let $P_\mu = (L_\mu[1,1], L_\mu[3,3], \ldots)$ or $P_\mu = (L_\mu[2,2], L_\mu[4,4], \ldots)$,
5. Use the output of $\phi((D_\nu, t_\nu^{(k)}, P_\mu); \theta)$ to define a monotonically increasing spline $S : [0,1] \to [0,1]$ with $S(0) = 0$ and $S(1) = 1$,
6. Compute $L'_\mu = \exp(2i\pi \cdot S(\frac{\log L_\mu}{2i\pi}))$, where we take the branch of the complex logarithm yielding values in the range $[0, 2\pi)$, and update $U'_\mu = V_\mu L'_\mu V_\mu^\dagger$ accordingly.

With many of these layers composed, we can update every dimension μ by considering both parities for P_μ. Furthermore, observe that this update also maintains gauge equivariance, since all inputs to ϕ are gauge invariant and the eigenvectors remain constant. In particular, under the gauge update given by Equation (2.8) and proper canonicalization,

$$V_\mu^{[G]} \leftarrow G V_\mu C \tag{3.10}$$

for some matrix

$$C = \begin{pmatrix} e^{i\varphi_1} & 0 & 0 \\ 0 & e^{i\varphi_2} & 0 \\ 0 & 0 & e^{i\varphi_3} \end{pmatrix} \tag{3.11}$$

with $\varphi_1 + \varphi_2 + \varphi_3 = 0$ due to freedom in selecting eigenvector phases during diagonalization. Although the neural network inputs D_ν are not used in [3], they are nevertheless gauge invariant:

$$\begin{aligned}
D_\nu^{[G]} &= ((V_\mu^{[G]})^\dagger U_\nu^{[G]} V_\mu^{[G]}) \odot ((V_\mu^{[G]})^\dagger (U_\nu^{[G]})^\dagger V_\mu^{[G]})^T && (3.12)\\
&= ((GV_\mu C)^\dagger (GU_\nu G^\dagger)(GV_\mu C)) \odot ((GV_\mu C)^\dagger (GU_\nu G^\dagger)^\dagger (GV_\mu C))^T && (3.13)\\
&= (C^\dagger V_\mu^\dagger U_\nu V_\mu C) \odot (C^\dagger V_\nu^\dagger U_\nu^\dagger V_\mu C)^T && (3.14)\\
&= (M \odot (V_\mu^\dagger U_\nu V_\nu)) \odot (M^T \odot (V_\mu^\dagger U_\nu^\dagger V_\nu)^T) && (3.15)\\
&= (V_\mu^\dagger U_\nu V_\mu) \odot (V_\nu^\dagger U_\nu^\dagger V_\nu)^T && (3.16)\\
&= D_\nu, && (3.17)
\end{aligned}$$

where

$$M = \begin{pmatrix} 1 & e^{i(\varphi_2-\varphi_1)} & e^{i(\varphi_3-\varphi_1)} \\ e^{i(\varphi_1-\varphi_2)} & 1 & e^{i(\varphi_3-\varphi_2)} \\ e^{i(\varphi_1-\varphi_3)} & e^{i(\varphi_2-\varphi_3)} & 1 \end{pmatrix}. \tag{3.18}$$

Likewise, the values of $t_\nu^{(k)}$ and P_μ are also gauge invariant since traces and eigenvalues do not change under matrix conjugation.

However, these spectral-flow layers have a key drawback due to the fact that eigenvectors are not updated. While this maintains gauge equivariance like the potential-based layers, it also enforces that under the transform

$$U_\mu^{[G_\mu]} \leftarrow G_\mu U_\mu G_\mu^\dagger \quad \text{for all } \mu, \tag{3.19}$$

where G_μ may be different for different dimensions μ,

$$\frac{q((U')^{[G_\mu]})}{q(U')} = \frac{\mu_N(U^{[G_\mu]})}{\mu_N(U)} = 1. \tag{3.20}$$

However, this is not true in the distribution p defined by the TEK action as specified in Equation (2.6), so the expressivity of the spectral flow is limited with respect to learning p.

3.3 SU(2)-block layers

In this section, we present a flow architecture based on updating two rows of one link U_μ at a time through left-multiplication by an SU(2)-block matrix parameterized by U; that is, an $N \times N$ matrix of the form

$$U'_\mu \leftarrow f(U;\theta) = A(U;\theta)U_\mu, \tag{3.21}$$

$$A(U;\theta) = \begin{pmatrix} 1 & 0 & \dots & 0 & \dots & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 & \dots & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \phi_{11}(U;\theta) & \dots & \phi_{12}(U;\theta) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \phi_{21}(U;\theta) & \dots & \phi_{22}(U;\theta) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & \dots & 0 & \dots & 1 \end{pmatrix} \tag{3.22}$$

where

$$\phi(U;\theta) = \begin{pmatrix} \phi_{11}(U;\theta) & \phi_{12}(U;\theta) \\ \phi_{21}(U;\theta) & \phi_{22}(U;\theta) \end{pmatrix} \in \mathrm{SU}(2). \tag{3.23}$$

Note that $\phi(U;\theta) \in \mathrm{SU}(2)$ implies $A(U;\theta) \in \mathrm{SU}(N)$, so the updated link U'_μ also remains in $\mathrm{SU}(N)$. Furthermore, $\phi(U;\theta)$ is parameterized by a neural network so that arbitrary functions $\phi : \mathrm{SU}(N)^d \to \mathrm{SU}(2)$ can be learned.

The primary benefit of this update is a fast Jacobian computation; while each individual layer is not adequately expressive due to only updating two rows, the efficient Jacobian computation means that many of these layers can be composed to form an expressive deep model. To further increase model expressivity, the spectral-flow layers discussed in Section 3.2 may also be composed with these SU(2)-block layers.

3.3.1 Efficient Jacobian computation

The design of this SU(2)-block update mechanism allows for numerous simplifications in the Jacobian computation. Let $\mathbf{T}$ be as defined in Equations (3.3) and (3.4). Additionally, suppose that $\mathbf{v}_\mu$ and $\mathbf{v}'_\mu$ be the representations of U_μ and U'_μ in this basis; that is,

$$U_\mu = e^{i(\mathbf{v}_\mu)_a T_a}, \tag{3.24}$$

$$U'_\mu = e^{i(\mathbf{v}'_\mu)_a T_a}. \tag{3.25}$$

Then, the Jacobian $\mathbf{J}$ of the transform $U \to U'$ is given by

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \mathbf{v}'_1}{\partial \mathbf{v}_1} & \frac{\partial \mathbf{v}'_1}{\partial \mathbf{v}_2} & \cdots & \frac{\partial \mathbf{v}'_1}{\partial \mathbf{v}_d} \\ \frac{\partial \mathbf{v}'_2}{\partial \mathbf{v}_1} & \frac{\partial \mathbf{v}'_2}{\partial \mathbf{v}_2} & \cdots & \frac{\partial \mathbf{v}'_2}{\partial \mathbf{v}_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{v}'_d}{\partial \mathbf{v}_1} & \frac{\partial \mathbf{v}'_d}{\partial \mathbf{v}_2} & \cdots & \frac{\partial \mathbf{v}'_d}{\partial \mathbf{v}_d} \end{bmatrix}, \tag{3.26}$$

where each entry in Equation (3.26) is a matrix block

$$\frac{\partial \mathbf{v}'_\sigma}{\partial \mathbf{v}_\rho} = \begin{pmatrix} \frac{\partial (\mathbf{v}'_\sigma)_1}{\partial (\mathbf{v}_\rho)_1} & \frac{\partial (\mathbf{v}'_\sigma)_1}{\partial (\mathbf{v}_\rho)_2} & \cdots & \frac{\partial (\mathbf{v}'_\sigma)_1}{\partial (\mathbf{v}_\rho)_{N^2-1}} \\ \frac{\partial (\mathbf{v}'_\sigma)_2}{\partial (\mathbf{v}_\rho)_1} & \frac{\partial (\mathbf{v}'_\sigma)_2}{\partial (\mathbf{v}_\rho)_2} & \cdots & \frac{\partial (\mathbf{v}'_\sigma)_2}{\partial (\mathbf{v}_\rho)_{N^2-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial (\mathbf{v}'_\sigma)_{N^2-1}}{\partial (\mathbf{v}_\rho)_1} & \frac{\partial (\mathbf{v}'_\sigma)_{N^2-1}}{\partial (\mathbf{v}_\rho)_2} & \cdots & \frac{\partial (\mathbf{v}'_\sigma)_{N^2-1}}{\partial (\mathbf{v}_\rho)_{N^2-1}} \end{pmatrix}. \tag{3.27}$$

However, observe that for $\nu \neq \mu$,

$$\frac{\partial \mathbf{v}'_\nu}{\partial \mathbf{v}_\rho} = \begin{cases} I & \nu = \rho \\ 0 & \nu \neq \rho \end{cases} \tag{3.28}$$

since U_μ is the only updated link. As a result, $\mathbf{J}$ can be rearranged into a block-triangular matrix, which gives

$$|\det \mathbf{J}| = |\det \mathbf{J}_\mu| = \left|\det \frac{\partial \mathbf{v}'_\mu}{\partial \mathbf{v}_\mu}\right|. \tag{3.29}$$

To simplify the computation of $|\det \mathbf{J}|$, we introduce a new vector $\mathbf{p}_\mu$ such that

$$\hat{U}_\mu(\mathbf{p}_\mu) = E_\mu U_\mu = e^{i(\mathbf{p}_\mu)_a T_a} U_\mu, \tag{3.30}$$

$$\hat{U}'_\mu(\mathbf{p}_\mu) = f(\hat{U}_\mu(\mathbf{p}_\mu), U_\nu; \theta). \tag{3.31}$$

The following proposition, which is similar to Equation (26) in [2], then gives a way to compute $|\det \mathbf{J}|$:

Proposition 3.1. *Let $\hat{\mathbf{U}}'_\mu$ denote the elements of $\hat{U}'_\mu$ as an N^2-dimensional vector such that*

$$\hat{\mathbf{U}}'_\mu = ((U_\mu)_{1,1}, (U_\mu)_{1,2}, \ldots, (U_\mu)_{N^2,N^2}). \tag{3.32}$$

In addition, let $\sigma_1, \sigma_2, \ldots, \sigma_{N^2-1}$ be the singular values of $\frac{\partial \hat{\mathbf{U}}'_\mu}{\partial \mathbf{p}_\mu}$ evaluated at $\mathbf{p}_\mu = \mathbf{0}$. Then,

$$\left|\det \frac{\partial \mathbf{v}'_\mu}{\partial \mathbf{v}_\mu}\right| = \prod_{i=1}^{N^2-1} \sigma_i. \tag{3.33}$$

To evaluate the above expression $\frac{\partial \hat{\mathbf{U}}'_\mu}{\partial \mathbf{p}_\mu}$, we can use the product rule to obtain

$$\left.\frac{\partial (\hat{U}'_\mu)_{pq}}{\partial (\mathbf{p}_\mu)_r}\right|_{\mathbf{p}_\mu = \mathbf{0}} = \left.\frac{\partial (A(\hat{U}_\mu; \theta) E_\mu U_\mu)_{pq}}{\partial (\mathbf{p}_\mu)_r}\right|_{\mathbf{p}_\mu = \mathbf{0}} \tag{3.34}$$

$$= \frac{\partial A(\hat{U}_\mu;\theta)_{ps}}{\partial(\mathbf{p}_\mu)_r}(E_\mu)_{st}(U_\mu)_{tq}\bigg|_{\mathbf{p}_\mu=\mathbf{0}} \tag{3.35}$$

$$+ A(\hat{U}_\mu;\theta)_{ps}\frac{\partial(E_\mu)_{st}}{\partial(\mathbf{p}_\mu)_r}(U_\mu)_{tq}\bigg|_{\mathbf{p}_\mu=\mathbf{0}} \tag{3.36}$$

$$+ A(\hat{U}_\mu;\theta)_{ps}(E_\mu)_{st}\frac{\partial(U_\mu)_{tq}}{\partial(\mathbf{p}_\mu)_r}\bigg|_{\mathbf{p}_\mu=\mathbf{0}} \tag{3.37}$$

$$= \frac{\partial A(\hat{U}_\mu;\theta)_{ps}}{\partial(\mathbf{p}_\mu)_r}\bigg|_{\mathbf{p}_\mu=\mathbf{0}}(U_\mu)_{sq} + iA(U_\mu;\theta)_{ps}(T_r)_{st}(U_\mu)_{tq}. \tag{3.38}$$

Defining F such that

$$F_{wr} = F_{(p,q)r} = \frac{\partial(\hat{U}'_\mu)_{pq}}{\partial(\mathbf{p}_\mu)_r}\bigg|_{\mathbf{p}_\mu=\mathbf{0}} \tag{3.39}$$

with $w = N^2(p-1)+q$, we have that the product of singular values of F is exactly the desired value $|\det \mathbf{J}|$.

Additionally, we can take advantage of unitary matrix multiplication for further simplification. Suppose we define F' such that

$$F'_{wr} = F'_{(p,q)r} = \frac{\partial(\hat{U}'_\mu)_{ps}}{\partial(\mathbf{p}_\mu)_r}\bigg|_{\mathbf{p}_\mu=\mathbf{0}} M_{sq} \tag{3.40}$$

for some matrix $M \in \mathrm{SU}(N)$. Then, the product of singular values of F is equal to the product of singular values of F' because

$$F' = F \cdot \begin{bmatrix} M & 0 & \dots & 0 \\ 0 & M & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & M \end{bmatrix} \tag{3.41}$$

and multiplication by a unitary matrix does not change singular values. This similarly holds in the left-multiplication case, so it suffices to compute the product of singular values of G, where

$$G_{w'r} = G_{(p,q)r} = -i \cdot (A(U_\mu;\theta)^\dagger)_{ps} \frac{\partial(\hat{U}'_\mu)_{st}}{\partial(\mathbf{p}_\mu)_r}\bigg|_{\mathbf{p}_\mu=\mathbf{0}} (U_\mu^\dagger)_{tq} \tag{3.42}$$

$$= -i \cdot (A(U_\mu;\theta)^\dagger)_{ps} \frac{\partial A(\hat{U}_\mu;\theta)_{sq}}{\partial(\mathbf{p}_\mu)_r} + (T_r)_{pq}. \tag{3.43}$$

Without loss of generality, suppose that rows 1 and 2 are being updated in U_μ. Since reordering rows also does not change singular values, we may alter the row-mapping relation $(p,q) \mapsto w'$ so that the first 4 rows correspond to $(p,q) = (1,1),(2,2),(1,2),(2,1)$ in that order.

Because an arbitrary basis $\mathbf{T}$ can be selected for this Jacobian computation, we select the one that makes computation easiest, namely the generalization

$$T_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & -1 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix} \tag{3.44}$$

$$T_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix} \tag{3.45}$$

$$T_3 = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 & -i & 0 & \dots & 0 \\ i & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix} \tag{3.46}$$

$$\vdots \tag{3.47}$$

of the Pauli and Gell-Mann matrices.

Additionally, because $A(\cdot, \theta)$ is of the SU(2)-block form,

$$D_{pqr} = -i \cdot (A(U_\mu;\theta)^\dagger)_{ps} \frac{\partial A(\hat{U}_\mu;\theta)_{sq}}{\partial(\mathbf{p}_\mu)_r} \tag{3.48}$$

is only non-zero when $p, q \in \{1, 2\}$. In addition, it is also traceless and Hermitian since

$$D_{pqr} - (D^\dagger)_{pqr} = -i \cdot (A(U_\mu;\theta)^\dagger)_{ps} \frac{\partial A(\hat{U}_\mu;\theta)_{sq}}{\partial(\mathbf{p}_\mu)_r} - i \cdot \frac{\partial A((\hat{U}_\mu;\theta)_{ps})^\dagger}{\partial(\mathbf{p}_\mu)_r} (A(U_\mu;\theta))_{sq} \tag{3.49}$$

$$= -i \cdot \frac{\partial(A(\hat{U}_\mu;\theta)^\dagger A(\hat{U}_\mu;\theta))}{\partial(\mathbf{p}_\mu)_r} \tag{3.50}$$

$$= 0. \tag{3.51}$$

As a result, G is of the block-matrix form

$$G = \begin{bmatrix} H & K + L \\ 0 & M \end{bmatrix}, \tag{3.52}$$

where H $[4 \times 3]$ and K $[4 \times (N^2 - 4)]$ correspond to the flattened entries of D and the concatenation of L $[4 \times (N^2 - 4)]$ and M $[(N^2 - 4) \times (N^2 - 4)]$ correspond to the flattened entries of $\mathbf{T}$. Of particular note is the fact that the columns of H and K are linear combinations of T_1, T_2, T_3, and the columns of the concatenation of L and M are the remaining $N^2 - 4$ generators $T_4, T_5 \ldots, T_{N^2-1}$. We take advantage of this fact by using generator orthogonality in the proposition below.

Proposition 3.2. *Let the singular values of G be $\gamma_1, \gamma_2, \ldots, \gamma_{N^2-1}$, and let the singular values of H be η_1, η_2, η_3. Then,*

$$\prod_{i=1}^{N^2-1} \gamma_i = \eta_1 \cdot \eta_2 \cdot \eta_3. \tag{3.53}$$

Proof. Observe that

$$\prod_{i=1}^{N^2-1} \gamma_i = (\det(G^\dagger G))^{\frac{1}{2}} \tag{3.54}$$

$$= \left(\det \begin{bmatrix} H^\dagger H & H^\dagger K + H^\dagger L \\ K^\dagger H + L^\dagger H & K^\dagger K + L^\dagger L + K^\dagger L + L^\dagger K + M^\dagger M \end{bmatrix}\right)^{\frac{1}{2}} \tag{3.55}$$

$$= \left(\det \begin{bmatrix} H^\dagger H & H^\dagger K \\ K^\dagger H & K^\dagger K + I \end{bmatrix}\right)^{\frac{1}{2}} \tag{3.56}$$

since $\begin{bmatrix} H & 0 \end{bmatrix}^T$, $\begin{bmatrix} K & 0 \end{bmatrix}^T$ are orthogonal to $\begin{bmatrix} L & M \end{bmatrix}^T$ and $L^\dagger L + M^\dagger M = I$ due to the relation from Equation (3.4). Then, because columns of H and K are of the form

$$\begin{pmatrix} w & -w & x+iy & x-iy \end{pmatrix}^T, \tag{3.57}$$

we have that by Schur's formula

$$\det \begin{bmatrix} H^\dagger H & H^\dagger K \\ K^\dagger H & K^\dagger K + I \end{bmatrix} = \det(H^\dagger H)\det(K^\dagger K + I - K^\dagger H(H^\dagger H)^{-1}H^\dagger K) \tag{3.58}$$

$$= \det(H^\dagger H)\det(I) \tag{3.59}$$

$$= \det(H^\dagger H). \tag{3.60}$$

In particular, this follows because

$$\tilde{H} = H(H^\dagger H)^{-1}H^\dagger = \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} & 0 & 0 \\ -\frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{3.61}$$

and $K^\dagger \tilde{H} K = I$. As a result,

$$\prod_{i=1}^{N^2-1} \gamma_i = (\det(G^\dagger G))^{\frac{1}{2}} = (\det(H^\dagger H))^{\frac{1}{2}} = \eta_1 \cdot \eta_2 \cdot \eta_3, \tag{3.62}$$

as desired. □

Therefore, we can compute $|\log \mathbf{J}|$ by simply computing $G_{(1,1)1}, G_{(1,1)2} \dots, G_{(2,2)3}$ and finding the product of singular values. Notably, this only requires a constant number of partial derivaties of the form $\frac{\partial A(\hat{U}_\mu;\theta)_{sq}}{\partial \mathbf{p}_r}$, though the complexity of each partial derivative may depend on N.

3.3.2 Neural network design

There are also some considerations when defining the neural network $\phi(\cdot;\theta) : \mathrm{SU}(N)^d \to \mathrm{SU}(2)$. A natural output for the neural network is simply a 3-dimensional vector $\mathbf{g} = (g_1, g_2, g_3)$ so that we may define

$$\phi(U;\theta) = e^{ig_a T_a}. \tag{3.63}$$

In order to ensure that the transformation remains invertible, it is also helpful to impose a restriction on the magnitude of $\mathbf{g}$. To do so, we simply normalize the vector $\mathbf{g}$ by computing

$$\mathbf{g}' = c \cdot \frac{\mathbf{g}}{\sqrt{1+|\mathbf{g}|^2}}, \qquad \phi(U;\theta) = e^{ig'_a T_a} \tag{3.64}$$

for an appropriate constant c.

For the neural network inputs, we choose the loop values

$$P_{\mu\nu} = U_\mu U_\nu U_\mu^\dagger U_\nu^\dagger \tag{3.65}$$

for all $\nu \neq \mu$ instead of the link value directly. This performed much better emperically, possibly due to the fact that more accurate probability density updates may result from inputs found directly in the action of Equation (2.6).

3.3.3 Complexity optimizations

For efficiency considerations, we targeted a model complexity of $O(N^3)$ since this is the runtime of naïve matrix multiplication. Since computing the action for the MCMC step requires the multiplication of link matrices, any further improvement to model complexity will not result in a significant asymptotic improvement to sampling efficiency.

The transformation described above with loop values $P_{\mu\nu}$ as neural network inputs has complexity $O(N^2)$ due to partial derivative computation since

$$\frac{\partial A(\hat{U}_\mu;\theta)_{sq}}{\partial \mathbf{p}_r} = \frac{\partial A(\hat{U}_\mu;\theta)_{sq}}{\partial (\hat{P}_{\mu\nu})_{kl}} \cdot \frac{\partial (\hat{P}_{\mu\nu})_{kl}}{\partial (\hat{U}_\mu)_{ij}} \cdot \frac{\partial (\hat{U}_\mu)_{ij}}{\partial \mathbf{p}_r}. \tag{3.66}$$

So, if $\Theta(N^2)$ such layers were composed in order to do an SU(2)-block update for every pair of rows, the overall model complexity would be $O(N^4)$. However, it is also possible to update every matrix element using only an $\Theta(N)$ subset of row pairs, leading to an $O(N^3)$ model overall. Furthermore, neural network input computation must be done with some care since naïvely computing $P_{\mu\nu}$ would require $O(N^3)$ operations per layer. To avoid this cost, we simply compute

$$\Delta P_{\mu\nu} = P'_{\mu\nu} - P_{\mu\nu} \tag{3.67}$$

$$= ((A(U,\theta) - I)U_\mu)U_\nu((A(U,\theta) - I)U_\mu)^\dagger U_\nu^\dagger \tag{3.68}$$

at each layer to maintain the updated value of $P_{\mu\nu}$. This computation can be done in $O(N^2)$ due to the fact that $A(U;\theta) - I$ only has 4 non-zero elements.

A model complexity of $O(N^3)$ can also be achieved without decreasing the number of SU(2)-block updates from $O(N^2)$ if each layer requires only $O(N)$ time. This can be done by decreasing the number of neural network inputs to a constant number of loop elements as it correspondingly reduces the number of pairs k, l in Equation (3.66). Supposing that rows i and j are updated, the 4 elements $(P_{\mu\nu})_{ii}, (P_{\mu\nu})_{jj}, (P_{\mu\nu})_{ij}, (P_{\mu\nu})_{ji}$

are a natural choice for neural network inputs since the action depends on $\mathrm{Tr}(P_{\mu\nu})$ and

$$\begin{aligned}\mathrm{Tr}(P'_{\mu\nu}) - \mathrm{Tr}(P_{\mu\nu}) &= \mathrm{Tr}(((A(U,\theta) - I)U_\mu)U_\nu((A(U,\theta) - I)U_\mu)^\dagger U_\nu^\dagger) && (3.69)\\ &\approx \mathrm{Tr}((I - A(U,\theta))U_\mu U_\nu U_\mu^\dagger U_\nu^\dagger) && (3.70)\\ &= \mathrm{Tr}((I - A(U,\theta))P_{\mu\nu}) && (3.71)\\ &= (1-\phi_{11})(P_\mu)_{ii} + (1-\phi_{22})(P_\mu)_{jj} - \phi_{12}(P_\mu)_{ji} - \phi_{21}(P_\mu)_{ij}. && (3.72)\end{aligned}$$

Finally, updating $P_{\mu\nu}$ in $O(N^2)$ per layer is no longer sufficient, which is particularly concerning because copying $P_{\mu\nu}$ to maintain the computational graph already requires $O(N^2)$. However, we can alternatively update many pairs of rows in parallel since updating rows i and j of U_μ does not impact the values in rows $i' \neq i$ and $j' \neq j$. Furthermore, a group of parallel updates on disjoint rows can still be pushed to U_μ in $O(N^2)$. As a result, an updating with every pair of rows can still be performed in $O(N^3)$, by using $\Theta(n)$ of these $O(N^2)$ parallel updates.

Each of these optimizations – reducing the number of layers or reducing the number of neural network inputs – motivates a corresponding $O(N^3)$ model architecture. For clarity, we will call these the layer-SU(2) and input-SU(2) architectures:

- the layer-SU(2) architecture composes $O(N)$ SU(2)-block update layers f_k using all elements of $P_{\mu\nu}$ as inputs to the neural network ϕ_k,
- the input-SU(2) architecture composes $O(N^2)$ SU(2)-block update layers f_k using 4 elements of $P_{\mu\nu}$ as inputs to the neural network ϕ_k.

3.3.4 Gauge fixing

The design of the SU(2)-block layers also sacrifices the gauge equivariance present in the potential-based and spectral-flow layers. However, the issue can be remedied through gauge fixing when $d > 2$ by transforming to a canonical gauge before applying the update and transforming back to the original gauge after.

Selecting a gauge is somewhat tricky, however, due to the fact that matrix diagonalization is not unique. In particular, there are two sources of degrees of freedom in matrix diagonalization that need to be addressed in order to determine a unique canonical gauge – the ordering of eigenvalues and the phases of eigenvectors. The ordering of eigenvalues is easily addressed by defining a canonical ordering as done in the spectral-flow layers. Canonicalizing eigenvector phases is more challenging, but it can be done with the following proposition.

Proposition 3.3. *Suppose matrices $M_1, M_2 \in \mathrm{SU}(N)$ satisfy*

$$K_1 = V^\dagger M_1 V = D_1, \tag{3.73}$$

$$K_2 = V^\dagger M_2 V = W D_2 W^\dagger. \tag{3.74}$$

If D_1 and D_2 are diagonal matrices with entries canonically ordered, there exists $V, W \in \mathrm{SU}(N)$ satisfying

$$\mathrm{Im}(\log W_{11}) \in \left[0, \frac{2\pi}{N-2}\right), \tag{3.75}$$

$$W_{12}, W_{13}, \ldots, W_{1N}, W_{21}, W_{31}, \ldots, W_{N1} \in \mathbb{R}^{\geq 0}. \tag{3.76}$$

Furthermore, W is unique and K_1 and K_2 are gauge invariant.

Proof. First, we describe the construction of V and W. Let

$$M_1 = (L_1 C_1) D_1 (L_1 C_1)^\dagger, \tag{3.77}$$

$$M_1 = (L_2 C_2) D_2 (L_2 C_2)^\dagger, \tag{3.78}$$

where L_1, L_2 are eigenvector matrices and C_1, C_2 are arbitrary diagonal phase matrices illustrating the freedom of eigenvector phases. Then, $V = L_1 C_1$ for some C_1, and

$$W = V^\dagger (L_2 C_2) = C_1^\dagger L_1^\dagger L_2 C_2, \tag{3.79}$$

for some C_1, C_2. We will show that there exists a unique C_1, C_2, up to a constant factor in both, such that the desired relations hold.

To begin the construction of C_1, C_2, let

$$\tilde{W} = L_1^\dagger L_2. \tag{3.80}$$

We will first show that there exists $\tilde{C}_1, \tilde{C}_2$ such that $\tilde{C}_1^\dagger \tilde{W} \tilde{C}_2$ satisfies the condition in Equation (3.76).

Let

$$\begin{aligned} \tilde{C}_1 &= \operatorname{diag}(e^{i\varphi_1}, e^{i\varphi_2}, \dots, e^{i\varphi_N}), && (3.81)\\ \tilde{C}_2 &= \operatorname{diag}(e^{i\psi_1}, e^{i\psi_2}, \dots, e^{i\psi_N}). && (3.82)\end{aligned}$$

Then, $\tilde{W}_{pq} = e^{i(-\varphi_p + \psi_q)} W_{pq}$, so $\varphi_1, \varphi_2, \dots, \varphi_N, \psi_1, \psi_2, \dots, \psi_N$ must satisfy the system of equations

$$\begin{aligned} \varphi_1 - \psi_q &= \operatorname{Im}(\log W_{1q}) \quad \text{for all } 2 \le q \le N, && (3.83)\\ \varphi_p - \psi_1 &= \operatorname{Im}(\log W_{p1}) \quad \text{for all } 2 \le p \le N, && (3.84)\\ \sum_{p=1}^{N} \varphi_i &= 0, && (3.85)\\ \sum_{q=1}^{N} \psi_i &= 0. && (3.86)\end{aligned}$$

This is a system of $2N$ variables and $2N$ equations, so it has a unique solution; this solution can be used to create a matrix $\tilde{C}_1^\dagger \tilde{W} \tilde{C}_2$ that has positive real values along the top and left edges. To finish the construction, take

$$\begin{aligned} C_1 &= \tilde{C}_1 \cdot \operatorname{diag}\left(e^{\frac{N-1}{N(N-2)}2i\pi k}, e^{\frac{-1}{N(N-2)}2i\pi k}, \dots, e^{\frac{-1}{N(N-2)}2i\pi k}\right), && (3.87)\\ C_2 &= \tilde{C}_2 \cdot \operatorname{diag}\left(e^{\frac{-1}{N(N-2)}2i\pi k}, e^{\frac{N-1}{N(N-2)}2i\pi k}, \dots, e^{\frac{N-1}{N(N-2)}2i\pi k}\right), && (3.88)\end{aligned}$$

where k is the integer satisfying $k \le \frac{N-2}{2\pi} \operatorname{Im}(\log W_{11}) < k + 1$. As a result, $W = C_1^\dagger \tilde{W} C_2$ satisfies the constraint of Equation (3.75) while retaining the property specified by Equation (3.76).

Furthermore, this value of W is unique. Suppose that $W' = C_1' W C_2'$ also satisfies the constraints given by Equations (3.75) and (3.76), where

$$C_1' = \operatorname{diag}(e^{i\alpha_1}, e^{i\alpha_2}, \ldots, e^{i\alpha_N}), \tag{3.89}$$

$$C_2' = \operatorname{diag}(e^{i\beta_1}, e^{i\beta_2}, \ldots, e^{i\beta_N}). \tag{3.90}$$

Without loss of generality, suppose that all $\alpha_p, \beta_q \in [-\pi, \pi)$. Due to Equation (3.75), $\beta_1 \in [-\operatorname{Im}(\log W_{11}) + \alpha_1, \frac{2\pi}{N-2} - \operatorname{Im}(\log W_{11}) + \alpha_1)$, so $|\beta_1 - \alpha_1| < \frac{2\pi}{N-2}$. In addition, because $\alpha_p = \beta_1$ and $\beta_q = \alpha_1$ for $p, q \neq 1$,

$$\alpha_1 + (N-1)\beta_1 = \sum_{i=1}^{N} \alpha_i = 2\pi n_1, \tag{3.91}$$

$$(N-1)\alpha_1 + \beta_1 = \sum_{i=1}^{N} \beta_i = 2\pi n_2, \tag{3.92}$$

$$n_1, n_2 \in \mathbb{Z}. \tag{3.93}$$

Therefore, the difference $(N-2)(\beta_1 - \alpha_1)$ must be a multiple of 2π. Combining this with the fact that $|\beta_1 - \alpha_1| < \frac{2\pi}{N-2}$, gives $\alpha_1 = \beta_1$ as the only solution. This means that W' is necesssarily equal to W because both $C_1, C_2 = e^{i\alpha_1} I$, indicating that the value of W is unique.

Finally, under a gauge transformation $M_1^{[G]} = G M_1 G^\dagger$ and $M_2^{[G]} = G M_2 G^\dagger$, $D_1^{[G]} = D_1$ and $D_2^{[G]} = D_2$ since eigenvalues are gauge invariant. Furthermore, if V and W satisfy Equations (3.73) to (3.76) for M_1 and M_2, then $V^{[G]} = GV$ and $W^{[G]} = W$ satisfy Equations (3.73) to (3.76) for $M_1^{[G]}$ and $M_2^{[G]}$. So, W is also gauge invariant because $W^{[G]}$ is unique, which means that K_1 and K_2 are gauge invariant, as desired.

$\square$

As a result, transformations that update one link U_μ can be made gauge equivariant by computing V such that U_{ν_1} and U_{ν_2} with $\nu_1, \nu_2 \neq \mu$ are canonicalized according to Proposition 3.3. We then gauge-transform $U \to U^{[V^\dagger]}$, apply the update in the canonical gauge, and finally undo the gauge transform. While these gauge transformations have complexity $O(N^3)$, they do not hurt the overall model complexity because we only need to transform once for consecutive layers that update the same link U_μ.

3.4 Transferrable layers

Transfer learning is another prospect for increasing training efficiency. In particular, if we can design models such that parameters can be transferred between different values of N, then models trained quickly on small values of N can also be applied to larger values of N. The following subsections present two architectures that achieve this; one is based on the input-SU(2) layers from Section 3.3, and the other is based on Hermitian link projections with inspiration from [4].

3.4.1 SU(2)-block transferrable layers

In the input-SU(2) model, there are batches of $\Theta(N^2)$ consecutive layers updating different pairs of rows on a single link. The neural network $\phi(\cdot;\theta)$ for each of these layers is of constant size because they take 4 inputs and produce 3 outputs, leading to a total of $\Theta(N^2)$ model parameters. However, we can also share the neural network parameters among each batch of $\Theta(N^2)$ consecutive layers, yielding an architecture whose neural networks are independent of N. As a result, transfer learning may be applied because the same set of model parameters describes a valid transformation for all values of N in the same way that convolutional neural network layers are valid for all image sizes.

3.4.2 Projection-based transferrable layers

Another architecture that allows for transfer learning with respect to N consists of layers with the following update:

$$U'_\mu \leftarrow f(U;\theta)_\mu = e^{ic_a H_a} U_\mu, \tag{3.94}$$

$$U'_\nu \leftarrow f(U;\theta)_\nu = U_\nu, \quad \nu \neq \mu, \tag{3.95}$$

with $\mathbf{c} = (c_1, c_2, \ldots, c_k)$ a collection of trainable constants and $\mathbf{H} = (H_1, H_2, \ldots, H_k)$ a collection of gauge-equivariant traceless Hermitian matrices. Like the potential-based update defined in Section 3.1, a gauge-equivariant exponent leads the transformation being gauge-equivariant as well.

To construct these gauge-equivariant traceless Hermitian matrices, we simply consider the traceless Hermitian projection of general gauge-equivariant matrices with the projection operator

$$P(M) = M + M^\dagger - \frac{\mathrm{Tr}\,(M + M^\dagger)}{N} \cdot I. \tag{3.96}$$

For example, we may take

$$\mathbf{H} = \big(P(U_\mu), \ldots, P(U_\mu^2), \ldots, P(U_\mu U_\nu U_\mu^\dagger U_\nu^\dagger), \ldots\big)\,. \tag{3.97}$$

In particular, observe that the only trainable parameters in this layer are the constants in $\mathbf{c}$, so the number of parameters depends only on the number of traceless Hermitian features in $\mathbf{H}$. As a result, transfer learining is possible because models based on different values of N may be described by the same set of parameters.

Chapter 4

Results

In this section, we present results regarding the original methods discussed in Sections 3.3 and 3.4. We focus primarily on the main contribution of this thesis – the layer-SU(2) and input-SU(2) architectures motivated by Section 3.3.3. All models were implemented in Pytorch [20], and training was done on a single NVIDIA A100 GPU for each model.

4.1 SU(2)-block layer efficiency

In order to test how efficiency scales with N, training iteration time was recorded for 2-layer models trained using a batch size of 64, where "layer" here refers to the composition of all $\Theta(N)$ or $\Theta(N^2)$ row-pair updates for the SU(2)-block architectures. While these hyperparameters provide neither adequate expressivity nor optimized training dynamics, they suffice for comparing training times among different values of N. Figure 4-1 displays these times and provides a benchmark for the relative training times of the three compared models for N from 4 to 36.

Notably, the results in Figure 4-1 agree with the theoretical $O(N^3)$ complexity for the SU(2)-block architectures, but not the $\Omega(N^4)$ for the potential-based layers. This is due to the fact numerical evaluation of the Jacobian is not the limiting factor for the values of N tested; further examination reveals that Jacobian evaluation time per $\mathfrak{su}(N)$ generator is approximately constant for the values of N in Figure 4-1 instead

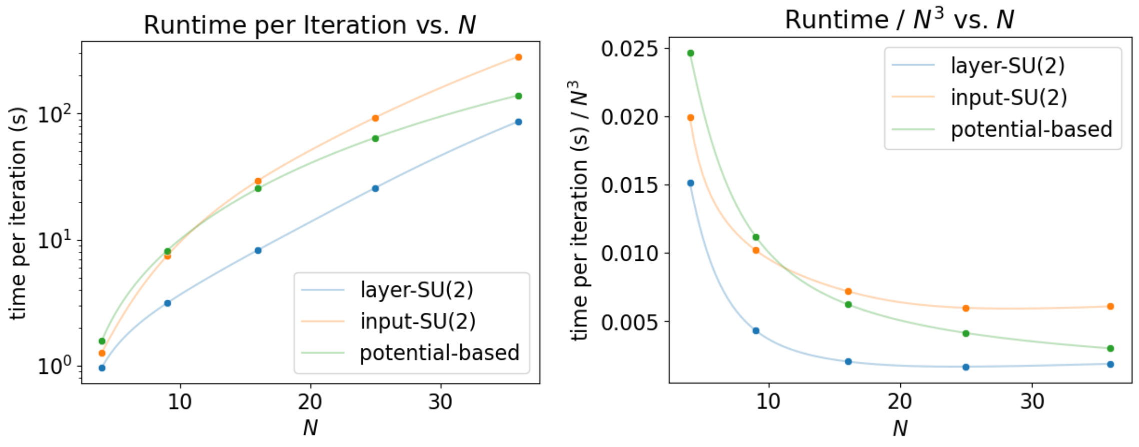

Figure 4-1: Runtime per gradient step of 2-layer layer-SU(2), input-SU(2), and potential-based models trained on a batch size of 64 in the two-dimensional $d = 2$ setting.

of the theoretical $\Omega(N^2)$. Additionally, it is also difficult to test significantly larger values of N for the potential-based layers due GPU memory limitations being unable to support the large computation graph.

4.2 SU(2)-block layer performance

We now compare the performance of the various SU(2)-block architectures using 16-layer models and a batch size of 4096. One useful metric to measure model quality is effective sample size (ESS), which is defined as

$$\text{ESS} = \frac{\left(\frac{1}{N}\sum_{i=1}^{n}\frac{p(U^{[i]})}{q(U^{[i]})}\right)^2}{\frac{1}{N}\sum_{i=1}^{n}\left(\frac{p(U^{[i]})}{q(U^{[i]})}\right)^2} \tag{4.1}$$

for a set of n samples $\{U^{[1]}, U^{[2]}, \ldots, U^{[n]}\}$, target distribution p, and model distribution q. In particular, it estimates that n samples drawn from the model with distribution q and MCMC has the same predictive power as $n' = \text{ESS} \cdot n$ samples drawn independently from p. As a result, values of ESS lie in the range $[\frac{1}{n}, 1]$, where 1 denotes a perfect model. Conveniently, calculating the ESS does not require knowledge of the partition function Z_{TEK} defined in Equation (2.15) since multiplying the numerator and denominator of Equation (4.1) by $(Z_{\text{TEK}})^{2n}$ yields the computable expression

$$\frac{\left(\frac{1}{N}\sum_{i=1}^{n}\frac{e^{-S_{\text{TEK}}(U^{[i]})}}{q(U^{[i]})}\right)^2}{\frac{1}{N}\sum_{i=1}^{n}\left(\frac{e^{-S_{\text{TEK}}(U^{[i]})}}{q(U^{[i]})}\right)^2}. \tag{4.2}$$

We first evaluate the SU(2)-block models in a setting where $d = 2$ dimensions and the coupling constant β used to define the action in Equation (2.6) is 0.4. Table 4.1 displays the ESS achieved by various SU(2)-block architectures after 30 hours of training for $N = 4, 5, 6, 7$ and 48 hours of training for $N = 9, 12$. The performance of the SU(2)-block models converged during this training for $N = 4, 5$, but not for larger values of N due to the longer iteration time. However, most of the ESS gain occurs at the start of training, so it is very unlikely that any model would improve its ESS significantly even if additional training time was given.

N	layer-SU(2)	layer-SU(2), spectral-flow	input-SU(2)	input-SU(2), spectral-flow	potential-based	Haar-sample
4	0.574	0.758	0.570	0.766	0.833	0.010
5	0.475	0.591	0.553	0.668	0.781	0.007
6	0.362	0.457	0.492	0.609	0.742	0.004
7	0.138	0.183	0.270	0.341	0.598	0.002
9	0.019	0.022	0.072	0.092	0.294	0.001
12	0.005	0.005	0.036	0.040	0.186	0.001

Table 4.1: ESS of the layer-SU(2) model, the layer-SU(2) model composed with spectral-flow layers, the input-SU(2) model, the input-SU(2) model composed with spectral-flow layers, the potential-based model, and samples directly from the Haar measure trained with a batch size of 4096 for the $d = 2$, $\beta = 0.4$ setting.

There are several key takeaways. First, the layer-SU(2) models provide a large improvement over simply performing MCMC with samples from the Haar measure for small N, but this advantage diminishes as N increases. Likewise, composing spectral-flow layers with SU(2)-block layers provides large ESS increases for small N while improving performance universally. Both architectures, and particularly the input-SU(2) models, also yield significant improvements over the Haar measure for all values of N. The asymptotically inefficient but expressive potential-based layer model remains the best performing model, as expected.

Additionally, the estimates of the partition function Z_{TEK} agree despite the relatively small sample size of 4096, while estimates generated with Metropolis-Hastings on the Haar measure tend to underestimate the value of Z_{TEK}. Results for partition function estimates are presented in Table 4.2.

N	layer-SU(2)	layer-SU(2), spectral-flow	input-SU(2)	input-SU(2), spectral-flow	potential-based	Haar-sample
4	2.443	2.444	2.462	2.452	2.442	2.421
5	3.519	3.515	3.521	3.518	3.515	3.458
6	5.064	5.062	5.063	5.063	5.061	4.925
7	7.226	7.225	7.224	7.224	7.222	6.738
9	12.558	12.569	12.575	12.564	12.571	10.547
12	22.257	22.290	22.473	22.466	22.481	16.427

Table 4.2: $\log Z_{\text{TEK}}$ estimates of the layer-SU(2) model, the layer-SU(2) model composed with spectral-flow layers, the input-SU(2) model, the input-SU(2) model composed with spectral-flow layers, the potential-based model, and samples directly from the Haar measure trained with a batch size of 4096 for the $d = 2$, $\beta = 0.4$ setting.

With estimates of the partition function Z_{TEK}, we can also compute the sample reverse KL-divergence estimates as defined in Equation (2.28) as another assessment of model quality. The KL-divergence results are displayed in Table 4.3, and they are consistent with the ESS values observed. Since the KL-divergence is less harsh than ESS as a metric, it is more evident here that the SU(2)-block models outperform the Haar-measure baseline by far even for models with low ESS.

We also evaluate the SU(2)-block models on a $d = 4$, $\beta = 0.1$ setting. Tables 4.4 to 4.6 displays the achieved ESS, Z_{TEK} estimate, and sample reverse KL-divergence estimate of the layer-SU(2) model, the input-SU(2) model, and the input-SU(2) model with gauge fixing. The performance of all models converged in this setting due to the smaller value of β requiring fewer training iterations.

Like the two-dimensional case, the SU(2)-block models greatly outperform the Haar-measure baseline, but they fall short of the potential-based layers. In addition, gauge fixing also improves model quality slightly. However, gauge-fixing begins to become unreliable at $N = 9$ due to numerical issues – while gauge fixing is theoretically sound, the gauge-fixing transformation presented in Section 3.3.4 is discontinous

N	layer-SU(2)	layer-SU(2), spectral-flow	input-SU(2)	input-SU(2), spectral-flow	potential-based	Haar-sample
4	0.277	0.133	0.275	0.121	0.092	3.220
5	0.354	0.247	0.298	0.197	0.123	4.213
6	0.506	0.389	0.375	0.253	0.157	5.594
7	0.957	0.812	0.668	0.537	0.260	7.313
9	2.234	2.130	1.299	1.199	0.577	10.946
12	4.563	4.415	2.033	1.988	0.880	16.618

Table 4.3: Sample reverse KL-divergence estimates of the layer-SU(2) model, the layer-SU(2) model composed with spectral-flow layers, the input-SU(2) model, the input-SU(2) model composed with spectral-flow layers, the potential-based model, and samples directly from the Haar measure trained with a batch size of 4096 for the $d = 2$, $\beta = 0.4$ setting.

N	layer-SU(2)	input-SU(2)	input-SU(2), gauge-fixed	potential-based	Haar-sample
4	0.907	0.911	0.926	0.941	0.074
5	0.877	0.924	0.931	0.940	0.046
6	0.801	0.896	0.909	0.926	0.025
7	0.704	0.856	0.864	0.908	0.014
9	0.504	0.773	—	0.862	0.005
12	0.201	0.611	—	0.760	0.002

Table 4.4: ESS of the layer-SU(2) model, the input-SU(2) model, the input-SU(2) model with gauge fixing, the potential-based model, and samples directly from the Haar measure trained with a batch size of 4096 for the $d = 4$, $\beta = 0.1$ setting. All three SU(2)-block models are composed with spectral-flow layers.

N	layer-SU(2)	input-SU(2)	input-SU(2), gauge-fixed	potential-based	Haar-sample
4	0.304	0.313	0.314	0.303	0.299
5	0.822	0.824	0.824	0.821	0.817
6	1.498	1.499	1.499	1.498	1.487
7	2.308	2.308	2.308	2.308	2.279
9	4.315	4.315	nan	4.316	4.171
12	8.253	8.255	nan	8.255	7.456

Table 4.5: $\log Z_{\text{TEK}}$ estimates of the layer-SU(2) model, the input-SU(2) model, the input-SU(2) model with gauge fixing, the potential-based model, and samples directly from the Haar measure trained with a batch size of 4096 for the $d = 4$, $\beta = 0.1$ setting. All three SU(2)-block models are composed with spectral-flow layers.

across the measure 0 set of matrices with degenerate eigenvalues. As a result, small numerical precision uncertainties may cause problems with invertibility that invali-

N	layer-SU(2)	input-SU(2)	input-SU(2), gauge-fixed	potential-based	Haar-sample
4	0.047	0.043	0.037	0.029	1.499
5	0.063	0.038	0.035	0.031	1.951
6	0.106	0.053	0.047	0.038	2.492
7	0.166	0.075	0.071	0.047	3.144
9	0.316	0.125	—	0.073	4.769
12	0.753	0.238	—	0.135	7.745

Table 4.6: Sample reverse KL-divergence estimates of the layer-SU(2) model, the input-SU(2) model, the input-SU(2) model with gauge fixing, the potential-based model, and samples directly from the Haar measure trained with a batch size of 4096 for the $d = 4$, $\beta = 0.1$ setting. All three SU(2)-block models are composed with spectral-flow layers.

date the flow model.

4.3 Transferrable layers

Finally, we demonstrate that architectures introduced in Section 3.4 allow transfer learning. To do so, we first train a model on a setting with N colors starting from a random initialization of weights. Then, the parameters of this trained model are loaded as the initial state of a N'-color model, where $N' > N$. Transfer learning viability can be assessed by comparing the training dynamics of this N'-color model versus the training dynamics of a randomly-initialized N'-color model as a result.

Figure 4-2 displays this comparison for the input-SU(2) transferrable models for transfers from a 4-color model to 5- and 7- color models. Similarly, Figure 4-3 displays the analogous results from the projection-based transferrable models.

In all cases, the pretrained model initializations lead to a substantially higher starting ESS. Furthermore, all models with pretrained initilaizations also achieve peak ESS significantly faster, which could greatly reduce training time when N is large. Finally, it should be noted that although the rudimentary projection-based architecture lacks expressivity due to the extremely small number of parameters, it manages to achieve its peak ESS upon initialization. This allows for transfers with N' much larger than N, which could be especially powerful if a more expressive

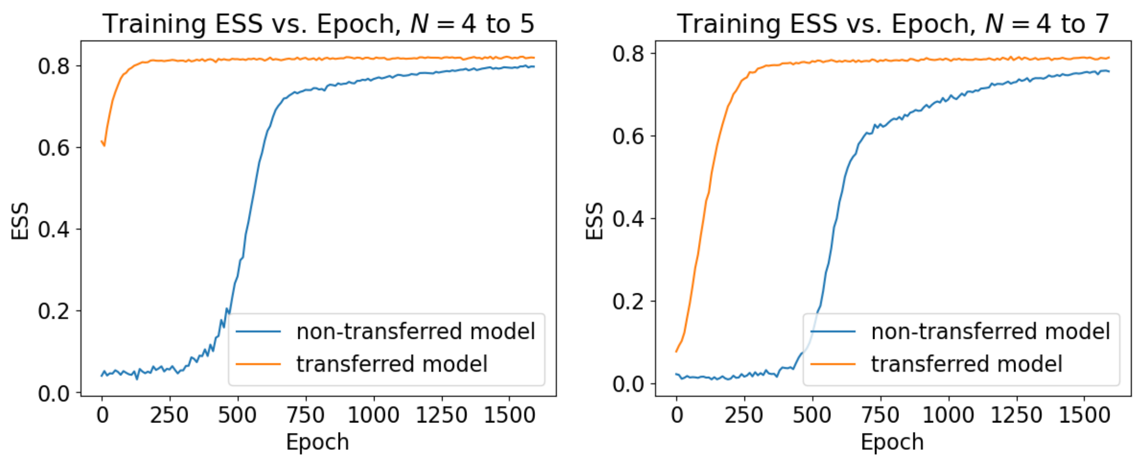

Figure 4-2: ESS vs. epoch during training of input-SU(2) transferrable models with randomly initialized weights and models with weights pretrained on a smaller N in the $d = 2$, $\beta = 0.1$ setting. The left plot displays a transfer from $N = 4$ to $N' = 5$, and the right plot displays a transfer from $N = 4$ to $N' = 7$.

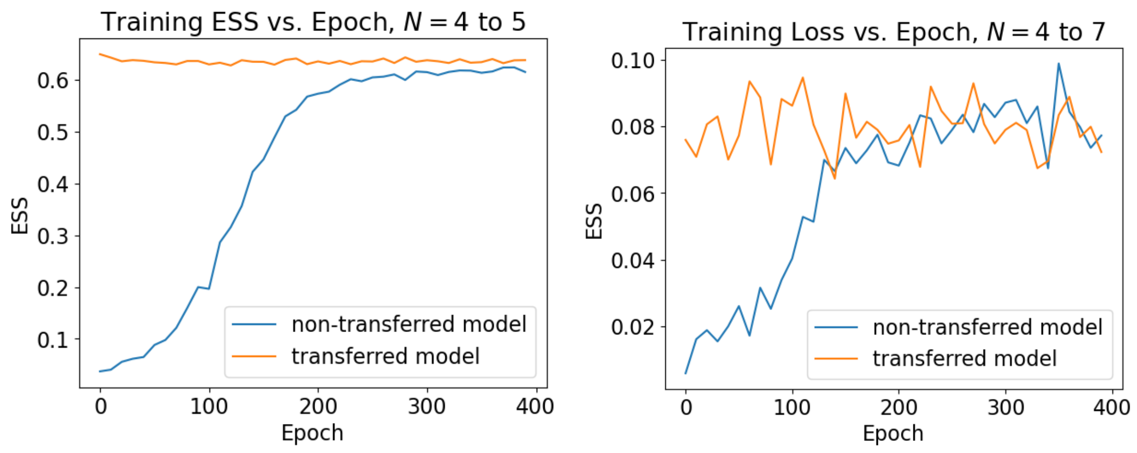

Figure 4-3: ESS vs. epoch during training of projection-based transferrable models with randomly initialized weights and models with weights pretrained on a smaller N in the $d = 2$, $\beta = 0.1$ setting. The left plot displays a transfer from $N = 4$ to $N' = 5$, and the right plot displays a transfer from $N = 4$ to $N' = 7$.

projection-based layer is designed.

Chapter 5

Conclusion

In this thesis, we present new, asymptotically faster flow-based generative model architectures for the TEK model by leveraging multiplication by SU(2)-block matrices for efficient Jacobian computation. Furthermore, we introduce the idea of transfer learning with respect to the number of colors N, and describe two architectures for which this is possible.

There are many areas of future work that build on ideas discussed in this thesis. First, running models at a larger scale through greater compute power or further code optimizations may provide additional insights, such as more clarity into asymptotic complexity and expressivity as a function of N. In addition, further investigation is required to either mitigate or resolve the numerical issues that arise in gauge fixing. Significant improvements to transfer learning architectures and methodology are also possible. For instance, the projection-based layers in their current form are still rather rudimentary; one improvement could be parameterizing each learned constant with a neural network. Furthermore, the relationship between model parameters and N remains unexplored, as it could be possible that weights should be scaled up or down when transfering from one value of N to another for optimal performance.

Whether through flow-based models or otherwise, we hope that the work presented provides inspiration for future sampling algorithms for the TEK model and large-N gauge theories.

Bibliography

[1] Ryan Abbott et al. Aspects of scaling and scalability for flow-based sampling of lattice QCD. *Eur. Phys. J. A*, 59(11):257, 2023.

[2] Ryan Abbott et al. Normalizing flows for lattice gauge theory in arbitrary space-time dimension. 5 2023.

[3] Ryan Abbott et al. Sampling QCD field configurations with gauge-equivariant flow models. *PoS*, LATTICE2022:036, 2023.

[4] Ryan Abbott et al. Applications of flow models to the generation of correlated lattice qcd ensembles. 2024.

[5] T. Aoyama et al. The anomalous magnetic moment of the muon in the Standard Model. *Phys. Rept.*, 887:1–166, 2020.

[6] Tom B. Brown et al. Language Models are Few-Shot Learners. 5 2020.

[7] Roger F. Dashen and Aneesh V. Manohar. 1/N(c) corrections to the baryon axial currents in QCD. *Phys. Lett. B*, 315:438–440, 1993.

[8] M. Daum, R. Frosch, and P. R. Kettle. The charged and neutral pion masses revisited. *Phys. Lett. B*, 796:11–14, 2019.

[9] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Phys. Lett. B*, 195:216–222, 1987.

[10] Tohru Eguchi and Hikaru Kawai. Reduction of Dynamical Degrees of Freedom in the Large N Gauge Theory. *Phys. Rev. Lett.*, 48:1063, 1982.

[11] Sam Foreman, Taku Izubuchi, Luchang Jin, Xiao-Yong Jin, James C. Osborn, and Akio Tomiya. HMC with Normalizing Flows. *PoS*, LATTICE2021:073, 2022.

[12] S. L. Glashow. Partial Symmetries of Weak Interactions. *Nucl. Phys.*, 22:579–588, 1961.

[13] Antonio Gonzalez-Arroyo and M. Okawa. The Twisted Eguchi-Kawai Model: A Reduced Model for Large N Lattice Gauge Theory. *Phys. Rev. D*, 27:2397, 1983.

[14] David J. Gross and Frank Wilczek. Ultraviolet Behavior of Nonabelian Gauge Theories. *Phys. Rev. Lett.*, 30:1343–1346, 1973.

[15] W. K. Hastings. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika*, 57:97–109, 1970.

[16] Fabian Heiße et al. High-precision measurement of the proton's atomic mass. *Phys. Rev. Lett.*, 119(3):033001, 2017.

[17] Paul Langacker and Ming-xing Luo. Implications of precision electroweak experiments for M_t, ρ_0, $\sin^2\theta_W$ and grand unification. *Phys. Rev. D*, 44:817–822, 1991.

[18] G. Peter Lepage and Stanley J. Brodsky. Exclusive Processes in Perturbative Quantum Chromodynamics. *Phys. Rev. D*, 22:2157, 1980.

[19] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 21:1087–1092, 1953.

[20] Adam Paszke et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. 12 2019.

[21] Danilo Jimenez Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. 5 2015.

[22] Abdus Salam. Weak and Electromagnetic Interactions. *Conf. Proc. C*, 680519:367–377, 1968.

[23] Stefan Schaefer, Rainer Sommer, and Francesco Virotta. Critical slowing down and error analysis in lattice QCD simulations. *Nucl. Phys. B*, 845:93–119, 2011.

[24] Gerard 't Hooft and M. J. G. Veltman. Regularization and Renormalization of Gauge Fields. *Nucl. Phys. B*, 44:189–213, 1972.

[25] Steven Weinberg. A Model of Leptons. *Phys. Rev. Lett.*, 19:1264–1266, 1967.

[26] Kenneth G. Wilson. Confinement of Quarks. *Phys. Rev. D*, 10:2445–2459, 1974.