# Gauge-equivariant neural networks as preconditioners in lattice QCD

*C. Lehner and T. Wettig*

arXiv:2302.05419v1 [hep-lat]

Hiroshi Ohno

Journal Club
June 9, 2023

# 概要

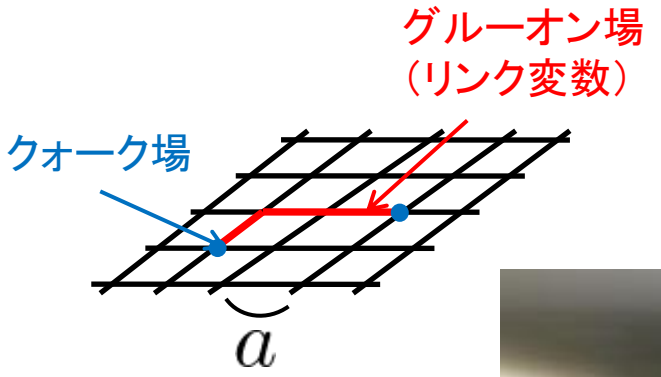
- 格子QCDのフェルミオン行列に対する連立一次方程式を解く際の前処理を、Gauge equivariant (= covariant) なニューラルネットで実現した。

- 学習に用いたゲージ配位とは別のゲージ配位についても、同じアンサンブル(シミュレーションパラメータ) ならば、少しの再学習をするだけで適用可能。

- 少しだけ違ったシミュレーションパラメータのゲージ配位についても適用できそう。

# 背景

- 格子QCD
  - 時空間を格子状に離散化 (可算有限自由度)
  - QCDの第一原理に基づく非摂動計算が可能
  - 大規模なモンテカルロシミュレーション




物理量 $\mathcal{O}$ の期待値

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}U \mathcal{O} \det D \, e^{-S_g} \rightarrow \frac{1}{N} \sum_{i=1}^{N} \mathcal{O}_i$$

4次元体積×内部自由度
= 数億次元の積分

$\frac{1}{Z} \det D \, e^{-S_g}$ の確率でゲージ配位 $U$ を生成

D: フェルミオン行列←大規模粗行列

多くの$D^{-1}$ の計算（CG法等）が必要
→ 計算のボトルネック

**フェルミオン行列の連立一次方程式を高速に解きたい！ → 前処理**

# 前処理

- CG法の反復回数
  - 行列 $A$ の条件数 $\kappa(A) = \left|\frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}\right|$ (最小固有値と最大固有値の比) に依存する
  - 条件数大 = 反復回数大 → 条件数は小さい方がよい

- 前処理
  - 条件数が小さい行列に変換して、より簡単に解ける問題にする

  → $Ax = b \ \rightarrow AM(M^{-1}x) = b, \kappa(AM) < \kappa(A)$ となる $M$ を見つける → $M \cong A^{-1}$

- 小さい固有値 (low mode) と大きい固有値 (high mode) それぞれの寄与について考える

# High mode 前処理

- フェルミオン行列 $D$ の High mode
  - 短距離の振る舞いと関係

- 短距離の振る舞いを取り入れた前処理 $M$ をニューラルネットで作る

- $M$ はゲージ対称性も考慮したものにしたい

  → Gauge-equivariance: $\varphi \to \Omega\varphi$ の時 $M\varphi \to \Omega M\varphi$

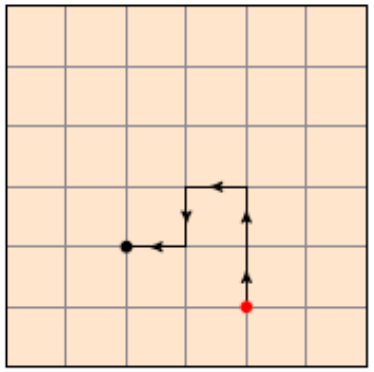
  → **Parallel-transport convolutions (PTC)**

# Parallel-transport convolutions (PTC)

- Parallel-transport operator

$$T_p = H_{p_{n_p}} \cdots H_{p_2} H_{p_1} \qquad H_{p_i}\varphi(x) = U^\dagger_{p_i}(x - \hat{p}_i)\varphi(x - \hat{p}_i)$$

サイト変数　　リンク変数



→ ゲージ変換

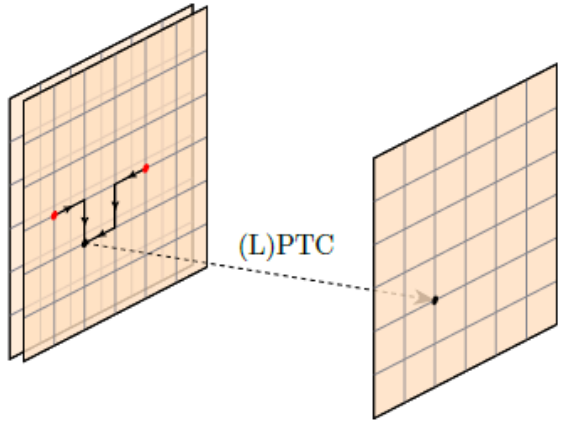
$$\varphi(x) \to \Omega(x)\varphi(x)\,,$$
$$U_\mu(x) \to \Omega(x)U_\mu(x)\Omega^\dagger(x+\hat{\mu})$$

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

- (L)PTC　　**重み *W*: 学習パラメータ**

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

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



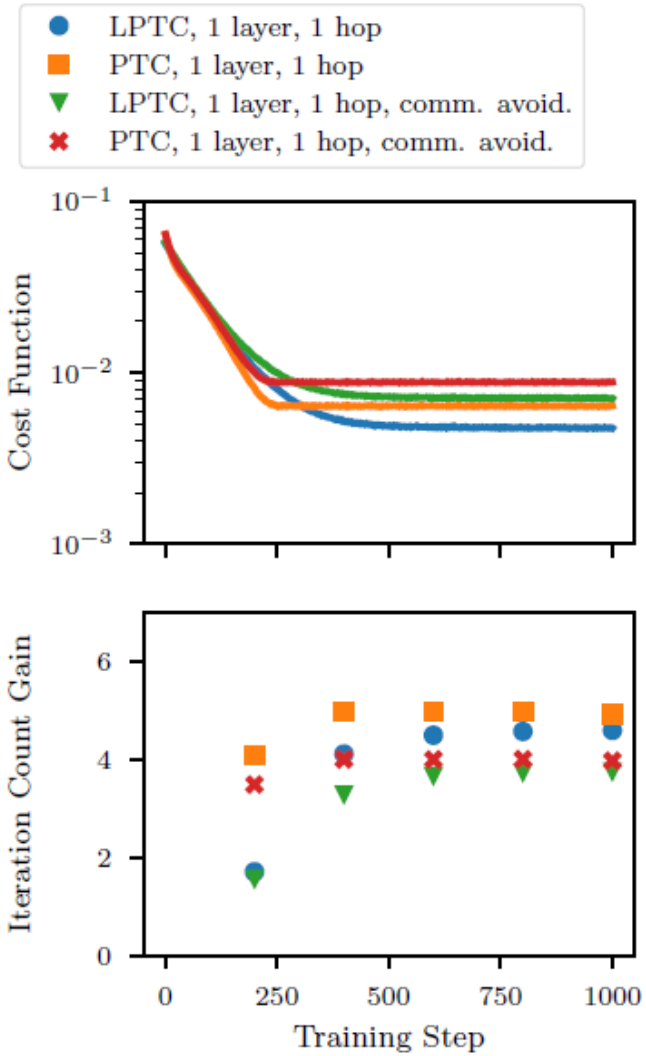
# High mode 前処理のモデル構築

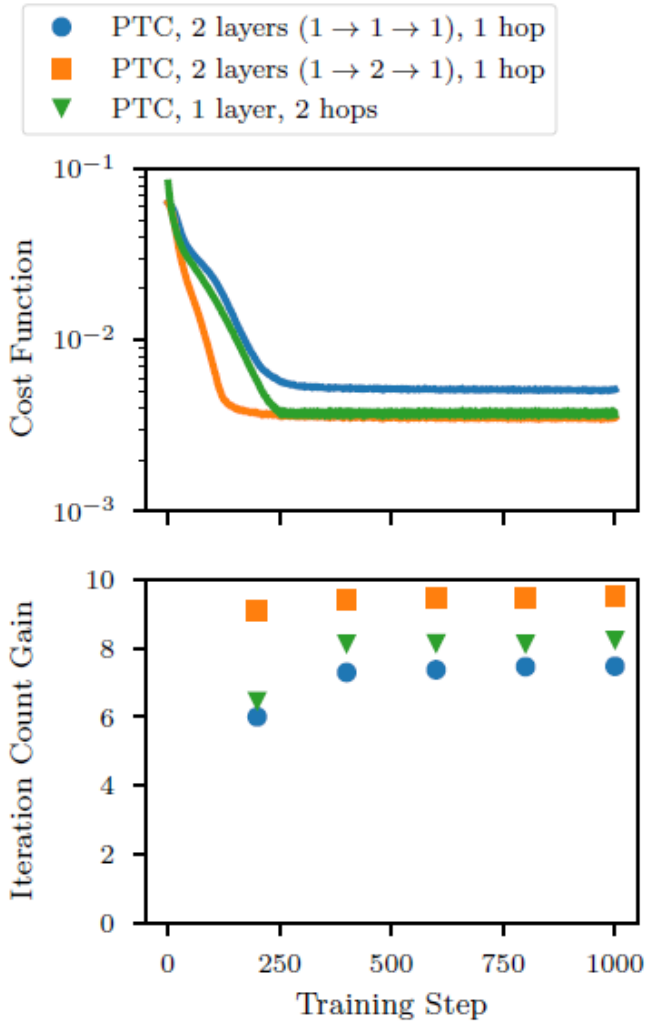
- (L)PTC を使って $M$ を作る
  - 短距離の振る舞いに興味があるので、短いホッピングのみ取り入れる
- 次の損失関数を最小化

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

  - $v$ は正規分布からランダムに生成
  - 様々な $v$ を使って学習 (損失関数を小さくするパラメータ $W$ の探索) を行う
    **($v$ は好きなだけ作れるので、学習データには困らない!)**
- 通信の回避
  - 並列化時に分割した領域をまたぐリンク変数を無視して通信を行わない

# High mode 前処理の結果 (1)



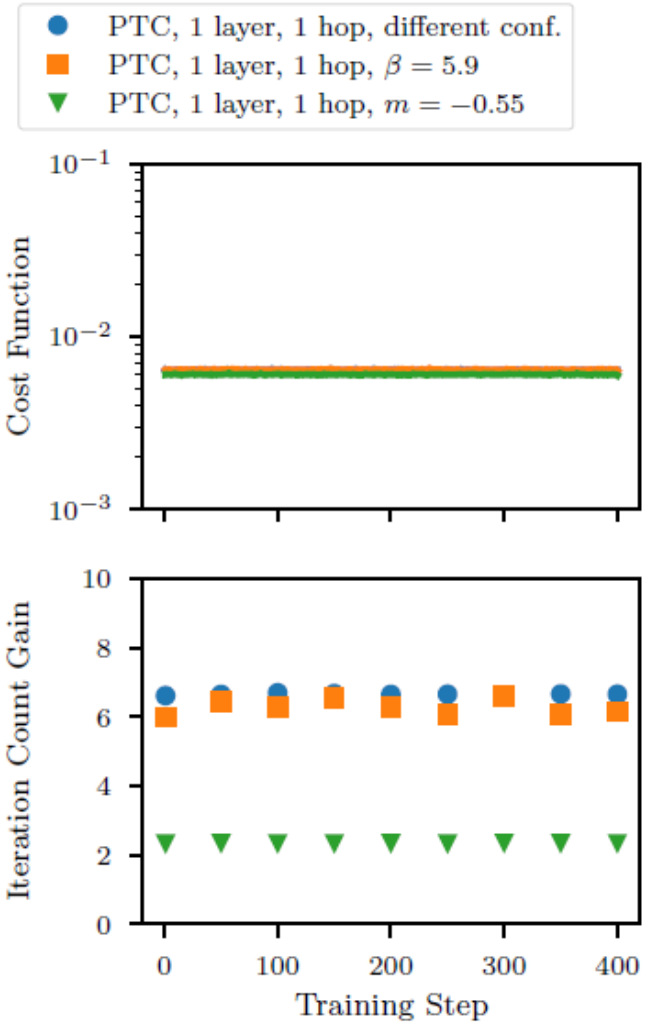

- LPTCはPTCよりパラメータ数がはるかに多い = 学習コストが高いが、より効果的なわけではない
- 通信を回避した学習でも、あまり効果が落ちない
- 多ホップ/多層のモデルを比較
  - 線形のモデルなので多層にしても表現力が高くなるとは限らない
  - 多ホップはパラメータ数が多く、学習コストが高くなる

  → 学習コストの面から、多層の方がよいかもしれない

$8^3 \times 16, \beta = 6.0, m = -0.6$ ($D$ の最小固有値の実部が0に近くなるように選択), Wilson gauge, Wilson-Clover fermion

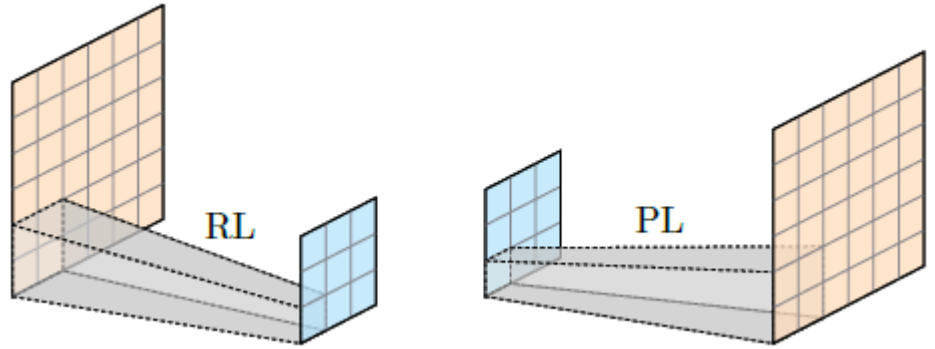
# High mode 前処理の結果 (2)

- 転移学習
  - 再学習はほとんど必要ない
  - 同じアンサンブルの別の配位についても効果がある
  - パラメータを少し変更しても効果がある
    ($m$ を変えたときに効果が小さく見えるのは、問題が簡単になってしまったため)



# Low mode 前処理

- フェルミオン行列 $D$ の low mode
  - 長距離の振る舞いと関係
  → 長いホッピングをすべて取り入れるのは無理
  → multi-grid の方法を用いる




- Multi-grid の方法
  1. high mode の寄与を落とし low mode の寄与を保った、より粗い間隔の格子へ移す (restriction、RL)
  2. 粗い格子上にはゲージ場はいないとする
  3. 元の細かい間隔の格子へ戻す (prolongation、PL)

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

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

$$\sum_{x \in B(y)} W(y,x)W(y,x)^{\dagger} = \mathbb{1}_{\tilde{V}_I}$$ → 内部自由度の空間

**この重み $W$ は low mode の寄与が大きくなるようにきめるので、学習パラメータではない (学習する方法もある → 後ほど紹介)**

# RL/PL の具体的な方法

1. フェルミオン行列の零空間のベクトルを内部自由度の数 $(s)$ だけ用意

$$Du_i \approx 0 \quad i \in \{1, \ldots, s\}$$

   - 連立一次方程式を解いて求める必要がある (初期ベクトルはランダムに用意)
   - High mode の寄与を取り除くことができる

2. 粗い格子上の座標 $y$ に関するブロックを抜き出す $u_1^y, \ldots, u_s^y$

3. 抜き出したブロックについて直行化 $\bar{u}_1^y, \ldots, \bar{u}_s^y$

4. 線形写像 $W(y,x)^\dagger$ を以下で与える

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

$u_i$ が low mode の線形結合になっていて、各係数がゲージ不変ならば、RL/PLは gauge equivariant ($Du_i = 0$ を解くときの初期ベクトルについて統計平均をとればそうなっている。)

# Low mode 前処理のモデル構築

- 粗い格子上のフェルミオン行列 $\tilde{D} = RD_{\mathrm{WC}}P$ に対して、(L)PTC を使って $\tilde{M}$ を作る
  - 1ホップのみ考える

- 次の損失関数を最小化

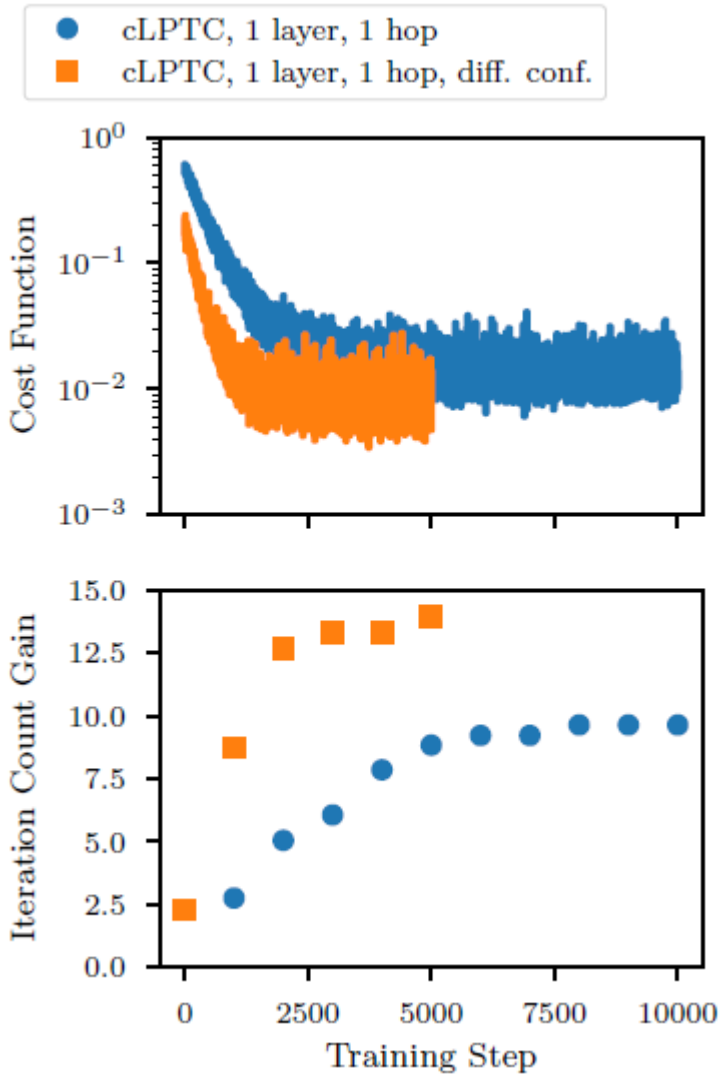
$$C = |\tilde{M}\tilde{D}v - v|^2$$

  - 学習方法は High mode の場合と同様

$C' = |\tilde{M}v - \tilde{D}^{-1}v|^2$ の方がlow mode の寄与がより強調されてよいが、学習中に連立一次方程式を解かなければならない

# Low mode 前処理の結果

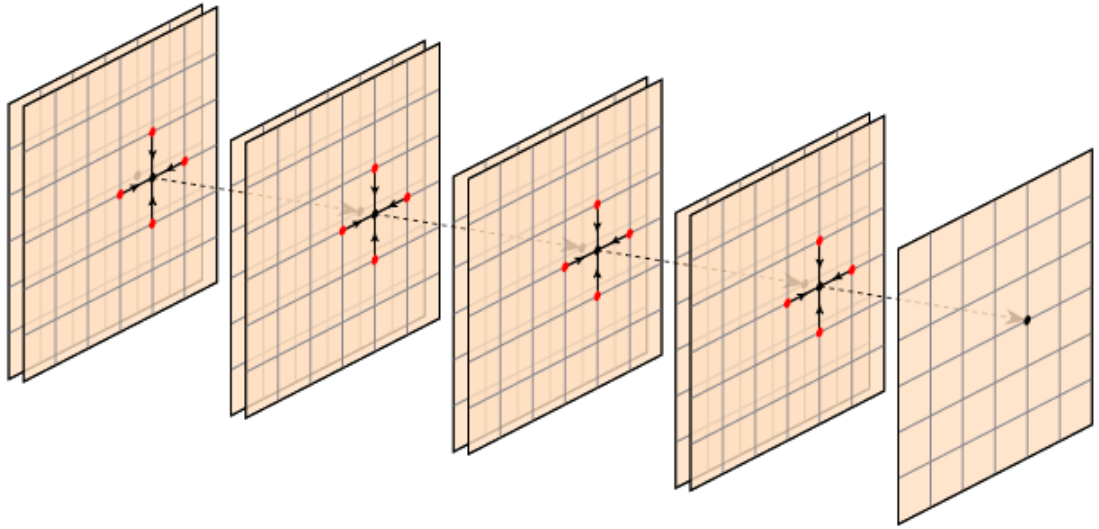
- 粗い格子 $2^3 \times 4$
- High mode の場合と比べて、非常に長い学習時間が必要
- 比較的少ない転移学習コストで、別の配位にも適用可能
  - $\widetilde{D}$ の定義を固定するため、$u_i$ を計算するときの初期ベクトルも固定



# Multi-gird モデルの構築 (1): 短距離モデル

- 短距離モデル
  - Smoother の役割
  - $Du = b$ を近似的に解くことを考える
  - 次の反復計算で解の精度を高める

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

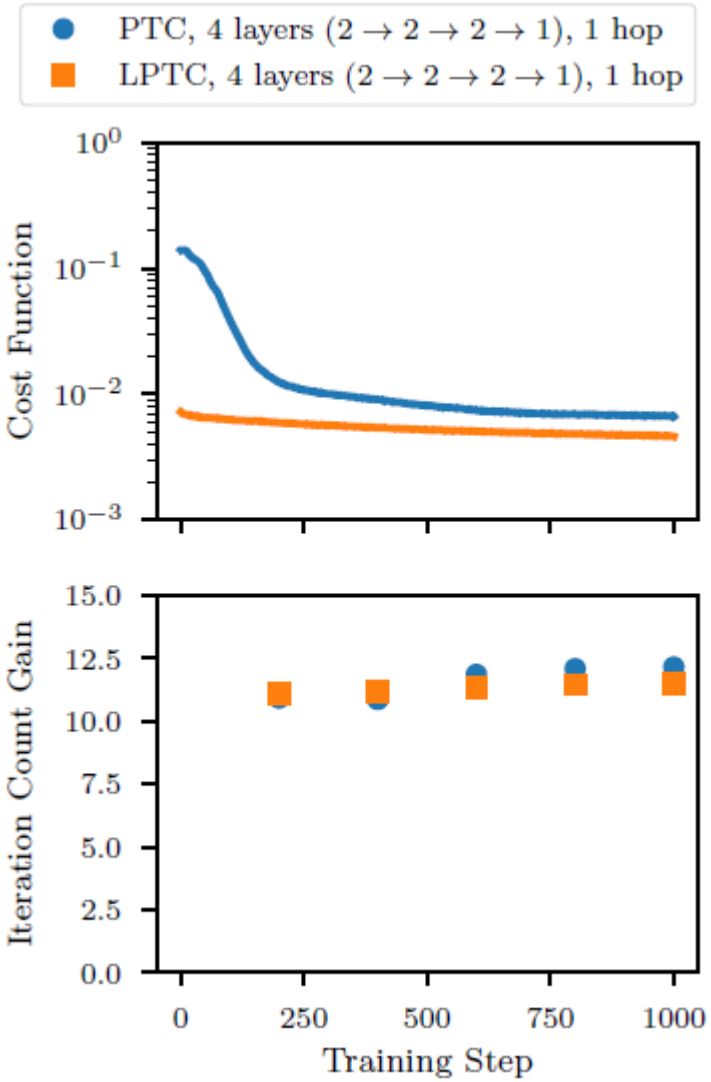


  - 長距離モデル (粗い格子) からの出力を初期値とする
  - 2つの(L)PTC層を用いて表現
- 学習: 次の損失関数を小さくする

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

# 短距離モデルの結果

- $r = 2$ を選択
  - $r = 1$ より大幅に性能がよかった
- $(u_k, b)$ はランダムなものを用いて学習
- 推論は $u = 0$ を初期ベクトルとして用いる
- High mode モデルの約2倍の効果
  - $r = 2$ としたためと思われる
- LPTC は PTC とほとんど変わらない



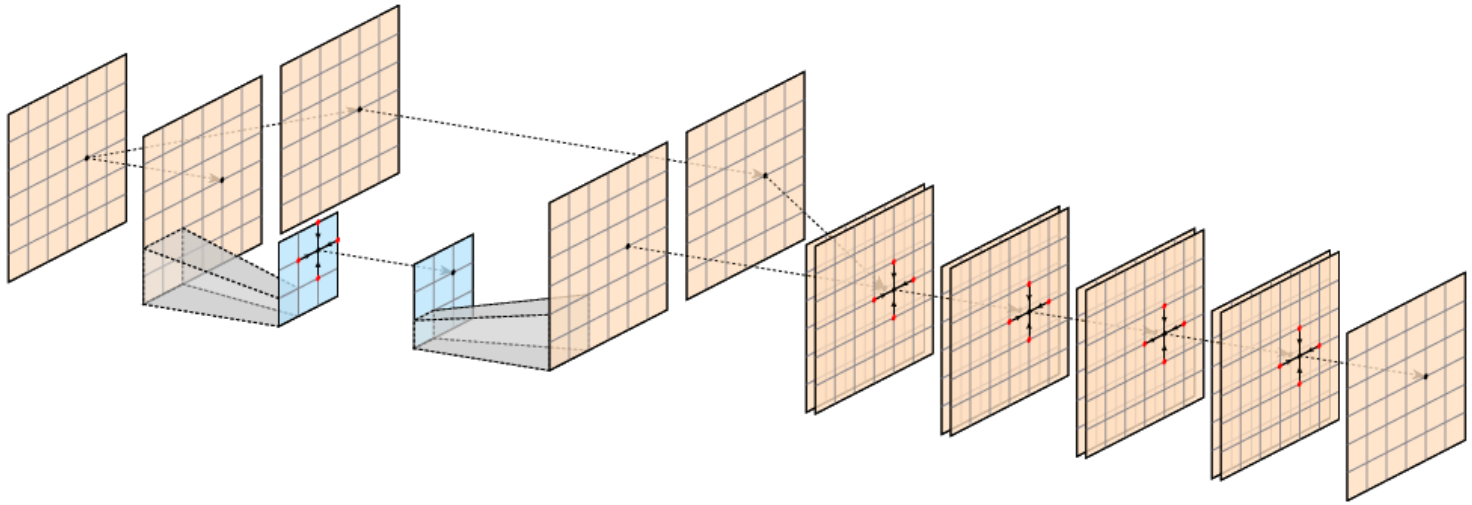
# Multi-grid モデルの構築 (2): 短距離 + 長距離モデル

- 長距離用と短距離用それぞれのために入力をコピー
- 最初は長距離・短距離モデルを別々に学習させる
- 最後に以下の損失関数を用いて全体を学習させる

$$C = |Mb_h - u_h|^2 + |Mb_\ell - u_\ell|^2$$

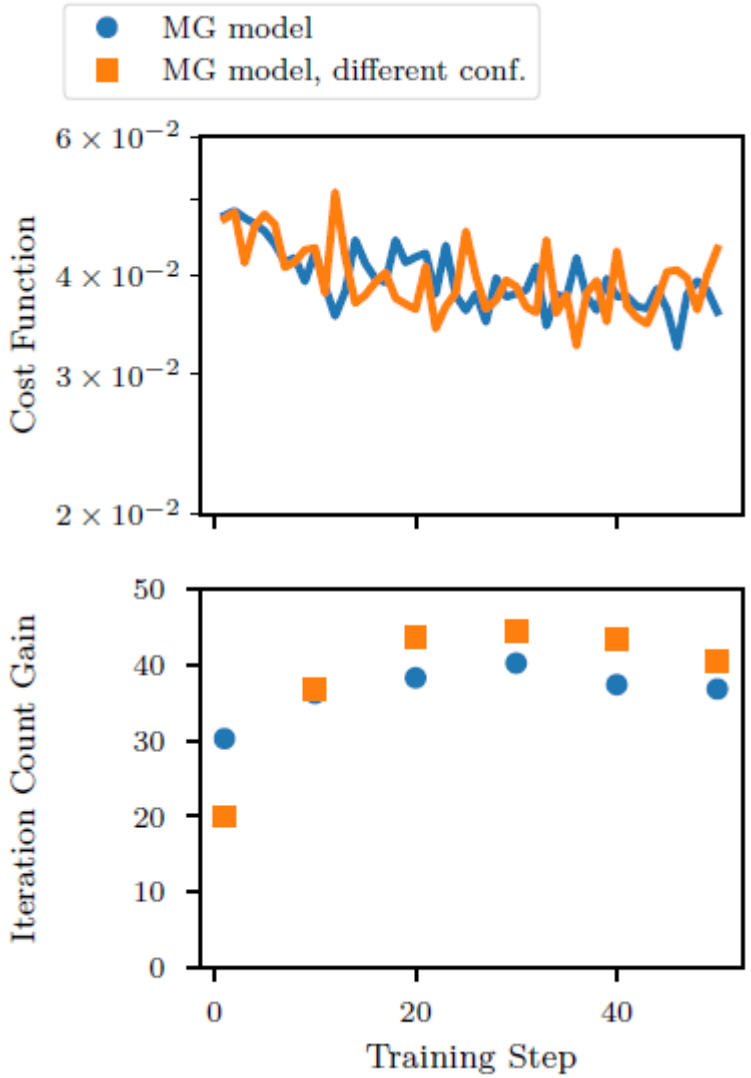
$$b_h = D_{\mathrm{WC}} v_1,\ u_h = v_1,\ b_\ell = v_2,\ u_\ell = D_{\mathrm{WC}}^{-1} v_2$$

$v_1$ と $v_2$ は $|b_1| = |b_2| = 1$ となるように規格化されたランダムなベクトル



# Multi-grid モデルの結果

- 短距離・長距離モデルは個別に学習済み
- 短距離+長距離モデル全体の学習コストを調査
- 少ない学習で40倍近い gain が出ている
- 別の配位についても少ない再学習で適用可能



# Gauge equivariant な RL/PL

arXiv:2304.10438 [hep-lat]

- RL/PL を pooling と subsampling に分ける

$$\mathrm{RL} = \mathrm{SubSample} \circ \mathrm{Pool} \qquad \mathrm{PL} = \mathrm{Pool}^{\dagger} \circ \mathrm{SubSample}^{\dagger}$$

- Pooling

$W_q$: 学習パラメータ

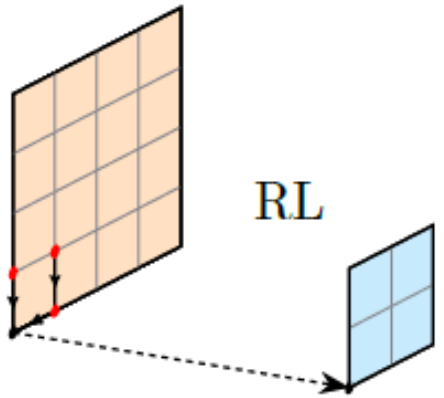
$$\mathrm{Pool}\varphi(x) = \sum_{q \in Q} W_q(x) T_q \varphi(x)$$

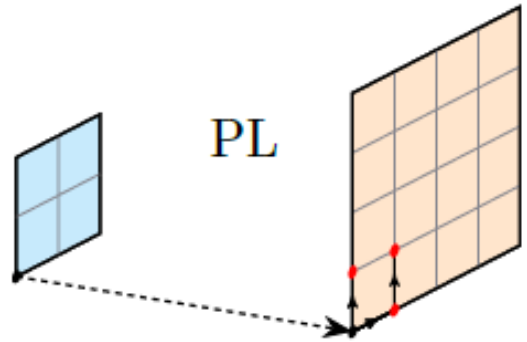
$q = (p, \bar{U})$
$p$: 細かい格子上で定義したブロック内の経路
$\bar{U}$: 細かい格子上のリンク変数
(変換性が同じなら元のリンク変数 $U$ そのものでなくてもよい)

- Subsampling

$$\mathrm{SubSample}\varphi(y) = \varphi(B_r(y))$$

$B_r(y)$: 粗い格子上の点 $y$ に対する細かい格子上の参照点



# 粗い格子上でのゲージ場の構成

- Plain coarse-link model
  - 細かい格子上での参照点同士を結ぶ最短経路で定義

$$B_r(y') - B_r(y) = b\hat{\mu}$$

$$\tilde{U}_\mu(y) = U_\mu(B_r(y)) \cdots U_\mu(B_r(y) + (b-1)\hat{\mu})$$

- Galerkin model
  - 粗い格子上でのディラック演算子を用いて定義

$$\tilde{D} = \mathrm{RL} \circ D \circ \mathrm{PL}$$

$$\tilde{U}_\mu(y) = \tilde{D}(y, y + \hat{\mu})$$

# RL/PL の学習

- $\mathrm{PL} \circ \mathrm{RL}$ は low mode に対する自己符号化器だとみなす
  → low mode を入力すると low mode を出力する
- また、$\mathrm{PL} \circ \mathrm{RL}$ は low mode の空間への射影演算子とする
  → high mode を入力すると 0 になる
- $\mathrm{RL} \circ \mathrm{PL} = \mathbb{1}$ を大体満たすようにする
- RLとPL に用いる重みは共通にする
  → $\mathrm{PL} = \mathrm{RL}^\dagger$
- 損失関数

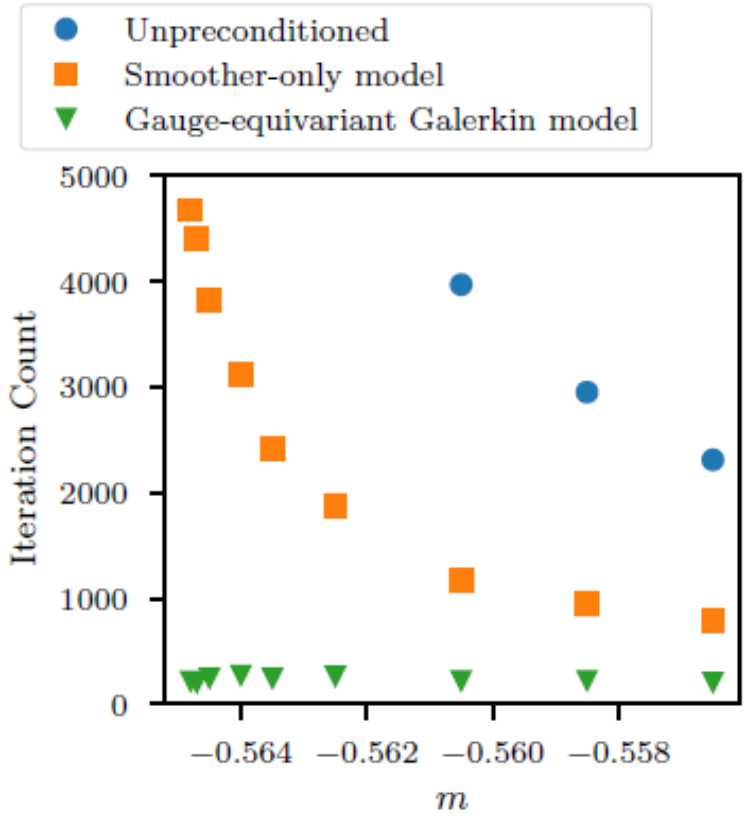
$$C = |\mathrm{PL} \circ \mathrm{RL} v_\ell - P_\ell v_\ell|^2 + |\mathrm{PL} \circ \mathrm{RL} v_h - P_\ell v_h|^2 + |\mathrm{RL} \circ \mathrm{PL} v_c - v_c|^2 .$$

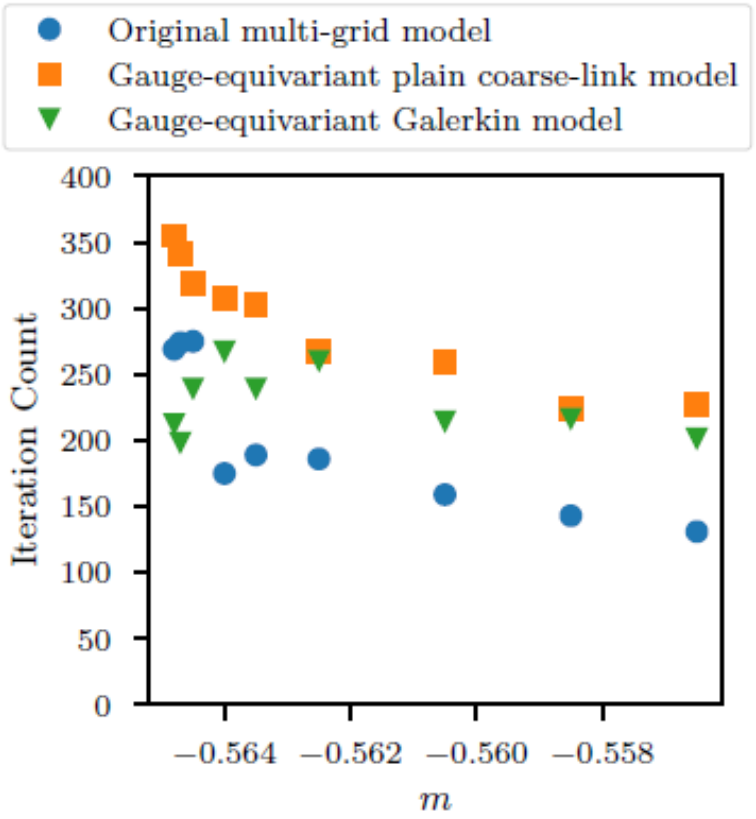
$v_l$: 近似零空間 $\{u_1, \cdots, u_s\}$ からランダムに選択
$v_h, v_c$: 正規分布からランダムに生成

$P_\ell = W^\dagger W$ : low mode への射影演算子

# 結果

- Galerkin モデルは critical slowing down を回避できている
- Plain coarse-link モデルは他の方法より悪そう



# まとめ

- Gauge-equivariant なニューラルネットを使って、multi-grid 前処理を作れた
- 同じアンサンブルや少しだけ違うシミュレーションパラメータのゲージ配位については、少ない転移学習コストで適用できそう
- 疑問点
  - 既存手法 (w/ and w/o 機械学習) との比較
  - 計算時間は速くなっているのか？
  - より大きな格子ではどうか？