Gauge-equivariant neural networks as preconditioners in lattice QCD

C. Lehner[*] and T. Wettig

Department of Physics, University of Regensburg, 93040 Regensburg, Germany

We demonstrate that a state-of-the-art multigrid preconditioner can be learned efficiently by gauge-equivariant neural networks. We show that the models require minimal retraining on different gauge configurations of the same gauge ensemble and to a large extent remain efficient under modest modifications of ensemble parameters. We also demonstrate that important paradigms such as communication avoidance are straightforward to implement in this framework.

I. INTRODUCTION

Our current understanding of nature at the most fundamental level is to a large extent based on quantum field theories. In particle physics, quantum chromodynamics (QCD) explains, for example, how the proton is made up of smaller constituents, quarks and gluons. To describe current and future experiments, and to search for physics beyond the Standard Model, we need to be able to solve QCD to high precision. Lattice QCD constitutes a systematically improvable tool to solve QCD in the nonperturbative regime by numerically simulating the theory on a finite space-time lattice. It has evolved over more than four decades and is now of direct phenomenological relevance, see [1] and references therein. It is also very compute-intensive and employs the largest supercomputers worldwide [2]. Therefore much research is focused on improving the algorithms that dominate the run time of these simulations.

The most time-consuming element, both in the generation of gauge-field configurations and in the computation of physical observables, is typically the solution of the Dirac equation in the presence of a given gauge field. For physical values of the light quark masses and large lattice volumes, the condition number of the matrix representing the Dirac operator becomes very large, and consequently very sophisticated methods are required to solve the Dirac equation in a feasible time frame. The current state of the art is to use a suitable preconditioner inside a Krylov subspace solver. The construction of the preconditioner is a complicated problem whose solution requires deep knowledge of the underlying physics. The aim of this paper is to reformulate the problem in the language of gauge-equivariant (or, equivalently, gauge-covariant) neural networks and to show that such networks can learn the general paradigms of state-of-the-art preconditioners and efficiently reduce the iteration count of the outer solver. We also provide a flexible implementation interface in the Grid Python Toolkit (GPT) [3] that allows for experimentation and further studies.

We briefly relate this paper to previous work. We will concentrate on multigrid preconditioners [4–10] and refer to [11] for an introduction. The idea of learning the elements of multigrid preconditioners with neural networks has been pursued in a number of earlier publications, see, e.g., [12–18]. These works differ in the details of their approaches, e.g., the choice of the loss function, the network architecture, and the kind of learning (supervised or unsupervised). The main difference in our work is that we have to address the gauge degrees of freedom. More precisely, our approach must be gauge-equivariant, i.e., the map implemented by the neural network must commute with local gauge transformations [19,20].

A number of papers have introduced gauge-equivariant neural networks in the context of lattice quantum field theory: Refs. [21–24] mainly addressed the question of gauge-field sampling in several different theories, while Ref. [25] showed how any gauge-covariant function on the lattice can be approximated by neural networks. Our work builds on and extends these papers.

We note that the use of neural networks as preconditioners in a lattice gauge theory was recently also explored in Ref. [26] for a two-dimensional U(1) gauge theory.

The structure of this paper is as follows. In Sec. II, we introduce gauge-equivariant layers as the building blocks of the models we study in this work. In Sec. III, we discuss the problem of solving the preconditioned Dirac equation with the Wilson-clover Dirac operator. In Sec. IV, we construct preconditioner models that address the high-mode component of the Dirac operator. In Sec. V,

---

[*]Corresponding author: christoph.lehner@ur.de

we discuss a model to address the low-mode component of the Dirac operator. In Sec. VI, we combine the specialized models to a multigrid model that addresses both the low-mode and high-mode components. We conclude in Sec. VII, where we also give an outlook to future work.

II. GAUGE-EQUIVARIANT LAYERS

In this section, we define the building blocks of the gauge-equivariant neural networks considered in this work and explain their properties in detail. We begin with a discussion of the concepts of parallel transport and gauge equivariance.

A. Parallel transport and gauge equivariance

We consider a discrete d-dimensional space-time lattice with L_μ sites in dimension $\mu \in \{1, ..., d\}$ and $d \in \mathbb{N}$. The canonical unit vector in dimension μ is denoted by $\hat{\mu}$. The set of all lattice sites shall be $S = \{(x_1, ..., x_d) | x_\mu \in \{1, ..., L_\mu\}\}$. Consider a field $\varphi \colon S \to V_I, x \mapsto \varphi(x)$ with internal vector space V_I. The internal vector space shall be a product of a gauge vector space $V_G = \mathbb{C}^N$ and a non-gauge vector space $V_{\bar{G}} = \mathbb{C}^{\bar{N}}$ with $N, \bar{N} \in \mathbb{N}$, i.e.,

$$V_I = V_G \otimes V_{\bar{G}}. \tag{1}$$

We also consider gauge fields $U_\mu \colon S \to \mathrm{SU}(N), x \mapsto U_\mu(x)$ with $\mathrm{SU}(N)$ acting on V_G. The set of fields φ shall be $\mathcal{F}_\varphi$, and the set of fields U_μ shall be $\mathcal{F}_U$.

We define the parallel-transport operator $T_p \colon \mathcal{F}_\varphi \to \mathcal{F}_\varphi, \varphi \mapsto T_p \varphi$ as

$$T_p = H_{p_{n_p}} \cdots H_{p_2} H_{p_1} \tag{2}$$

for a path p defined as the sequence $p_1, ..., p_{n_p}$ with $n_p \in \mathbb{N}$ and $p_i \in \{\pm 1, \pm 2, ..., \pm d\}$. The operator $H_{p_i} \colon \mathcal{F}_\varphi \to \mathcal{F}_\varphi, \varphi \mapsto H_{p_i} \varphi$ acts on a field according to[1]

$$H_{p_i} \varphi(x) = U_{p_i}^\dagger(x - \hat{p}_i) \varphi(x - \hat{p}_i) \tag{3}$$

so as to transport information by a single hop in direction $\hat{p}_i$. Here, we introduced the convention $\hat{\nu} = -\hat{\mu}$ for $\nu = -\mu$, and we identify $U_{-\mu}(x) = U_\mu^\dagger(x - \hat{\mu})$. Addition and subtraction of coordinate tuples are defined component-wise. Note that a single path p defines the transport for any site $x \in S$ to

$$x' = x + \sum_{i=1}^{n_p} \hat{p}_i \tag{4}$$

and may be illustrated using a representative starting point. If $x' = x$, the path is closed. Note that the trivial path 0 with $n_0 = 0$ and $T_0 = \mathbb{1}$ is allowed as well.

A field $\varphi \in \mathcal{F}_\varphi$ acquires a phase θ_μ when translated by L_μ in direction $\hat{\mu}$, i.e.,

$$\varphi(x + L_\mu \hat{\mu}) = e^{i\theta_\mu} \varphi(x) \tag{5}$$

for any coordinate tuple x. A gauge field $U_\mu \in \mathcal{F}_U$ is periodic in all dimensions, i.e.,

$$U_\mu(x + L_\nu \hat{\nu}) = U_\mu(x) \tag{6}$$

with $\nu \in \{1, ..., d\}$. These equations define $\varphi(x)$ and $U_\mu(x)$ for all sites x outside of S.

In Fig. 1, we illustrate the transport from the red starting point along a path p to the black site. This path corresponds to

$$T_p = H_{-1} H_{-2} H_{-1} H_2 H_2, \tag{7}$$

where $\hat{1}$ and $\hat{2}$ is the horizontal and vertical unit vector, respectively, in Fig. 1.

A gauge transformation is parametrized by a field $\Omega \colon S \to SU(N), x \mapsto \Omega(x)$ that acts on all $\varphi \in \mathcal{F}_\varphi$ and $U_\mu \in \mathcal{F}_U$ by

$$\varphi(x) \to \Omega(x)\varphi(x), \tag{8}$$

$$U_\mu(x) \to \Omega(x)U_\mu(x)\Omega^\dagger(x + \hat{\mu}). \tag{9}$$

It is straightforward to show that under such a gauge transformation we have

$$T_p \varphi(x) \to \Omega(x)T_p \varphi(x) \tag{10}$$

for any path p, i.e., the parallel-transport operator T_p commutes with gauge transformations, and thus it is a
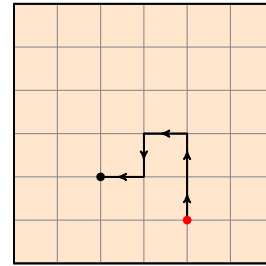

FIG. 1. The path p defining a parallel-transport operator T_p can be visualized as a sequence of hops from a starting point (red) to an end point (black).

---

[1]Note that the operator H_{p_i} does not act on the numerical value $\varphi(x)$. Rather, it acts on the field φ, resulting in the new field $H_{p_i}\varphi$, which is then evaluated at x. The same comment applies to T_p in Eq. (10). Note also that in Eq. (3), the information is transported from $x - \hat{p}_i$ to x.

gauge-equivariant operator. For a comprehensive discussion of gauge equivariance we refer to Ref. [20].

B. Parallel-transport convolutions

The models discussed in this work will be composed of individual layers that map n input features $\varphi_1, \ldots, \varphi_n \in \mathcal{F}_\varphi$ to m output features $\psi_1, \ldots, \psi_m \in \mathcal{F}_\varphi$.

We consider a parallel-transport convolution (PTC) layer defined by[2]

$$\psi_a(x) \overset{\text{PTC}}{=} \sum_{b=1}^{n} \sum_{p \in P} W_a^{bp} T_p \varphi_b(x) \qquad (11)$$

for $a = 1, \ldots, m$, with a set of paths P and an endomorphism $W_a^{bp} \in \text{End}(V_{\bar{G}})$. This extends the definition of Ref. [23] from nearest-neighbor hops to a sum over arbitrary paths. For closed paths p, we recover the case discussed in Ref. [25]. Note that in lattice QCD W_a^{bp} is a 4×4 spin matrix.

We also consider a local parallel-transport convolution (LPTC) layer defined by

$$\psi_a(x) \overset{\text{LPTC}}{=} \sum_{b=1}^{n} \sum_{p \in P} W_a^{bp}(x) T_p \varphi_b(x) \qquad (12)$$

with $W_a^{bp} \colon S \to \text{End}(V_{\bar{G}}), x \mapsto W_a^{bp}(x)$. Such a layer is also gauge equivariant and may be able to better address localized features. In the following we refer to the elements of W as layer weights.

Since we intend to learn a *linear* preconditioner in this work, we do not apply an activation function in these layers. The expressivity of a deep network composed of such layers is therefore equivalent to a single layer with a larger set P. Nevertheless, it may be computationally more efficient for a given problem to compose multiple layers with smaller sets P.

In Fig. 2, we provide a graphical representation of a (L)PTC layer with two input features and one output feature and $P = \{p_1, p_2\}$ with

$$T_{p_1} = H_{-1} H_{-2} H_{-1}, \qquad T_{p_2} = H_{-2} H_1. \qquad (13)$$

C. Restriction and prolongation layers

In order to let information propagate efficiently over long distances in terms of sites $x \in S$, we make use of the multigrid paradigm [4,5]. To this end, we consider a coarse grid with lattice sites $\tilde{S}$ and a coarse field $\tilde{\varphi} \colon \tilde{S} \to \tilde{V}_I, y \mapsto \tilde{\varphi}(y)$ with coarse internal vector space $\tilde{V}_I$. The set

---

[2]Equation (11) is a convolution with kernel W and input φ, whose argument is shifted by T_p.
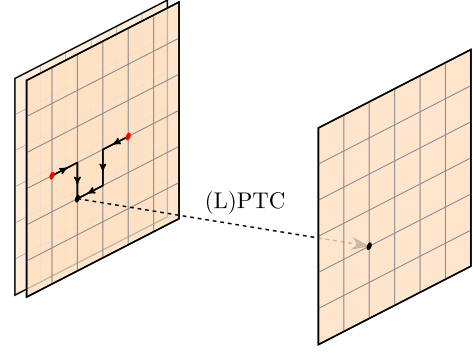

FIG. 2. Graphical representation of a (L)PTC layer with two input features and one output feature. The planes represent the features. The layer is represented by the paths drawn and the dashed arrow.

of such fields is denoted by $\mathcal{F}_{\tilde{\varphi}}$. Note that there are no gauge degrees of freedom in $\tilde{V}_I$.

We define a restriction layer mapping a $\varphi \in \mathcal{F}_\varphi$ to a $\tilde{\psi} \in \mathcal{F}_{\tilde{\varphi}}$ by

$$\tilde{\psi}(y) \overset{\text{RL}}{=} \sum_{x \in B(y)} W(y, x) \varphi(x) \qquad (14)$$

with $W \colon \tilde{S} \times S \to \text{Hom}(V_I, \tilde{V}_I)$ and block map $B \colon \tilde{S} \to \mathcal{P}(S)$, where $\mathcal{P}$ denotes the power set. We also define a corresponding prolongation layer mapping a $\tilde{\varphi} \in \mathcal{F}_{\tilde{\varphi}}$ to a $\psi \in \mathcal{F}_\varphi$ by

$$\psi(x) \overset{\text{PL}}{=} W(y, x)^\dagger \tilde{\varphi}(y) \qquad (15)$$

for $x \in B(y)$. In practice, we choose B corresponding to a blocking in all dimensions. The linear maps W satisfy

$$\sum_{x \in B(y)} W(y, x) W(y, x)^\dagger = \mathbb{1}_{\tilde{V}_I}, \qquad (16)$$

where $\mathbb{1}_{\tilde{V}_I}$ is the identity in $\tilde{V}_I$. These layers are straightforward to extend to the case of multiple input and output features.

The linear maps W can be considered layer weights and are constructed from a list of vectors that are block-wise orthonormal, see Sec. V for details. The restriction and prolongation layers are gauge equivariant if

$$W(y, x) \to W(y, x) \Omega(x)^\dagger \qquad (17)$$

under a gauge transformation. Note that since $\tilde{V}_I$ does not have gauge degrees of freedom there is no $\Omega(y)$ on the coarse grid. We provide a graphical representation of the restriction and prolongation layers in Fig. 3.
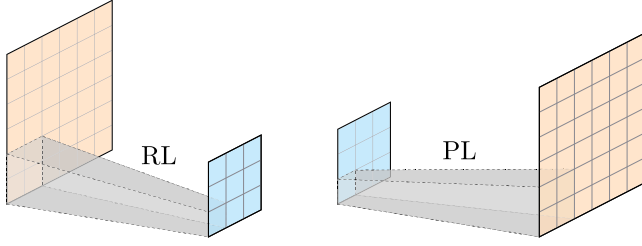
FIG. 3. Graphical representation of the restriction layer (left) and prolongation layer (right) for a single feature. The layers are represented by the gray square frustums, while the input and output features are represented by the planes.

D. Parallel and identity layers

In this work, we consider models that act on a given input feature with multiple layers in parallel. Consider applying a layer L_i to input features $\varphi_1, \ldots, \varphi_n$ mapping to output features $\psi_{i1}, \ldots, \psi_{im_i}$. For several layers $L_1, \ldots, L_\ell$, we concatenate the output features $\psi_{11}, \ldots, \psi_{1m_1}, \ldots, \psi_{\ell 1}, \ldots, \psi_{\ell m_\ell}$. The combination of layers $L_1, \ldots, L_\ell$ being applied in parallel can then be considered to be a single layer that maps features $\varphi_1, \ldots, \varphi_n$ to features $\psi_{11}, \ldots, \psi_{1m_1}, \ldots, \psi_{\ell 1}, \ldots, \psi_{\ell m_\ell}$.

We also introduce an identity layer that maps the input features without modification to output features (which implies $m = n$). Such a layer is represented graphically by a single dashed arrow pointing from the input features to the output features.

We provide a graphical representation for the case of $n = 1$, $\ell = 2$, and $m_1 = m_2 = 1$ in Fig. 4.

E. Communication avoidance

In practice, the performance of a given model in terms of execution time is crucial. For problem sizes of interest to the lattice QCD community, a single problem will be distributed over multiple compute nodes that are connected
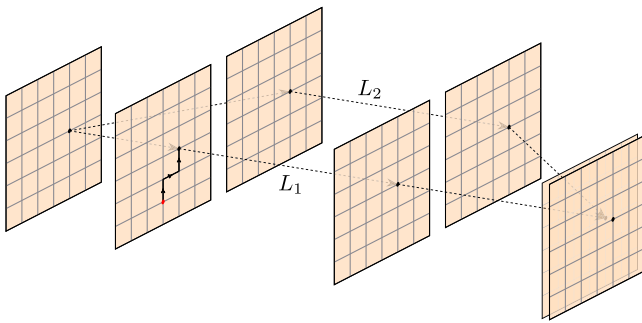

FIG. 4. Graphical representation of two parallel layers L_1 and L_2 being applied to a single input feature and mapping to two output features. As before, the features are represented by planes. An identity layer (i.e., a copy operation) is represented by a dashed arrow. In this example, the only nontrivial layer is L_1, which includes a single path in (11) or (12).

by a communication network. It is not uncommon that the time needed to exchange information between nodes exceeds the time each node spends performing floating-point operations. Therefore it is an important paradigm in lattice QCD to investigate approaches that avoid communication between nodes even if it possibly increases the computational effort within a given node [27–30]. In this work, we also investigate layers which do not communicate between different sub-volumes that would typically be mapped to multiple nodes in an MPI job. We perform such investigations by setting the gauge links U_μ that connect one such subvolume to another to zero. For such a modified model, we can then avoid the communication step between nodes altogether.

III. THE WILSON DIRAC OPERATOR

The main objective of this work is to precondition the Dirac equation

$$Du = b \tag{18}$$

with Dirac operator $D \colon \mathcal{F}_\varphi \to \mathcal{F}_\varphi$, source $b \in \mathcal{F}_\varphi$, and solution $u \in \mathcal{F}_\varphi$. It is useful to interpret Eq. (18) as a matrix equation with $u, b \in \mathbb{C}^k$ and invertible complex $k \times k$ matrix D with

$$k = L_1 \cdots L_d N\bar{N}. \tag{19}$$

We train a model to play the role of an invertible complex $k \times k$ preconditioner matrix M in

$$(DM)M^{-1}u = b, \tag{20}$$

where we attempt to improve the condition number of DM compared to D. Ideally, DM is close to the identity matrix up to a trivial scaling factor. The Dirac matrix transforms as

$$D \to \Omega D \Omega^\dagger \tag{21}$$

under a gauge transformation with block-diagonal matrix $\Omega = \oplus_{x \in S} \Omega(x) \otimes \mathbb{1}_{V_{\bar{G}}}$, which motivates the use of gauge-equivariant layers to construct M.

We first consider the Wilson Dirac operator [31]

$$D_{\mathrm{W}} = \frac{1}{2} \sum_{\mu=1}^{4} \gamma_\mu (H_{-\mu} - H_{+\mu}) + m$$
$$- \frac{1}{2} \sum_{\mu=1}^{4} (H_{-\mu} + H_{+\mu} - 2) \tag{22}$$

with mass $m \in \mathbb{R}$ and Euclidean gamma matrices $\gamma_1, \ldots, \gamma_4$ satisfying the anticommutation relation $\gamma_\mu\gamma_\nu + \gamma_\nu\gamma_\mu = 2\delta_{\mu\nu}$

with Kronecker delta $\delta_{\mu\nu}$. This operator can be mapped to a single PTC layer with a zero-hop path and eight one-hop paths.

We add a clover term that includes closed paths consisting of four hops using

$$Q_{\mu\nu} = H_{-\mu}H_{-\nu}H_{+\mu}H_{+\nu} + H_{-\nu}H_{+\mu}H_{+\nu}H_{-\mu}$$
$$+ H_{+\nu}H_{-\mu}H_{-\nu}H_{+\mu} + H_{+\mu}H_{+\nu}H_{-\mu}H_{-\nu} \quad (23)$$

to obtain the Wilson-clover Dirac operator [32]

$$D_{\mathrm{WC}} = D_{\mathrm{W}} - \frac{c_{\mathrm{sw}}}{4} \sum_{\mu,\nu=1}^{4} \sigma_{\mu\nu} F_{\mu\nu} \quad (24)$$

with $c_{\mathrm{sw}} \in \mathbb{R}$,

$$F_{\mu\nu} = \frac{1}{8}(Q_{\mu\nu} - Q_{\nu\mu}), \quad (25)$$

and

$$\sigma_{\mu\nu} = \frac{1}{2}(\gamma_\mu \gamma_\nu - \gamma_\nu \gamma_\mu). \quad (26)$$

The operator D_{WC} can also be mapped to a single PTC layer, however, paths up to four hops are needed.

For the numerical experiments presented in the following sections, we use gauge group SU(3) and the D_{WC} operator tuned to near criticality, i.e., the mass parameter is chosen such that the real part of the smallest eigenvalue is close to zero. This provides a challenging problem even for the small lattice volume with $L_1 = L_2 = L_3 = 8$ and $L_4 = 16$ used in this work. We set $m = -0.6$ and $c_{\mathrm{sw}} = 1$ on a pure Wilson gauge configuration [31] with coupling parameter $\beta = 6.0$. We use periodic boundary conditions also for the fields in $\mathcal{F}_\varphi$, i.e., $\theta_\mu = 0$ in Eq. (5). We show the spectrum of D_{WC} on a representative single gauge configuration in Fig. 5.

We quantify the improvement achieved using the preconditioner M by the reduction of the iteration count to solve Eq. (20) to 10^{-8} precision in the preconditioned flexible generalized minimal residual method (FGMRES) [33]. We quote the iteration count gain defined as the iteration count of the unpreconditioned solve divided by the iteration count of the preconditioned solve. The unpreconditioned solve took 4749 iterations.

In future work we will investigate how the iteration count gain translates to a reduction in wall-clock time for a highly-optimized model. As a first approximation, we note that a coarse-grid layer has negligible cost compared to the fine-grid layer, which in turn has similar cost to the application of D_{WC}.

The methods developed in this work also extend to other Dirac matrices. However, particular challenges exist in some cases. For example, in the case of domain-wall fermions [34,35] the spectrum encircles the origin [9,10,36–38],
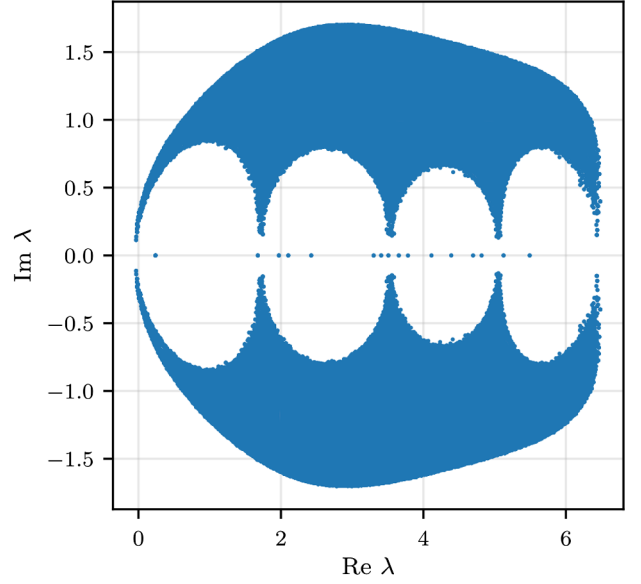

FIG. 5. Eigenvalues λ of the Wilson-clover Dirac operator with $m = -0.6$ and $c_{\mathrm{sw}} = 1$ on a pure-Wilson-gauge configuration with $\beta = 6$, $L_1 = L_2 = L_3 = 8$, and $L_4 = 16$. The mass m is tuned to near criticality for the experiments in this work. We computed the boundaries of the spectrum using the Arnoldi method applied to $(D - \lambda)^{-1}$ for several carefully selected values of λ and filled in the bulk of the spectrum by hand for illustrative purposes.

which limits the convergence of unpreconditioned solves of $Du = b$ using Krylov-subspace methods.

IV. HIGH-MODE PRECONDITIONERS

We want to learn a preconditioner M that approximates D^{-1}. For this purpose it is useful to consider an eigendecomposition of D and first construct optimal models for the high-mode and low-mode components separately. We study the high-mode component in this section and the low-mode component in Sec. V. We then combine the corresponding models in Sec. VI.

A. Model setup and training strategy

The high-mode part of the spectrum of D_{WC} is related to the short-distance behavior. Therefore we expect a single layer with paths up to one hop to already show a gain in iteration count. We consider a linear model M mapping a vector x to Mx. We employ a supervised learning approach and describe a single training step in the following.

We first pick a random vector v with components drawn from a Gaussian distribution with mean zero and unit standard deviation. We then construct the cost function[3]

---

[3]Note that in Eq. (20) we use DM, while in Eq. (27) we use MD. If DM is close to the identity, then so is MD, and thus Eq. (27) is a suitable cost function.

$$C = |MD_{\mathrm{WC}}v - v|^2 \qquad (27)$$

and its derivatives with respect to the model weights using backpropagation. This corresponds to a batch of a single training tuple $(D_{\mathrm{WC}}v, v)$, where the model learns to map the first to the second component. This cost function is dominated by the high modes of D_{WC} and is therefore similar in spirit to using the spectral radius [12,14]. Since we use a different random vector at every iteration our training dataset is unbounded in size and there is no need to add a regulator. This holds even for LPTC layers with a large number of model weights. We then apply a single iteration of the Adam optimizer [39] with parameters $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\alpha = 10^{-3}$ that gave good performance for the models considered in this work. This process is repeated until the model weights are converged sufficiently.

All layers and optimizers are implemented in the Grid Python Toolkit (GPT) [3], and corresponding code samples are provided in Appendix.

B. Locality and communication avoidance

In Fig. 6 we compare the performance of single-layer models with a maximum of one hop. They correspond to a

version of Fig. 2 with a single input and output feature and nine paths corresponding to

$$
\begin{aligned}
T_0 &= \mathbb{1}, \\
T_1 &= H_1, & T_2 &= H_2, \\
T_3 &= H_3, & T_4 &= H_4, \\
T_5 &= H_{-1}, & T_6 &= H_{-2}, \\
T_7 &= H_{-3}, & T_8 &= H_{-4}. \qquad (28)
\end{aligned}
$$

We also investigate communication-avoiding versions with local volume $4^3 \times 8$. We find that the LPTC models do not perform better in terms of iteration count gain than the PTC models. However, the LPTC models require more training compared to the PTC models. The slower convergence is expected due to the much larger number weights in the LPTC models. We find that eliminating communication between sub-volumes, as described in Sec. II E, only leads to a modest reduction in performance. After translating the iteration count gain to a reduction in time-to-solution, we may therefore find the communication-avoiding models to perform best.

C. Multiple hops and deep networks

In Fig. 7, we investigate models with multiple hops either in a single layer or distributed over two layers.
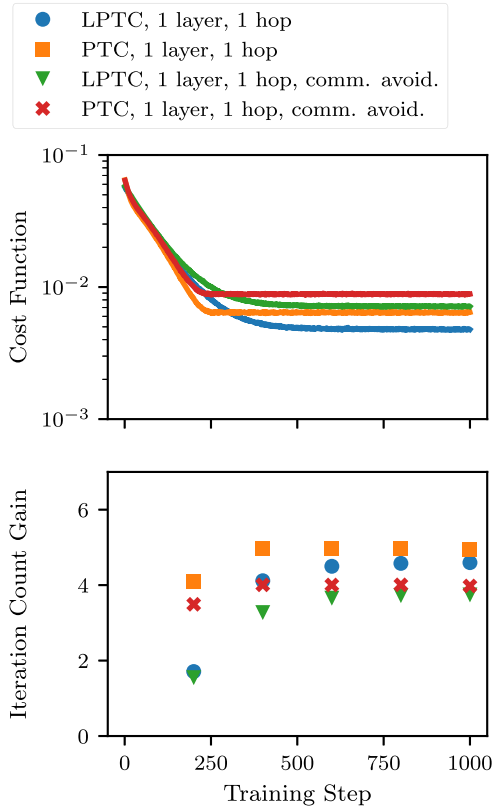

FIG. 6. Convergence of the cost function (27) and iteration count gain for one-layer and one-hop high-mode preconditioners. The lattice volume is $8^3 \times 16$, and the local volume for the communication-avoiding version is $4^3 \times 8$.
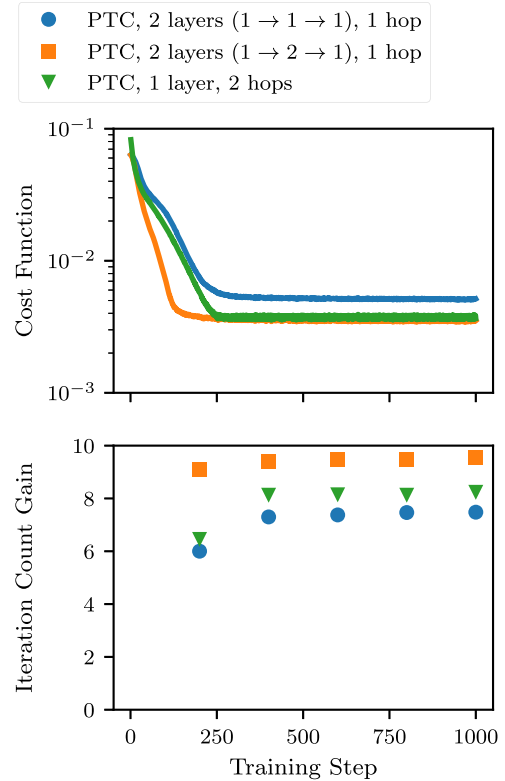

FIG. 7. Convergence of the cost function (27) and iteration count gain for two-layer and two-hop high-mode preconditioners.

We use one-hop layers with paths defined in Eq. (28) as well as a two-hop layer extending this set by all combinations

$$H_a H_b \tag{29}$$

for $a, b \in \{-4, -3, -2, -1, 1, 2, 3, 4\}$ with $a \neq -b$. The two-hop layer therefore has 65 distinct paths compared to the 9 paths of the one-hop layer.

The first model that we investigate stacks two one-hop layers with one input and one output feature back-to-back. We denote this model as "2 layers $(1 \to 1 \to 1)$, 1 hop." The second model is similar but has two output features in the first layer and correspondingly two input features in the second layer. We denote this model as "2 layers $(1 \to 2 \to 1)$, 1 hop." The third model consists of a single two-hop layer as described above.

We find that the second model performs best and gives approximately twice the iteration count gain of the corresponding single-layer models with a maximum of one hop shown in Fig. 6. Since the layers are linear, the two-layer models are not more expressive compared to the single-layer model with two hops. We therefore expect the third model to be able to match the performance of the second model with a sufficiently improved training procedure. It is not surprising that the second model can be trained more efficiently compared to the third model given that it has a smaller number of weights. We conclude that while deep models do not increase expressivity, the computational effort needed to train deep models may be reduced compared to a corresponding shallow model with more paths.

D. Transfer learning

In Fig. 8 we investigate how well the one-layer one-hop PTC model of Fig. 6 that was trained on a given gauge configuration with $\beta = 6.0$ and $m = -0.6$ performs when it is used in the case of (i) a different gauge configuration of the same gauge ensemble, (ii) a gauge configuration of a different ensemble with $\beta = 5.9$, and (iii) the same gauge configuration but with a different mass $m = -0.55$. In all cases, we investigate the performance without retraining and after additional retraining steps following the same procedure as for the initial training. We find that the high-mode preconditioner model does not require retraining to efficiently perform in all three cases. Once such a model is trained, it can be used efficiently for different gauge configurations of the same and similar ensembles. We note that the maximum iteration count gain for mass $m = -0.55$ is significantly reduced. In this case, however, the spectrum is not well tuned to criticality and the initial problem is therefore less challenging. Comparing with Fig. 6, we also observe a modest fluctuation in iteration count gain between different configurations.
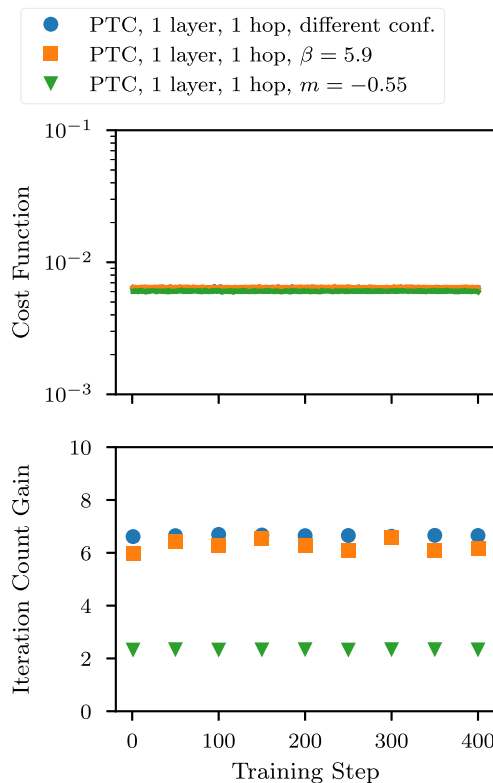

FIG. 8. Convergence of the cost function (27) and iteration count gain for one-layer and one-hop high-mode preconditioners. We retrain the model of Fig. 6 for a different gauge configuration in the same ensemble, for a different value of $\beta = 5.9$, and for a different mass value of $m = -0.55$. The network performs well in all cases even without retraining.

V. LOW-MODE PRECONDITIONERS

We now turn to the low-mode component in the eigendecomposition of D. Since the low-mode component corresponds to the long-distance behavior of the Dirac operator D, it is not efficient to use the layers discussed in Sec. IV since a rather deep network composed of such layers would be needed to propagate information over sufficiently long distances. The multigrid paradigm, however, is ideally suited to address this issue. In this section, we focus solely on the low-mode component and then combine low modes and high modes in Sec. VI.

A. Model setup and training strategy

In the multigrid approach, we define an additional coarser version of the lattice as well as restriction and prolongation operations that map between the fine and coarse lattices. These operations must preserve the low-mode component of D [40].

To achieve this, we follow standard procedures for the multigrid setup [4–6]. We first find vectors $u_1, \ldots, u_s$ in the near-null space of D, i.e., vectors that satisfy

$$Du_i \approx 0 \tag{30}$$

with null vector 0 and $i \in \{1, ..., s\}$ for $s = \dim(\tilde{V}_I)$. These vectors are then blocked such that one site $y \in \tilde{S}$ on the coarse lattice corresponds to a set of sites, or block, $B(y) \subset S$ on the fine lattice. Let us denote such a blocked vector, which lives on the sites $B(y)$, by u_i^y. One then defines an inner product within each block $B(y)$ and orthonormalizes the vectors $u_1^y, ..., u_s^y$ within each block according to this inner product. The resulting vectors are labeled $\bar{u}_1^y, ..., \bar{u}_s^y$. The linear map $W^\dagger$ discussed in Sec. II C is then defined as

$$W(y, x)^\dagger = \sum_{i=1}^{s} \bar{u}_i^y(x) \hat{e}_i^\dagger \tag{31}$$

with standard basis $\hat{e}_1, ..., \hat{e}_s$ of $\tilde{V}_I$ and $x \in B(y)$.

In practice a good approximation of such vectors u_i can be found by applying the FGMRES solver for matrix D with source vector 0 and a random vector as initial guess. This procedure removes high-mode components in u_i, leaving a linear combination of low-modes. We follow this approach in the numerical experiments presented in the following. While high precision is not needed, we solve to 10^{-8} precision to avoid an additional tuning step. We use a coarse grid of size $2^3 \times 4$ and a list of 12 near-null vectors $u_1, ..., u_{12}$.

We define a coarse-grid operator

$$\tilde{D} = R D_{\mathrm{WC}} P \tag{32}$$

with restriction matrix R and prolongation matrix P that are defined according to Eqs. (14) and (15). We then train a coarse-grid model $\tilde{M}$ that contains a single LPTC layer with gauge fields $U_\mu = \mathbb{1}$, $V_G = \mathbb{C}^1$, $V_{\tilde{G}} = \tilde{V}_I$, and use only zero-hop and one-hop paths corresponding to $\{H_1, H_2, H_3, H_4, H_{-4}\}$. We omit the H_{-1}, H_{-2}, and H_{-3} paths since they are redundant on a $2^3 \times 4$ coarse grid with periodic boundary conditions. The gauge fields are replaced with the identity since the coarse fields do not have a gauge degree of freedom. We refer to this special case of the LPTC layer as cLPTC in the following.

We follow the training procedure described in Sec. IV A but replace the cost function with

$$C = |\tilde{M} \tilde{D} v - v|^2. \tag{33}$$

It is worth noting that one could have considered a different cost function

$$C' = |\tilde{M} v - \tilde{D}^{-1} v|^2 \tag{34}$$

in order to project more strongly on the low modes of $\tilde{D}$. In this case, however, the training tuples require the somewhat

costly (approximate) inversion of $\tilde{D}$. We find that the cost function Eq. (33) is sufficient for the purpose of training the coarse-grid model. This point will be revisited when we train a combined multigrid model in Sec. VI.

Note that the gauge equivariance of the restriction and prolongation layers is guaranteed if every vector u_i is a linear combination of eigenmodes of D with gauge-invariant coefficients. In our procedure the coefficients are gauge invariant in the statistical average over random initial guess vectors. Furthermore, note that the weights W of the restriction and prolongation layers could also be learned directly [12,14]. We leave the systematic study of learning the restriction and prolongation layers, including explicitly gauge-equivariant versions, to future work.

B. Results

In Fig. 9, we show the cost function (33) and the iteration count gain for the training of the coarse-grid model $\tilde{M}$. In this case, we consider the iteration count gain for the inverse of $\tilde{D}$. We find that a significantly longer training process is needed compared to the high-mode preconditioner models of Sec. IV.
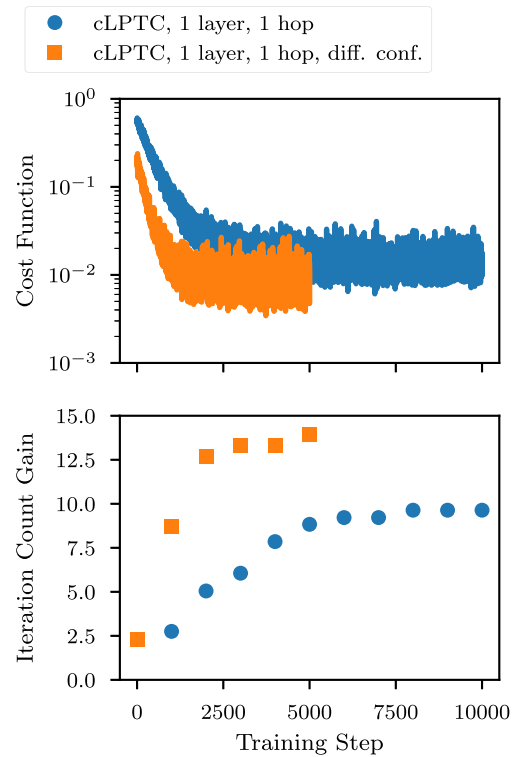

FIG. 9. Convergence of the cost function (33) and iteration count gain for one-layer and one-hop low-mode preconditioners. We show both the initial training in blue as well as the performance of the trained model on a different gauge field of the same gauge ensemble in orange. We find that after a moderate amount of retraining, the model performs well on a different gauge configuration.

We also investigate using the fully trained model from a given gauge configuration and applying it to a different gauge configuration. We use the same definition of the restriction and prolongation layers on the different gauge configuration to preserve the definition of $\tilde{D}$. For the same reason we also use the same seeds for the random number generator to generate the initial guess for the fields $u_1, \ldots, u_{12}$. We find that after a modest amount of retraining the model performs very well on the different gauge configuration. The retraining phase is significantly shorter compared to the initial training phase. We note that the maximum iteration count gain again differs to some degree between configurations.

VI. MULTIGRID PRECONDITIONERS

In the previous sections we successfully trained separate models M to approximate the short-distance and long-distance features of D^{-1}. In this section we combine them to obtain a model that approximates D^{-1} over a wide range of distances.

A. Smoother model setup and training strategy

We first create a version of the short-distance model that accepts a second input feature, which provides an initial guess. This model plays the role of a smoother in the multigrid paradigm. The initial guess is provided by the long-distance model acting on the coarse grid.

Concretely, we aim to find a sequence of u_k that approximately solve $Du = b$ such that the equation becomes exact in the $k \to \infty$ limit. The smoother then maps the tuple (u_k, b) to u_{k+1}. If we have a high-mode model M_{h} that approximates D^{-1} sufficiently well this can be achieved by the iterative relaxation approach

$$
\begin{aligned}
u_{k+1} &= (\mathbb{1} - M_{\mathrm{h}}D)u_k + M_{\mathrm{h}}b \\
&= u_k + M_{\mathrm{h}}(b - Du_k).
\end{aligned} \tag{35}
$$

This approach is also commonly referred to as defect correction with defect $b - Du_k$.

Since both D and the high-mode model M_{h} can be represented by (L)PTC layers we should be able to train a model M_{s} only composed of (L)PTC layers to map (u_k, b) to a u_{k+r} for $r \in \mathbb{N}^+$. Such a model has two input features and one output feature. We may construct M_{s} using $2r$ (L) PTC layers stacked back-to-back since each iteration of Eq. (35) corresponds to two (L)PTC layers. All but the final layer need two output features.

In order to choose a reasonable value for r, we studied the performance of the final multigrid preconditioner described below and found that $r = 2$ performed significantly better than $r = 1$. We therefore train the model M_{s} for $r = 2$ using the cost function

$$
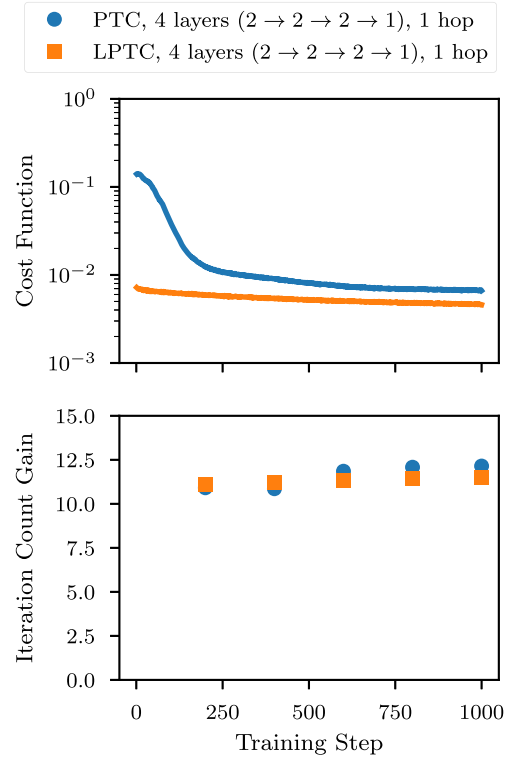C = |M_{\mathrm{s}}(u_k, b) - u_{k+r}|^2 \tag{36}
$$

FIG. 10. Convergence of the cost function (36) and iteration count gain for four-layer and one-hop smoother. The iteration count gain is studied for the case of zero initial guess. We first train the PTC model and use the result as initial weights for the LPTC model.

with random vectors (u_k, b) and u_{k+r} given by Eq. (35). We use the same optimizer as in Secs. IV and V.

In Fig. 10, we show the training progress. The iteration count gain is obtained by using M_{s} with initial guess zero as a preconditioner for $Du = b$. We use both PTC and LPTC layers with zero-hop and one-hop paths. We expect these models to yield an iteration count gain of approximately twice the iteration count gain of the corresponding high-mode models shown in Fig. 6 because of $r = 2$. We find that this expectation is satisfied by our data. In Fig. 10, we first train the PTC model and then use the model weights as initial values for the LPTC model (using the same value for every site x). We find no additional benefit by using the LPTC model.

B. Multigrid model setup and training strategy

We are now ready to combine the individual models to a complete multigrid model M as shown in Fig. 11. We start by duplicating the input feature. One copy is preserved for the smoother, while the other copy is restricted to the coarse grid, where we apply the coarse-grid model of Sec. V. The result is then prolonged to the fine grid, and both the copy of the initial feature and the result of the coarse-grid model are combined to two input features for
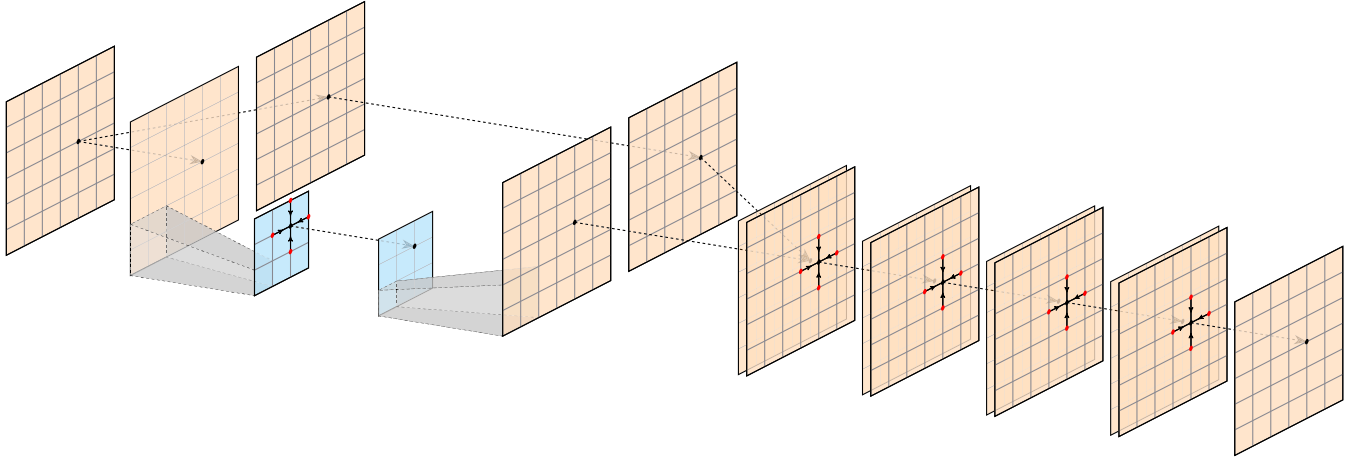
FIG. 11. The combined two-level multigrid model studied in this work. The use of the multigrid paradigm allows for the efficient transport of information over both short and long-distances. Additional levels can be introduced by recursively replacing the coarse-grid layer (limited by the blue features) by the entire model as presented above.

the last four layers. These layers are the smoother that we have learned in Sec. VI A.

We may expect this combined model to work well by using the weights obtained in the training of the respective model components. The model performance may, however, be further improved by continued training of the complete multigrid model M. For such additional training, we need to modify the cost function of Secs. V and IV such that both the low-mode and high-mode components of D constrain the model in the training phase. To this end, we use

$$C = |Mb_h - u_h|^2 + |Mb_\ell - u_\ell|^2 \qquad (37)$$

with $b_h = D_{\text{WC}}v_1$, $u_h = v_1$, $b_\ell = v_2$, and $u_\ell = D_{\text{WC}}^{-1}v_2$. Here, v_1 and v_2 are random vectors normalized such that $|b_h| = |b_\ell| = 1$. We therefore use a batch size of two with one training tuple geared toward the high-mode component and the other training tuple geared toward the low-mode component of D_{WC}. We can shift the focus of the training between both components by adding a relative weight factor to Eq. (37).

C. Results

In Fig. 12, we show the performance of the multigrid (MG) model with initial weights taken from the trained model components as well as progress achieved by continued training of the combined model M. From the start, the model performs substantially better than the smoother by itself. Continued training of the combined model further improves the iteration count gain to approximately 40. Such continued training converges within the first 20 training steps.

We also study using the multigrid model trained on one configuration applied to a different gauge configuration of the same gauge ensemble. In Fig. 12, we show that after a brief retraining phase of only 20 training steps,

the model performs optimally on the different gauge configuration as well.

Note that for concreteness we only present results for a two-level multigrid preconditioner in this work.
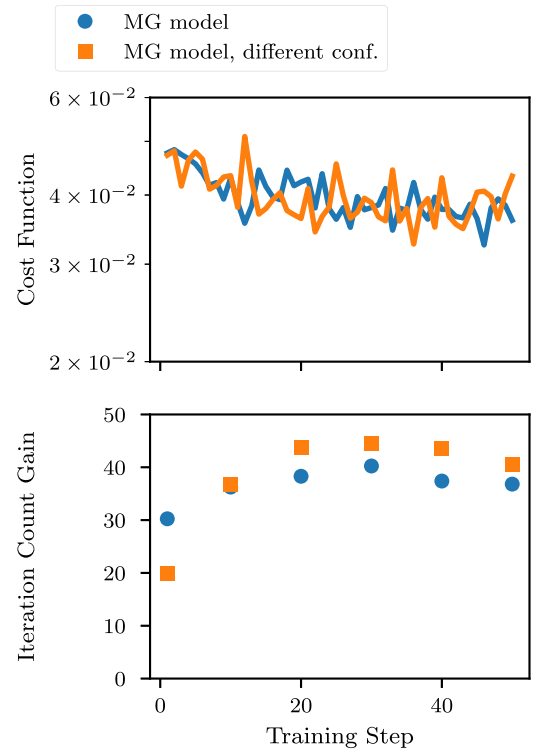

FIG. 12. Convergence of the cost function (37) and iteration count gain for the complete multigrid model. We use the weights of the individually trained model components as starting point and show further improvement by training the combined model. The model also performs well on a different gauge configuration and quickly converges to optimum performance after a modest amount of retraining.

The extension to multiple levels is straightforward. In Fig. 11, one merely has to replace the coarse-grid layer limited by the blue features by the entire model as presented in Fig. 11. By repeating this process n times, one obtains an $(n + 2)$–level multigrid preconditioner.

Also note that we use a rather small lattice volume of $8^3 \times 16$ in this work. In future work, we will investigate multigrid models in more challenging large-volume simulations, where even larger iteration count gains should be achievable.

D. Beyond the Krylov subspace

We note that any polynomial of D_{WC} can be represented by a network of PTC layers with suitable weights. Therefore our models include Krylov-subspace preconditioners as a special case. However, our models are sufficiently expressive to represent solutions outside of the Krylov subspace as well. This statement applies both to the fine-grid and the coarse-grid models. As mentioned in Sec. III, the flexibility to directly approximate D_{WC}^{-1} without being constrained to the Krylov subspace may play an important role for Dirac operators with a more complex spectrum such as domain-wall fermions. This will be studied in future work.

VII. SUMMARY AND OUTLOOK

In this paper we have initiated a program to use gauge-equivariant neural networks to learn preconditioners in lattice QCD. We introduced a number of building blocks from which suitable models can be constructed: (i) parallel-transport convolution layers that can include arbitrary paths, with either global or local weights, (ii) restriction and prolongation layers that implement the multigrid paradigm, and (iii) parallel layers that act on a single input feature.

To solve the Dirac equation for the Wilson-clover Dirac operator we have first constructed models that approximate the high-mode and low-mode component of the operator separately. We then combined these models in a two-level multigrid model, which can be extended straightforwardly to an arbitrary number of levels. In all cases we found that the models reduce the iteration count of the outer solver significantly, e.g., by up to $O(40)$ in the multigrid model. We also found that transfer learning works: If we consider another gauge configuration (for the same or a slightly different value of β) or a slightly different quark mass, only a modest amount of retraining (or none at all) is required for the model to perform efficiently again.

We also introduced a communication-avoiding algorithm in which layers do not transfer information between sub-volumes assigned to different MPI processes. In our numerical experiments we found that the performance, i.e., the iteration count gain, of the corresponding model is only slightly reduced. We expect that on large supercomputers,

the wall-clock time saved by avoiding communication more than compensates for this modest reduction.

There are many interesting directions which we plan to explore in future work. For example, we will attempt to learn the weights W of the restriction and prolongation layers directly, without computing the near-null vectors explicitly. Also, we will investigate the space of possible models that can be constructed from our building blocks in a more comprehensive manner. Furthermore, we plan to perform benchmarks that measure the cost of (re) training and applying our models and compare the overall wall-clock time to standard state-of-the-art multigrid methods. It would also be worthwhile to apply our ideas to Dirac operators whose spectrum encircles the origin, such as in the case of domain-wall fermions. Finally, our finding that very little, if any, retraining is needed between configurations suggests that the present approach could also be beneficial in the generation of gauge-field configurations by Markov chain Monte Carlo.

APPENDIX: GPT CODE LISTINGS

In this appendix, we provide Grid Python Toolkit (GPT) [3] code listings to implement the models used in this work. We first import the library and load a gauge field U:

```
1   import gpt as g
2
3   # load gauge field
4   U = g.load("gauge_field")
5   grid = U[0].grid
```

The layer drawn in Fig. 2 corresponds to

```
1   # object types for QCD
2   ot_i = g.ot_vector_spin_color(4,3)
3   ot_w = g.ot_matrix_spin(4)
4
5   # two distinct paths
6   paths = [
7     g.path().f(0).f(1).f(0),
8     g.path().f(1).b(0)
9   ]
10
11  # define an abbreviation
12  l = g.ml.layer
13
14  # define the layer of Fig. 2
15  fig2 = l.parallel_transport_convolution(
16    grid, U, paths, ot_i, ot_w, 2, 1
17  )
```

in the case of lattice QCD. Next, we define restriction and prolongation layers with a coarse grid of size 4^4 defined using vectors $\bar{u}_i$ as

```
1   # define coarse grid
2   coarse_grid = g.grid([4,4,4,4], g.double)
3
4   # load \bar{u}_i vectors
5   u_bar = g.load("u_bar")
6
7   # create blocking map
8   b = g.block.map(coarse_grid, u_bar)
9
10  # create restriction and prolongation
        layers
11  restrict = l.block.project(b)
12  prolong = l.block.promote(b)
```

Note that in the numerical work in this paper, we used a $2^3 \times 4$ coarse grid, while we present the 4^4 case here since it lifts the degeneracy of paths mentioned in Sec. V.

The complete multigrid preconditioner model of Fig. 11 corresponds to

```
1       # define abbreviations
2       lptc = l.local_parallel_transport_
            convolution
3       ptc = l.parallel_transport_convolution
4
5       # identities on coarse grid
6       one=g.complex(coarse_grid)
7       one[:]=1
8
9       I=[g.copy(one) for i in range(4)]
10
11      # coarse-grid vector space
12      cot_i = g.ot_vector_complex_additive_
            group(
13          len(u_bar)
14      )
15      cot_w = g.ot_matrix_complex_additive_
            group(
```

(Table continued)

```
16          len(u_bar)
17      )
18
19      # consider only nearest-neighbor hops
20      paths = [
21          g.path().forward(i)
22          for i in range(4)
23      ] + [
24          g.path().backward(i)
25          for i in range(4)
26      ]
27
28      # coarse-grid layer
29      def coarse_lptc(n_in, n_out):
30        return lptc(
31          coarse_grid, I, paths,
32          cot_i, cot_w, n_in, n_out
33        )
34
35      # fine-grid layer
36      def fine_ptc(n_in, n_out):
37        return ptc(
38          grid, U, paths, ot_i,
39          ot_w, n_in, n_out
40        )
41
42      # combined multigrid model
43      model_multi_grid = g. ml.model.sequence(
44        l.parallel(
45          l.sequence(),
46          l.sequence(
47            restrict,
48            coarse_lptc(1, 1),
49            prolong
50          )
51        ),
52        fine_ptc(2, 2),
53        fine_ptc(2, 2),
54        fine_ptc(2, 2),
55        fine_ptc(2, 1),
56      )
```

[1] A. S. Kronfeld *et al.* (USQCD Collaboration), Lattice QCD and particle physics, arXiv:2207.07641.

[2] P. Boyle *et al.*, Lattice QCD and the computational frontier, arXiv:2204.00039.

[3] C. Lehner *et al.*, Grid Python Toolkit (GPT), https://github.com/lehner/gpt.

[4] J. Brannick, R. C. Brower, M. A. Clark, J. C. Osborn, and C. Rebbi, Adaptive Multigrid Algorithm for Lattice QCD, Phys. Rev. Lett. **100,** 041601 (2008).

[5] R. Babich, J. Brannick, R. C. Brower, M. A. Clark, T. A. Manteuffel, S. F. McCormick, J. C. Osborn, and C. Rebbi, Adaptive Multigrid Algorithm for the Lattice Wilson-Dirac Operator, Phys. Rev. Lett. **105,** 201602 (2010).

[6] A. Frommer, K. Kahl, S. Krieg, B. Leder, and M. Rottmann, Adaptive aggregation based domain decomposition multigrid for the lattice Wilson Dirac operator, SIAM J. Sci. Comput. **36,** A1581 (2014).

[7] J. Brannick, A. Frommer, K. Kahl, B. Leder, M. Rottmann, and A. Strebel, Multigrid preconditioning for the overlap operator in lattice QCD, Numer. Math. **132,** 463 (2016).

[8] R. C. Brower, M. A. Clark, A. Strelchenko, and E. Weinberg, Multigrid algorithm for staggered lattice fermions, Phys. Rev. D **97,** 114513 (2018).

[9] R. C. Brower, M. A. Clark, D. Howarth, and E. S. Weinberg, Multigrid for chiral lattice fermions: Domain wall, Phys. Rev. D **102,** 094517 (2020).

[10] P. Boyle and A. Yamaguchi, Comparison of domain wall fermion multigrid methods, arXiv:2103.05034.

[11] U. Trottenberg, C. Oosterlee, and A. Schuller, *Multigrid* (Elsevier Science, New York, 2000).

[12] A. Katrutsa, T. Daulbaev, and I. Oseledets, Deep Multigrid: Learning prolongation and restriction matrices, arXiv:1711.03825.

[13] J. He and J. Xu, MgNet: A unified framework of multigrid and convolutional neural network, Sci. China Math. **62,** 1331 (2019).

[14] D. Greenfeld, M. Galun, R. Basri, I. Yavneh, and R. Kimmel, Learning to optimize multigrid PDE solvers, in *Proceedings of the 36th International Conference on Machine Learning* (2019), pp. 2415–2423, arXiv:1902.10248.

[15] I. Luz, M. Galun, H. Maron, R. Basri, and I. Yavneh, Learning algebraic multigrid using graph neural networks, in *Proceedings of the 37th International Conference on Machine Learning* (2020), pp. 6489–6499, arXiv:2003.05744.

[16] M. Eliasof, J. Ephrath, L. Ruthotto, and E. Treister, MGIC: Multigrid-in-channels neural network architectures, arXiv:2011.09128.

[17] R. Huang, R. Li, and Y. Xi, Learning optimal multigrid smoothers via neural networks, SIAM J. Sci. Comput. **45,** S199 (2021).

[18] A. van Betteray, M. Rottmann, and K. Kahl, MGiaD: Multigrid in all dimensions. Efficiency and robustness by coarsening in resolution and channel dimensions, arXiv:2211.05525.

[19] T. Cohen and M. Welling, Group equivariant convolutional networks, in *Proceedings of The 33rd International Conference on Machine Learning* (2016), pp. 2990–2999, arXiv:1602.07576.

[20] T. S. 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* (2019), pp. 1321–1330, arXiv:1902.04615.

[21] G. Kanwar, M. S. Albergo, D. Boyda, K. Cranmer, D. C. Hackett, S. Racanière, D. J. Rezende, and P. E. Shanahan, Equivariant Flow-Based Sampling for Lattice Gauge Theory, Phys. Rev. Lett. **125,** 121601 (2020).

[22] D. Boyda, G. Kanwar, S. Racanière, D. J. Rezende, M. S. Albergo, K. Cranmer, D. C. Hackett, and P. E. Shanahan, Sampling using $SU(N)$ gauge equivariant flows, Phys. Rev. D **103,** 074504 (2021).

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

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

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

[26] S. Calì, D. C. Hackett, Y. Lin, P. E. Shanahan, and B. Xiao, Neural-network preconditioners for solving the Dirac equation in lattice gauge theory, Phys. Rev. D **107,** 034508 (2023).

[27] M. Lüscher, Solution of the Dirac equation in lattice QCD using a domain decomposition method, Comput. Phys. Commun. **156,** 209 (2004).

[28] Y. Osaki and K.-I. Ishikawa, Domain Decomposition method on GPU cluster, Proc. Sci. LATTICE2010 (**2010**) 036.

[29] R. Babich, M. A. Clark, B. Joo, G. Shi, R. C. Brower, and S. Gottlieb, Scaling Lattice QCD beyond 100 GPUs, in *SC11 International Conference for High Performance Computing, Networking, Storage and Analysis* (2011), arXiv:1109.2935.

[30] J. Tu, M. A. Clark, C. Jung, and R. Mawhinney, Solving DWF Dirac equation using multi-splitting preconditioned conjugate gradient with tensor cores on NVIDIA GPUs, in *PASC '21: Proceedings of the Platform for Advanced Scientific Computing Conference* (2021), pp. 1–11, arXiv:2104.05615.

[31] K. G. Wilson, Confinement of quarks, Phys. Rev. D **10,** 2445 (1974).

[32] B. Sheikholeslami and R. Wohlert, Improved continuum limit lattice action for QCD with Wilson fermions, Nucl. Phys. **B259,** 572 (1985).

[33] Y. Saad, A flexible inner-outer preconditioned GMRES algorithm, SIAM J. Sci. Comput. **14,** 461 (1993).

[34] Y. Shamir, Chiral fermions from lattice boundaries, Nucl. Phys. **B406,** 90 (1993).

[35] V. Furman and Y. Shamir, Axial symmetries in lattice QCD with Kaplan fermions, Nucl. Phys. **B439,** 54 (1995).

[36] N. Nachtigal, S. Reddy, and L. Trefethen, How fast are nonsymmetric matrix iterations?, SIAM J. Matrix Anal. Appl. **13,** 778 (1992).

[37] P. A. Boyle, Hierarchically deflated conjugate gradient, arXiv:1402.2585.

[38] A. Yamaguchi and P. Boyle, Hierarchically deflated conjugate residual, Proc. Sci. LATTICE2016 (**2016**) 374.

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

[40] M. Lüscher, Local coherence and deflation of the low quark modes in lattice QCD, J. High Energy Phys. 07 (2007) 081.