Gauge-equivariant multigrid neural networks

Daniel Knüttel, Christoph Lehner and Tilo Wettig[*]

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

E-mail: tilo.wettig@ur.de

We show how multigrid preconditioners for the Wilson-clover Dirac operator can be constructed using gauge-equivariant neural networks. For the multigrid solve we employ parallel-transport convolution layers. For the multigrid setup we consider two versions: the standard construction based on the near-null space of the operator and a gauge-equivariant construction using pooling and subsampling layers. We show that both versions eliminate critical slowing down. We also show that transfer learning works and that our approach allows for communication-avoiding algorithms on large machines. In the outlook we discuss how the multigrid setup can be accelerated using gauge-invariant properties of the gauge field.

The 40th International Symposium on Lattice Field Theory (Lattice 2023)
July 31st - August 4th, 2023
Fermi National Accelerator Laboratory

[*]Speaker

1. Introduction

The wall-clock time of lattice QCD simulations is typically dominated by the solution of the Dirac equation. The iteration count of an iterative solver is determined by the condition number of the Dirac matrix D, which increases dramatically in the continuum limit and for physical quark mass ("critical slowing down"). Let us assume that the residual vector in the iterative solver is expanded in the eigenvectors of D. This expansion contains contributions from the high and low eigenmodes of D. A good preconditioner can reduce both contributions and thus eliminate critical slowing down. The current state of the art is represented by multigrid algorithms [1–3], which consist of two components: a smoother and a coarse-grid correction that reduce the contributions of the high and low modes to the residual, respectively. Here, we show that such preconditioners can be learned by gauge-equivariant neural networks. For details we refer to [4, 5].

2. Parallel-transport convolution layers and high-mode preconditioner

An important guiding principle in the construction of a neural network (called "model" in the following) for lattice QCD is gauge equivariance, i.e., the requirement that the transformation implemented by the model commutes with gauge transformations [6–8]. Building gauge equivariance into the model implies that the model does not have to learn the gauge symmetry and thus can achieve the same expressivity with fewer weights. To this end we define a parallel-transport operator T_p for a path $p = (p_1, \dots, p_{n_p})$ consisting of a sequence of hops $p_i = \pm\hat{\mu}$ in a single direction by

$$T_p = H_{p_{n_p}} \cdots H_{p_1} \quad \text{with} \quad H_\mu \varphi(x) = U_\mu^\dagger(x-\hat{\mu})\varphi(x-\hat{\mu}) , \tag{1}$$

where we employed the usual lattice notation and φ denotes a field with both gauge and non-gauge degrees of freedom. (In lattice QCD with gauge group SU(3) and Wilson fermions, φ has $3 \cdot 4 = 12$ d.o.f. at every lattice site.) T_p acts on the field φ and returns a new field $T_p\varphi$. By construction, φ and $T_p\varphi$ transform in the same way under a gauge transformation. Generalizing [7], we define a gauge-equivarant parallel-transport convolution layer by its action on a field,

$$\psi_a(x) \overset{\text{PTC}}{=} \sum_b \sum_{p\in P} W_{ab}^p T_p \varphi_b(x) \quad \text{or} \quad \psi_a(x) \overset{\text{LPTC}}{=} \sum_b \sum_{p\in P} W_{ab}^p(x) T_p \varphi_b(x) , \tag{2}$$

where a and b are feature indices, P denotes a set of paths, the W_{ab}^p are trainable layer weights that act on the non-gauge indices of the field, and L stands for local (i.e., W depends on x). In the context of lattice QCD, the W_{ab}^p are 4×4 spin matrices. We do not include an activation function since we want to learn a linear preconditioner. Graphically, we represent a feature by a plane and a layer by an arrow between planes, with the paths defining the layer drawn on the plane on which the layer acts (see Fig. 1 below). Note that our approach naturally allows for communication-avoiding algorithms: On a large machine we can choose not to communicate information between subvolumes by setting the links $U_\mu(x)$ connecting subvolumes to zero. The resulting (small) loss in performance may be amortized by the time saved on communication, leading to an overall gain in wall-clock time.

Since the high eigenmodes of the Dirac matrix are related to the short-distance behavior, we can construct a PTC-based high-mode preconditioner consisting of one or two layers with a small number of hops. Our performance measure is the iteration count gain, defined as the ratio of the iteration counts of the outer solver (GMRES in our case) without and with preconditioner. For our numerical tests we use the Wilson-clover Dirac operator D_{WC} with $c_{\mathrm{SW}} = 1$ on an $8^3 \times 16$ pure-gauge configuration with $\beta = 6.0$. The quark mass m was tuned to near criticality to make the solution of the Dirac equation a challenging problem. We train the model M by generating random vectors v and optimizing the cost function

$$C = |M D_{\mathrm{WC}} v - v|^2 , \tag{3}$$

which is dominated by the high modes of D_{WC}. Since our training data set is arbitrarily large we do not need to add a regulator to avoid overfitting. Here and below we used the Adam optimizer [9]. We obtained an iteration count gain of about 5 (for a single layer with zero- and one-hop paths) to 10 (for two layers with zero- and one-hop paths or one layer with up to two-hop paths). We also observed that transfer learning worked with no or very little retraining for (i) a different configuration from the same ensemble, (ii) a configuration with a different β and (iii) a different quark mass.

3. Low-mode preconditioner: standard and gauge-equivariant construction

To obtain a low-mode preconditioner we could, in principle, construct a deep network of (L)PTC layers to propagate information over long distances. However, we choose to follow the multigrid paradigm instead and try to preserve the low-mode part of the Dirac spectrum on a coarse lattice. For this we need restriction and prolongation operators ("multigrid setup") to take us back and forth between fine and coarse grid. We present two versions: the standard multigrid setup [4] and a gauge-equivariant version [5]. We also describe how the Dirac equation is solved approximately on the coarse grid in each version.

In the standard multigrid setup, there are no gauge degrees of freedom (and thus no gauge transformations) on the coarse grid. We define restriction and prolongation layer (with RL = PL†) by

$$\tilde{\psi}(y) \overset{\mathrm{RL}}{=} \sum_{x \in B(y)} W(y,x)\varphi(x) \quad \text{and} \quad \psi(x) \overset{\mathrm{PL}}{=} W(y,x)^\dagger \tilde{\varphi}(y) , \tag{4}$$

where the tilde denotes fields on the coarse grid, x and y are sites on the fine and coarse grid, respectively, and $B(y)$ denotes the block of sites on the fine grid corresponding to site y on the coarse grid. The $W(y,x)$ are layer weights. In the standard setup they are not trainable but constructed from a set of vectors u_i in the near-null space of D. Specifically,

$$W(y,x)^\dagger = \sum_{i=1}^{s} \bar{u}_i^y(x) \hat{e}_i^\dagger , \tag{5}$$

where the $\hat{e}_i$ are the canonical unit vectors on the coarse grid and the $\bar{u}_i^y$ are the u_i blocked on the sites of $B(y)$ and then orthonormalized within each block. The coarse-grid operator is defined as

$$\tilde{D} = \mathrm{RL} \circ D_{\mathrm{WC}} \circ \mathrm{PL} . \tag{6}$$

For the approximate solution of the Dirac equation on the coarse grid we use a preconditioner $\tilde{M}$ consisting of a single LPTC layer with zero- and one-hop paths and gauge fields replaced by $\mathbb{1}$. We use the same training strategy as before, but now with cost function $C = |\tilde{M}\tilde{D}v - v|^2$.

We now turn to the gauge-equivarant version (for earlier related work see, e.g., [10–13]). The coarse grid is the same as before, but the coarse fields now have the same gauge indices as on the fine grid (and non-gauge indices as well). For each block $B(y)$ on the fine grid we define a reference site $B_r(y) \subset B(y)$. Given a gauge transformation Ω on the fine grid, we require that the coarse fields transform as $\tilde{\varphi}(y) \to \tilde{\Omega}(y)\tilde{\varphi}(y)$, where $\tilde{\Omega}(y) = \Omega(B_r(y))$. This can be achieved if we define restriction and prolongation layer by

$$\mathrm{RL} = \mathrm{SubSample} \circ \mathrm{Pool} \quad \text{and} \quad \mathrm{PL} = \mathrm{Pool}^\dagger \circ \mathrm{SubSample}^\dagger \tag{7}$$

with pooling layer and subsampling layer given by

$$\mathrm{Pool}\,\varphi(x) = \sum_{q\in Q} W_q(x) T_q \varphi(x) \quad \text{and} \quad \mathrm{SubSample}\,\varphi(y) = \varphi(B_r(y))\,. \tag{8}$$

Here, $q = (p, \bar{U})$ denotes a path p and a (possibly smeared) gauge field $\bar{U}$, and the $W_q(x)$ are trainable layer weights acting on the non-gauge indices of φ (in lattice QCD they are 4×4 spin matrices). A set of paths is called complete if it connects every site in $B(y)$ to $B_r(y)$ exactly once, and we require Q to contain one or more such complete sets of paths. The pooling layer corresponds to a gauge-equivariant averaging procedure. The subsampling layer then simply picks out the field after pooling at the reference site. To train the layer weights we used the cost function

$$C = |\,\mathrm{PL} \circ \mathrm{RL}\, v_\ell - v_\ell|^2 + |\,\mathrm{PL} \circ \mathrm{RL}\, v_h - P_\ell v_h|^2 + |\,\mathrm{RL} \circ \mathrm{PL}\, v_c - v_c|^2\,, \tag{9}$$

where v_ℓ denotes fine-grid vectors from the near-null space, v_h and v_c are random vectors on fine and coarse grid, respectively, and $P_\ell = W^\dagger W$ is the blocked low-mode projector with W from Eq. (5). The first term in the cost function trains $\mathrm{PL} \circ \mathrm{RL}$ to be an autoencoder that preserves the low modes, the second term trains $\mathrm{PL} \circ \mathrm{RL}$ to project high eigenmodes to zero, and the third term encourages $\mathrm{RL} \circ \mathrm{PL} = \mathbb{1}$ so that $\mathrm{PL} \circ \mathrm{RL}$ is a proper projection operator.

For the approximate coarse-grid solve we employ a similar LPTC model as in the standard version, but now with coarse-grid gauge fields $\tilde{U}_\mu(y)$ (instead of $\mathbb{1}$). There are several options to define coarse-grid gauge fields that transform correctly under gauge transformations. We investigated two options: "plain" and "Galerkin" coarse-grid gauge fields. In the first case, we assume that for two neighboring sites y and y' on the coarse grid, the corresponding reference sites on the fine grid are connected by a straight path that aligns with a coordinate axis, i.e., $B_r(y') - B_r(y) = b\hat{\mu}$ with $b \in \mathbb{N}^+$. Then $\tilde{U}_\mu(y) = U_\mu(B_r(y)) \cdots U_\mu(B_r(y) + (b-1)\hat{\mu})$. In the second case, we define $\tilde{U}_\mu(y) = \tilde{D}(y, y+\hat{\mu})$ with $\tilde{D} = \mathrm{RL} \circ D_{\mathrm{WC}} \circ \mathrm{PL}$. Note that in this case the coarse-grid gauge fields are, in general, no longer elements of the original gauge group.

4. Multigrid model and results

We now combine the high- and low-mode models to learn a multigrid model that approximates the short- and long-distance features of D^{-1}, see Fig. 1. We first construct a short-distance model

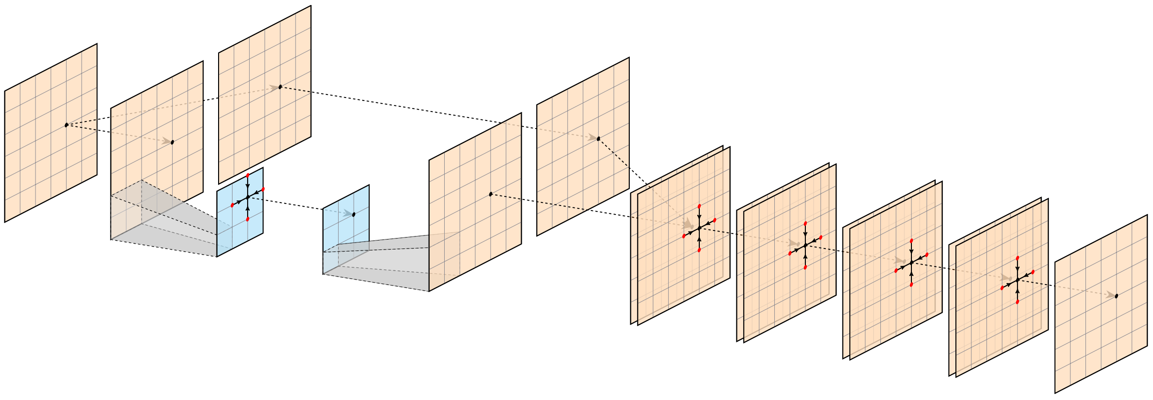

Figure 1: Gauge-equivariant multigrid model. The smoother is represented by the four double planes and arrows, while the coarse-grid correction is represented by the gray frustums and the blue planes and arrows.

that accepts a second input feature (an initial guess). This model plays the role of the smoother in the multigrid method, with the initial guess from the long-distance model acting on the coarse grid. Assuming that we have a high-mode model M_h approximating D^{-1}, the smoother maps the tuple (u_k, b) to u_{k+1} according to

$$u_{k+1} = (\mathbb{1} - M_\mathrm{h} D)u_k + M_\mathrm{h} b = u_k + M_\mathrm{h}(b - Du_k)\,. \tag{10}$$

Hence the smoother model M_s must have two input features and one output feature. To have another control parameter we choose M_s to map (u_k, b) to a u_{k+r} with $r \in \mathbb{N}^+$. (In the full multigrid model $r = 2$ performed better than $r = 1$.) Since both D and M_h can be represented by (L)PTC layers, every smoother iteration corresponds to $2r$ successive layers. The cost function for the training of M_s is $C = |M_s(u_k, b) - u_{k+r}|^2$ with random vectors u_k, b and u_{k+r} given by Eq. (10).

In the combined multigrid model in Fig. 1, we first duplicate the input feature and preserve one copy for smoother. We then restrict the other copy to the coarse grid, apply our coarse-grid model, and prolongate the result to the fine grid. The copy of the initial feature and the result of the coarse-grid model are the two input features of the smoother, represented by the last $2r = 4$ layers. Additional multigrid levels can be implemented by recursively replacing the coarse-grid layer by the entire model.

In Fig. 2 we present numerical results (for more results and details of the training, see [4, 5]). The iteration count on the vertical axis refers to the outer solver GMRES to precision 10^{-8}. In the plot on the left we observe that without preconditioner the iteration count increases dramatically as the quark mass is tuned to a critical value. Using the smoother-only model as a preconditioner reduces the iteration count but still exhibits critical slowing down (CSD), while the gauge-equivarant Galerkin model eliminates CSD. In the plot on the right we see the same data for the Galerkin model but on a different scale on the vertical axis. The gauge-equivariant multigrid model based on plain coarse-grid gauge fields performs well but shows small remnants of CSD. The performance of the multigrid model based on the standard multigrid setup (called “Original” in the title of the plot) is similar to that of the Galerkin model.

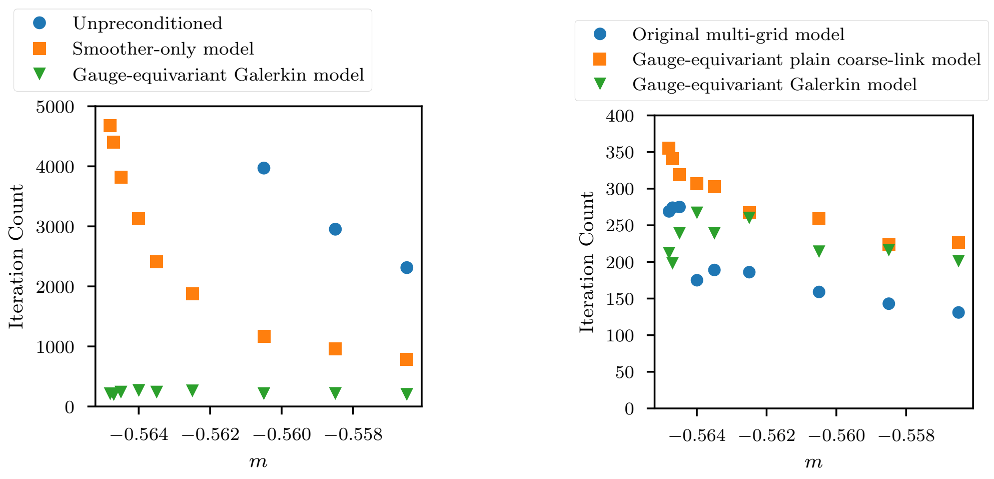

Figure 2: Original model and gauge-equivariant Galerkin model both eliminate critical slowing down.

5. Summary and outlook

We have reformulated the problem of constructing a (multigrid) preconditioner in the language of gauge-equivariant neural networks. We find that such networks can learn the general paradigms of multigrid, significantly reduce the iteration count of the outer solver, and eliminate critical slowing down. This is true for both the standard and the gauge-equivariant multigrid setup. We have also seen that transfer learning works, i.e., very little or no extra training is needed if we change the gauge-field configuration or system parameters like κ and β. Furthermore, we can implement communication avoidance naturally. All layers presented here are implemented in GPT [14].

An important problem, which we will address in future work, is to reduce the cost of the multigrid setup. In the standard version of the setup, restriction and prolongation layer do not have trainable weights, and hence the setup cost is comparable to the current state of the art. However, in the gauge-equivariant version, there are trainable weights. They are gauge-invariant spin matrices that could be learned from gauge-invariant features of the gauge field such as energy density or topological charge density, see Fig. 3.

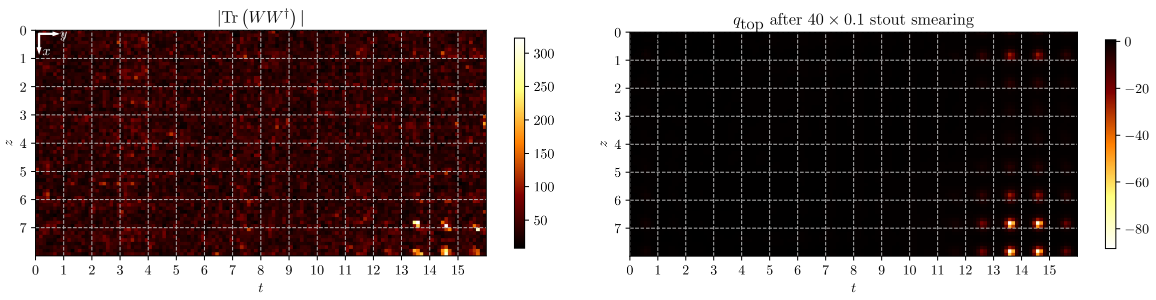

Figure 3: We compare the site-dependent norm of one of our trained pooling layer weights (left) with the topological charge density after 40 steps of $\rho = 0.1$ stout smearing (right). The plot shows the entire four-dimensional fields as two-dimensional t–z grids of two-dimensional y–x grids, thereby covering all $8^3 \times 16$ sites.

References

[1] J. Brannick, R.C. Brower, M.A. Clark, J.C. Osborn and C. Rebbi, *Adaptive Multigrid Algorithm for Lattice QCD*, *Phys. Rev. Lett.* **100** (2008) 041601 [0707.4018].

[2] R. Babich, J. Brannick, R.C. Brower, M.A. Clark, T.A. Manteuffel, S.F. McCormick et al., *Adaptive multigrid algorithm for the lattice Wilson-Dirac operator*, *Phys. Rev. Lett.* **105** (2010) 201602 [1005.3043].

[3] 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** (2014) A1581 [1303.1377].

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

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

[6] T. Cohen and M. Welling, *Group Equivariant Convolutional Networks*, in *Proceedings of The 33rd International Conference on Machine Learning (ICML)*, pp. 2990–2999, 2016 [1602.07576].

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

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

[9] D.P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, in *3rd International Conference on Learning Representations, ICLR 2015*, 2015 [1412.6980].

[10] A. Hulsebos, J. Smit and J.C. Vink, *Multigrid Monte Carlo for a Bose Field in an External Gauge Field*, *Nucl. Phys. B* **331** (1990) 531.

[11] R.C. Brower, C. Rebbi and E. Vicari, *Projective multigrid method for propagators in lattice gauge theory*, *Phys. Rev. D* **43** (1991) 1965.

[12] T. Kalkreuter, *Multigrid methods for the computation of propagators in gauge fields*, *Int. J. Mod. Phys. C* **5** (1994) 629 [hep-lat/9212021].

[13] R. Ben-Av, M. Harmatz, P.G. Lauwers and S. Solomon, *Parallel transported multigrid for inverting the Dirac operator: Variants of the method and their efficiency*, *Nucl. Phys. B* **405** (1993) 623.

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