Gauge-equivariant neural networks as preconditioners in lattice QCD

Christoph Lehner and Tilo Wettig (University of Regensburg)
arXiv:2302.05419 [hep-lat]

RIKEN R-CCS Workshop
Challenges and opportunities in Lattice QCD simulations and related fields
Kobe, February 15, 2023

Outline

Introduction
Gauge-equivariant layers
Wilson-clover Dirac operator
High-mode preconditioners
Low-mode preconditioners
Multi-grid preconditioners
Summary and outlook

Introduction

Take-home messages

- We reformulate 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 and significantly reduce the iteration count of the outer solver.
- Transfer learning: If we change the configuration or parameters like κ and β, only very little or no extra training is needed.
- We can implement communication avoidance naturally.
- We provide a flexible implementation interface (GPT) for experimentation and further studies.

Preconditioning

- In lattice QCD, wall-clock time is typically dominated by solution of Dirac equation

$$Du = b$$

  Usually done by an iterative solver (here, FGMRES)

- Time to solution is determined by condition number of Dirac matrix
  - Condition number increases dramatically in physical quark-mass and continuum limit
- Way out: Preconditioning
  - Find a preconditioner M such that $M \approx D^{-1}$
  - Define $v = M^{-1}u$ and use

$$DMM^{-1}u = (DM)v = b$$

    to solve for v with preconditioned matrix DM (smaller condition number)
  - Then $u = Mv$

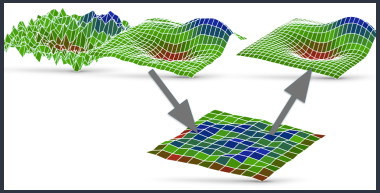
Low and high modes

- Consider the eigendecomposition of D

$$D = \sum_n \lambda_n |n\rangle\langle n|$$

  → Preconditioner should approximate low-mode and high-mode components of D^{-1}

- State-of-the-art algorithms (multigrid) are designed to do this
- We follow this paradigm, but here we learn the preconditioner



Source: https://summerofhpc.prace-ri.eu/multithreading-the-multigrid-solver-for-lattice-qcd

Related work

Related work falls into three categories

1. Neural networks for multigrid (but not for gauge theories), e.g.,
   - Katrutsa, Daulbaev, Oseledets — arXiv:1711.03825 [math.NA]
   - He & Xu — arXiv:1901.10415 [cs.CV]
   - Greenfeld, Galun, Basri, Yavneh, Kimmel — arXiv:1902.10248 [cs.LG]
   - Eliasof, Ephrath, Ruthotto, Treister — arXiv:2011.09128 [cs.CV]
   - Huang, Li, Xi — arXiv:2102.12071 [math.NA]
2. Gauge-equivariant neural networks (but not for solving Dirac equation), e.g.,
   - Cohen, Weiler, Kicanaoglu, Welling — arXiv:1902.04615 [cs.LG]
   - Finzi, Stanton, Izmailov, Wilson — arXiv:2002.12880 [stat.ML]
   - Luo, Carleo, Clark, Stokes — arXiv:2012.05232 [cond-mat.str-el]
   - Kanwar et al. — arXiv:2003.06413 [hep-lat]
   - Boyda et al. — arXiv:2008.05456 [hep-lat]
   - Favoni, Ipp, Müller, Schuh — arXiv:2012.12901 [hep-lat]
   - Abbott et al. — arXiv:2207.08945 [hep-lat]

Related work

3. Multigrid algorithms in lattice QCD
   - Brannick, Brower, Clark, Osborn, Rebbi — arXiv:0707.4018 [hep-lat]
   - R. Babich et al. — arXiv:1005.3043 [hep-lat]
   - Frommer et al. — arXiv:1303.1377 [hep-lat]
   - Brannick et al. — arXiv:1410.7170 [hep-lat]
   - Brower, Clark, Strelchenko, Weinberg — arXiv:1801.07823 [hep-lat]
   - Brower, Clark, Howarth, Weinberg — arXiv:2004.07732 [hep-lat]
   - Boyle & Yamaguchi — arXiv:2103.05034 [hep-lat]

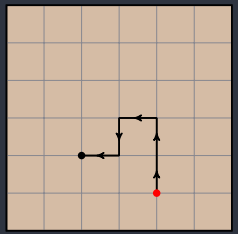
Gauge-equivariant layers

Parallel transport

- Consider a field $\varphi(x)$ with $x \in S$ (space-time lattice, dim $= d$) and $\varphi \in V_I = V_G \otimes V_{\bar{G}}$ (gauge space: $V_G = \mathbb{C}^N$, non-gauge space: $V_{\bar{G}} = \mathbb{C}^{\bar{N}}$)
- Also consider an SU(N) gauge field $U_\mu(x)$ acting on V_G
- Define the parallel-transport operator for a path $p = p_1, \ldots, p_{n_p}$ with $p_i \in \{\pm 1, \ldots, \pm d\}$

$$T_p = H_{p_{n_p}} \cdots H_{p_2} H_{p_1} \quad \text{with} \quad H_\mu \varphi(x) = U_\mu^\dagger(x - \hat{\mu})\varphi(x - \hat{\mu})$$

- H_μ transports information by a single hop in direction $\hat{\mu}$
- H_μ acts on field; new field $H_\mu \varphi$ is evaluated at x
- Example: $T_p = H_{-1} H_{-2} H_{-1} H_2 H_2$



Gauge equivariance

- A gauge transformation by $\Omega(x) \in \mathrm{SU}(N)$ acts in the usual way

$$\varphi(x) \to \Omega(x)\varphi(x)$$
$$U_\mu(x) \to \Omega(x)U_\mu(x)\Omega^\dagger(x+\hat{\mu})$$

- Such gauge transformations commute with T_p for any path p

$$T_p\varphi(x) \to \Omega(x)T_p\varphi(x)$$

- This is an example of gauge equivariance:

An object (here: φ) and the transformed object (here: $T_p\varphi$) transform in the same way under a gauge transformation.

Parallel-transport convolutions

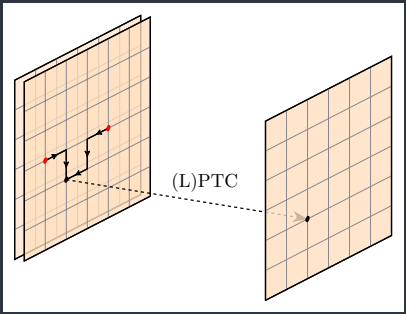
- Parallel-transport convolution layer and local parallel-transport convolution layer

$$\psi_a(x) \overset{\text{PTC}}{=} \sum_b \sum_{p \in P} W_a^{bp} T_p \varphi_b(x)$$

$$\psi_a(x) \overset{\text{LPTC}}{=} \sum_b \sum_{p \in P} W_a^{bp}(x) T_p \varphi_b(x)$$

  - $a =$ output feature index
  - $b =$ input feature index
  - $P =$ set of paths
  - W_a^{bp} acts in $V_{\tilde{G}}$ (here: 4×4 spin matrix)
  - Elements of W: "layer weights"

- Layers are gauge-equivariant
- No activation function since we want to learn a linear preconditioner

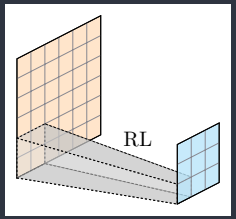


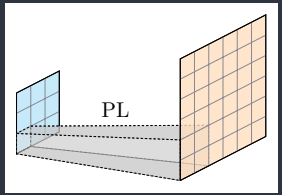
Restriction and prolongation layers

- On the coarse grid $\tilde{S}$ we have fields $\tilde{\varphi}(y)$ with $y \in \tilde{S}$ and $\tilde{\varphi} \in \tilde{V}_I$
- $\tilde{V}_I$ has no gauge degrees of freedom → no gauge transformations on $\tilde{V}_I$
- Restriction and prolongation layer (with $B =$ block map from $\tilde{S}$ to S)

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

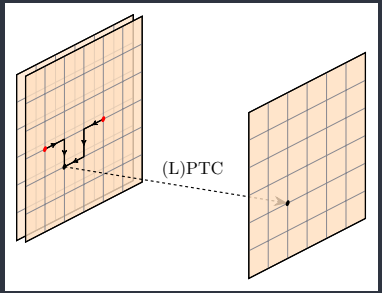
$$\psi(x) \overset{\text{PL}}{=} W(y,x)^{\dagger}\tilde{\varphi}(y)$$

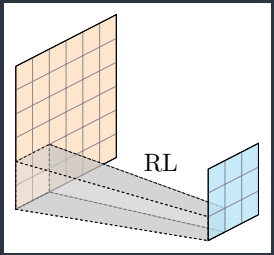




Graphical conventions for features and layers

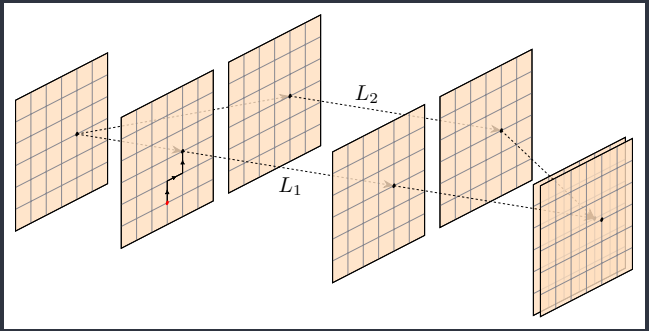
- A feature is represented by a plane
- A layer sits between planes and is represented by paths plus a dashed arrow
- For restriction/prolongation, a layer is represented by a square frustum





Parallel and identity layers

- Parallel layers act on the same input feature in parallel
- Identity layer (dashed arrow w/o paths): simple copy operation, i.e., output = input
- Example: (All layers except L_1 are identity layers)



Communication avoidance

- On machines with many nodes, subvolumes are assigned to different MPI processes
- We also consider models where no information is communicated between subvolumes (by setting the links $U_\mu(x)$ connecting subvolumes to zero)
- We find that the performance of these models (in terms of iteration count gain) is close to those with communication
  $\rightarrow$ Overall wall-clock time could be lower since no time is spent on communication

Wilson-clover Dirac operator

Dirac operator

- The Wilson Dirac operator can be written in terms of single hops:

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

- For Wilson-clover, consider closed paths with four hops and define

$$Q_{\mu\nu} = H_{-\mu}H_{-\nu}H_{+\mu}H_{+\nu} + H_{-\nu}H_{+\mu}H_{+\nu}H_{-\mu} + H_{+\nu}H_{-\mu}H_{-\nu}H_{+\mu} + H_{+\mu}H_{+\nu}H_{-\mu}H_{-\nu}$$

Then

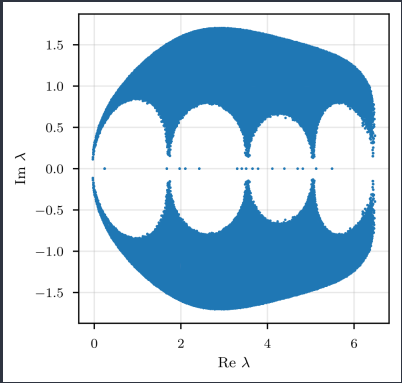
$$D_{\mathrm{WC}} = D_{\mathrm{W}} - \frac{c_{\mathrm{sw}}}{4}\sum_{\mu,\nu=1}^{4}\sigma_{\mu\nu}F_{\mu\nu}$$

with

$$F_{\mu\nu} = \frac{1}{8}(Q_{\mu\nu} - Q_{\nu\mu}) \qquad \sigma_{\mu\nu} = \frac{1}{2}(\gamma_\mu\gamma_\nu - \gamma_\nu\gamma_\mu)$$

Numerical details and eigenvalue spectrum

- $V = 8^3 \times 16$, $\beta = 6.0$ (pure gauge), $c_{\mathrm{SW}} = 1$, periodic boundary conditions for all fields
- $m = -0.6$ is chosen so that D_{WC} is tuned to near criticality (i.e., real part of smallest eigenvalue ≈ 0) $\rightarrow$ solution of $Du = b$ is challenging problem



High-mode preconditioners

Model setup and training strategy

- High-mode part of Dirac spectrum is related to short-distance behavior
  → Expect one or two layers with small number of hops to show gain in iteration count
- Consider a linear model M mapping a vector x to Mx
- Supervised learning approach with training step as follows:
  - Pick random vector v from Gaussian distribution (mean zero, standard deviation 1)
  - Compute training tuple $(D_{\mathrm{WC}}v, v)$ and optimize cost function

$$C = |MD_{\mathrm{WC}}v - v|^2$$

  → Model learns to map $D_{\mathrm{WC}}v$ to v (and hence $M \approx D_{\mathrm{WC}}^{-1}$)
  - Optimizer is Adam Kingma & Ba, arXiv:1412.6980 [cs.LG]
  - Derivatives w.r.t. model weights computed using backpropagation
- Training data set is unbounded in size → No need to add a regulator
- Cost function is dominated by high modes

Models chosen for high-mode preconditioner

- One layer, one hop (i.e., 9 paths)

$$T_0 = \mathbb{1}\,,\ T_1 = H_1\,,\ T_2 = H_2\,,\ T_3 = H_3\,,\ T_4 = H_4\,,\ T_5 = H_{-1}\,,\ T_6 = H_{-2}\,,\ T_7 = H_{-3}\,,\ T_8 = H_{-4}$$

- One layer, two hops: extend the above by 56 two-hop paths

$$H_a H_b \qquad \text{with} \qquad a, b \in \{-4, -3, -2, -1, 1, 2, 3, 4\} \qquad (a \neq -b)$$

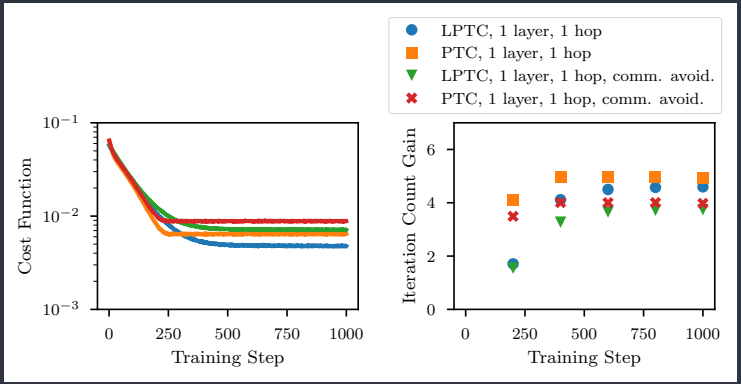
- "Deep" network of two one-hop layers:
  - $1 \to 1 \to 1$: Two successive layers with one hop each
  - $1 \to 2 \to 1$: Two output features in first layer, two input features in second layer
- PTC (layer weights constant) and LPTC (layer weights depend on x)
- Communication avoidance: $U_\mu(x) \equiv 0$ between subvolumes of size $4^3 \times 8$

Measure of performance

$$\text{Iteration count gain} = \frac{\text{Iteration count without preconditioner}}{\text{Iteration count with preconditioner}}$$

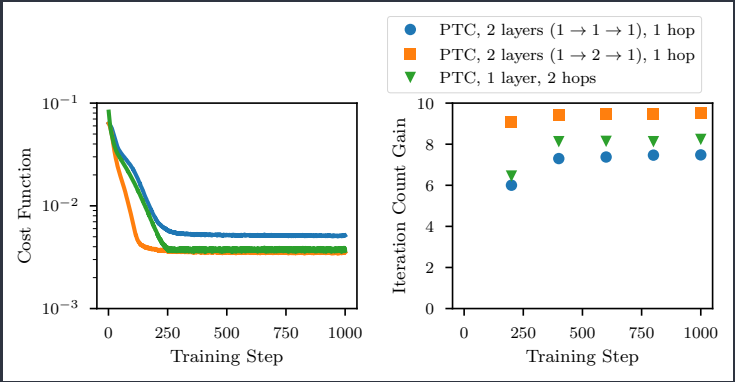
- Iteration count refers to outer solver (here, FGMRES)

Results for high-mode preconditioner (one layer, one hop)



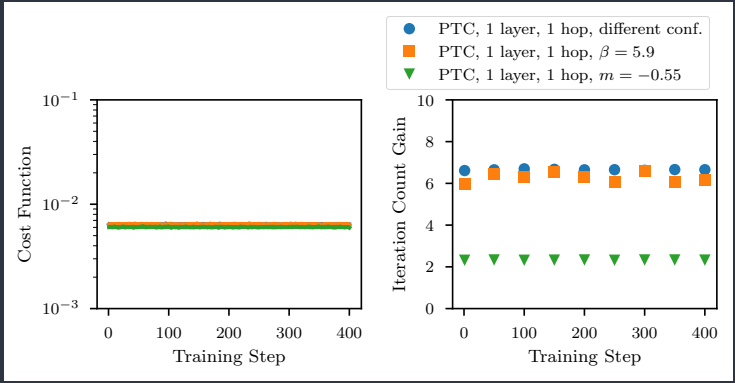
- No gain from LPTC (and they require more training)
- Communication-avoiding version only slightly worse (could be amortized)

Results for high-mode preconditioner (deep network or multiple hops)



- $1 \to 2 \to 1$ model performs best (and gives $\sim$ twice the gain of 1 layer/1 hop model)
- Since layers are linear, deep models are not more expressive than shallow models with same number of hops (but easier to train b/o smaller number of weights)
  $\to$ 2-hop model should reach similar performance with improved training procedure

Transfer learning



- No retraining required for (i) different configuration from same ensemble, (ii) configuration with different β, (iii) different mass
- $m = -0.55$ is not tuned to criticality → Easier initial problem → Smaller gain
- Performance varies slightly between configurations

Low-mode preconditioners

Possible approaches

- Low-mode part of Dirac spectrum is related to long-distance behavior
  → Need deep network of (L)PTC layers to propagate information over long distances
- Alternative: Use multigrid paradigm
  - Define coarse version of the lattice
  - Define restriction and prolongation operations (= layers)
  - Preserve low-mode part of Dirac spectrum Lüscher arXiv:0706.2298 [hep-lat]

Model setup

- Reminder: coarse lattice $= \tilde{S}$, internal vector space $= \tilde{V}_I$ with $s = \dim(\tilde{V}_I)$
- Find s vectors in the near-null space of D

$$Du_i \approx 0 \qquad (i = 1, \ldots, s)$$

  - Apply FGMRES for D with source vector $= 0$ and random initial guess (solve to 10^{-8})
  - This removes high-mode components and leaves linear combination of low modes
- Block the u_i
  - One site $y \in \tilde{S}$ corresponds to a set of sites (or block) $B(y) \in S$
  - Blocked vector u_i^y lives on the sites of $B(y)$
- Orthonormalize the u_i^y within each block $\rightarrow \bar{u}_i^y$
- Then the prolongation map (see slide 10) is defined by

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

with $x \in B(y)$ and $\hat{e}_i$ the canonical unit vectors of $\tilde{V}_I$

Training strategy

- Coarse-grid operator is defined as

$$\tilde{D} = RD_{\mathrm{WC}}P$$

  with R and P defined according to restriction and prolongation layers (slide 10)

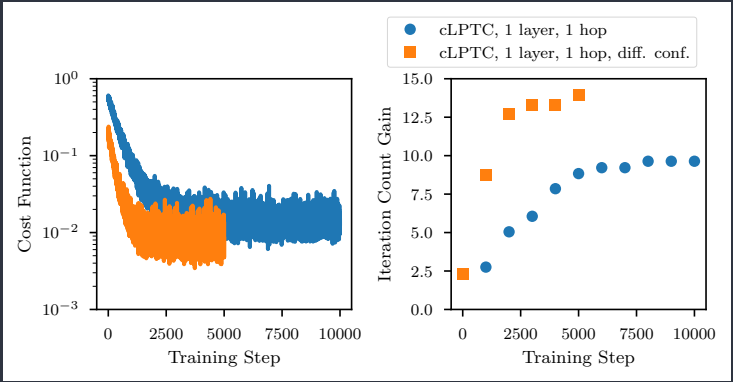
- Coarse-grid model $\tilde{M}$ contains single LPTC layer with zero- and one-hop paths and gauge fields replaced by $\mathbb{1}$ (layer is denoted by cLPTC)

- Same training strategy as before, with cost function

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

  (this avoids costly inversion of $\tilde{D}^{-1}$)

Results for low-mode preconditioner (cLPTC layer)



- Iteration count gain refers to inversion of $\tilde{D}$ (we use $\tilde{S} = 2^3 \times 4$ and $s = 12$)
- Longer training period compared to high-mode preconditioner
- Transfer learning works with moderate retraining

Multi-grid preconditioners

Model setup

- Combine the high- and low-mode models to learn a model M that approximates the short- and long-distance features of D^{-1}
- First create a short-distance model that accepts a second input feature (initial guess)
  - Model plays role of smoother in multigrid paradigm
  - Initial guess from long-distance model acting on coarse grid

Smoother model setup and training strategy

- Find a sequence of u_k that approximately solve $Du = b$ (exact solution for $k \to \infty$)
  - Assume we have a high-mode model M_h that approximates D^{-1}
  - Smoother maps the tuple (u_k, b) to u_{k+1}

$$\begin{aligned} u_{k+1} &= (\mathbb{1} - M_\mathrm{h} D) u_k + M_\mathrm{h} b \\ &= u_k + M_\mathrm{h}(b - D u_k) \qquad (*) \end{aligned}$$

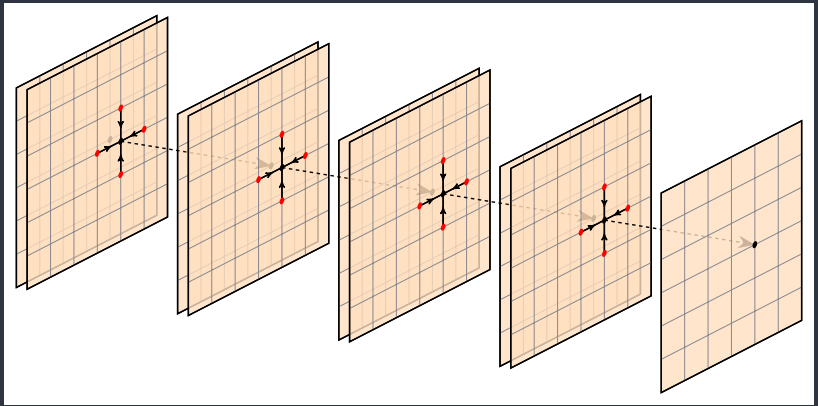
  ("iterative relaxation approach" or "defect correction" with defect $b - Du$)

- Both D and high-mode model M_h can be represented by (L)PTC layers
  $\to$ Train a model M_s to map (u_k, b) to a u_{k+r} (with $r \in \mathbb{N}^+$)
  - Model must have two input features and one output feature
  - Every iteration of $(*)$ corresponds to two (L)PTC layers
    $\to$ Construct M_s using $2r$ successive layers (here with up to one hop each)
  - We use $r = 2$ since it performed better than $r = 1$ in full multigrid model

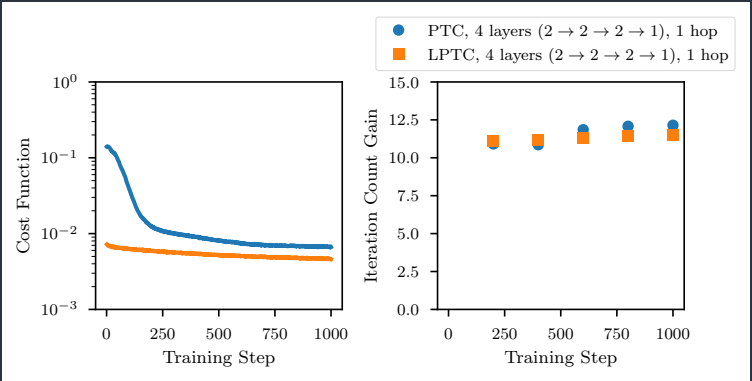
- Cost function (with random vectors u_k, b)

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

Smoother model

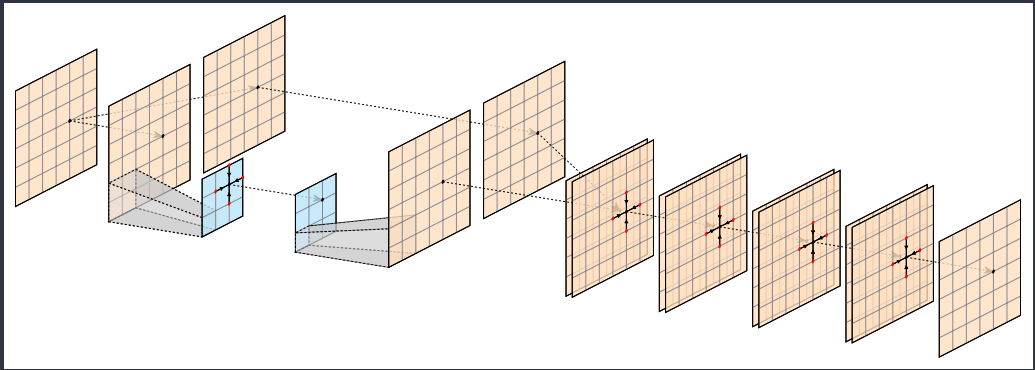


Results for smoother



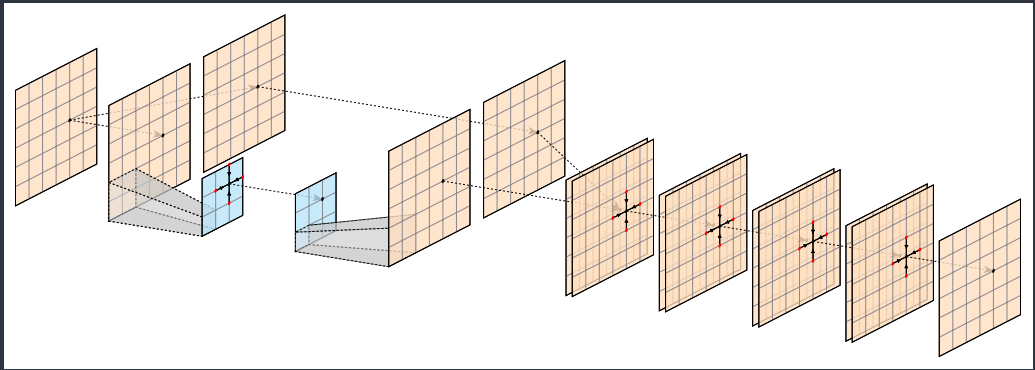
- Iteration count gain from using M_s as preconditioner for $Du = b$ with initial guess zero
- Performance is $\sim$ twice that of M_h with 1 layer/1 hop (since $r = 2$)
- Trained PTC model is used as initial weights for LPTC model (but no benefit from LPTC)

Combined two-level multigrid model



- Duplicate the input feature and preserve one copy for smoother
- Restrict other copy to coarse grid and apply our coarse-grid model
- Prolongate result to fine grid
- Combine copy of initial feature and result of coarse-grid model to two input features for smoother (= last four layers)

Combined two-level multigrid model



- Allows for efficient transport of information over both short and long-distances
- Additional multigrid levels: Recursively replace coarse-grid layer by entire model

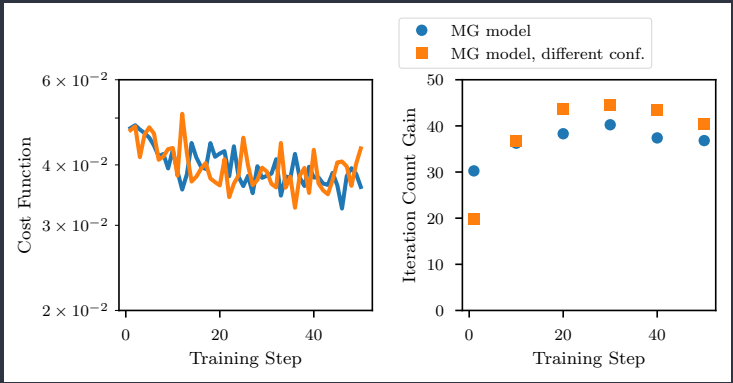
Training strategy for multigrid model

- In principle, model should work well with layer weights from individual models
- Performance can be further improved by continued training with cost function

$$C = |M\, b_h - u_h|^2 + |M\, b_\ell - u_\ell|^2 \qquad (*)$$

  - $b_h = D_{\mathrm{WC}} v_1$, $u_h = v_1$, $b_\ell = v_2$, $u_\ell = D_{\mathrm{WC}}^{-1} v_2$
  - v_1 and v_2 are random vectors with $|b_h| = |b_\ell| = 1$
- Focus on high- or low modes could be shifted by relative prefactor in $(*)$

Results for full multigrid model



- Performance greatly improved over individual high-/low-mode models
- Continued training converges very quickly
- Transfer learning works again after brief retraining

Summary and outlook

Take-home messages

- We reformulate 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 and significantly reduce the iteration count of the outer solver.
- Transfer learning: If we change the configuration or parameters like κ and β, only very little or no extra training is needed.
- We can implement communication avoidance naturally.
- We provide a flexible implementation interface (GPT) for experimentation and further studies.

Outlook

Many interesting directions for future work, e.g.,

- Learn the weights W of the restriction and prolongation layers directly (without computing the near-null vectors)
- Investigate the space of possible models more comprehensively
- Perform benchmarks on large lattices and compare to state-of-the-art multigrid (larger volumes should lead to larger iteration count gain)
- Apply our ideas to Dirac operators whose spectrum encircles the origin (e.g., DWF)
- Since no or very little retraining is needed between configurations: Apply our ideas to gauge-field generation (HMC)