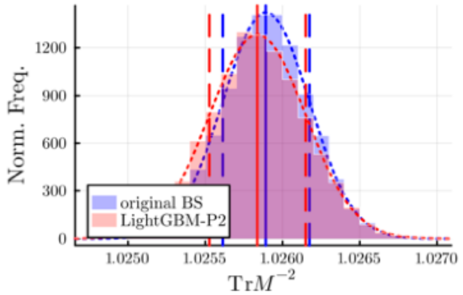
ML meets LFT
Pre-LATTICE 2024 Workshop
Swansea University

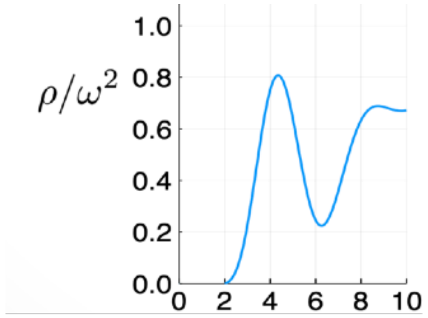
MLPhys in Japan and Developments of CASK: Gauge Symmetric Transformer

Akio Tomiya (Lecturer/Jr Associate prof)
Tokyo Woman's Christian University
(I moved in this April)

Outline of my talk

MLPhys





Main part

CASK

Covariant transformer

What is MLPhys

My team: LQCD + ML

“Machine Learning Physics Initiative”

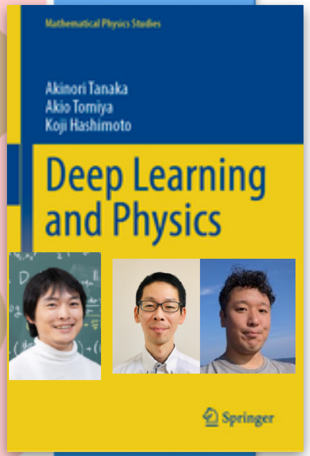
MLPhYs

2022-2027, 10M USD, 70 researchers

Director : K. Hashimoto

- B01 A.Tanaka: Math and Application of DL
- B02 Y.Kabashima: Statistical data ML
- B03 K.Fukushima: Topology and Geometry of ML +...
- A01 A.Tomiya: Computational physics +...
- A02 M.Nojiri: High Energy Physics
- A03 T.Ohtsuki: Condensed Matter Physics
- A04 K.Hashimoto: Quantum and Gravity Physics +...

ML Phys

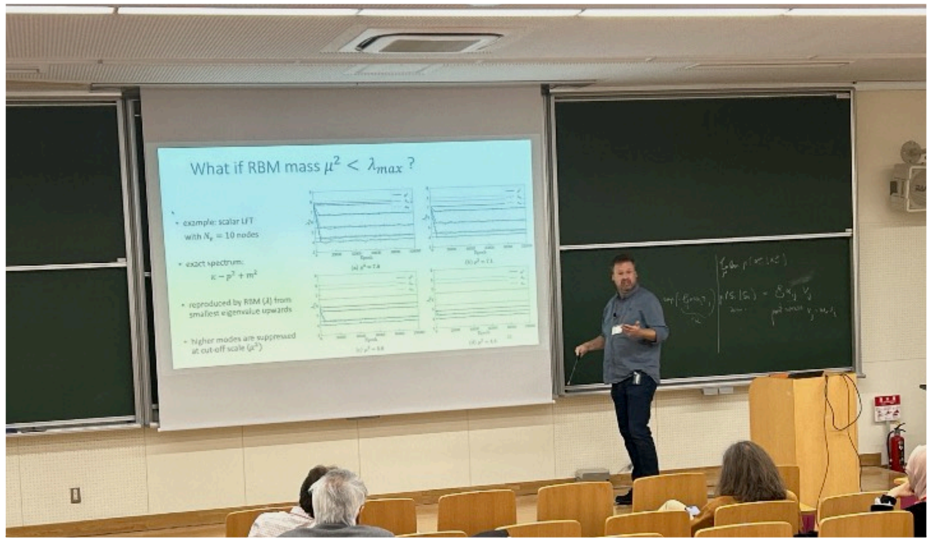


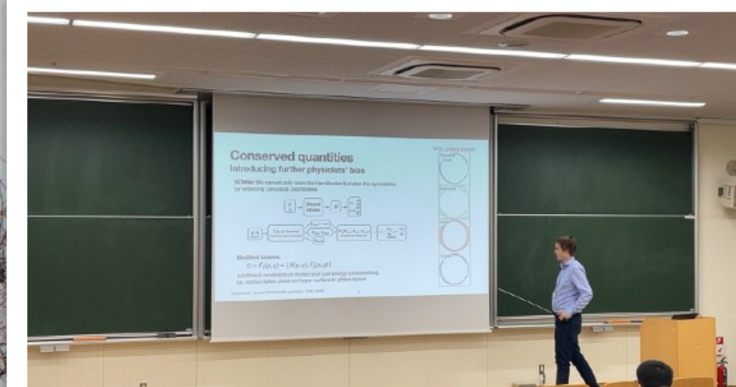
2021

FY2022-2026 MEXT -KAKENHI- Grant-in-Aid for Transformative Research Areas (A)

科研費 KAKENHI

International conference in 2023 at Kyoto, Japan





Yukawa Institute, Kyoto Japan

 https://mlphys.scphys.kyoto-u.ac.jp/ic_mlphys/

String data 2024

https://mlphys.scphys.kyoto-u.ac.jp/stringdata2024/

Confirmed invited speakers

- Yago Bea (University of Barcelona)
- Gabriel Lopes Cardoso (Lisbon, IST)
- François Charton (META AI)
- Sergei Gukov (Caltech)
- James Halverson (Northeastern University)
- Song He (Jilin University / Max Planck Institute Potsdam)
- Edward Hirst (Queen Mary, University of London)
- Vishunu Jejjala (University of the Witwatersrand in Johannesburg)
- Hyun-Sik Jeong (Institute for Theoretical Physics UAM-CSIC in Madrid)
- Keun-Young Kim (GIST)
- Sven Krippendorf (Arnold Sommerfeld Center for Theoretical Physics, LMU Munich)
- Anindita Maiti (Perimeter Institute)
- Fabian Ruehle (Northeastern University)
- Matthew Schwartz (Harvard University)
- Rak-Kyeong Seong (UNIST)
- Eva Silverstein (Stanford)

Organizers

Hashimoto, Koji (Kyoto University, chair)
Yoshida, Kentaroh (Saitama University)
Murata, Masaki (Saitama Institute of Technology)
Sugishita, Sotaro (Kyoto University)
Hirono, Yuji (Osaka University)
Sannai, Akiyoshi (Kyoto University)
Yoda, Takuya (Kyoto University)
Hikida, Yasuyuki (Kyoto University)
Tanahashi, Norihiro (Kyoto University)

My team (A01): LQCD + ML

MLPhYs

PI: Akio Tomiya (Me)
TWCU
LQCD, ML

Kouji Kashiwa
Fukuoka Institute of Technology
LQCD, ML

Hiroshi Ohno
U. of Tsukuba
LQCD

Tetsuya Sakurai
U. of Tsukuba
Computation

Yasunori Futamura
U. of Tsukuba
Computation

post-docs
& external members

B. J. Choi
U. of Tsukuba

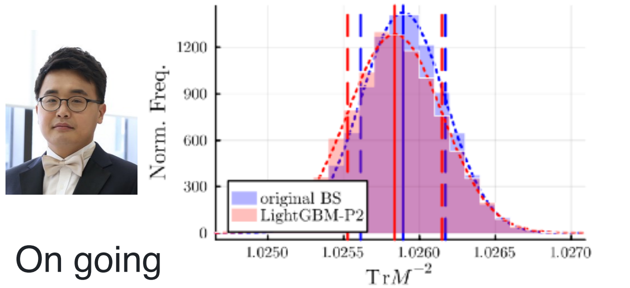
J. Takahashi
Meteorological College

Y. Nagai
U. of Tokyo

- **Apply machine learning techniques on LQCD (To increase what we can do)**
- **Find physics-oriented ML architecture**
- **Making codes for LQCD + ML**

ML Phys A01

measurement with BDT



On going

LatticeQCD.jl

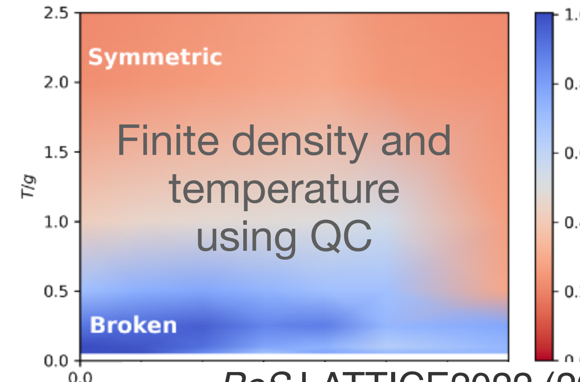
Open source

LQCD (+ML) with julia

This covers most of modern tech

https://github.com/akio-tomiya/LatticeQCD.jl

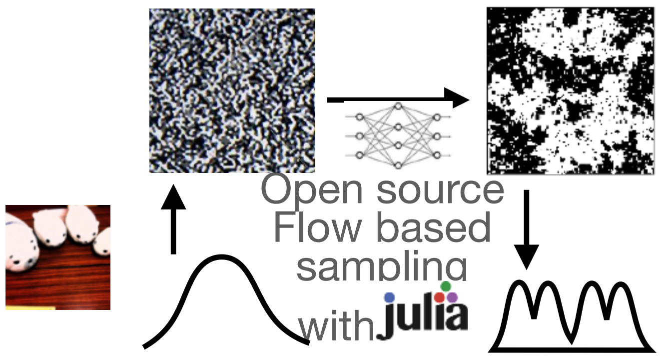
(and associated sub-libraries)



Finite density and temperature using QC

PoS LATTICE2022 (2023) 039

ML + QC: Quantum thermodynamics using Density matrix and MADE



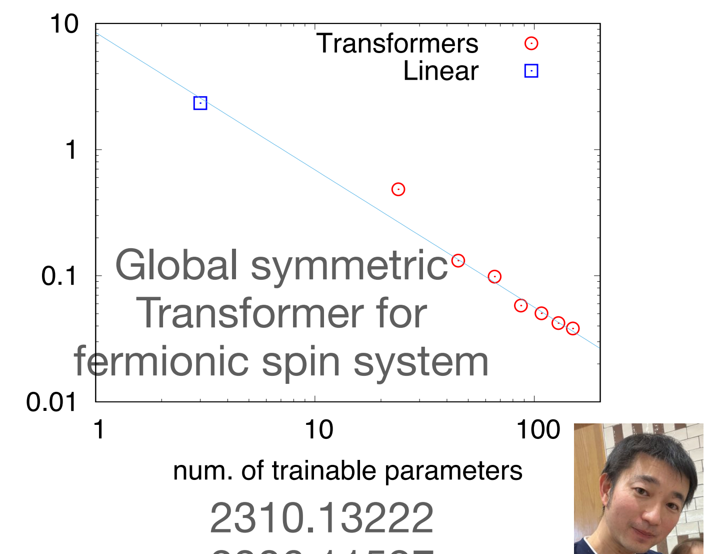
Open source Flow based sampling with julia

arXiv: 2208.08903

$$\frac{dU_\mu^{(t)}(n)}{dt} = \mathcal{G}^{\bar{\theta}}(U_\mu^{(t)}(n))$$

Gauge covariant neural net

arXiv: 2103.11965

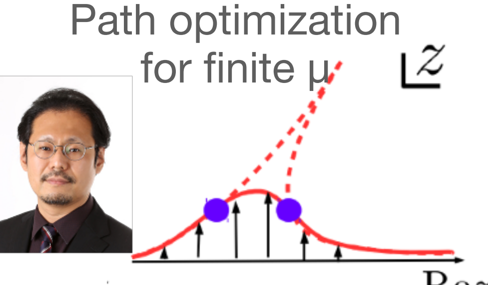


Global symmetric Transformer for fermionic spin system

2310.13222

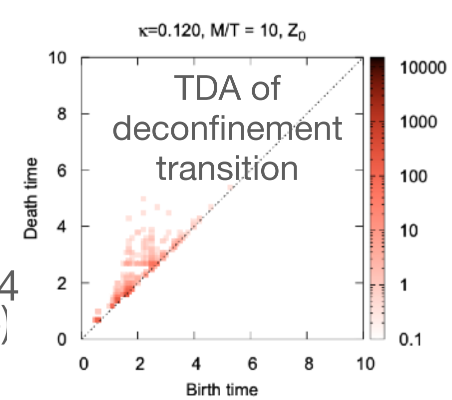
2306.11527

Path optimization for finite μ



Phys. Rev. D 108, 094504

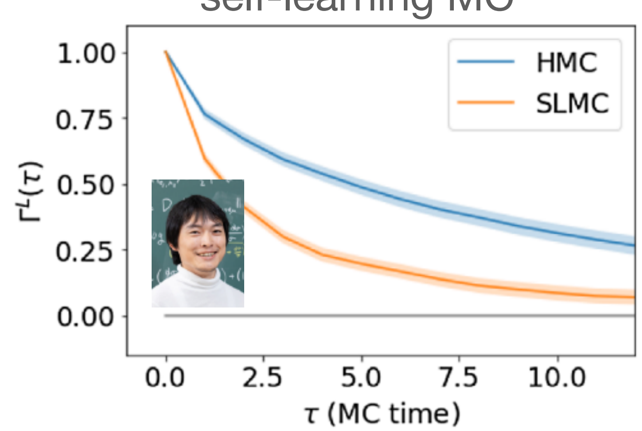
(Figure from 1812.11506)



TDA of deconfinement transition

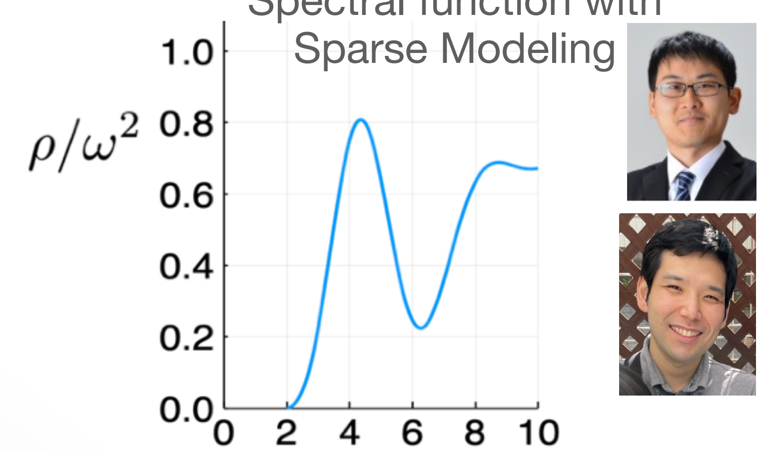
1810.07635

Gauge invariant self-learning MC



Phys. Rev. D 107, 054501

Spectral function with Sparse Modeling



(arXiv: 2311.15233)

+ on going

Lattice QCD code for generic purpose

Open source LQCD code in Julia Language

Akio Tomiya


AT & Y. Nagai (A01, A03)

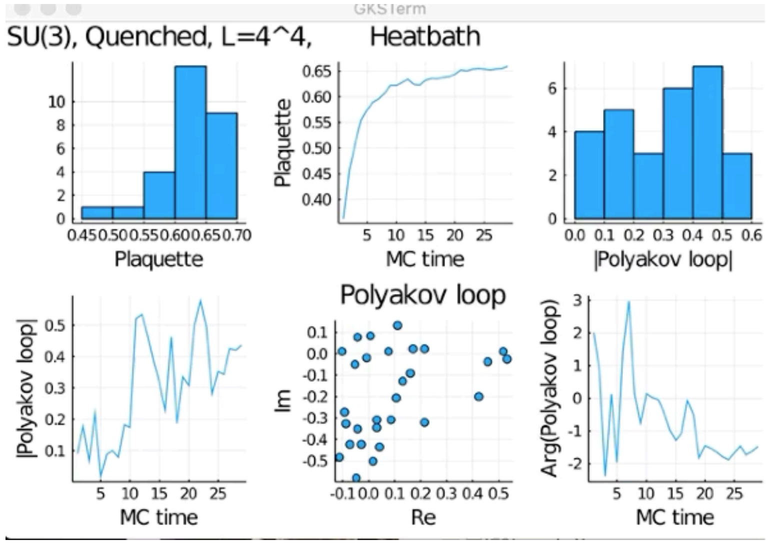
LatticeQCD.jl Open source (Julia Official package, Now updated to v1.0)
Fast as a fortran code

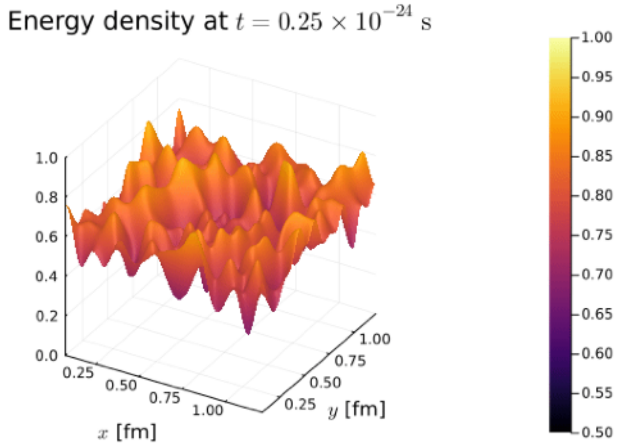
Machines: Laptop/desktop/Jupyter/Supercomputers

Functions: SU(Nc)-heatbath, (R)HMC, Self-learning HMC, SU(Nc) Stout
Dynamical Staggered, Dynamical Wilson, Dynamical Domain-wall
Measurements, Gauge covariant net, Auto-grad for gauge fields,

Start LQCD in **5 min** **(super-easy)**	1. Download Julia binary 2. Add this package through Julia package manager 3. Execute! (no explicit compile is needed)

https://github.com/akio-tomiya/LatticeQCD.jl





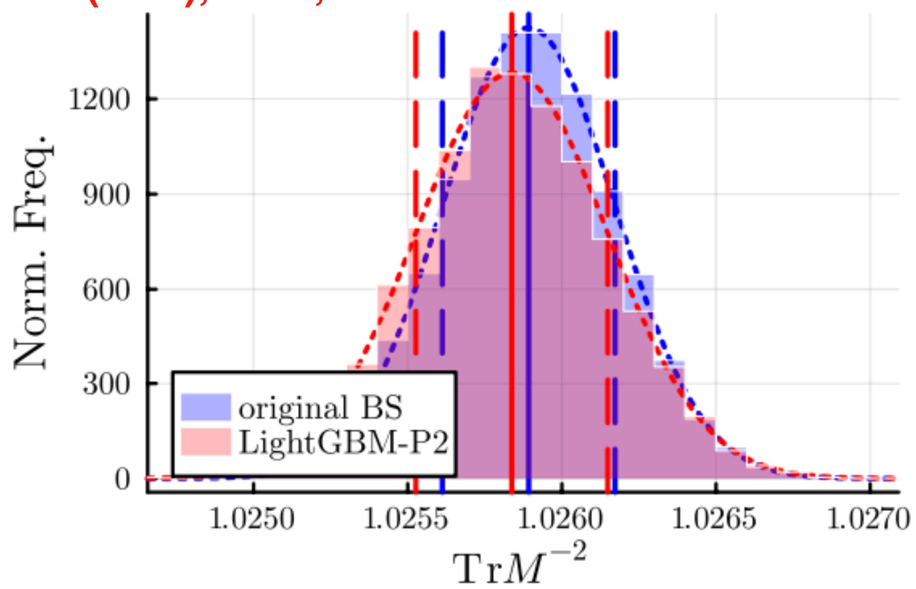
Two new textbooks + review are publishing

- An introduction to “machine learning physics” (PhysML members)
  - Published in this autumn in Japanese
    this will be translated to English (probably) using LLM
  - Contents: Statistical estimation, Basics of neural nets
    Transformers, language model, Mean field theory for neural nets,
    Neural net wave functions, …
- Introduction to lattice QCD (Kashiwa, Ohno, Tomiya)
  - Published in this Winter in Japanese (I’m not sure about English ver)
- I am writing a review paper LQCD/QFT + ML, please let me know if you have things worth to write. This will be published in JPSJ.

Two related talks in Lattice2024

Other than my talk

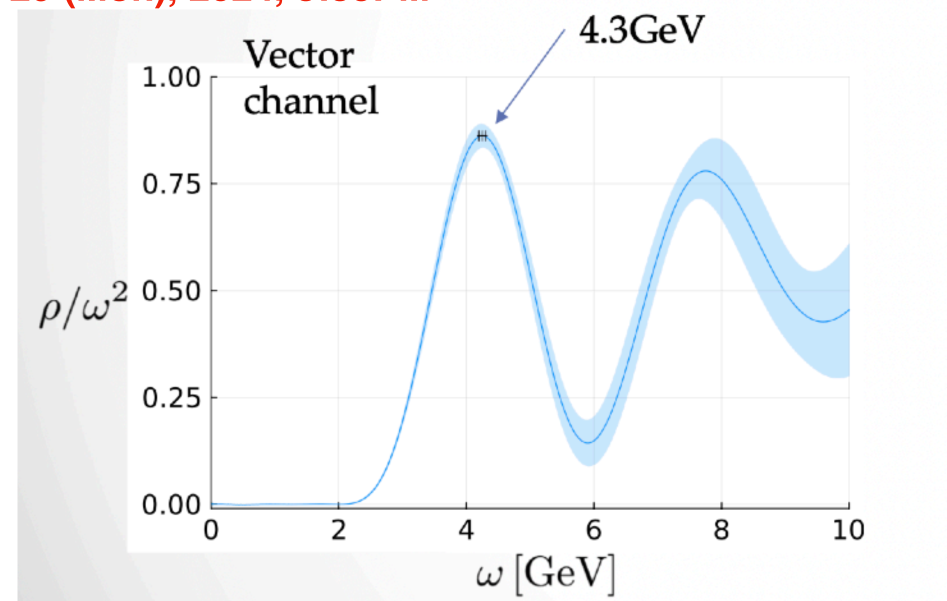
Algorithms and artificial intelligence
Jul 29 (Mon), 2024, 11:15AM



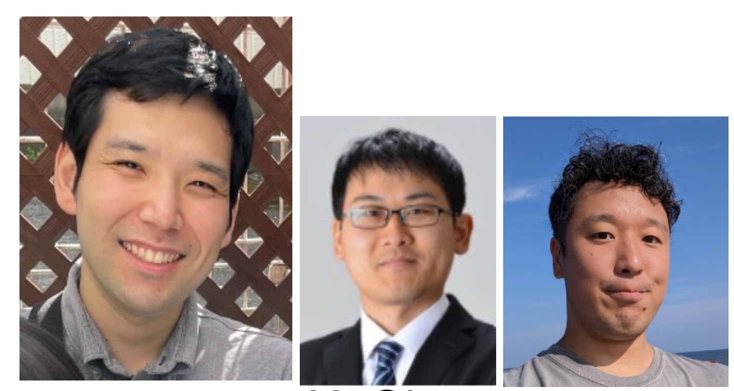
Using idea based on B. Yoon+ 1807.05971,
we estimate higher order of 1/D using ML.
Impact of bias correction will be discussed

B. J. Choi H. Ohno
U. of Tsukuba

Algorithms and artificial intelligence
Jul 29 (Mon), 2024, 3:35PM



Reconstruction of spectral function
using machine learning (sparse modeling)



J. Takahashi H. Ohno
Meteorological College

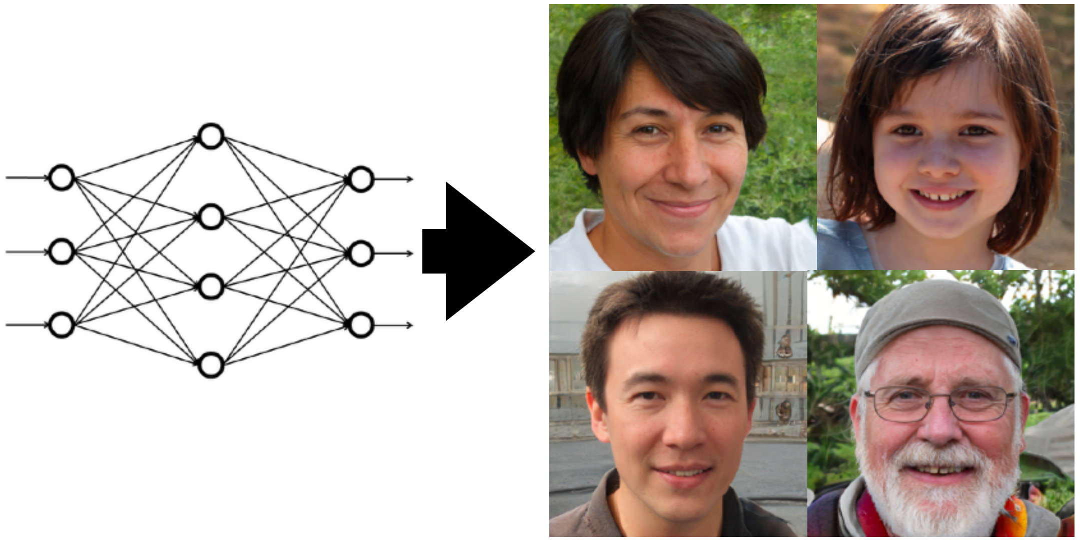
Two previous works to realize Gauge symmetric Transformer for LQCD

1. Gauge covariant net arXiv: 2103.11965 AT+
2. Transformer for fermion-spin systems

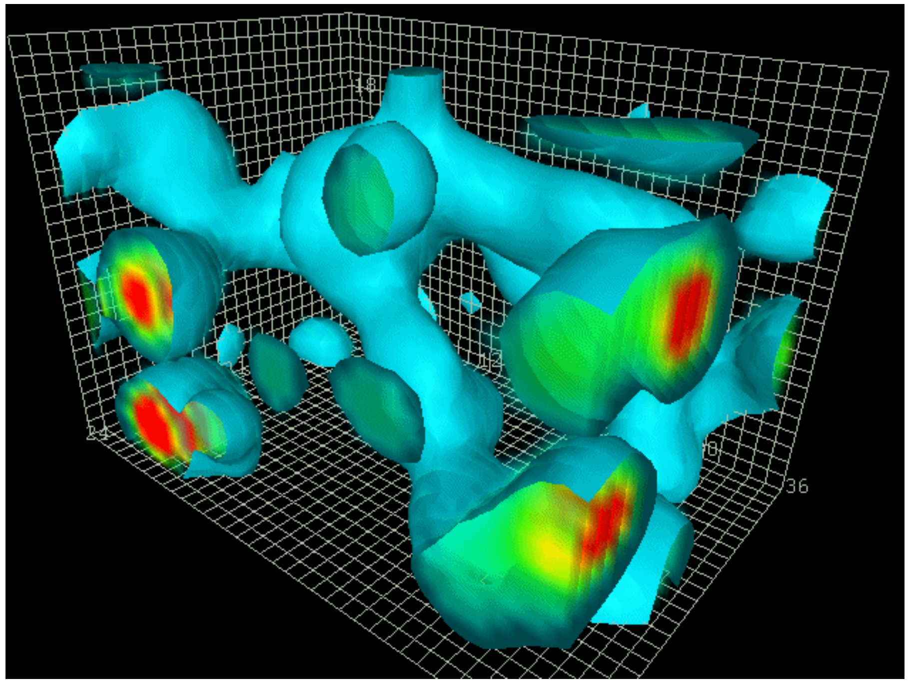
2310.13222 AT+
2306.11527 AT+

ML for LQCD is needed

- Neural networks
  - Data processing techniques mainly for 2d image (a picture = pixels = a set of real #)
  - Neural network helps data processing e.g. AlphaFold3
- Lattice QCD requires numerical effort but is more complicated than pictures
  - 4 dimension
  - **Non-abelian gauge d.o.f. and symmetry**
  - Fermions (Fermi-Dirac statistics)
  - Exactness of algorithm is necessary
- Q. How can we deal with neural nets?



thispersondoesnotexist.com



http://www.physics.adelaide.edu.au/theory/staff/leinweber/VisualQCD/QCDvacuum/

What is the neural networks?

Akio Tomiya

Attempts to gauge symmetry

7,8 years! 😯

In my paper for fields generation using ML (1712.03893),

> If we want to use generative models as lattice QCD sampler, we must guarantee the gauge symmetry of a probability distribution for the model. This is because, configurations which are generated by a algorithm must

We have created several architectures:

2010.11900, AT+: Gauge *invariant* self-learning MC for 4d LQCD

2103.11965, AT+ Gauge *covariant* self-learning HMC for 4d LQCD
(Covariant NN = adaptive gradient flow = adaptive stout)

(2310.13222, AT+: Global symmetric transformer for fermion-spin system)

This work, AT+: *Gauge* symmetric transformer for 4d LQCD

There are several realization of gauge covariant maps arXiv:2012.12901 arXiv: 2305.02402

Overview/outline

Gauge covariant transformer for LQCD

Two conditions/restrictions in LQCD:

Gauge symmetry
U(x, x+μ)

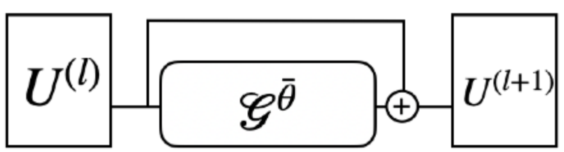
Non-locality from
pseudo-fermions
(1/D) ~ non-local

(I want to mimic
this by NN)

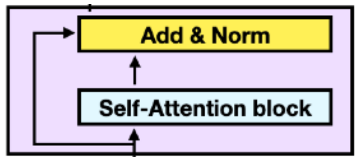
Solutions in neural net:

arXiv: 2103.11965 AT+

1. Gauge covariant net
(adaptive stout)



2. Transformer with global symmetry
(Heisenberg spin + electron)



2310.13222 AT+
2306.11527 AT+

3. Gauge symmetric Transformer for LQCD

This talk

Related topics in this meeting

- Wednesday
  - Alessandro Nada, Sampling SU(3) pure gauge theory with out-of-equilibrium evolutions and stochastic normalizing flows
- Thursday
  - Ryan Abbott, Progress in normalizing flows for 4d gauge theories
  - Fernando Romero Lopez, Applications of flow models to the generation of correlated lattice QCD ensembles
  - Mathis Gerdes, Exploring continuous normalizing flows for gauge theories

Overview/outline

Gauge covariant transformer for LQCD

Two conditions/restrictions in LQCD:

Gauge symmetry
U(x, x+μ)

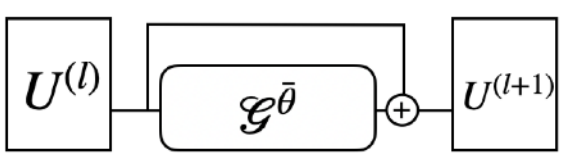
Non-locality from
pseudo-fermions
(1/D) ~ non-local

(I want to mimic
this by NN)

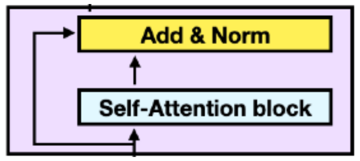
Solutions in neural net:

arXiv: 2103.11965 AT+

1. **Gauge covariant net**
(adaptive stout)



2. Transformer with global symmetry
(Heisenberg spin + electron)



2310.13222 AT+
2306.11527 AT+

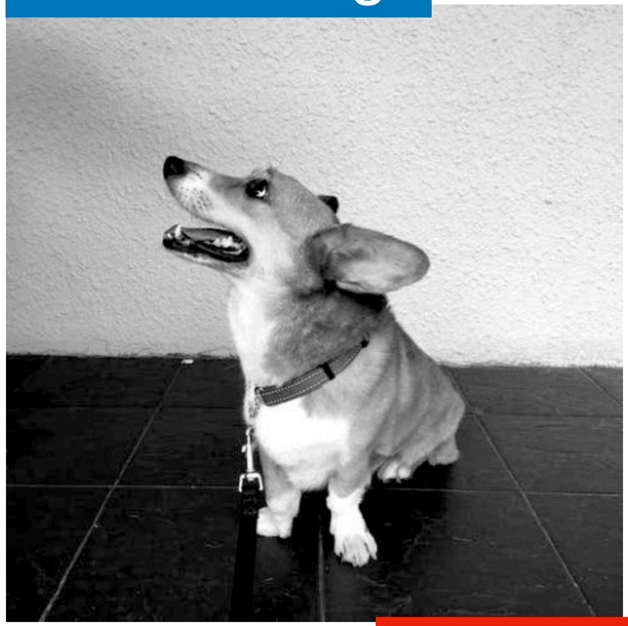
3. Gauge symmetric Transformer for LQCD

This talk

What is conv. neural networks?

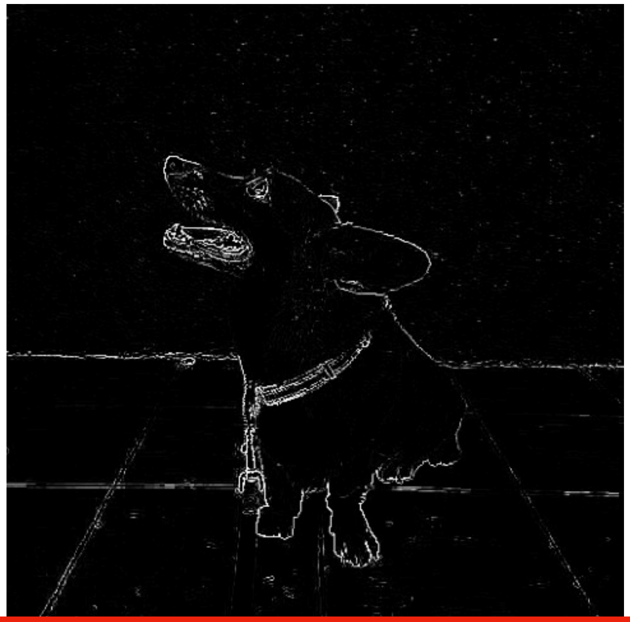
The convolution layer can treat a translation transformation

Filter on image



$$* \quad \begin{array}{c}\textbf{Laplacian filter}\\ \begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 1 & -2 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array}\end{array} \quad =$$

(Discretization of ∂^2)



Edge detection

IMPORTANT: If inputs are shifted to right, outputs are shifted to right

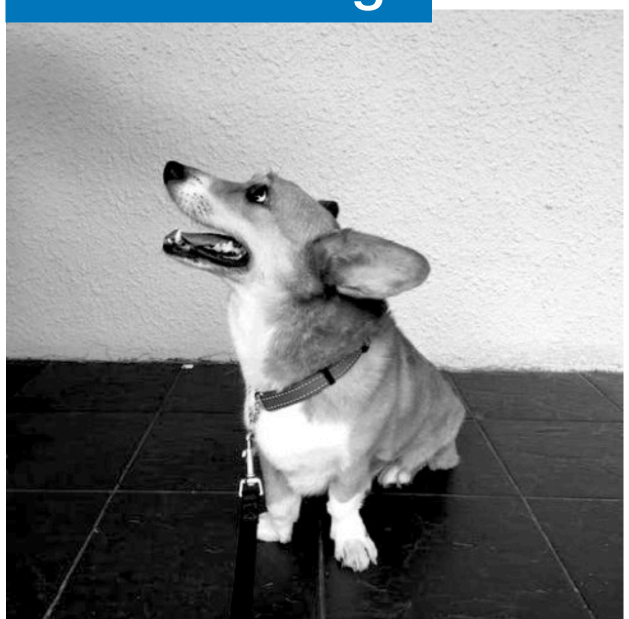
= translationally equivaliant (similar to covariance, operation just commute)

What is conv. neural networks?

Akio Tomiya

Convolution layer = trainable filter

Filter on image



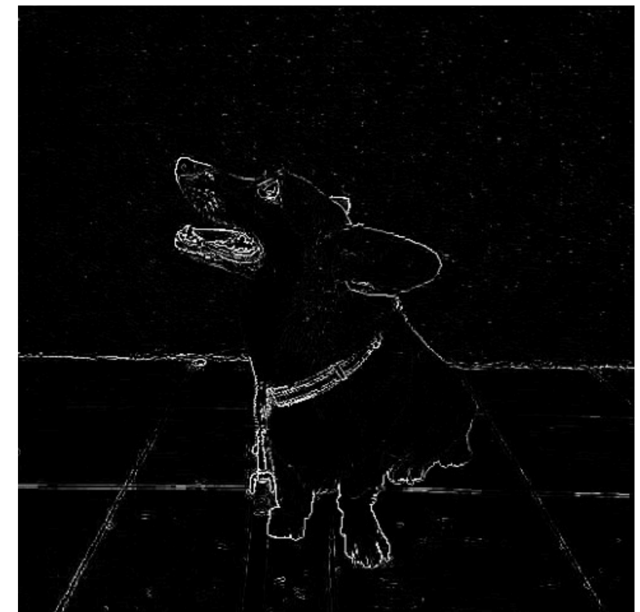
$*$

Laplacian filter

0	1	0
1	-2	1
0	1	0

(Discretization of ∂^2)

$=$



Edge detection

IMPORTANT: If inputs are shifted to right, outputs are shifted to right

= translationally equivaliant (similar to covariance, operation just commute)

Convolution layer

Fukushima, Kunihiko (1980)
Zhang, Wei (1988) + a lot!

$*$

Trainable filter

W_{11}	W_{12}	W_{13}
W_{21}	W_{22}	W_{23}
W_{31}	W_{32}	W_{33}

$\rightarrow$

Edge detection

Smoothing
(Gaussian filter)

…

Gaussian filter

$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

This can be any filter which helps feature extraction but still transitionally equivariant!

Smearing

Smoothing improves global properties

Eg.

Coarse image

Numerical derivative is unstable

Gaussian filter

$$\frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 1 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \longrightarrow$$

Smoothened image

Numerical derivative is stable

We want to smoothen gauge configurations with keeping gauge symmetry

Two types:

APE-type smearing

Stout-type smearing

M. Albanese+ 1987
R. Hoffmann+ 2007
C. Morningster+ 2003

Smearing

Akio Tomiya

Smoothing with gauge symmetry, APE type

M. Albanese+ 1987
R. Hoffmann+ 2007

APE-type smearing

Covariant sum

$$U_\mu(n) \to U_\mu^{\text{fat}}(n) = \mathcal{N}\left[(1-\alpha)U_\mu(n) + \frac{\alpha}{6}V_\mu^\dagger[U](n)\right]$$

Normalization

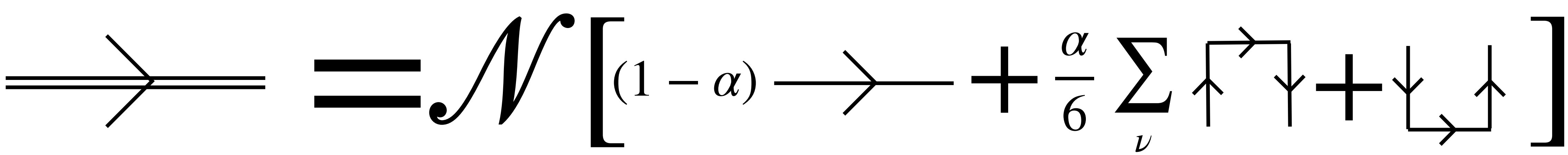
$$\mathcal{N}[M] = \frac{M}{\sqrt{M^\dagger M}}$$

Or projection

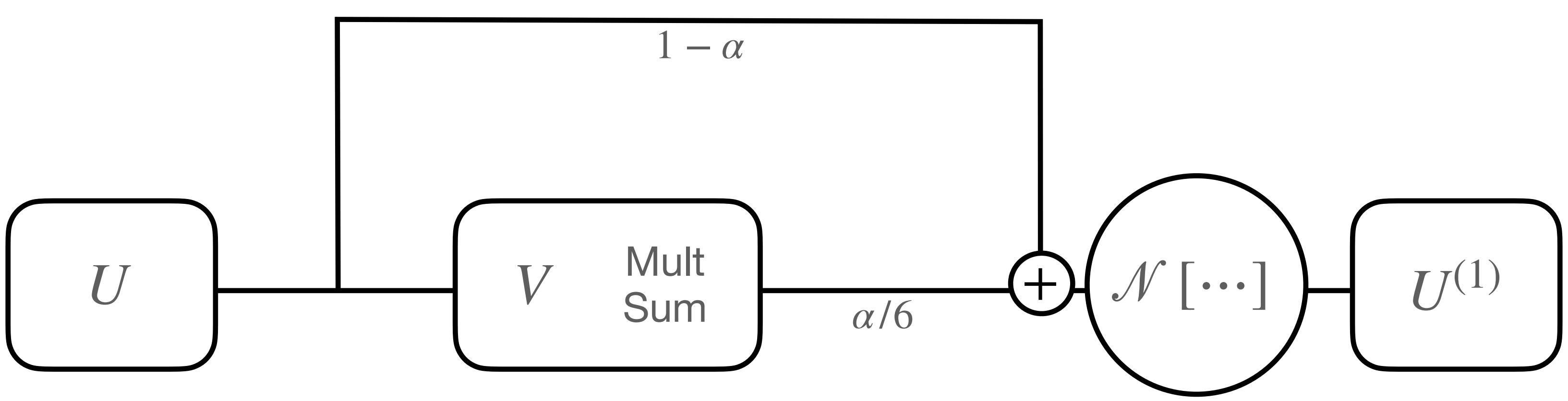
$$V_\mu^\dagger[U](n) = \sum_{\mu\neq\nu} U_\nu(n)U_\mu(n+\hat{\nu})U_\nu^\dagger(n+\hat{\mu}) + \cdots$$

$V_\mu^\dagger[U](n)$& $U_\mu(n)$ shows same transformation
→$U_\mu^{\text{fat}}[U](n)$ is as well

Schematically,

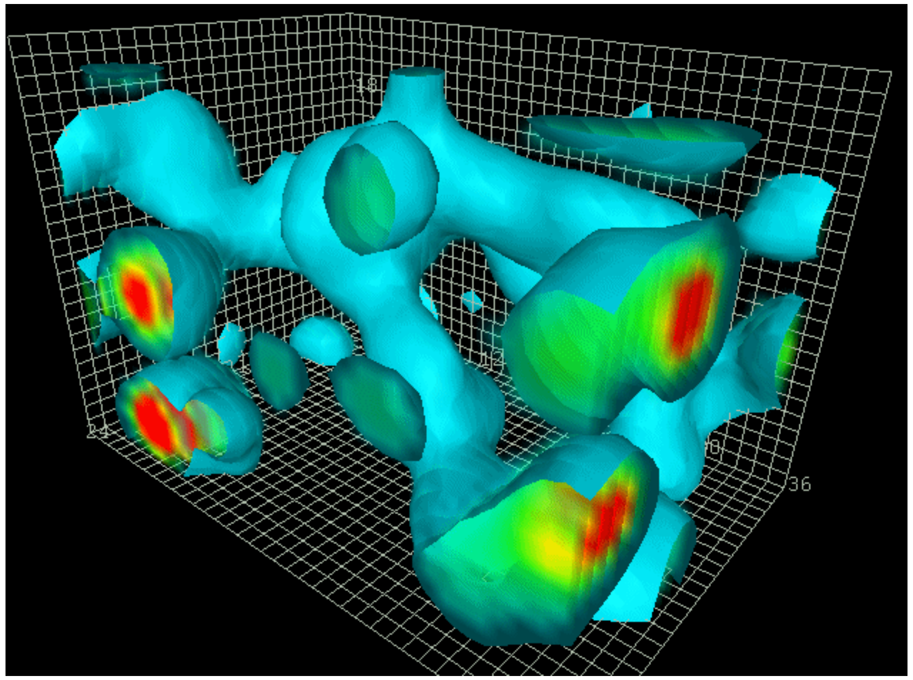


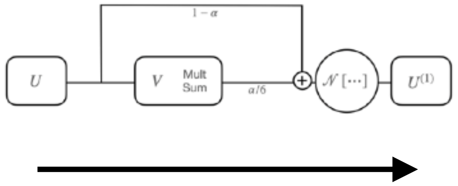
In the calculation graph,

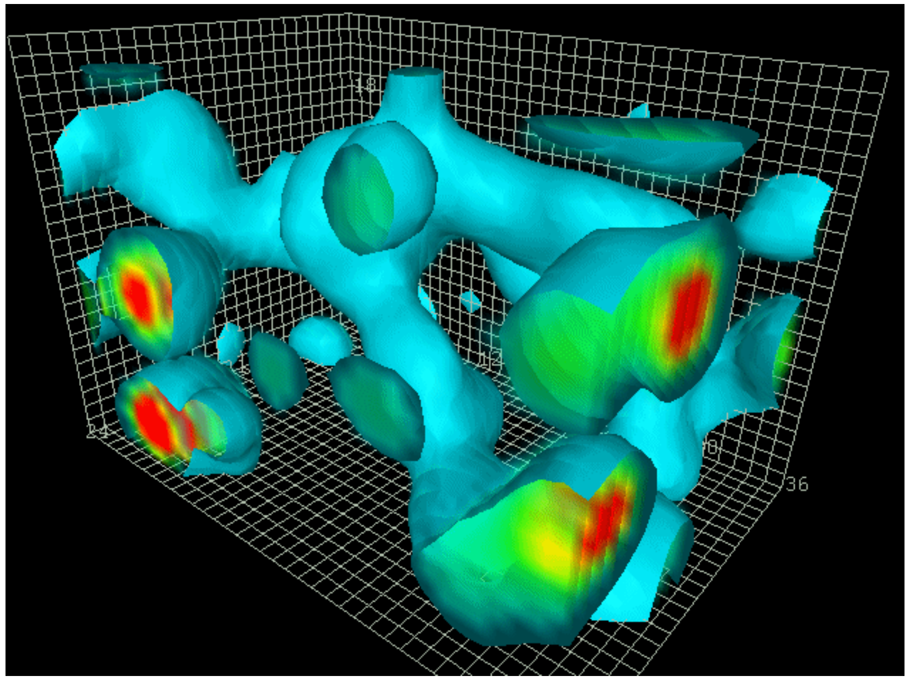


Smearing is a map with gauge covariance

Smearing makes a map between configurations, works as a *filter*







Gauge covariant neural network

= trainable smearing (= residual flow)

Akio Tomiya

AT Y. Nagai arXiv: 2103.11965

Smearing = gauge covariant way of transform gauge configurations

Covariant sum

staple

$$U_\mu(n) \to U_\mu^{\text{smr}}(n) = \mathcal{N}\left[(1-\alpha)U_\mu(n) + \frac{\alpha}{6}V_\mu^\dagger[U](n)\right]$$

$$V_\mu^\dagger[U](n) = \sum_{\mu\neq\nu} U_\nu(n)U_\mu(n+\hat{\nu})U_\nu^\dagger(n+\hat{\mu}) + \cdots$$

Normalization

$$\mathcal{N}[M] = \frac{M}{\sqrt{M^\dagger M}}$$

Or projection

Gauge covariant neural network = smearing with tunable parameters w

$$\begin{cases} z_\mu^{(l)}(n) = w_1 U_\mu(n) + w_2 V_\mu^\dagger[U](n) \\ U_\mu^{(l+1)}(n) = \mathcal{N}(z_\mu^{(l)}(n)) \end{cases}$$

Trainable param

link-wise projection/normalization (local)

Gauge covariant NN: $U_\mu^{\text{NN}}(n)[U] = U_\mu^{(4)}(n)\left[U_\mu^{(3)}(n)\left[U_\mu^{(2)}(n)\left[U_\mu(n)\right]\right]\right]$

Gauge covariant variational map: $U_\mu(n) \mapsto U_\mu^{\text{NN}}(n) = U_\mu^{\text{NN}}(n)[U]$

Stout type can be constructed in the same way

There are several realization of gauge covariant maps arXiv:2012.12901 arXiv: 2305.02402

Gauge covariant neural network
= trainable smearing (= residual flow)

AT Y. Nagai arXiv: 2103.11965

Stout-type

$$U_\mu(n) \to U_\mu^{\rm smr}(n) = e^{\sum_i \rho_i L_i[U]} U_\mu(n)$$

staple

$$V_\mu^\dagger[U](n) = \sum_{\mu\neq\nu} U_\nu(n) U_\mu(n+\hat{\nu}) U_\nu^\dagger(n+\hat{\mu}) + \cdots$$

Trainable param

Training done by the back-prop
(extension to the stout paper [1])

Following results using this stout type

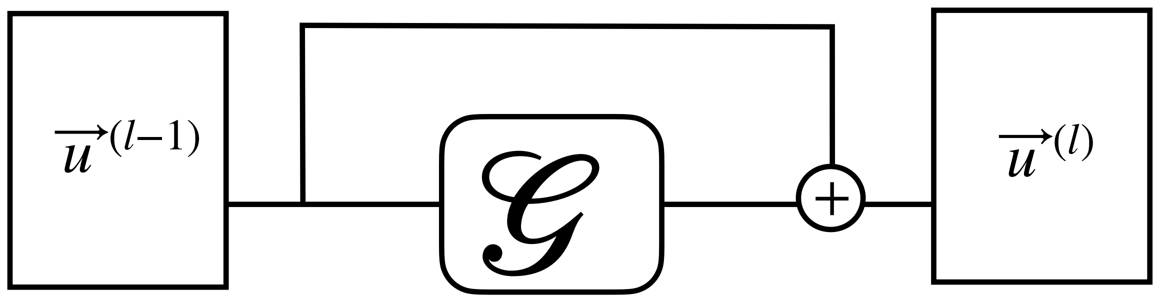
[1] C. Morningster+ 2003

There are several realization of gauge covariant maps arXiv:2012.12901 arXiv: 2305.02402

Gauge covariant neural network

Neural ODE of Cov-Net = "gradient flow"

ResNet



arXiv: 1512.03385

Continuum Layer Limit

Neural ODE

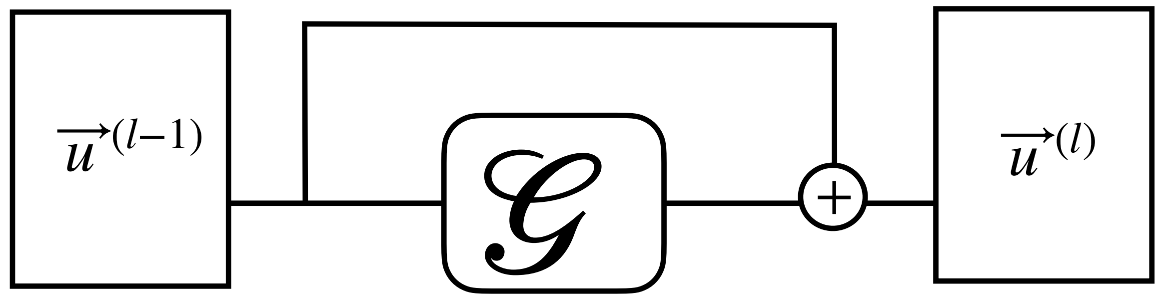
$$\frac{d\vec{u}^{(t)}}{dt} = \mathcal{G}(\vec{u}^{(t)})$$

arXiv: 1806.07366
(Neural IPS 2018 best paper)

Gauge covariant neural network

Neural ODE of Cov-Net = “gradient flow”

ResNet



arXiv: 1512.03385

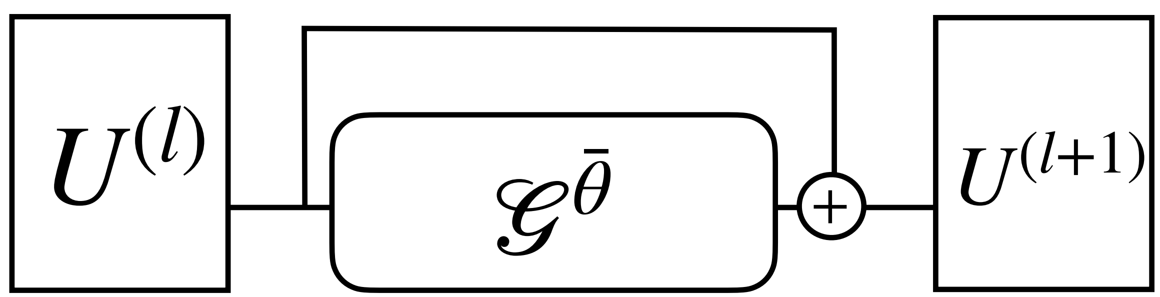
Continuum
Layer
Limit

Neural ODE

$$\frac{d\vec{u}^{(t)}}{dt} = \mathcal{G}(\vec{u}^{(t)})$$

arXiv: 1806.07366
(Neural IPS 2018 best paper)

Gauge-cov net



AT Y. Nagai arXiv: 2103.11965

Continuum
Layer
Limit

Neural ODE
for Gauge-cov NN

$$\frac{dU_\mu^{(t)}(n)}{dt} = \mathcal{G}^{\bar{\theta}}(U_\mu^{(t)}(n))$$

“Gradient” flow
(not has to be gradient of S)

“Continuous stout smearing is the Wilson flow”

2010 M. Luscher
AT Y. Nagai arXiv: 2103.11965
cf. 2212.11387 AT+

Gauge covariant neural network

= trainable smearing

Akio Tomiya

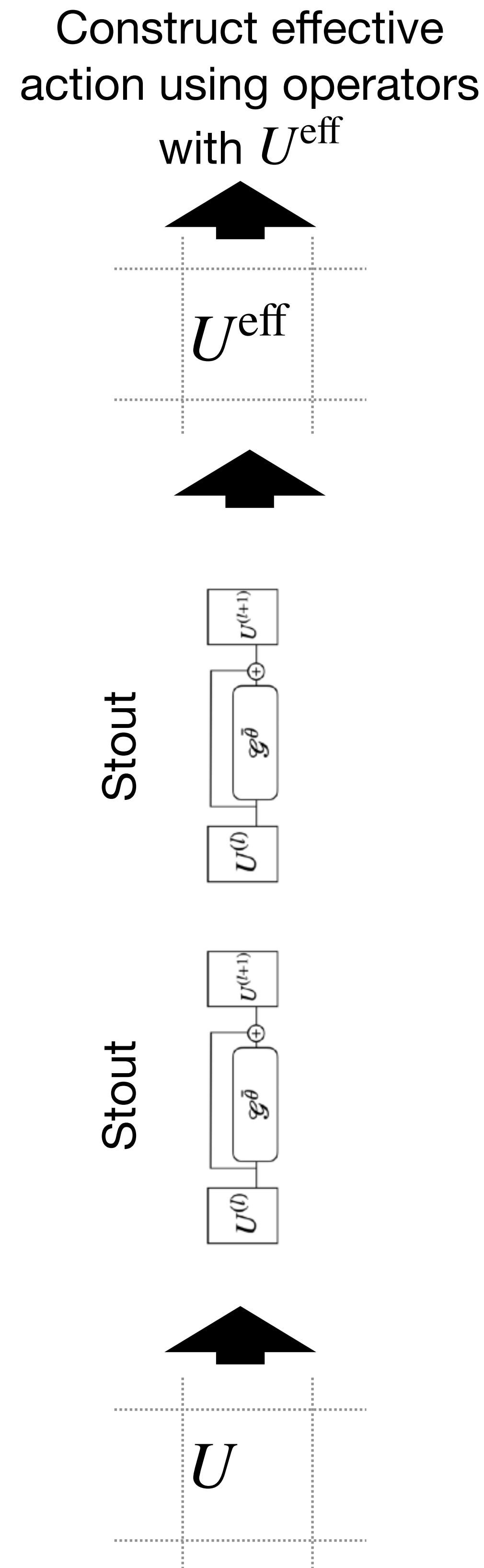
AT Y. Nagai arXiv: 2103.11965

Dictionary	(convolutional) Neural network	Gauge Covariant Neural network	
Input	Image (2d data, structured)	gauge config (4d data, structured)	
Output	Image (2d data, structured)	gauge config (4d data, structured)	
Symmetry	Translation	Translation, rotation(90°), Gauge sym.	
with Fixed param	Image filter	(APE/stout …) Smearing	
Local operation	Summing up nearest neighbor with weights	Summing up staples with weights	
Activation function	Tanh, ReLU, sigmoid, …	projection/normalization in Stout/HYP/HISQ	
Formula for chain rule	Backprop	"Smeared force calculations" (Stout)	**Well-known**
Training?	Backprop + Delta rule	AT Nagai 2103.11965	

(Index i in the neural net corresponds to n & μ in smearing. Information processing with NN is evolution of scalar field)

Gauge covariant neural net

Simulation parameter



- Self-learning HMC (1909.02255, 2021 AT+), an exact algorithm
  - Exact Metropolis test and MD with effective action
- Target S : $m = 0.3$, dynamical staggered fermion, Nf=2, $L^4 = 4^4$, SU(2), $\beta = 2.7$
- Effective action in MD (S^{eff})
  - Same gauge action
  - $m_{\text{eff}} = 0.4$ dynamical staggered fermion, Nf=2
  - U links are replaced by U^{eff} in D_{stag}
- "Adaptively reweighted HMC"

Details (skip)

Network: trainable stout (plaq+poly)

arXiv: 2103.11965

Structure of NN

(Polyakov loop+plaq in the stout-type)

$$\Omega_\mu^{(l)}(n) = \rho_{\text{plaq}}^{(l)} O_\mu^{\text{plaq}}(n) + \begin{cases} \rho_{\text{poly},4}^{(l)} O_4^{\text{poly}}(n) & (\mu = 4), \\ \rho_{\text{poly,s}}^{(l)} O_i^{\text{poly}}(n), & (\mu = i = 1,2,3) \end{cases}$$

All ρ is weight

O meas an loop operator

$$Q_\mu^{(l)}(n) = 2[\Omega_\mu^{(l)}(n)]_{\text{TA}}$$

TA: Traceless, anti-hermitian operation

$$U_\mu^{(l+1)}(n) = \exp(Q_\mu^{(l)}(n)) U_\mu^{(l)}(n)$$

$$U_\mu^{\text{NN}}(n)[U] = U_\mu^{(2)}(n)\Big[U_\mu^{(1)}(n)\Big[U_\mu(n)\Big]\Big]$$

2- layered stout

with 6 trainable parameters

Neural network Parametrized action:

$$S_\theta[U] = S_{\text{g}}\big[U\big] + S_{\text{f}}\big[\phi, U_\theta^{\text{NN}}[U]; m_{\text{h}} = 0.4\big],$$

Action for MD is built by gauge covariant NN

Loss function:

$$L_\theta[U] = \frac{1}{2}\left| S_\theta[U,\phi] - S[U,\phi] \right|^2,$$

Invariant under, rot, transl, gauge trf.

Training strategy:

1. Train the network in prior HMC (online training+stochastic gr descent)
2. Perform SLHMC with fixed parameter

Details (skip)

Results: Loss decreases along with the training

arXiv: 2103.11965

Loss function:

$$L_\theta[U] = \frac{1}{2}\left| S_\theta[U,\phi] - S[U,\phi] \right|^2,$$

Intuitively, e^(-L) is understood as Boltzmann weight or reweighting factor.

Prior HMC run (training)

$$\frac{\partial S}{\partial \rho_i^{(l)}} = 2\,\mathrm{Re}\sum_{\mu',m} \mathrm{tr}\left[U_{\mu'}^{(l)\dagger}(m)\Lambda_{\mu',m}\frac{\partial C}{\partial \rho_i^{(l)}}\right]$$

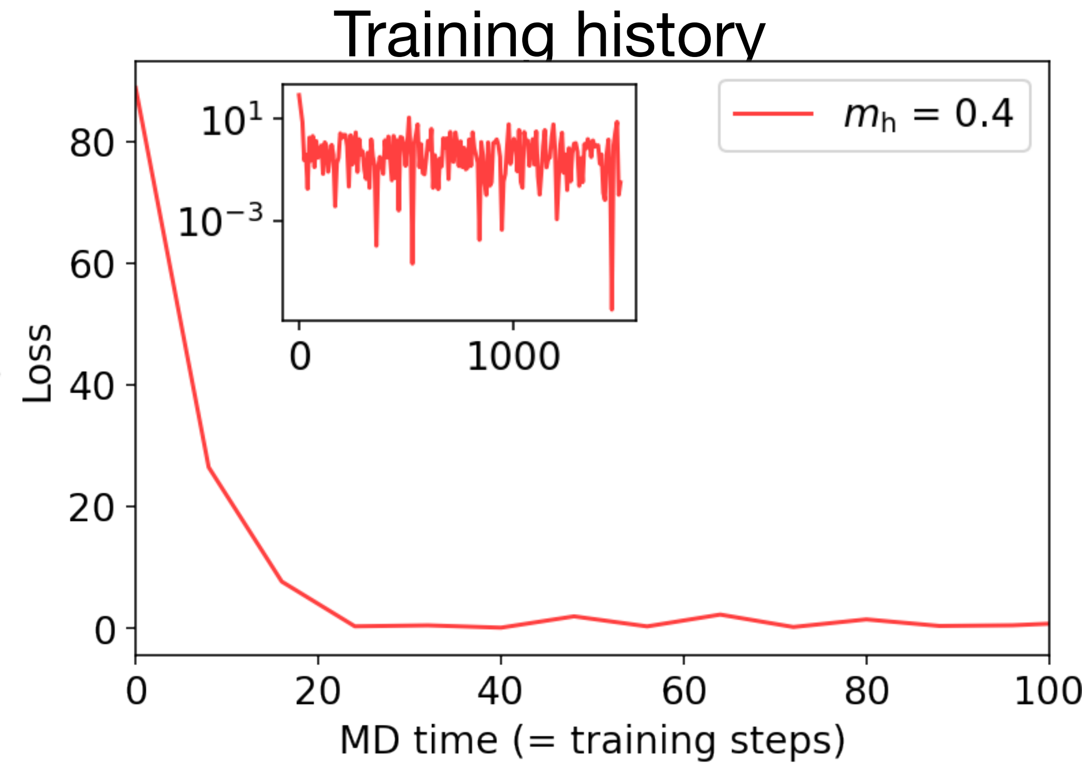
Ω: sum of un-traced loops

C: one U removed Ω

Λ: A polynomial of U. (Same object in stout)

$$\theta \leftarrow \theta - \eta\frac{\partial L_\theta(\mathcal{D})}{\partial \theta},$$

$$\frac{\partial L_\theta(\mathcal{D})}{\partial w_i^{(L-1)}} = \frac{\partial L_\theta(\mathcal{D})}{\partial S_\theta}\frac{\partial S_\theta}{\partial w_i^{(L-1)}}$$



Without training, e^(-L)<< 1,
this means that candidate with approximated action never accept.
After training, e^(-L) ~1, and we get practical acceptance rate!

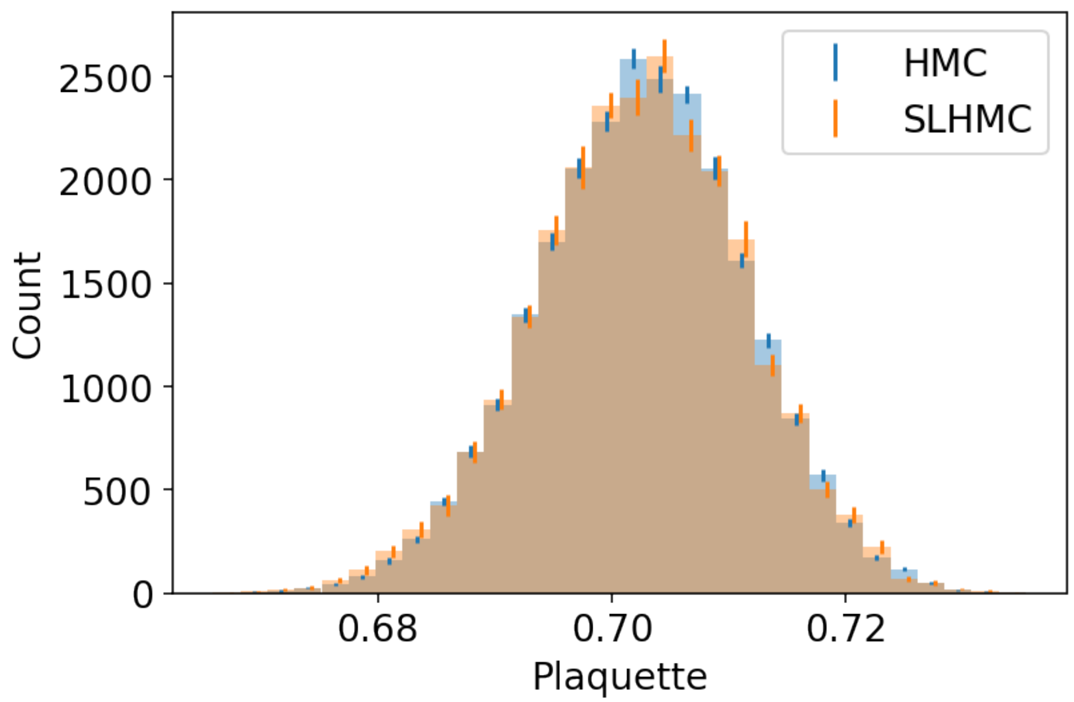
We perform SLHMC with these values!

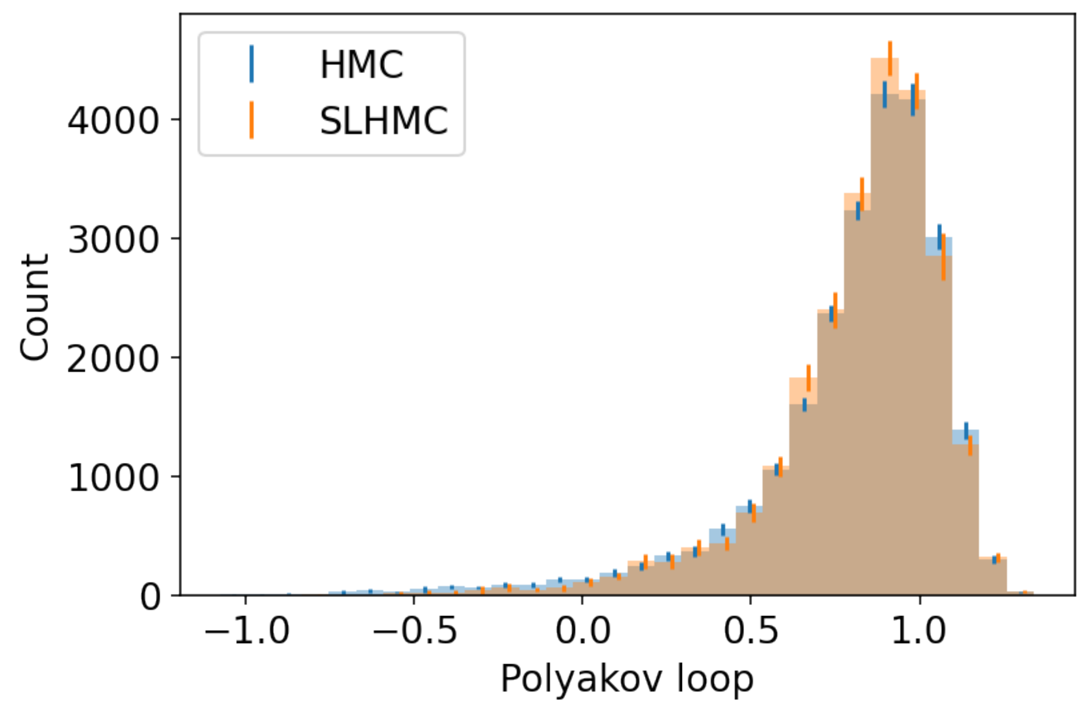
Application for the staggered in 4d

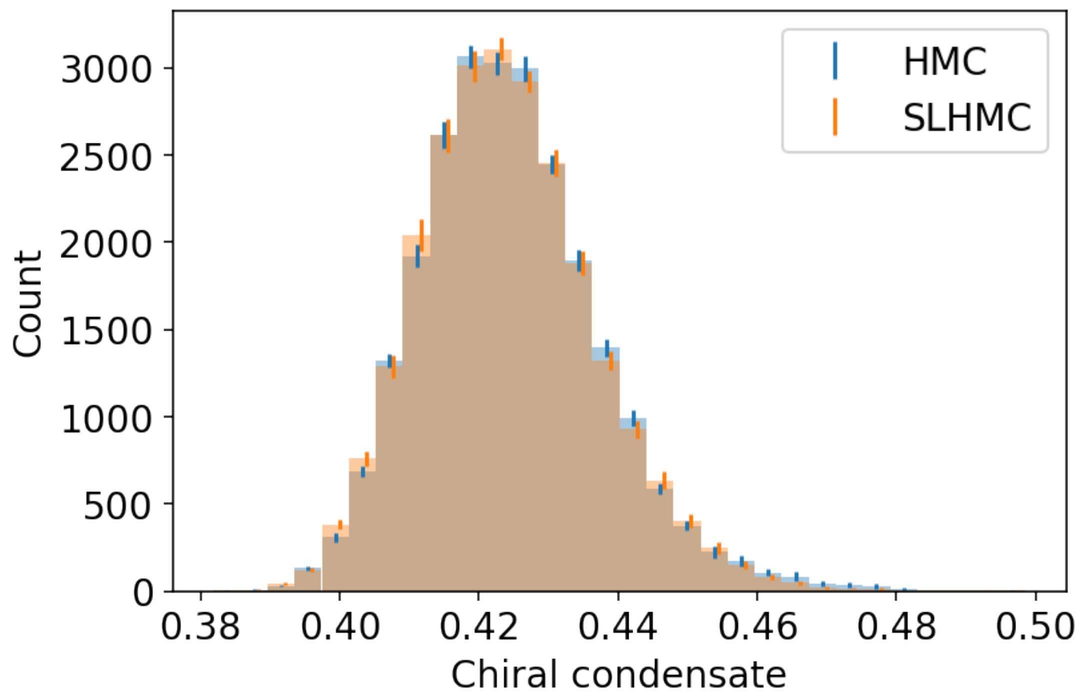
Akio Tomiya

Results are consistent with each other (stout-type used)

arXiv: 2103.11965







Expectation value

Algorithm	Observable	Value
HMC	Plaquette	0.7025(1)
SLHMC	Plaquette	0.7023(2)
HMC	|Polyakov loop|	0.82(1)
SLHMC	|Polyakov loop|	0.83(1)
HMC	Chiral condensate	0.4245(5)
SLHMC	Chiral condensate	0.4241(5)

Implemented by LatticeQCD.jl julia

Overview/outline

Gauge covariant transformer for LQCD

Two conditions/restrictions in LQCD:

Gauge symmetry
U(x, x+μ)

Non-locality from
pseudo-fermions
(1/D) ~ non-local
(I want to mimic
this by NN)

Solutions in neural net:

1. Gauge covariant net
(adaptive stout)
arXiv: 2103.11965 AT+

2. **Transformer with global symmetry**
(Heisenberg spin + electron)
2310.13222 AT+
2306.11527 AT+

3. Gauge symmetric Transformer for LQCD
This talk

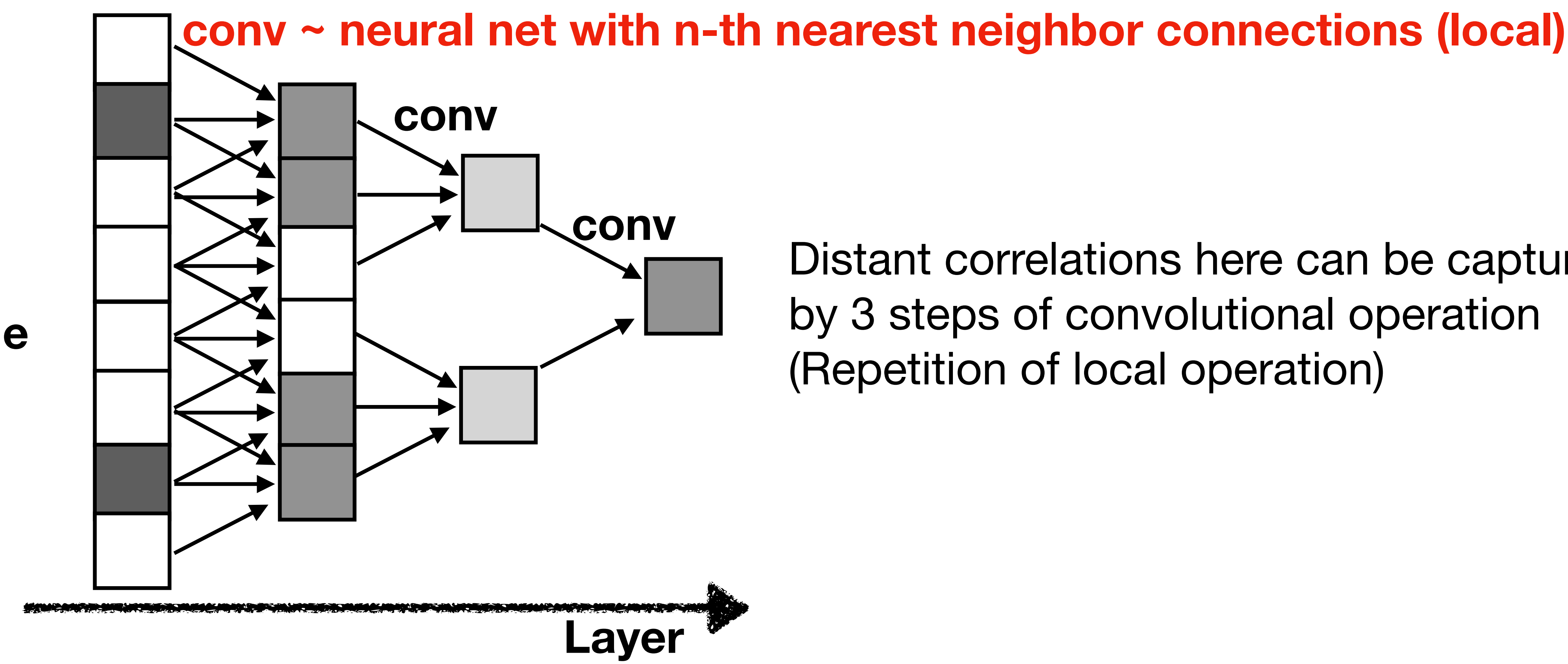
Equivariance and convolution

Akio Tomiya

Convolutional Neural network have been good job but local

Convolutional neural layers in neural networks keep translational symmetry,
it can be generalized to any continuous/discrete symmetry in the theory. It helps generalization.

e.g.
1d Input image



Distant correlations here can be captured by 3 steps of convolutional operation (Repetition of local operation)

However, 1 step of convolutional layer can pick up only local correlation and representability of neural networks is limited. Global correlations are sometimes important.
How can we overcome these difficulties?

Transformer and Attention

Attention layer used in Transformers (GPT, Bard)

Akio Tomiya

arXiv:1706.03762

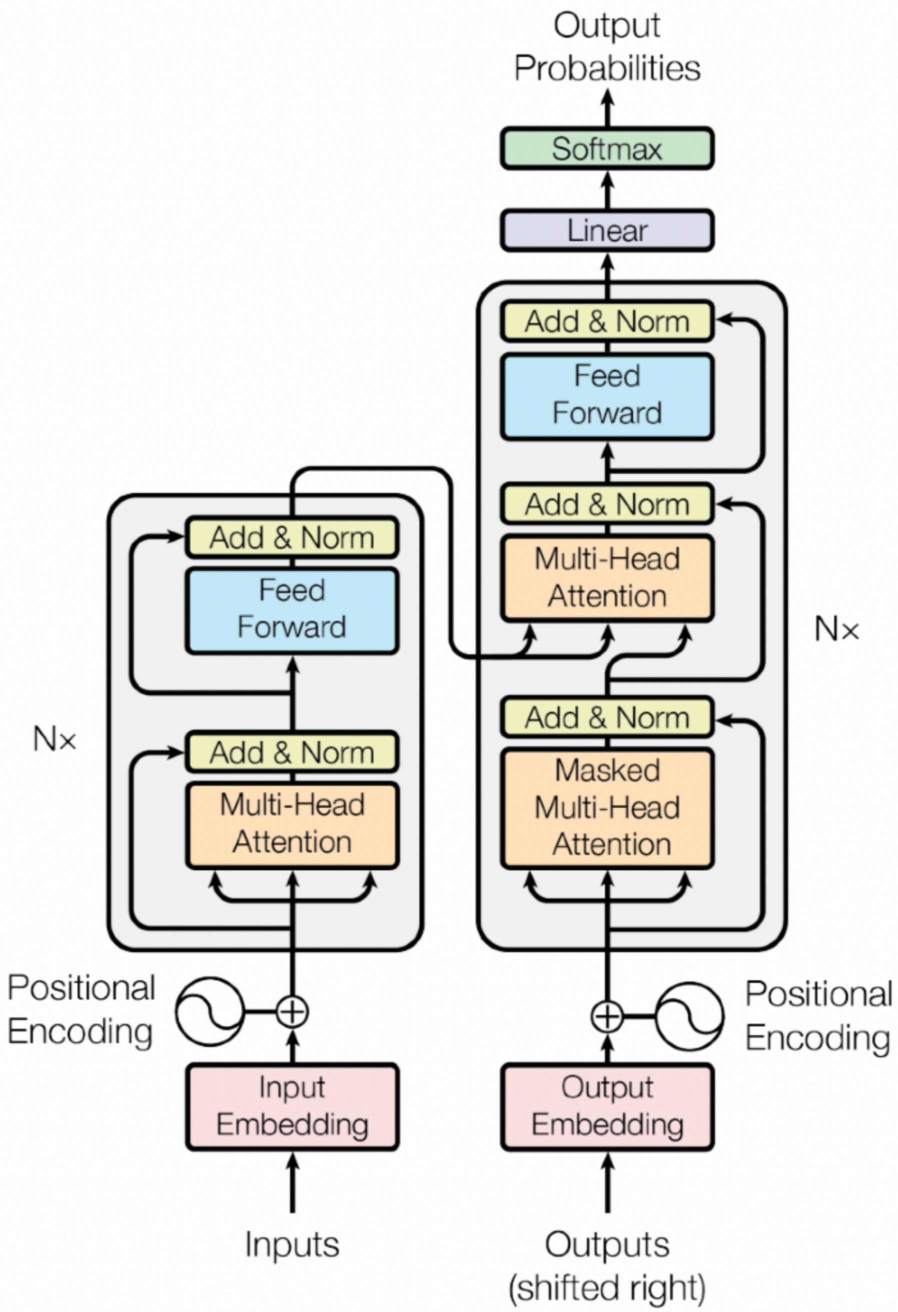


Figure 1: The Transformer - model architecture.

Attention layer (in transformer model) has been introduced in a paper titled **"Attention is all you need"** (1706.03762)

State of the art architecture of language processing.

Attention layer is essential.

Transformer and Attention

Attention layer can capture non-local correlations

arXiv:1706.03762

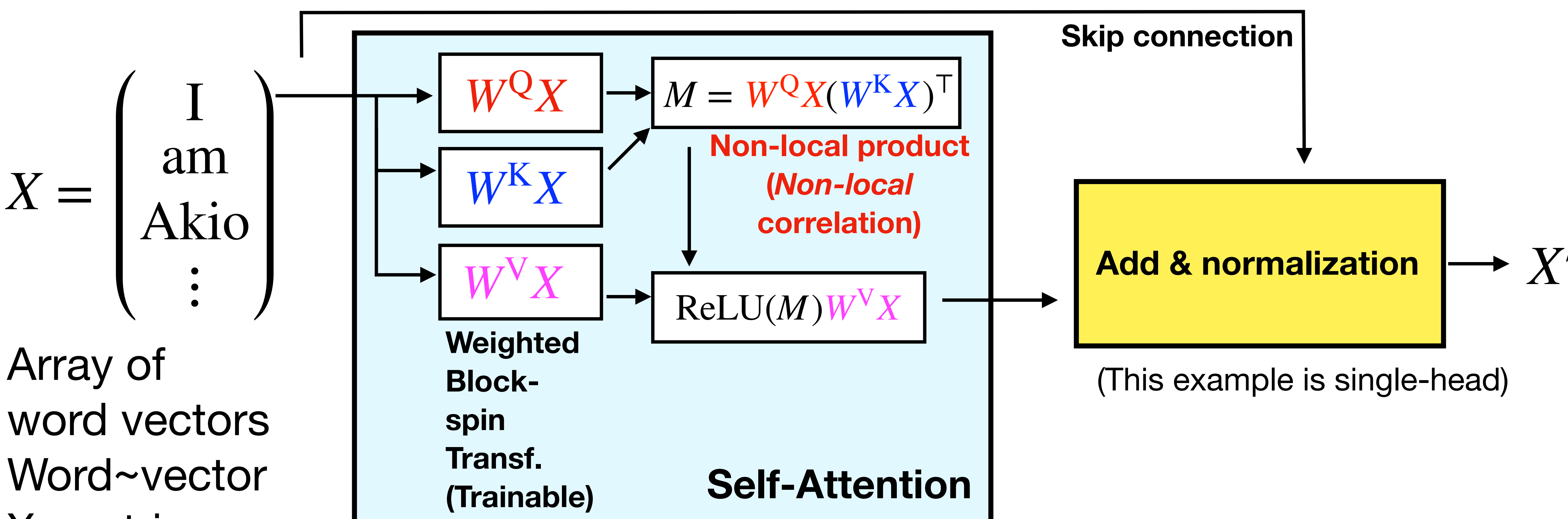
Modifier in language can be non-local

Eg. I am Akio Tomiya living in Japan, who studies machine learning and physics

In physics terminology, this is **non local correlation.**

The attention layer enables us to treat non-local correlation with a neural net!

Simplified version of Attention/Transformer



These can be repeated

Transformer and Attention

Akio Tomiya

Transformer shows scaling lows (power law)

arXiv: 2001.08361

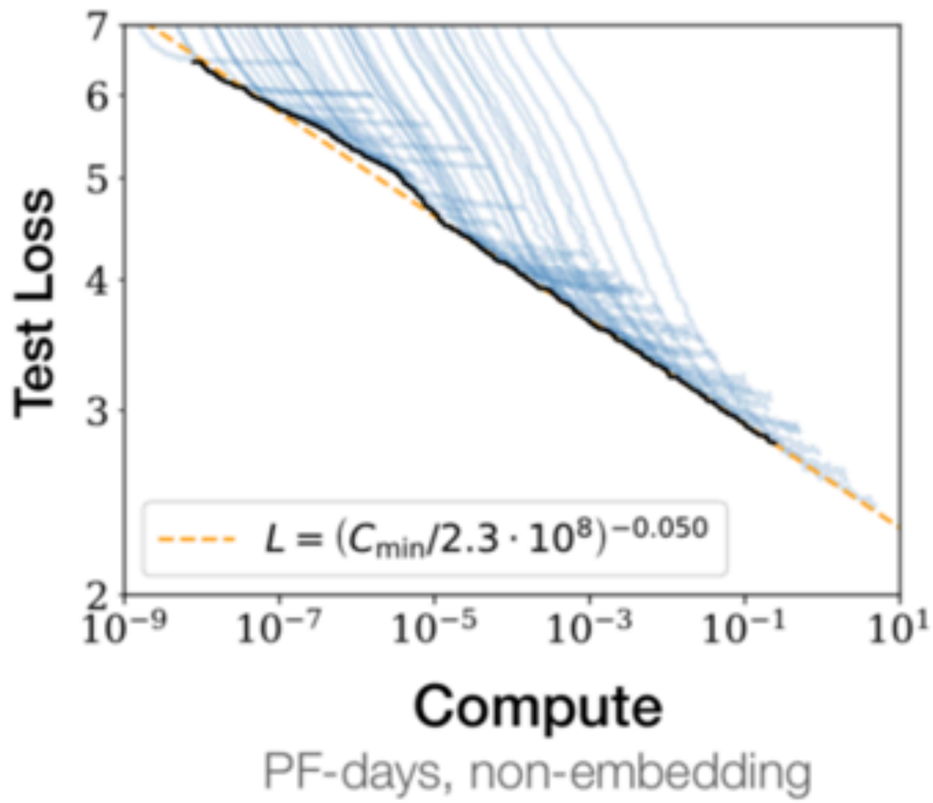


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

- **Transformers requires huge data**
  (e.g. GPT uses all electric books in the world)
  Because it has few inductive bias (no equivariance)
- **It can be improved systematically**

Transformer and Attention

Akio Tomiya

Physically symmetric Attention layer

Attention layer can capture global correlation
Equivariance reduces data demands for training

	Equivariance	**Capturable correlation**	**Data demmands**	**Applications**
Convolution ($\in$ equivariant layers)	Yes	Local	Low	Image recognition VAE, GAN Normalizing flow
Standard Attention layer	No	Global	Huge	ChatGPT GEMINI Vision Transformer arXiv:1706.03762
Physically *Equivariant* attention layer	Yes	Global	?	Kondo system (this work) arXiv: 2306.11527

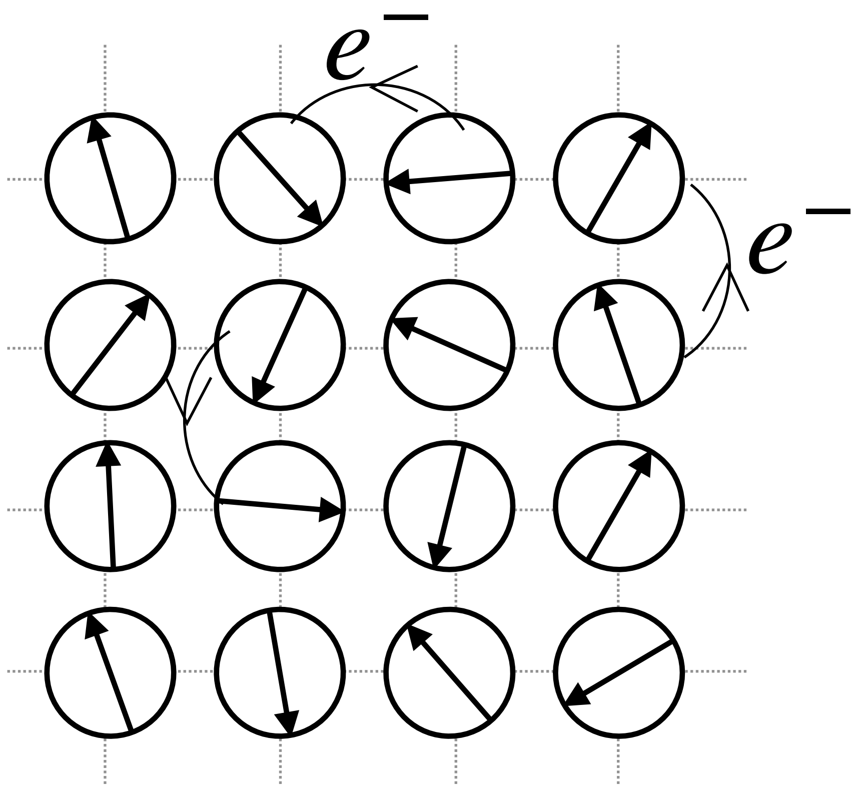
Self-learning Monte-Carlo

Akio Tomiya

Target: Double exchange model

Target system: Classical Heisenberg spin $\mathbf{S}_i$+ Fermion on 2d lattice

$$H = -t \sum_{\alpha,\langle i,j \rangle} (\hat{c}^\dagger_{i\alpha} \hat{c}_{j\alpha} + \mathrm{h.c.}) + \frac{J}{2} \sum_i \mathbf{S}_i \cdot \hat{\sigma}_i \qquad \text{(Kondo model)}$$



3d vectors on 2d lattice
Anti-ferro magnet

Two different phases
- Anti-ferromagnet (~staggered mag)
- Paramagnet (~normal metal)

(This system is similar to lattice QCD but easier)

Self-learning Monte-Carlo
Previous work

Akio Tomiya

Target system: Classical Heisenberg spin $\mathbf{S}_i$+ Fermion on 2d lattice

$$H = -t \sum_{\alpha,\langle i,j \rangle} (\hat{c}^\dagger_{i\alpha} \hat{c}_{j\alpha} + \mathrm{h.c.}) + \frac{J}{2} \sum_i \mathbf{S}_i \cdot \hat{\sigma}_i$$ (Kondo model)

Naive effective model:

$$H^{\mathrm{Linear}}_{\mathrm{eff}} = -\sum_{\langle i,j \rangle_n} J^{\mathrm{eff}}_n \mathbf{S}_i \cdot \mathbf{S}_j + E_0$$

J^{eff}_n: n-th nearest neighbor

J^{eff}_n is determined by regression (training) to improve approximation

Self-learning Monte-Carlo:

Update with H_{eff} , and Metropolis-Hastings with H & H_{eff}

Cancel inexactness. This is an exact algorithms

Self-learning Monte-Carlo

Previous work

Target system: Classical Heisenberg spin S_i+ Fermion on 2d lattice

$$H = -t \sum_{\alpha,\langle i,j\rangle} (\hat{c}^\dagger_{i\alpha}\hat{c}_{j\alpha} + \mathrm{h.c.}) + \frac{J}{2}\sum_i \mathbf{S}_i \cdot \hat{\sigma}_i$$

(Kondo model)

This has fermion det

Naive effective model:

$$H^{\mathrm{Linear}}_{\mathrm{eff}} = -\sum_{\langle i,j\rangle_n} J^{\mathrm{eff}}_n \mathbf{S}_i \cdot \mathbf{S}_j + E_0$$

J^{eff}_n: n-th nearest neighbor

$$H_{\mathrm{eff}} = -\sum_{\langle i,j\rangle_n} J^{\mathrm{eff}}_n \mathbf{S}^{\mathrm{NN}}_i \cdot \mathbf{S}^{\mathrm{NN}}_j + E_0$$

We replace this by
"translated" spin S^{NN}_i
with a transformer
and used in self-learning MC

mimics effects from fermions
with smeared spins

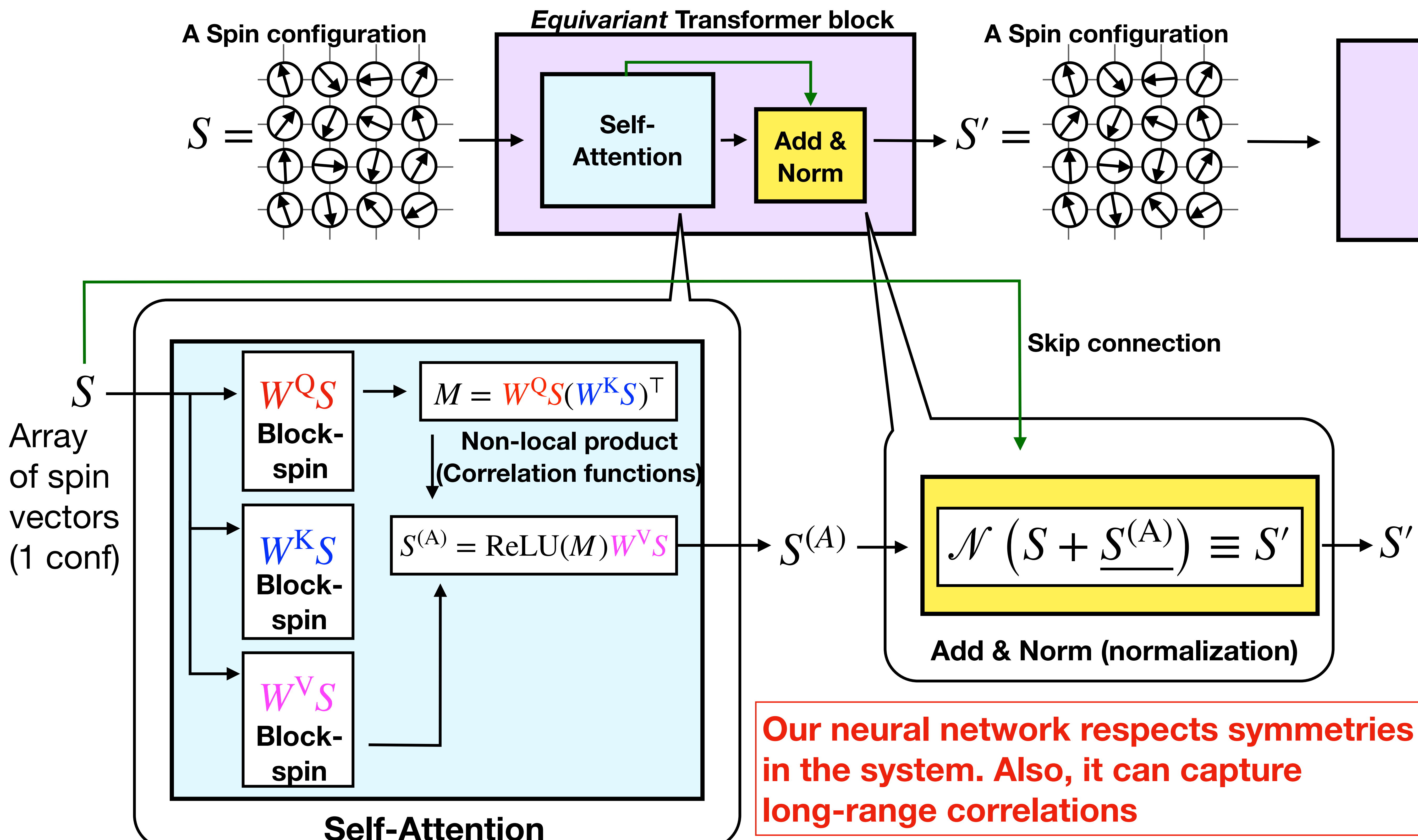
This doesn't have fermion det

Self-learning Monte-Carlo

Physically equivariant Attention layer/Transformer

Akio Tomiya

arXiv: 2306.11527.



Our neural network respects symmetries in the system. Also, it can capture long-range correlations

arXiv: 2306.11527.

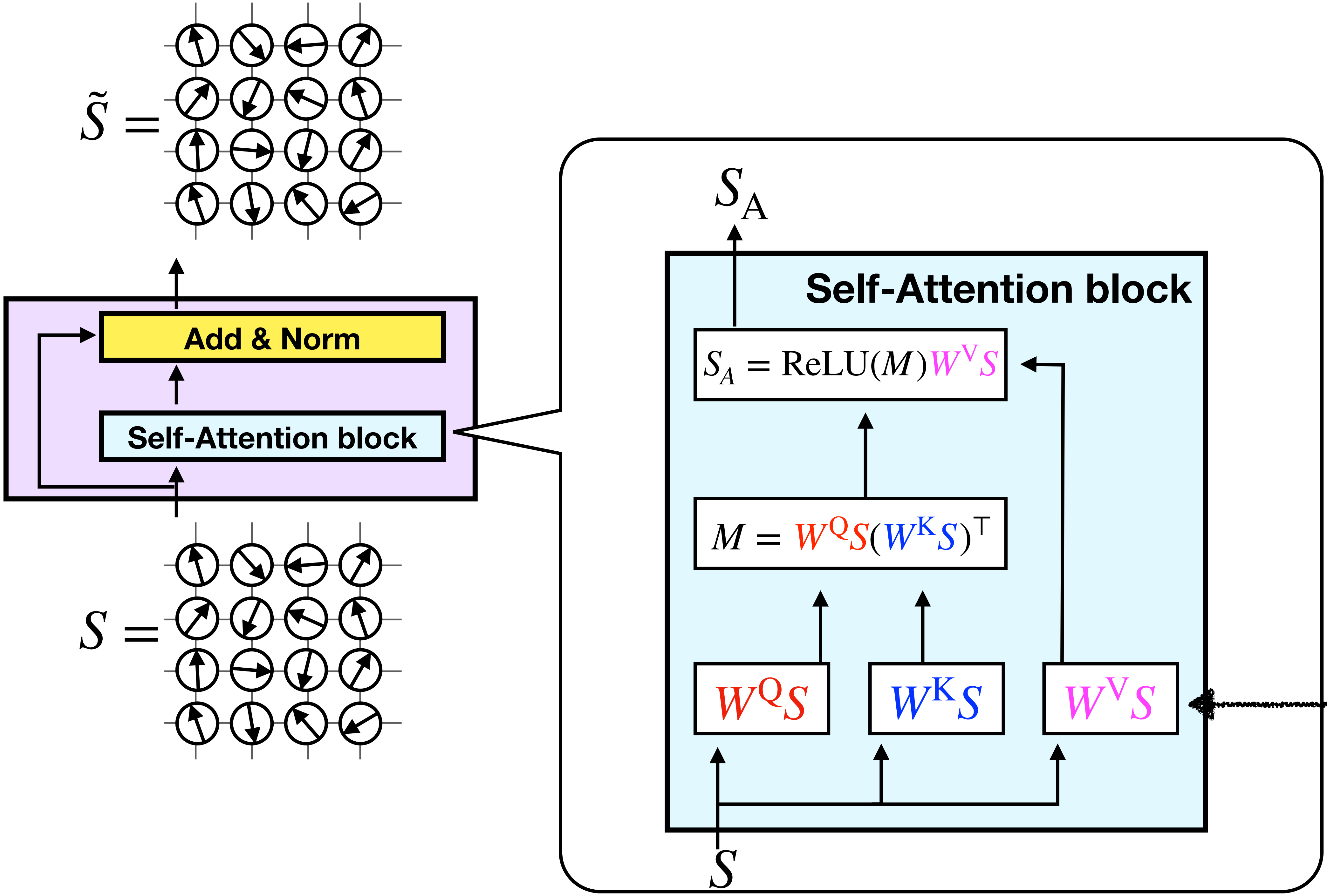
Equivariant attention

Self-learning Monte-Carlo

Akio Tomiya

arXiv: 2306.11527.

Attention block makes effective spin field with non-local BST



Smearing (BST)
Rot. equivariant
Trsl. equivariant
trainable!

Self-learning Monte-Carlo

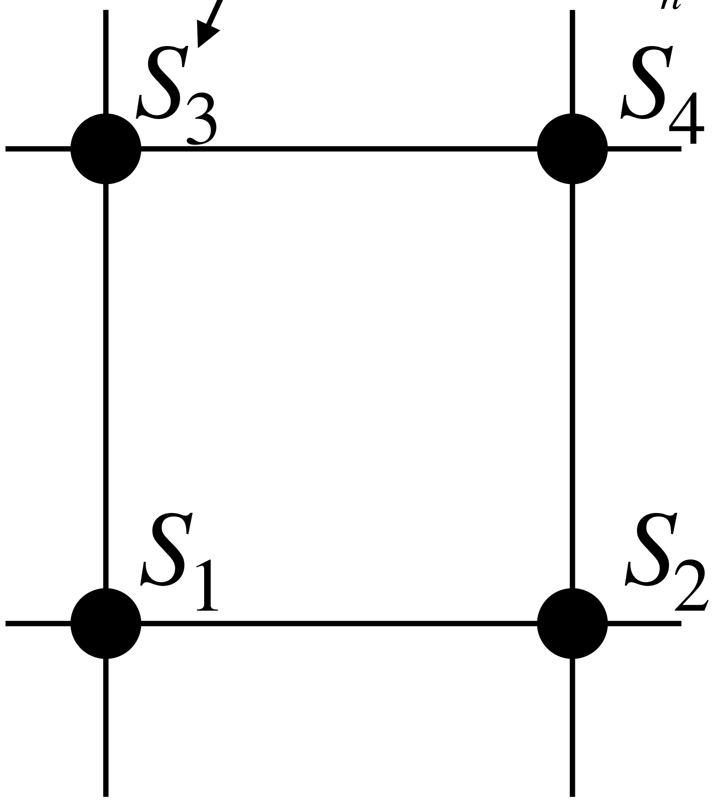
Equivariant under spin-rotation & translation

Akio Tomiya

arXiv: 2306.11527.

$$S_3^\top = \begin{pmatrix} s_3^1 & s_3^2 & s_3^3 \end{pmatrix}^\top$$

s_n^μ : O(3) vector



$$S_i^\top = \begin{pmatrix} s_i^1 & s_i^2 & s_i^3 \end{pmatrix}^\top$$

$$|S_i| = \sqrt{(s_i^1)^2 + (s_i^2)^2 + (s_i^3)^2}$$
$$= 1$$

3 component scalar, normalized

$$\mathbf{S} = \begin{pmatrix} S_1^\top & S_2^\top & S_3^\top & S_4^\top \end{pmatrix}^\top$$

- Local weighted sum over neighbors
= "Smeared spin" with parameters
~ "Block spin sum" with parameters

$$\tilde{S}_i^\alpha = \sum_{l=0} w_l^\alpha S_{i+l} \qquad \alpha = \mathrm{Q}, \mathrm{K}, \mathrm{V}$$

$w_l^\alpha \in \mathbb{R}$: trainable

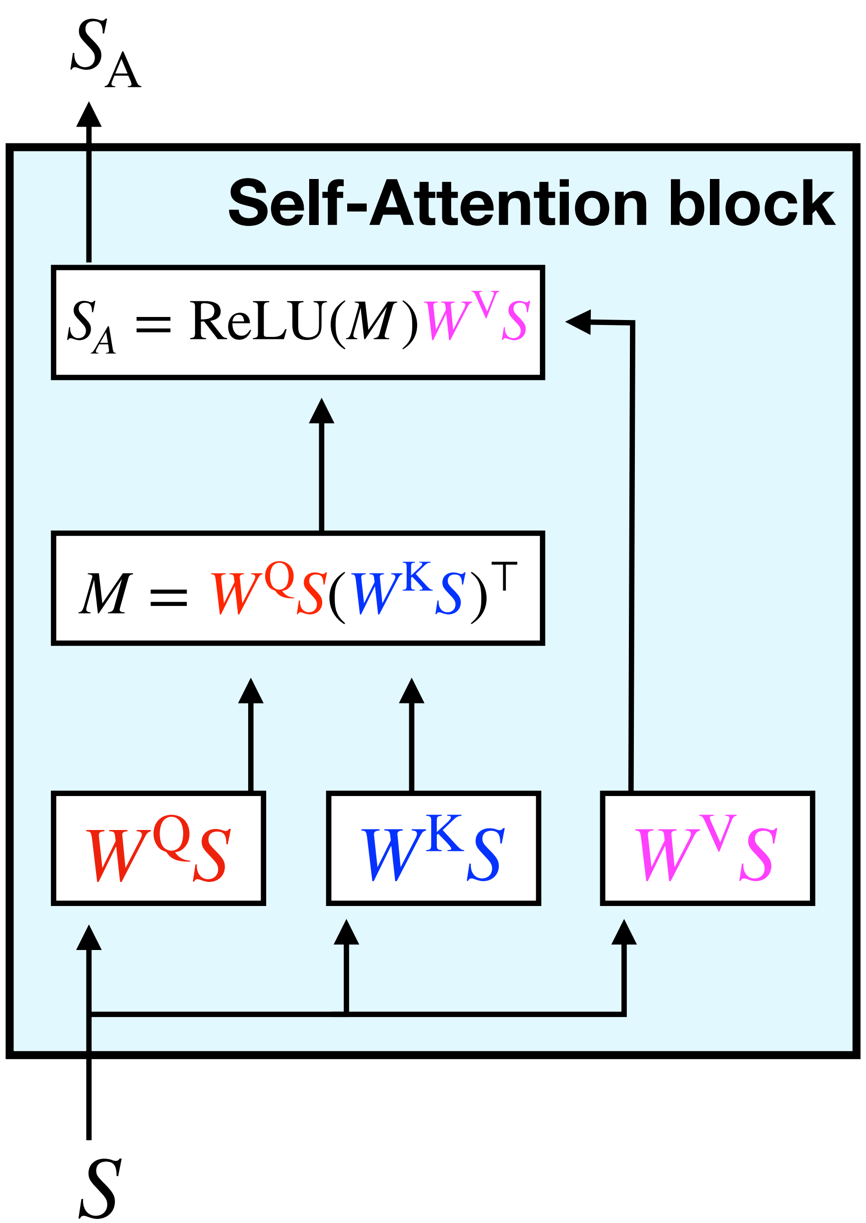
Translationally equivariant
Rotationally equivariant

Self-learning Monte-Carlo

Akio Tomiya

Equivariant under spin-rotation & translation

arXiv: 2306.11527.



$$\mathbf{S} = \begin{pmatrix} S_1^\top & S_2^\top & S_3^\top & S_4^\top \end{pmatrix}^\top$$

$$S_i^\top = \begin{pmatrix} s_i^1 & s_i^2 & s_i^3 \end{pmatrix}^\top$$

s_n^μ : O(3) vector

$$\tilde{S}_i^\alpha = W^\alpha S = \sum w_l^\alpha S_{i+l}$$

"averaged spin" by neighbors

Gram matrix with averaged spin

$$M = \tilde{G}^\alpha \equiv (\tilde{\mathbf{S}}^\alpha)^\top \tilde{\mathbf{S}}^\alpha \quad \alpha = \mathrm{Q}, \mathrm{K}, \mathrm{V}$$

$$G \equiv \mathbf{S}^\top \mathbf{S} = \begin{pmatrix} S_1^\top S_1 & S_1^\top S_2 & S_1^\top S_3 & S_1^\top S_4 \\ S_2^\top S_1 & S_2^\top S_2 & S_2^\top S_3 & S_2^\top S_4 \\ S_3^\top S_1 & S_3^\top S_2 & S_3^\top S_3 & S_3^\top S_4 \\ S_4^\top S_1 & S_4^\top S_2 & S_4^\top S_3 & S_4^\top S_4 \end{pmatrix}$$

Translationally covariant, Rotationally invariant

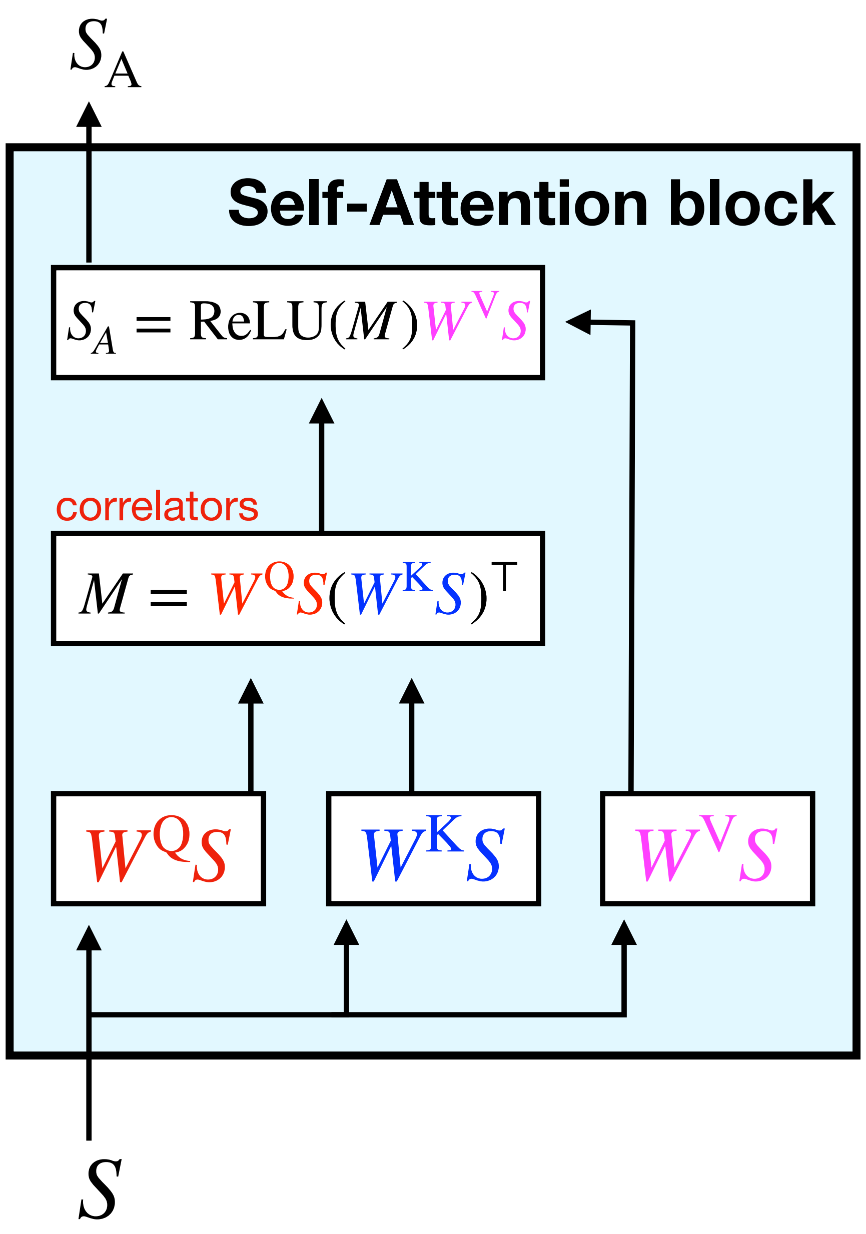
A set of correlators

Self-learning Monte-Carlo

Equivariant under spin-rotation & translation

Akio Tomiya

arXiv: 2306.11527.



$$\mathbf{S} = \begin{pmatrix} S_1^\top & S_2^\top & S_3^\top & S_4^\top \end{pmatrix}^\top$$

$$S_i^\top = \begin{pmatrix} s_i^1 & s_i^2 & s_i^3 \end{pmatrix}^\top$$

s_n^μ : O(3) vector

$$\tilde{S}_i^\alpha = W^\alpha S = \sum w_l^\alpha S_{i+l}$$

"averaged spin" by neighbors

Gram matrix with averaged spin

$$M = \tilde{G}^\alpha \equiv (\tilde{\mathbf{S}}^\alpha)^\top \tilde{\mathbf{S}}^\alpha \quad \alpha = \mathrm{Q}, \mathrm{K}, \mathrm{V}$$

Translationally covariant
Rotationally invariant

$$S_A = \mathrm{ReLU}(M) W^\mathrm{V} S$$
$$= \mathrm{ReLU}(M) \tilde{S}^\mathrm{V}$$

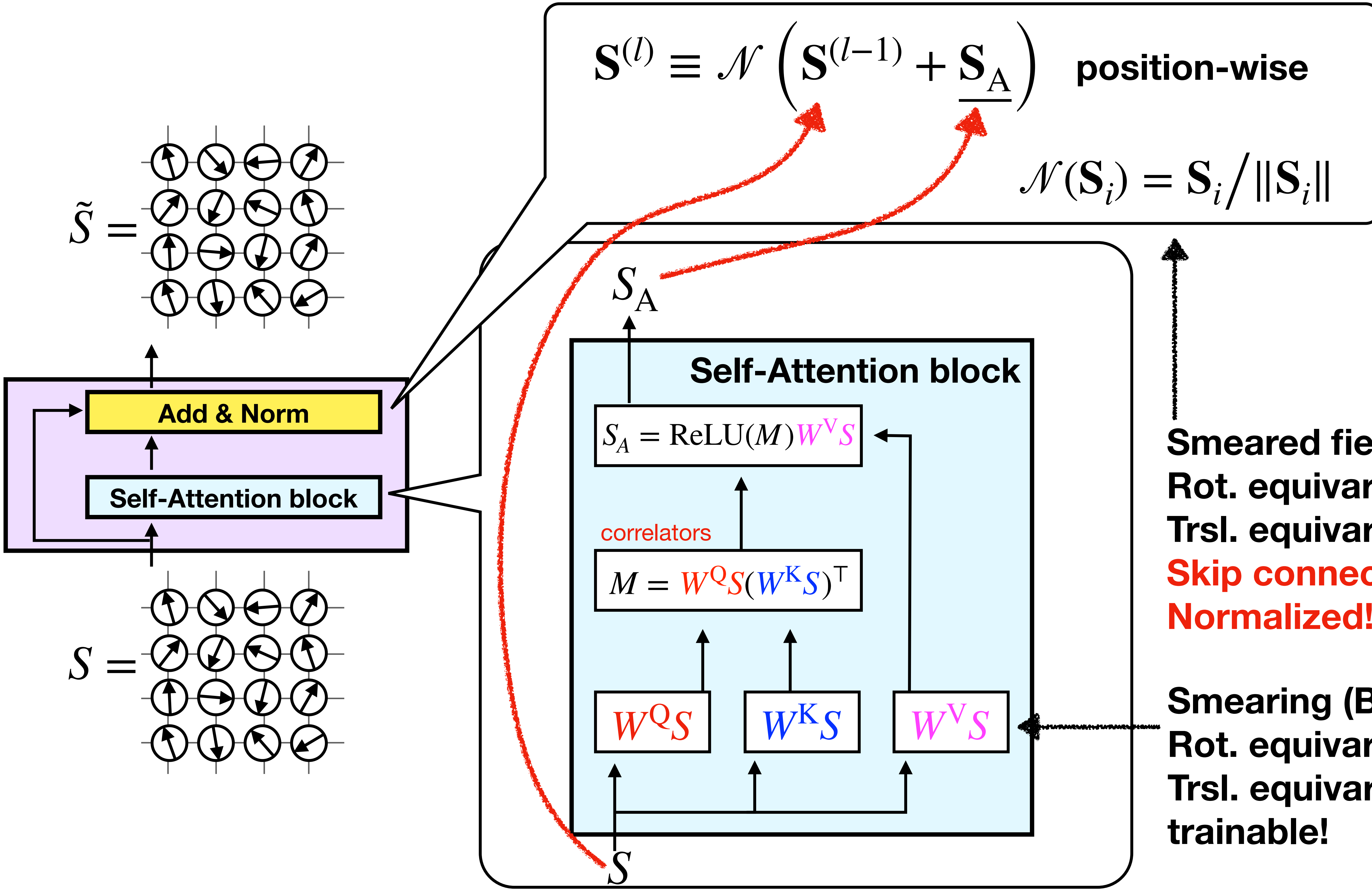
A set of correlators

Self-learning Monte-Carlo

Akio Tomiya

arXiv: 2306.11527.

Attention block makes effective spin field with non-local BST



Smeared fields
Rot. equivariant
Trsl. equivariant
Skip connection
Normalized!

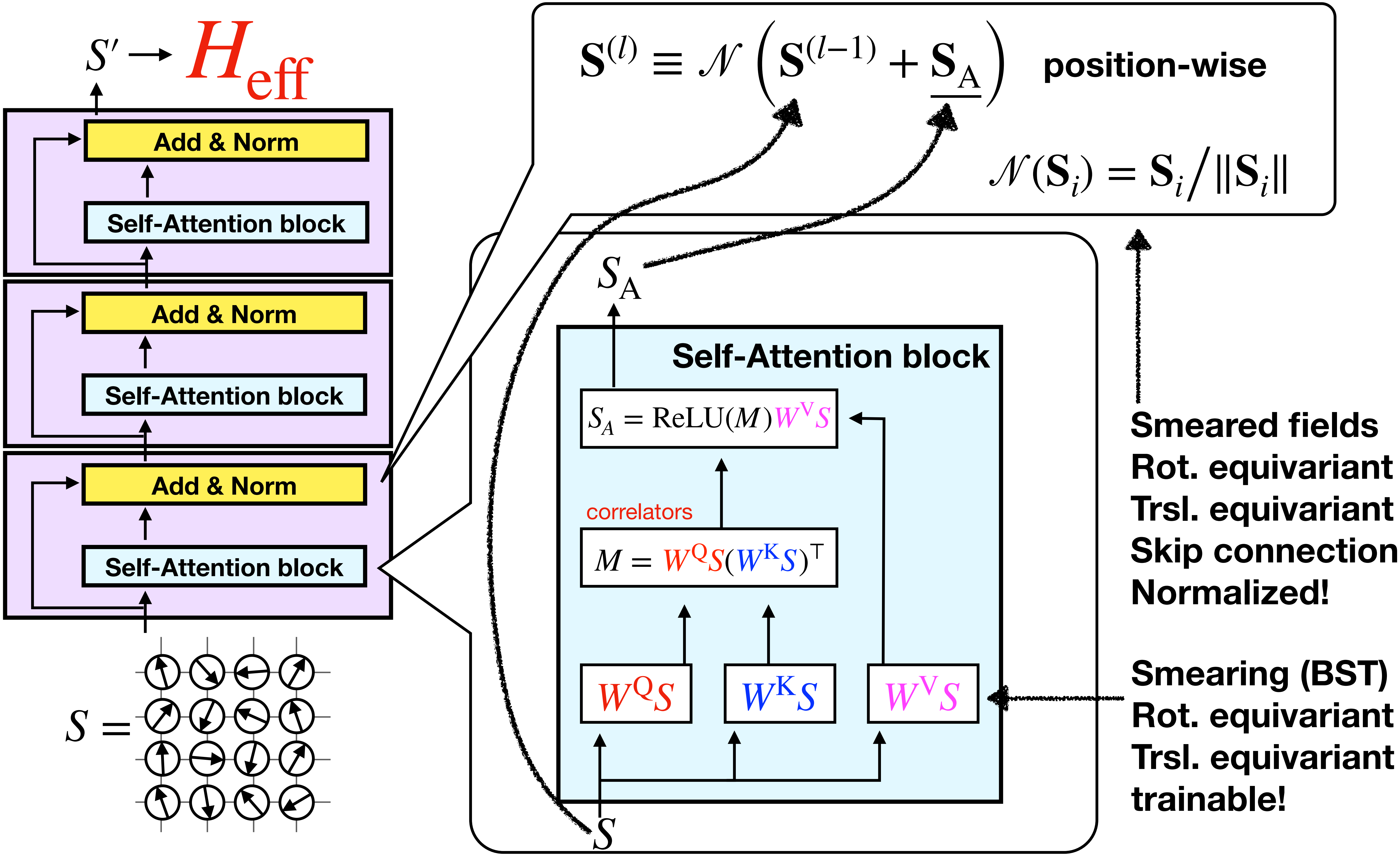
Smearing (BST)
Rot. equivariant
Trsl. equivariant
trainable!

Self-learning Monte-Carlo

Akio Tomiya

arXiv: 2306.11527.

Variational Hamiltonian with Equivariant Attention layers



Self-learning Monte-Carlo

Akio Tomiya

SLMC = MCMC with an effective model/ Adaptive rew

arXiv:1610.03137+

For statistical spin system, we want to calculate expectation value with

$$W(\{\mathbf{S}\}) \propto \exp[-\beta H(\{\mathbf{S}\})]$$

On the other hand, an effective model $H_{\mathrm{eff}}(\{\mathbf{S}\})$ can compose in MCMC,

$\{\mathbf{S}\} \xrightarrow{\text{eff}} \{\mathbf{S}\} \xrightarrow{\text{eff}} \{\mathbf{S}\} \xrightarrow{\text{eff}} \{\mathbf{S}\}$ this distributes $W_{\mathrm{eff}}(\{\mathbf{S}\}) \propto \exp[-\beta H_{\mathrm{eff}}(\{\mathbf{S}\})]$

if the update「→」 satisfies the detailed balance. We can employ Metropolis test like

$$A_{\mathrm{eff}}(\{\mathbf{S}'\}, \{\mathbf{S}\}) = \min\left(1, W_{\mathrm{eff}}(\{\mathbf{S}'\})/W_{\mathrm{eff}}(\{\mathbf{S}\})\right).$$

Self-learning Monte-Carlo

Akio Tomiya

SLMC = MCMC with an effective model/ Adaptive rew

arXiv:1610.03137+

For statistical spin system, we want to calculate expectation value with

$$W(\{\mathbf{S}\}) \propto \exp[-\beta H(\{\mathbf{S}\})]$$

On the other hand, an effective model $H_{\text{eff}}(\{\mathbf{S}\})$ can compose in MCMC,

$\{\mathbf{S}\} \xrightarrow{\text{eff}} \{\mathbf{S}\} \xrightarrow{\text{eff}} \{\mathbf{S}\} \xrightarrow{\text{eff}} \{\mathbf{S}\}$ this distributes $W_{\text{eff}}(\{\mathbf{S}\}) \propto \exp[-\beta H_{\text{eff}}(\{\mathbf{S}\})]$

if the update「→」 satisfies the detailed balance. We can employ Metropolis test like

$$A_{\text{eff}}(\{\mathbf{S}'\}, \{\mathbf{S}\}) = \min\left(1, W_{\text{eff}}(\{\mathbf{S}'\})/W_{\text{eff}}(\{\mathbf{S}\})\right).$$

SLMC: Self-learning Monte-Carlo

We can construct *double* MCMC with $H(\{\mathbf{S}\})$ and $H_{\text{eff}}(\{\mathbf{S}\})$

$$\{\mathbf{S}\} \xrightarrow{\text{eff}} \{\mathbf{S}\} \xrightarrow{\text{eff}} \{\mathbf{S}\} \xrightarrow{\text{eff}} \{\mathbf{S}\} \longrightarrow \{\mathbf{S}\} \xrightarrow{\text{eff}} \{\mathbf{S}\} \xrightarrow{\text{eff}} \{\mathbf{S}\} \xrightarrow{\text{eff}} \{\mathbf{S}\} \longrightarrow$$

with Metropolis-*Hastings* test: $A(\{\mathbf{S}'\}, \{\mathbf{S}\}) = \min\left(1, \frac{W(\{\mathbf{S}'\})}{W(\{\mathbf{S}\})} \frac{W_{\text{eff}}(\{\mathbf{S}\})}{W_{\text{eff}}(\{\mathbf{S}'\})}\right).$

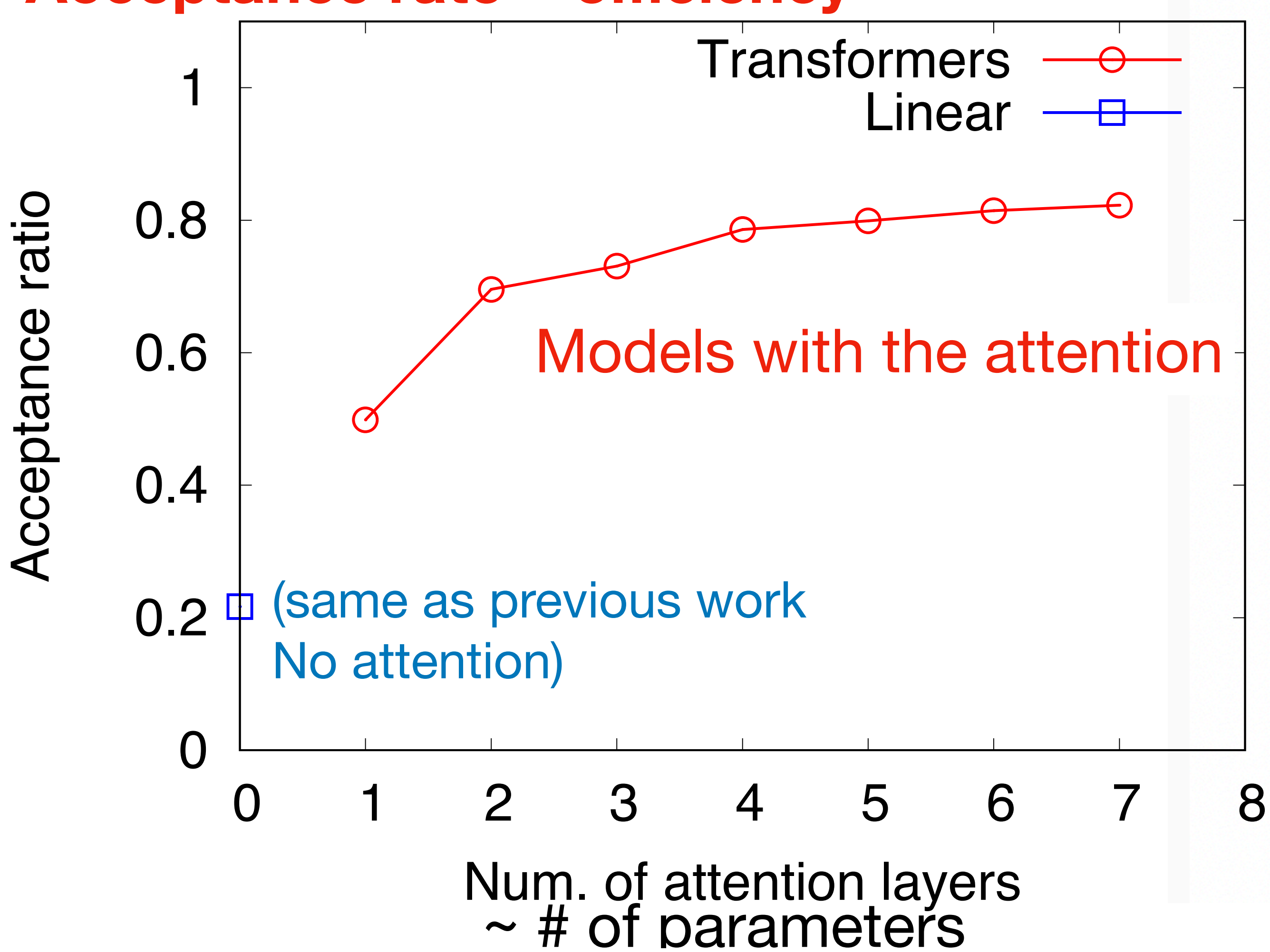
- **Effective model can have fit parameters**
- **Exact! It satisfies detailed balance with $W(\{\mathbf{S}\})$ (exact)**
- **It has been used for full QCD too (arXiv: 2010.11900, 2103.11965)**

Transformer and Attention

Akio Tomiya
arXiv: 2306.11527 + update

Application to O(3) spin model with fermions

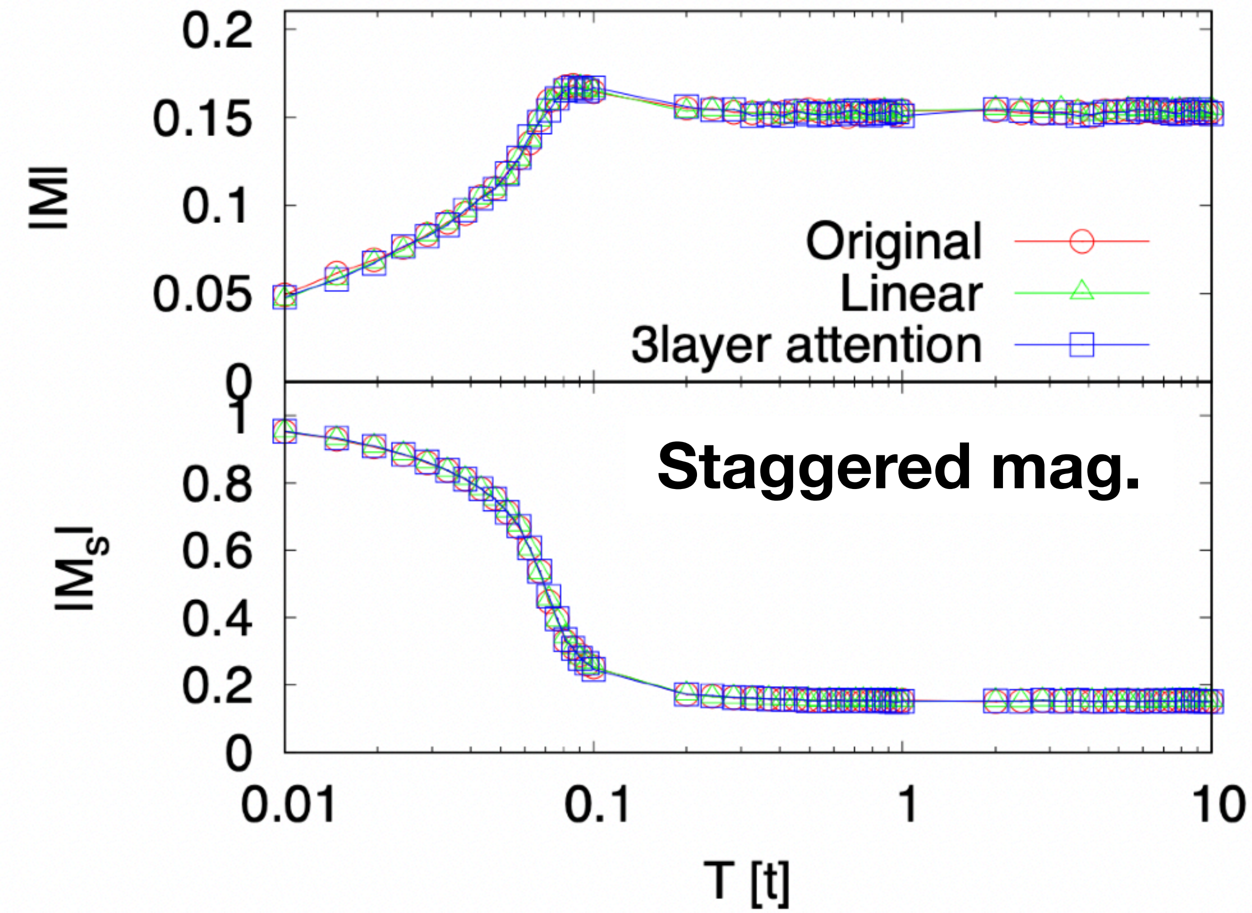
Acceptance rate ~ efficiency



Note: As far as we tested,
CNN-type does not work in this case.

No improvements with increase of layers.
(Global correlations of fermions from
Fermi-Dirac statistics make acceptance bad?)

Observables



Physical values are consistent
(as we expected)

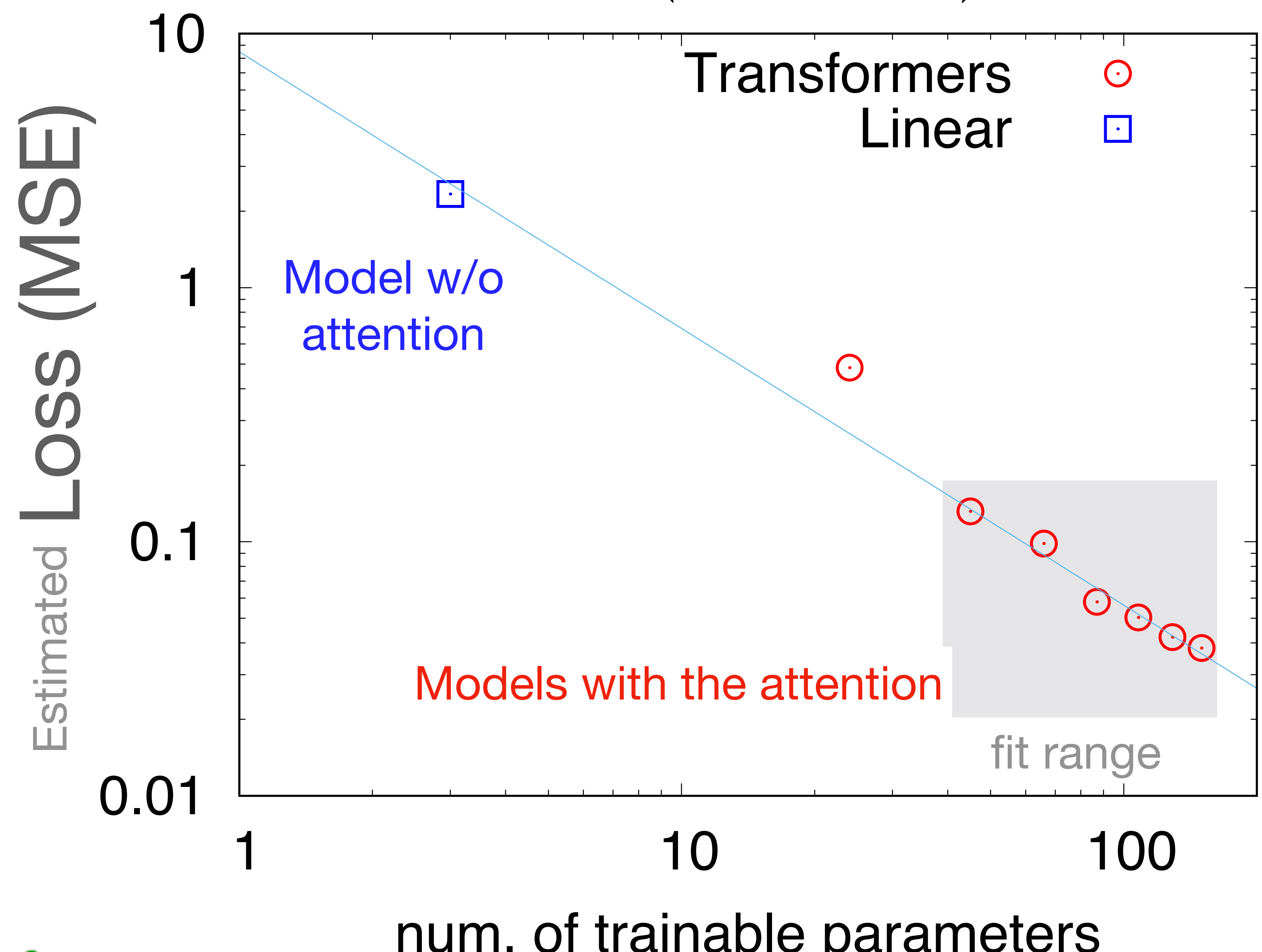
Nx=Ny=6
(Lattice sites)

Transformer and Attention

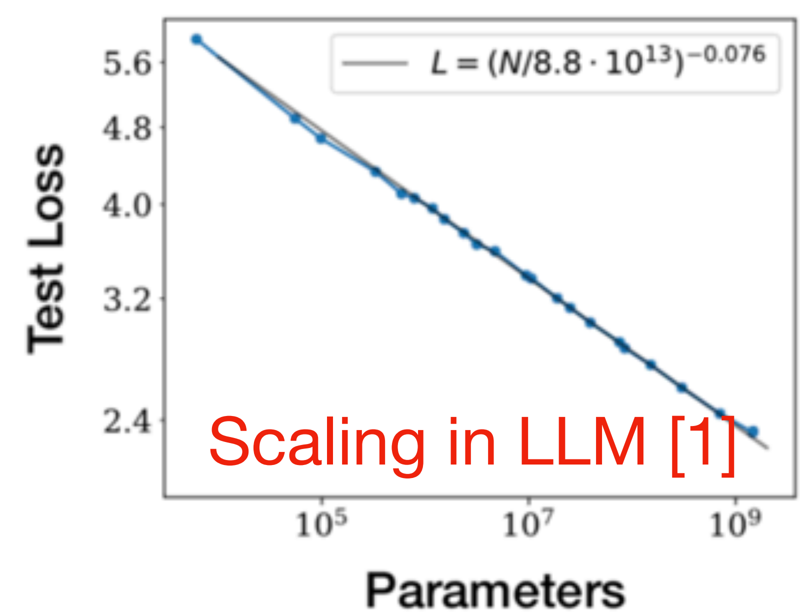
Loss function shows Power-type scaling law as LLM

arXiv: 2306.11527 + update

$$\text{Acceptance rate} = \exp\left(-\sqrt{\text{MSE}}\right)$$



(1 layer ~ 30 parameters)



Line is just for guiding eyes (no meaning)

fit ~(7.1/x)^(1.1)

[1] arXiv:2001.08361

Gauge covariant transformer (*CASK*)

Work in progress



A. Tomiya, H. Ohno, Y. Nagai

Lattice 2024
Jul 29 (Mon), 2024, 11:55 AM
Algorithms and artificial intelligence

Akio Tomiya

Overview/outline

Gauge covariant transformer for LQCD

Two conditions/restrictions in LQCD:

Gauge symmetry
U(x, x+μ)

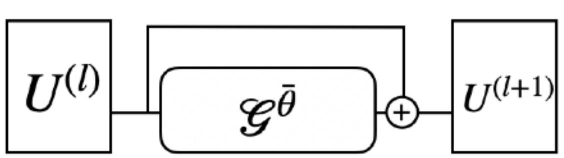
Non-locality from
pseudo-fermions
(1/D) ~ non-local

(I want to mimic
this by NN)

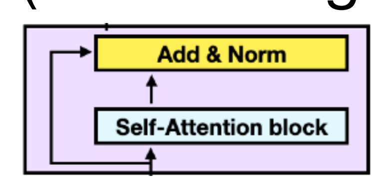
Solutions in neural net:

arXiv: 2103.11965 AT+

1. Gauge covariant net
(adaptive stout)



2. Transformer with global symmetry
(Heisenberg spin + electron)



2310.13222 AT+
2306.11527 AT+

3. **Gauge symmetric Transformer for LQCD**

This talk

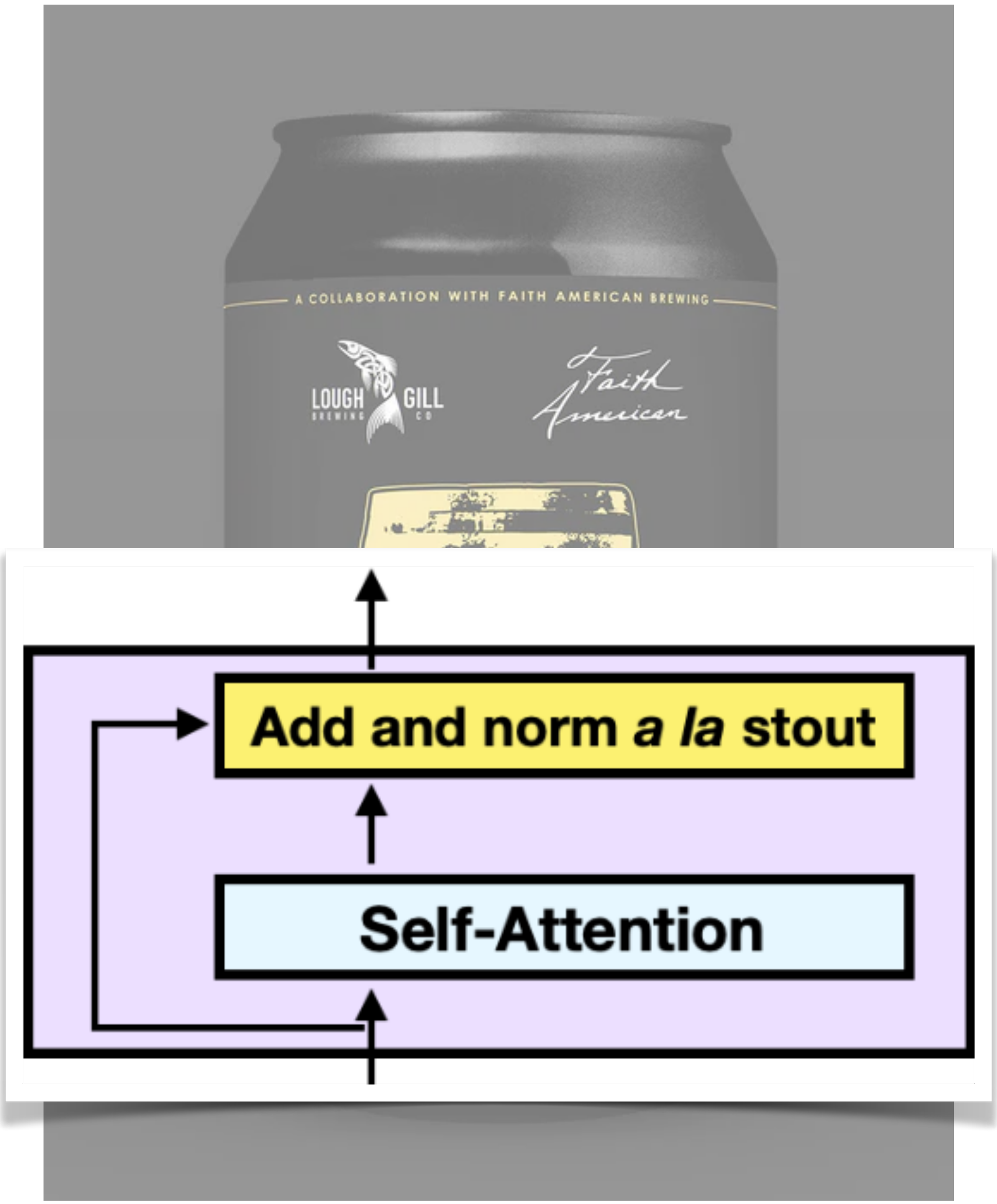
Gauge covariant transformer

CASK?

Cask stout
(Whisky Barrel-Aged Stout beer)
= stout beer in a cask

Gauge covariant transformer
= CASK

Akio Tomiya



Cask stout
(Whisky Barrel-Aged Stout beer)
= stout beer in a cask

Covariant attention block
CASK = Covariant Attention
with Stout Kernel

It is named in an obvious reason😜

Gauge covariant transformer

Akio Tomiya

Collection of ML/LQCD

Lattice	**ML(Framework)**	**ML/Lattice**
• Demon method (inverse MC) arXiv1508.04986 AT+ • Hopping parameter	Linear regression	Phys. Rev. D 107, 054501 AT+ Gauge inv. SLMC Trivializing with SD eq a la Luscher 2212.11387 AT+
Stout & Flow	CNN/Equivariant NN	Gauge covariant net 2021 AT+
(nothing. mean field?)	Transformer - GPT	• Global symmetric Transformer 2306.11527 AT+ • CASK (this talk)

Gauge covariant transformer

Idea: Attention must be covariant

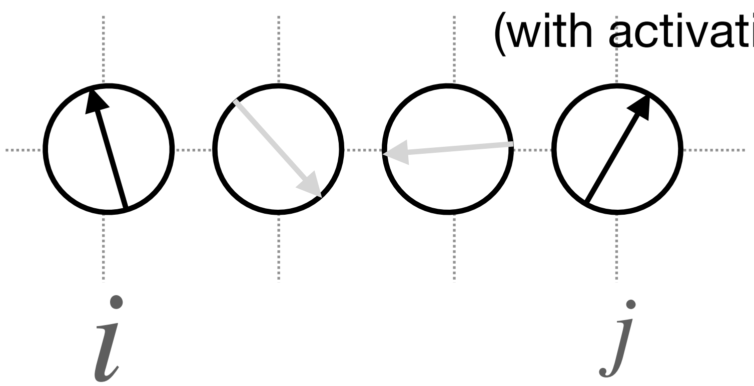
Akio Tomiya

Attention matrix in transformer ~ correlation function (with block-spin transformed spin)

-> we replace it with “correlation function for links” in a **covariant** way

$$a_{ij} \sim \vec{S}_i \cdot \vec{S}_j$$

(with activation)

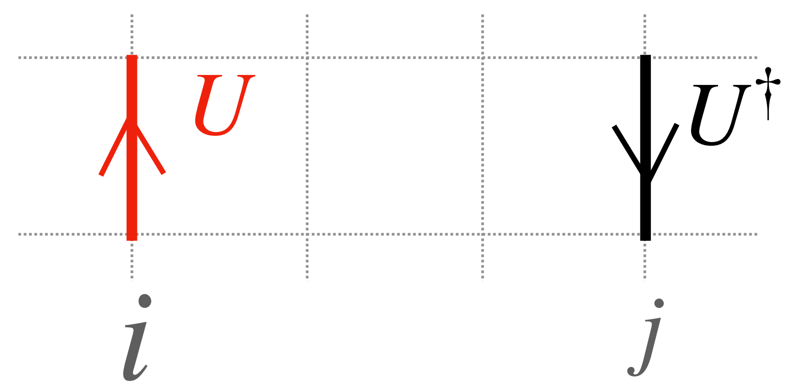


invariant
under global O(3)

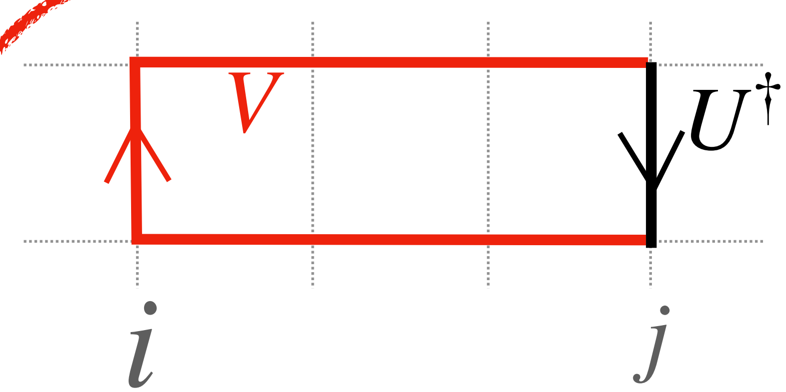
$$a_{ij} \sim (R\vec{S})_i^\top R\vec{S}_j = \vec{S}_i^\top \vec{S}_j$$

In total, output is covariant

$$a_{i\mu,j\mu} \sim \mathrm{Re}\,\mathrm{tr}\, U_\mu(i) U_\mu^\dagger(j)$$



not invariant
(cannot be used)



invariant under
local SU(N)

$$a_{i\mu,j\mu} \sim \mathrm{Re}\,\mathrm{tr}\, V_\mu(i) U_\mu^\dagger(j)$$

(with activation)

In total, output is covariant

Gauge covariant transformer

Akio Tomiya

Structure of gauge symmetric attention using stout

[1] 2021 AT+

Procedure in three steps:

0. U^{in} : Input configuration/Links

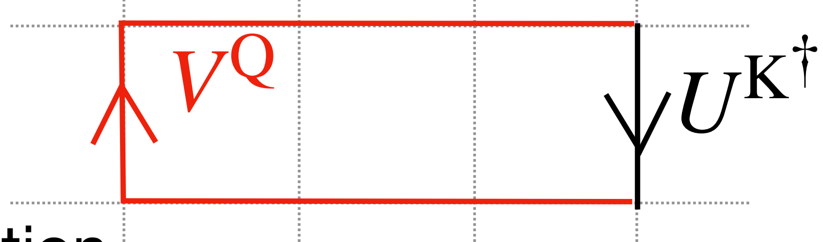
1. 3 types of (trainable) stout [1] -> $U^{\mathrm{Q}}, U^{\mathrm{K}}, U^{\mathrm{V}}$ (they have different weights)

$$U^{\alpha} = \exp[\rho^{\alpha} L[U^{\mathrm{in}}]]U^{\mathrm{in}} \qquad \alpha = \mathrm{Q}, \mathrm{K}, \mathrm{V}$$

weights

Loop operator projected on Lie algebra

2. Construct attention matrix (Rectangular Wilson loop) using $U^{\mathrm{Q}}, U^{\mathrm{K}}$ -> $a_{(*,*)}$



$\sim a_{(*,*)}$ (with activation)

cf. sparse attention

3. Construct "stout smeared" [1] link with weight $a_{(*,*)}$ and U^{V}, U (as matrix mult)

$$U^{\mathrm{out}} = \exp[a_{(*,*)} L[U^{\mathrm{V}}]]U^{\mathrm{in}} \qquad \text{Covariant}$$

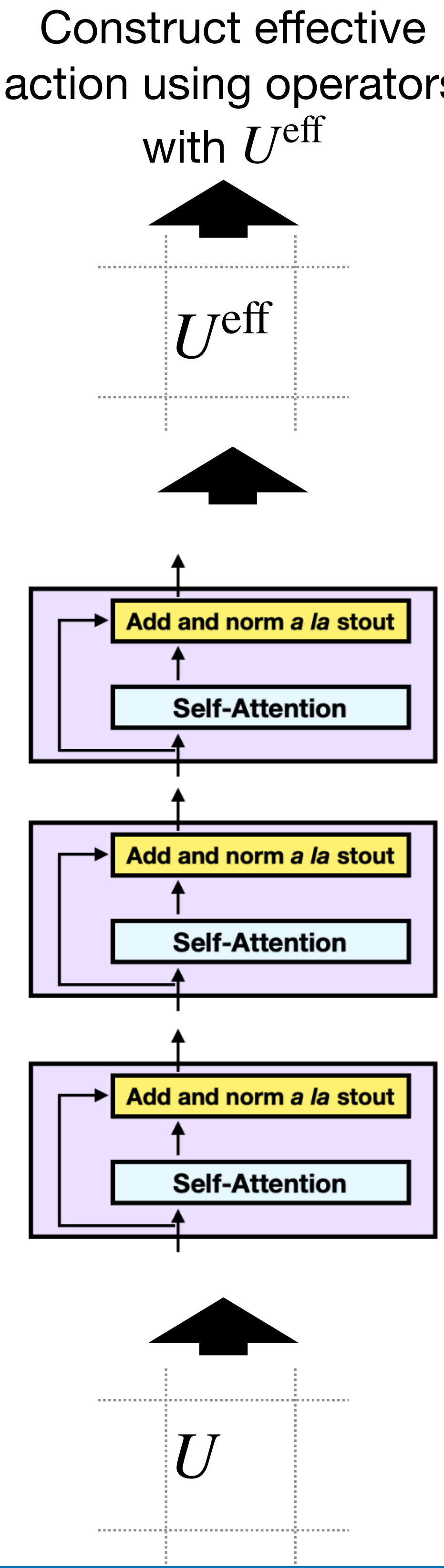
(This can be extend to have multi-head trivially)

Loop operator projected on Lie algebra

Gauge covariant transformer

Akio Tomiya

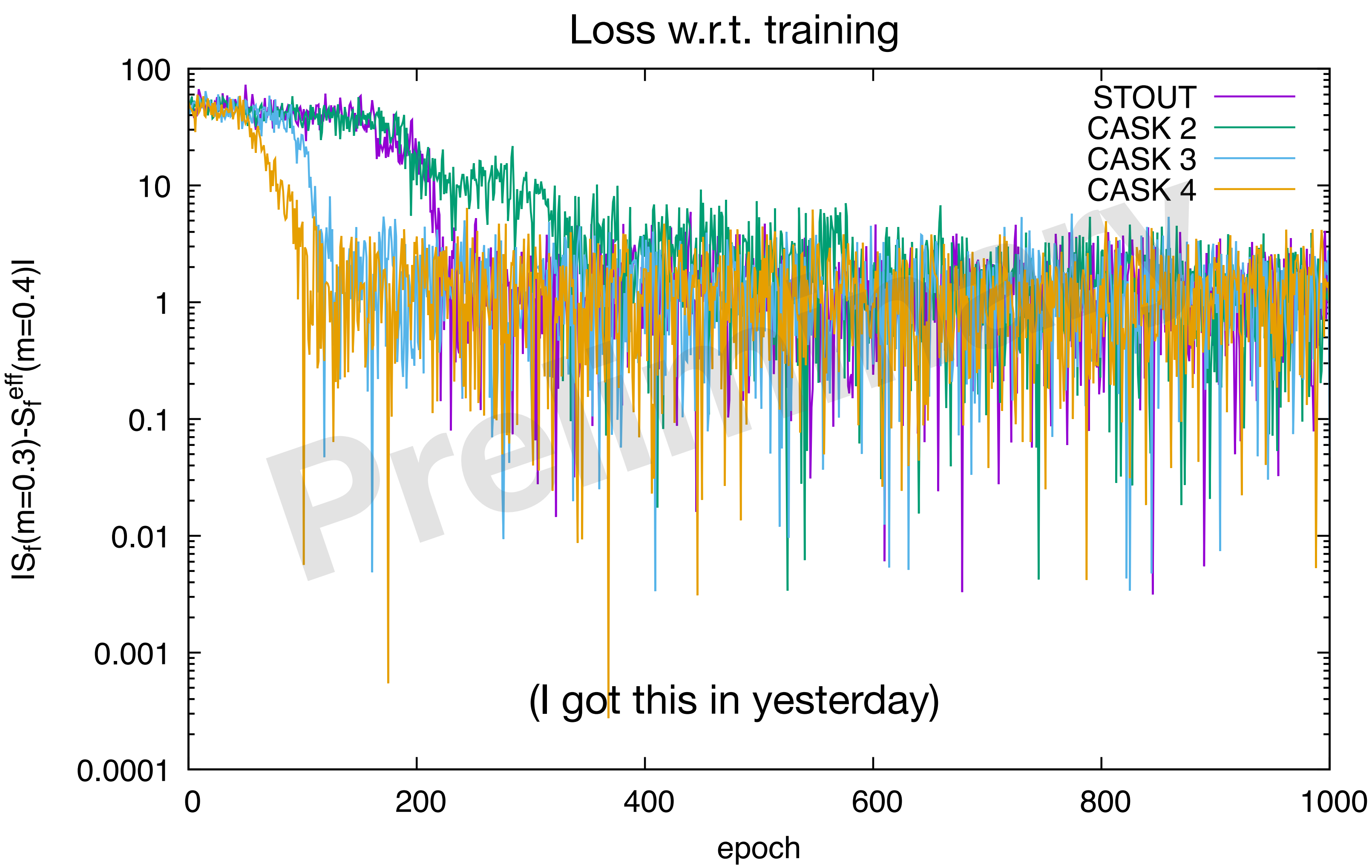
Simulation parameter



- Self-learning HMC (1909.02255, 2021 AT+), an exact algorithm
  - Exact Metropolis test and MD with effective action
- Target S : $m = 0.3$, dynamical staggered fermion, Nf=2, $L^4 = 4^4$, SU(2), $\beta = 2.7$
- Effective action in MD ($S^{\rm eff}$)
  - Same gauge action
  - $m_{\rm eff} = 0.4$ dynamical staggered fermion, Nf=2
  - CASK with plaquette covariant kernel
    - Attention = 7-links rect staple (=3 plaq)
  - U links are replaced by $U^{\rm eff}$ in $D_{\rm stag}$
- “Adaptively reweighted HMC”

Gauge covariant transformer

Loss = difference of action

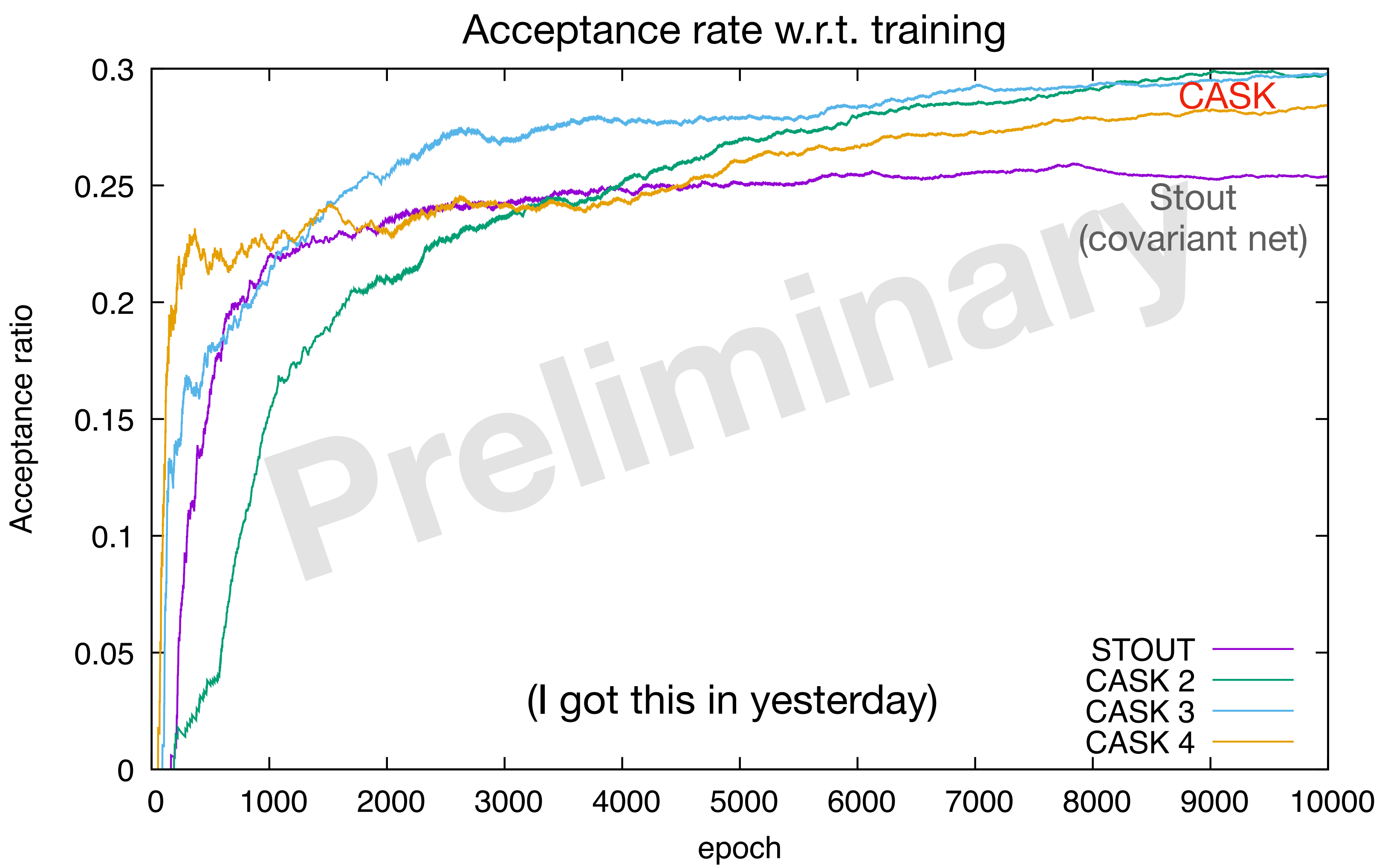


= MC steps

- Loss decreases along with the training steps
- it works as same as the stout (covariant net)
- No gain?

Gauge covariant transformer

Some gain



= MC steps

- In terms of acceptance, CASK has some gain
- It is still improving

Summary

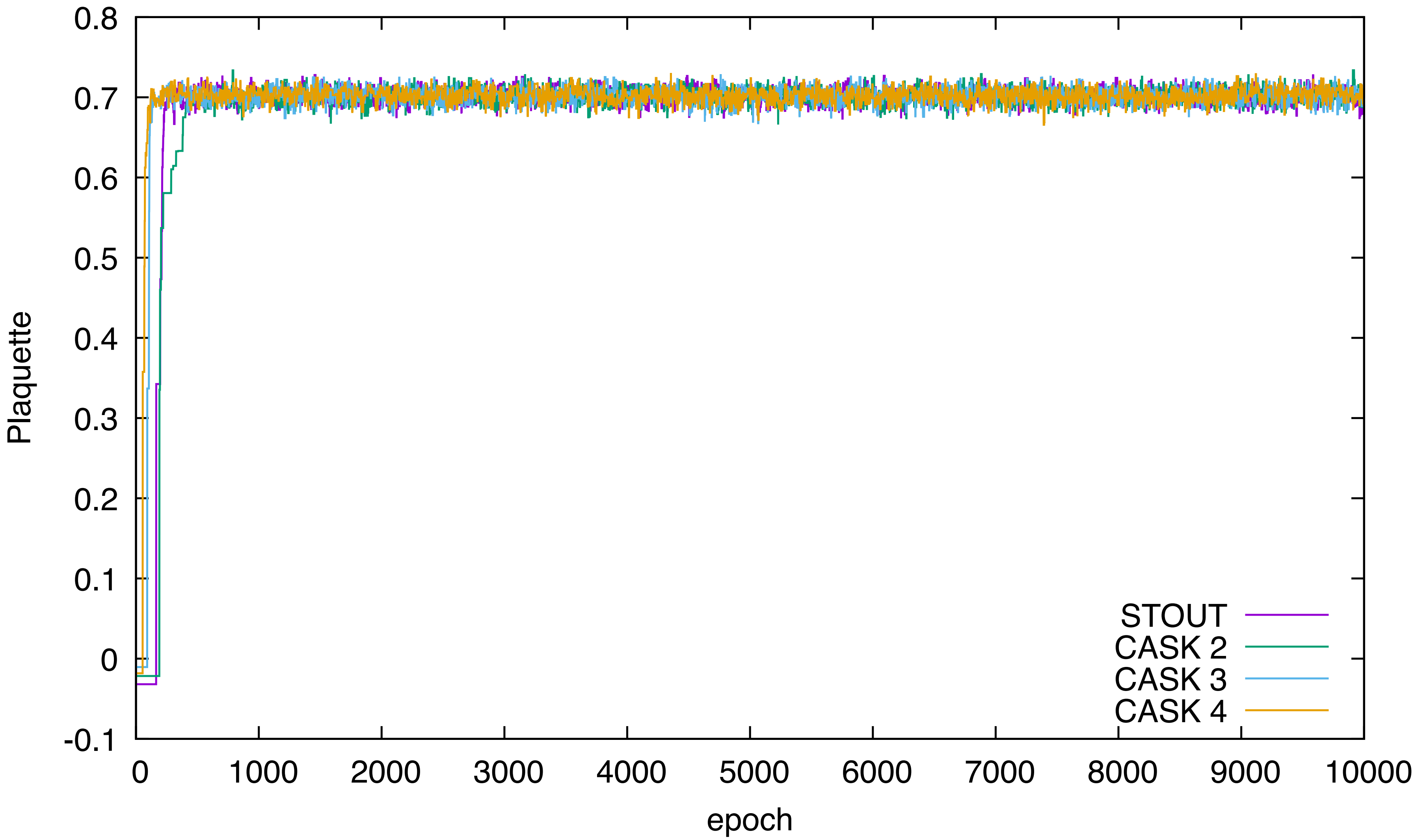
Transformer NN for Lattice QCD

Akio Tomiya

- Gauge covariant attention layer (CASK) has been developed
  - Test case for 4d SU(N) with dynamical fermions in tiny lattice
  - it is implemented with julia
  - Training is done using back-prop for gauge fields
  - It works as covariant nn and it has some gain 😀
- It is still working in progress
  - Scaling law for model size (and system size?)
  - Removing pseudo-fermions? (as same as the spin 2306.11527 AT+)
  - Optimization of architecture
    - Sparse-attention/star-attention/etc
  - Bigger model? Applications?

MLPhys Foundation of "Machine Learning Physics" Grant-in-Aid for Transformative Research Areas (A)
Program for Promoting Researches on the Supercomputer Fugaku Large-scale lattice QCD simulation and development of AI technology
KAKENHI: 20K14479, 22H05112, 22H05111, 22K03539

Thanks!



- CASK gives consistent results with other gauge cov net (as expected)

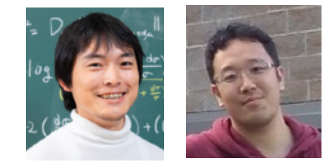
Applications

Configuration generation with machine learning is developing

Configuration generation for 2d scalar

Restricted Boltzmann machine + HMC: 2d scalar — A. Tanaka, AT 2017

The first challenge, machine learning + configuration generation. Wrong at critical pt. Not exact.



GAN (Generative adversarial network): 2d scalar — J. Pawlowski+ 2018, G. Endrodi+ 2018

Results look OK. No proof of exactness

Exact algorithm, gauge symmetry

Flow based model: 2d scalar, pure U(1), pure SU(N) — MIT, Google Brain 2019, 2020, 2021

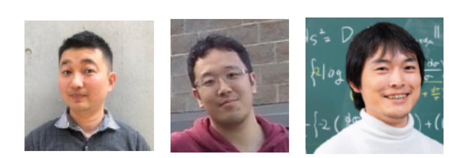
Mimicking a trvializing map using a neural net which is reversible and has tractable Jacobian.
Exact algorithm, no dynamical fermions. SU(N) is treated with diagonalization.

L2HMC for 2d U(1) (Sam Foreman+ 2021)

Dynamical fermions, 4 Dimension

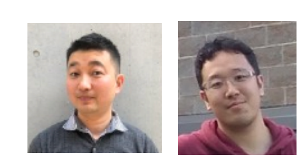
Self-learning Monte Carlo (SLMC) for lattice QCD — arxiv 2010.11900 Y. Nagai, AT, A. Tanaka

Non-abelian gauge theory with dynamical fermion in 4d
Using gauge invariant action with linear regression
Exact. Costly (Diagonalize Dirac operator)



Self-learning Hybrid Monte Carlo for lattice QCD (SLHMC, This talk) — arxiv 2103.11965 Y. Nagai, AT

Non-abelian gauge theory with dynamical fermion in 4d
Using covariant neural network to parametrize the gauge invariant action
Exact



Application for the staggered in 4d

Problems to solve

arXiv: 2103.11965

Our neural network enables us to **parametrize** gauge symmetric action **covariant way.**

e.g.

$$S^{\mathrm{NN}}[U] = S_{\mathrm{plaq}}\left[U_\mu^{\mathrm{NN}}(n)[U]\right]$$

$$S^{\mathrm{NN}}[U] = S_{\mathrm{stag}}\left[U_\mu^{\mathrm{NN}}(n)[U]\right]$$

Test of our neural network?

Can we mimic a different Dirac operator using neural net?

Artificial example for HMC:

Target action $S[U] = S_{\mathrm{g}}\left[U\right] + S_{\mathrm{f}}\left[\phi, U; m = 0.3\right],$

Action in MD $S_\theta[U] = S_{\mathrm{g}}\left[U\right] + S_{\mathrm{f}}\left[\phi, \underline{U_\theta^{\mathrm{NN}}[U]}; m_{\mathrm{h}} = 0.4\right],$

Q. Simulations with approximated action can be exact?
-> Yes! with SLHMC (Self-learning HMC)

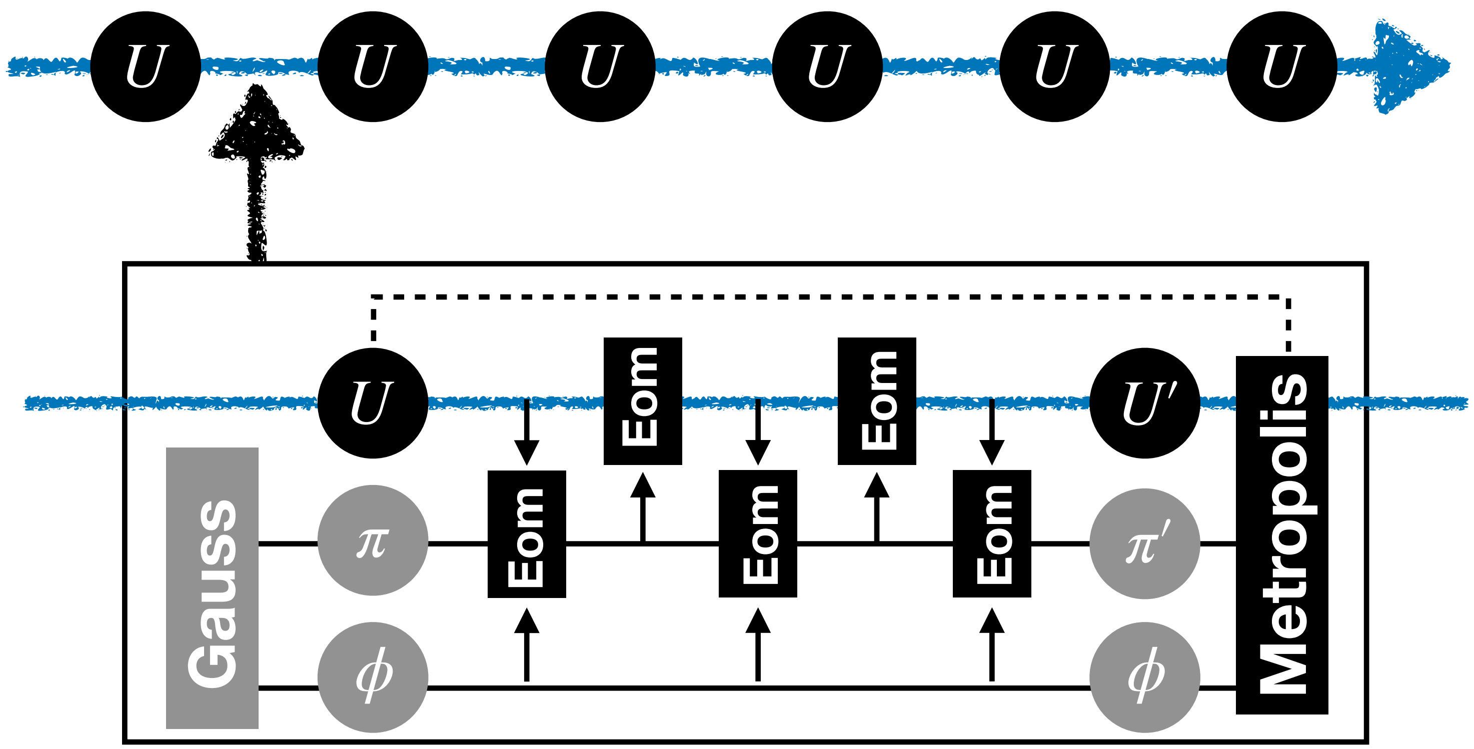
SLHMC = Exact algorithm with ML

SLHMC for gauge system with dynamical fermions

Akio Tomiya

arXiv: 2103.11965 and reference therein

HMC



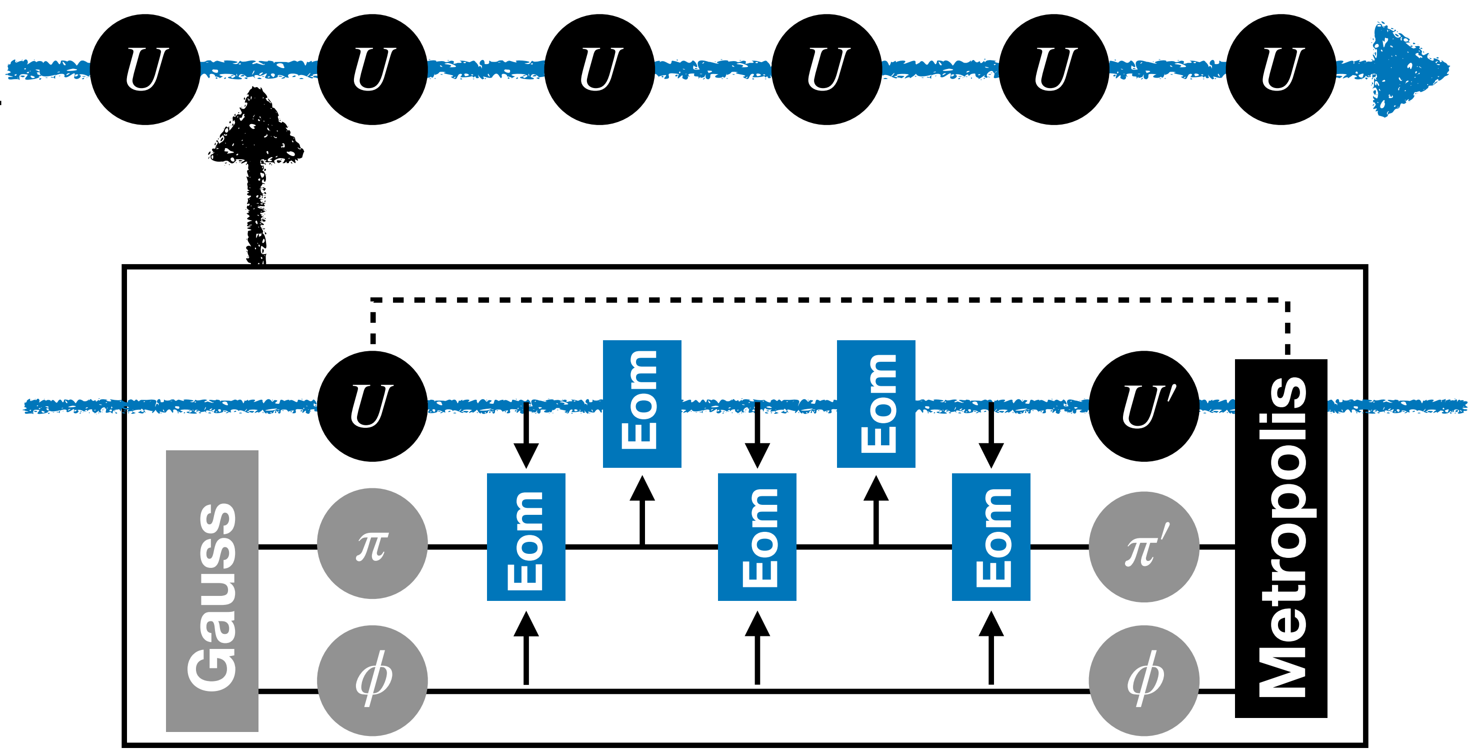
Eom **Metropolis**

Both use

$$H_{\mathrm{HMC}} = \frac{1}{2}\sum \pi^2 + S_{\mathrm{g}} + S_{\mathrm{f}}$$

Non-conservation of H cancels since the molecular dynamics is reversible

SLHMC



Metropolis

$$H = \frac{1}{2}\sum \pi^2 + S_{\mathrm{g}} + S_{\mathrm{f}}[U]$$

Eom

$$H = \frac{1}{2}\sum \pi^2 + S_{\mathrm{g}} + S_{\mathrm{f}}[U^{\mathrm{NN}}[U]]$$

Neural net approximated fermion action but exact

Application for the staggered in 4d

Lattice setup and question

arXiv: 2103.11965

Target Two color QCD (plaquette + staggered)

Algorithms SLHMC, HMC (comparison)

Parameter Four dimension, L=4, m = 0.3, beta = 2.7, Nf=4 (non-rooting)

Target action $S[U] = S_{\mathrm{g}}[U] + S_{\mathrm{f}}[\phi, U; m = 0.3],$ **For Metropolis Test**

Action in MD (for SLHMC) $S_{\theta}[U] = S_{\mathrm{g}}[U] + S_{\mathrm{f}}[\phi, U_{\theta}^{\mathrm{NN}}[U]; m_{\mathrm{h}} = 0.4],$

Observables Plaquette, Polyakov loop, Chiral condensate $\langle \bar{\psi}\psi \rangle$

Code Full scratch,
fully written in Julia lang.

LatticeQCD.jl

AT+ (in prep)

(But we added some functions on the public version)

Lattice QCD code

We made a public code in Julia Language

Akio Tomiya

AT & Y. Nagai in prep

What is julia?

1. Open source scientific language (Just in time compiler)
2. Fast as C/Fortran (sometime, faster)
3. Productive as Python
4. Machine learning friendly (Julia ML packages + Python libraries w/ PyCall)
5. Supercomputers support Julia

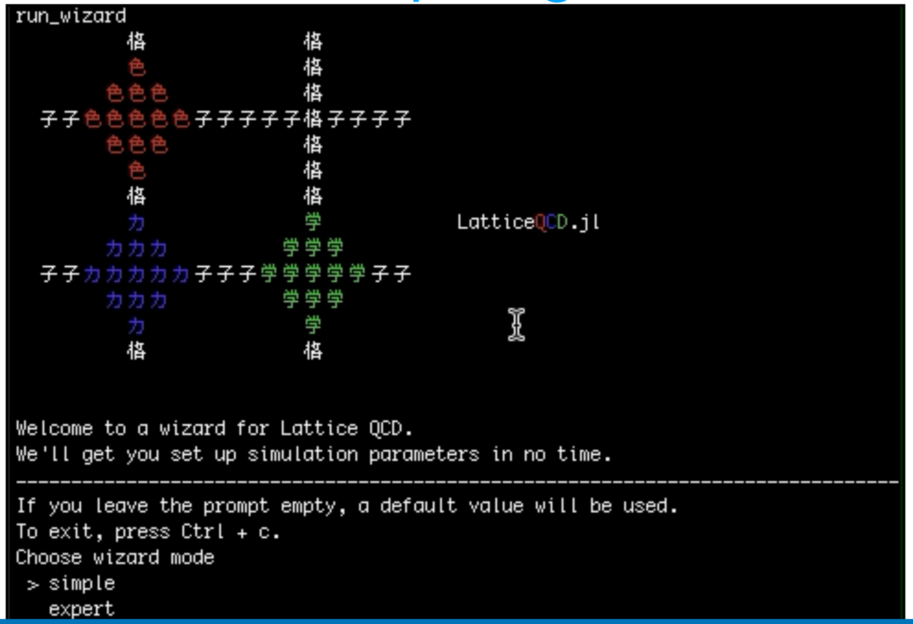
LatticeQCD.jl (Official package) : Laptop/desktop/PC-cluster/Jupyter (Google colab)

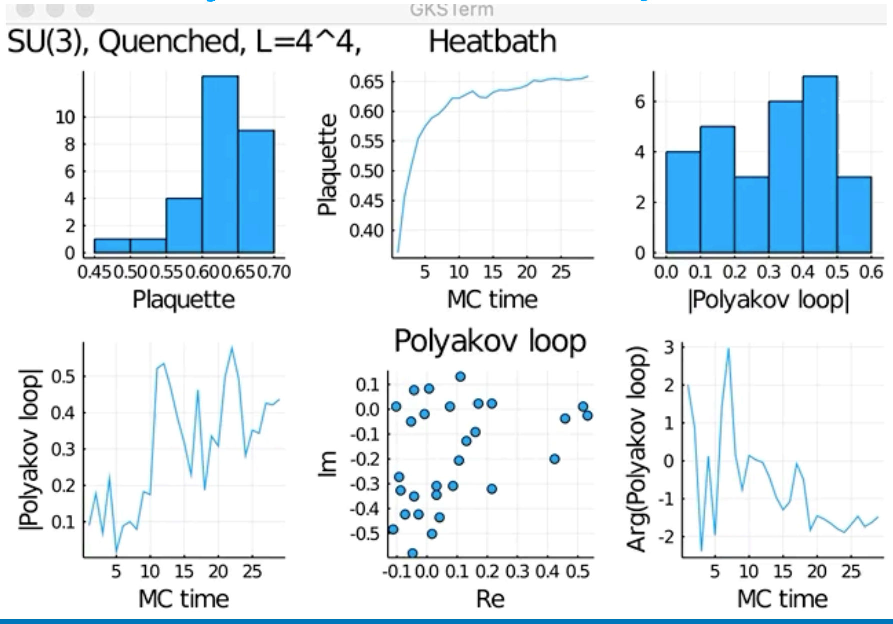
SU(Nc)-heatbath/SLHMC/SU(Nc) Stout/(R)HMC/staggered/Wilson-Clover
Domain-wall (experimental) + Measurements

3 steps in 5 min

1. Download Julia binary
2. Add the package through Julia package manager
3. Execute!

https://github.com/akio-tomiya/LatticeQCD.jl





Details (skip)

Akio Tomiya

Network: trainable stout (plaq+poly)

arXiv: 2103.11965

Structure of NN

(Polyakov loop+plaq in the stout-type)

$$\Omega_\mu^{(l)}(n) = \rho_{\rm plaq}^{(l)} O_\mu^{\rm plaq}(n) + \begin{cases} \rho_{\rm poly,4}^{(l)} O_4^{\rm poly}(n) & (\mu = 4), \\ \rho_{\rm poly,s}^{(l)} O_i^{\rm poly}(n), & (\mu = i = 1,2,3) \end{cases}$$

All ρ is weight

O meas an loop operator

$$Q_\mu^{(l)}(n) = 2[\Omega_\mu^{(l)}(n)]_{\rm TA}$$

TA: Traceless, anti-hermitian operation

$$U_\mu^{(l+1)}(n) = \exp(Q_\mu^{(l)}(n)) U_\mu^{(l)}(n)$$

$$U_\mu^{\rm NN}(n)[U] = U_\mu^{(2)}(n)\Big[U_\mu^{(1)}(n)\Big[U_\mu(n)\Big]\Big]$$

2- layered stout
with 6 trainable parameters

Neural network Parametrized action:

$$S_\theta[U] = S_{\rm g}[U] + S_{\rm f}[\phi, U_\theta^{\rm NN}[U]; m_{\rm h} = 0.4],$$

Action for MD is built by gauge covariant NN

Loss function:

$$L_\theta[U] = \frac{1}{2}\left| S_\theta[U,\phi] - S[U,\phi] \right|^2,$$

Invariant under, rot, transl, gauge trf.

Training strategy:

1. Train the network in prior HMC (online training+stochastic gr descent)
2. Perform SLHMC with fixed parameter

Details (skip)

Results: Loss decreases along with the training

arXiv: 2103.11965

Loss function:

$$L_\theta[U] = \frac{1}{2}\left|S_\theta[U,\phi] - S[U,\phi]\right|^2,$$

Intuitively, e^(-L) is understood as Boltzmann weight or reweighting factor.

Prior HMC run (training)

$$\frac{\partial S}{\partial \rho_i^{(l)}} = 2\,\mathrm{Re}\sum_{\mu',m}\mathrm{tr}\left[U_{\mu'}^{(l)\dagger}(m)\Lambda_{\mu',m}\frac{\partial C}{\partial \rho_i^{(l)}}\right]$$

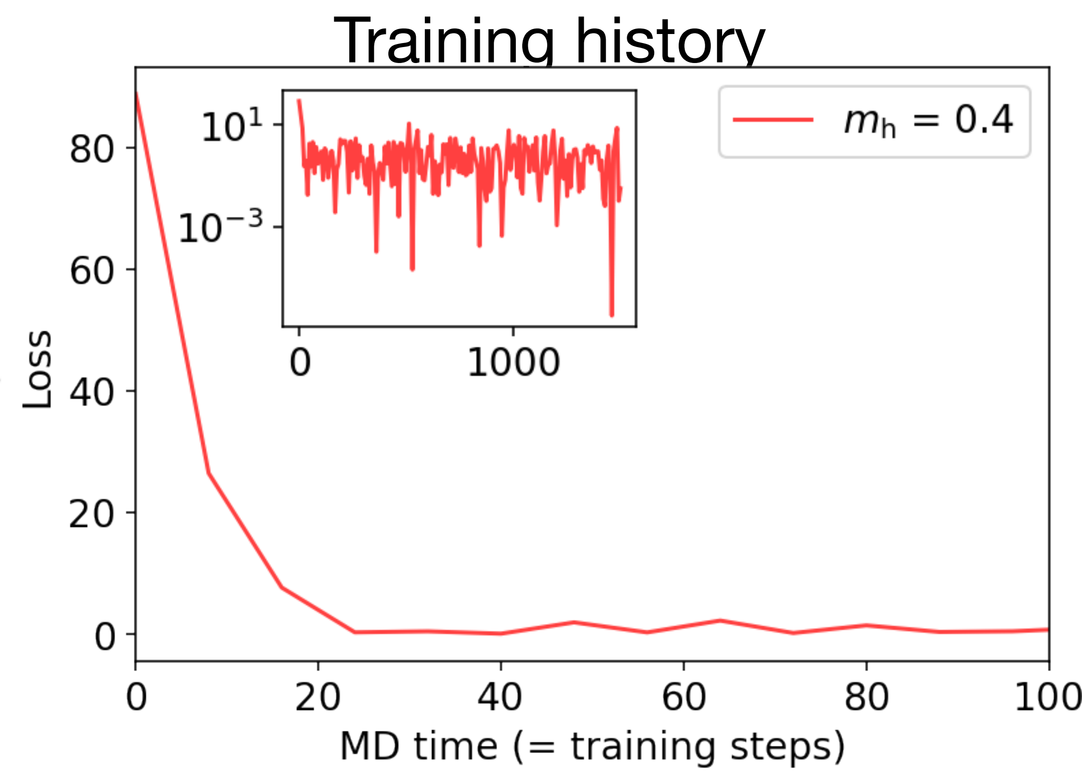
$$\theta \leftarrow \theta - \eta\frac{\partial L_\theta(\mathcal{D})}{\partial \theta},$$

$$\frac{\partial L_\theta(\mathcal{D})}{\partial w_i^{(L-1)}} = \frac{\partial L_\theta(\mathcal{D})}{\partial S_\theta}\frac{\partial S_\theta}{\partial w_i^{(L-1)}}$$

Ω: sum of un-traced loops

C: one U removed Ω

Λ: A polynomial of U. (Same object in stout)



Without training, e^(-L)<< 1,
this means that candidate with approximated action never accept.
After training, e^(-L) ~1, and we get practical acceptance rate!

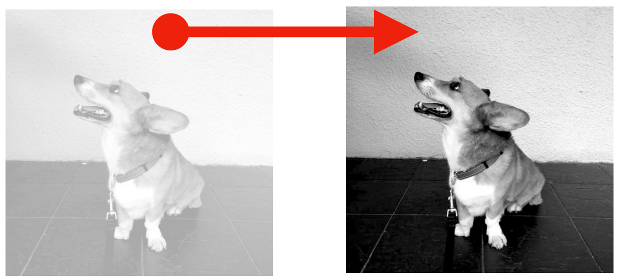
We perform SLHMC with these values!

Equivariance and convolution

Akio Tomiya

Knowledge ∋ Convolution layer = trainable filter, Equivariant

Filter on image



shift to right

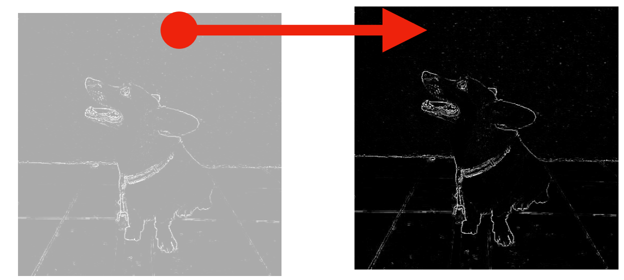
$*$

Laplacian filter

0	1	0
1	-2	1
0	1	0

(Discretization of ∂^2)

$=$

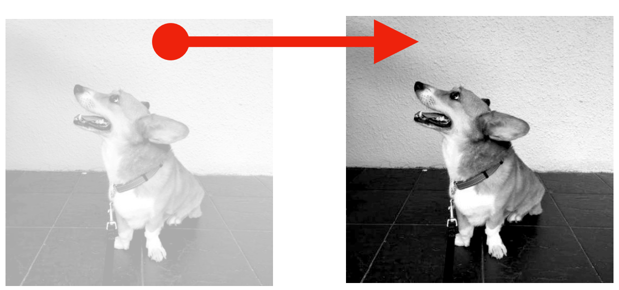


shift to right

Edge detection

Translational operation is *commutable* with filtering (equivariant)

Convolution layer



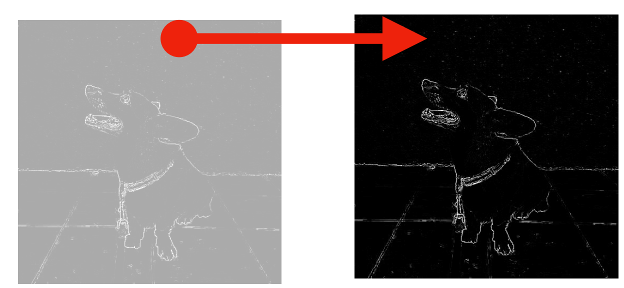
shift to right

$*$

Trainable filter

w_{11}	w_{12}	w_{13}
w_{21}	w_{22}	w_{23}
w_{31}	w_{32}	w_{33}

$=$



shift to right

Fukushima, Kunihiko (1980)
Zhang, Wei (1988) + a lot!

Translational operation is *commutable* with convolutional neurons (equivariant)

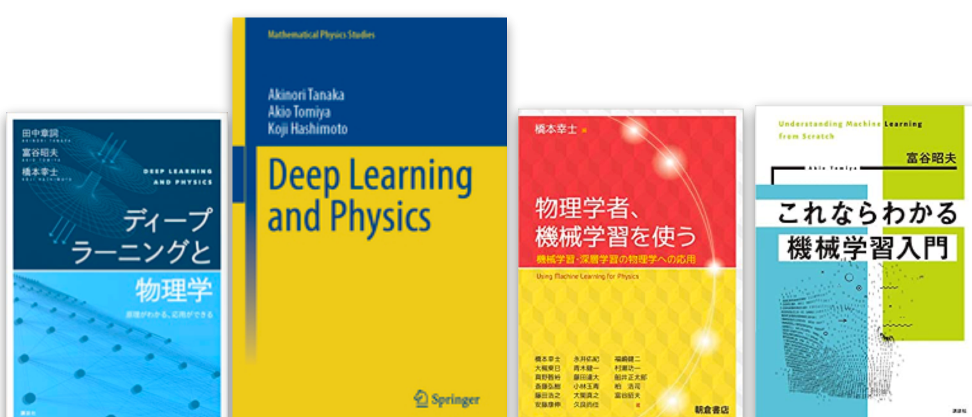
This can be any filter which helps feature extraction (minimizing loss)

Equivariance reduces data demands. Ensuring symmetry (plausible Inference)

Many of convolution are needed to capture global structures

Akio Tomiya

Machine learning for theoretical physics



Organizing “Deep Learning and physics”

https://cometscome.github.io/DLAP2020/

What am I?

I am a particle physicist, working on lattice QCD.

I want to apply machine learning on lattice QCD.

My papers https://scholar.google.co.jp/citations?user=LKVqy_wAAAAJ

Detection of phase transition via convolutional neural networks
A Tanaka, A Tomiya
Journal of the Physical Society of Japan 86 (6), 063001

Detecting phase transition

Digital quantum simulation of the schwinger model with topological term via adiabatic state preparation
B Chakraborty, M Honda, T Izubuchi, Y Kikuchi, A Tomiya
arXiv preprint arXiv:2001.00485

Quantum computing for quantum field theory

Biography

2006-2010 : University of Hyogo (Superconductor)
2015 : PhD in Osaka university (Particle phys)
2015 - 2018 : Postdoc in Wuhan (China)
2018 - 2021 : SPDR in Riken/BNL (US)
2021 - 2024 : Assistant prof. in IPUT Osaka (ML/AI)
2021 - 2024 : ML(ML/AI)

Kakenhi and others

Leader of proj A01 Transformative Research Areas, Fugaku

MLPhys Foundation of "Machine Learning Physics"
Grant-in-Aid for Transformative Research Areas (A)

Program for Promoting Researches on the Supercomputer Fugaku
Large-scale lattice QCD simulation and development of AI technology

+quantum computer

Others:

Supervision of Shin-Kamen Rider

The 29th Outstanding Paper Award of the Physical Society of Japan

14th Particle Physics Medal: Young Scientist Award

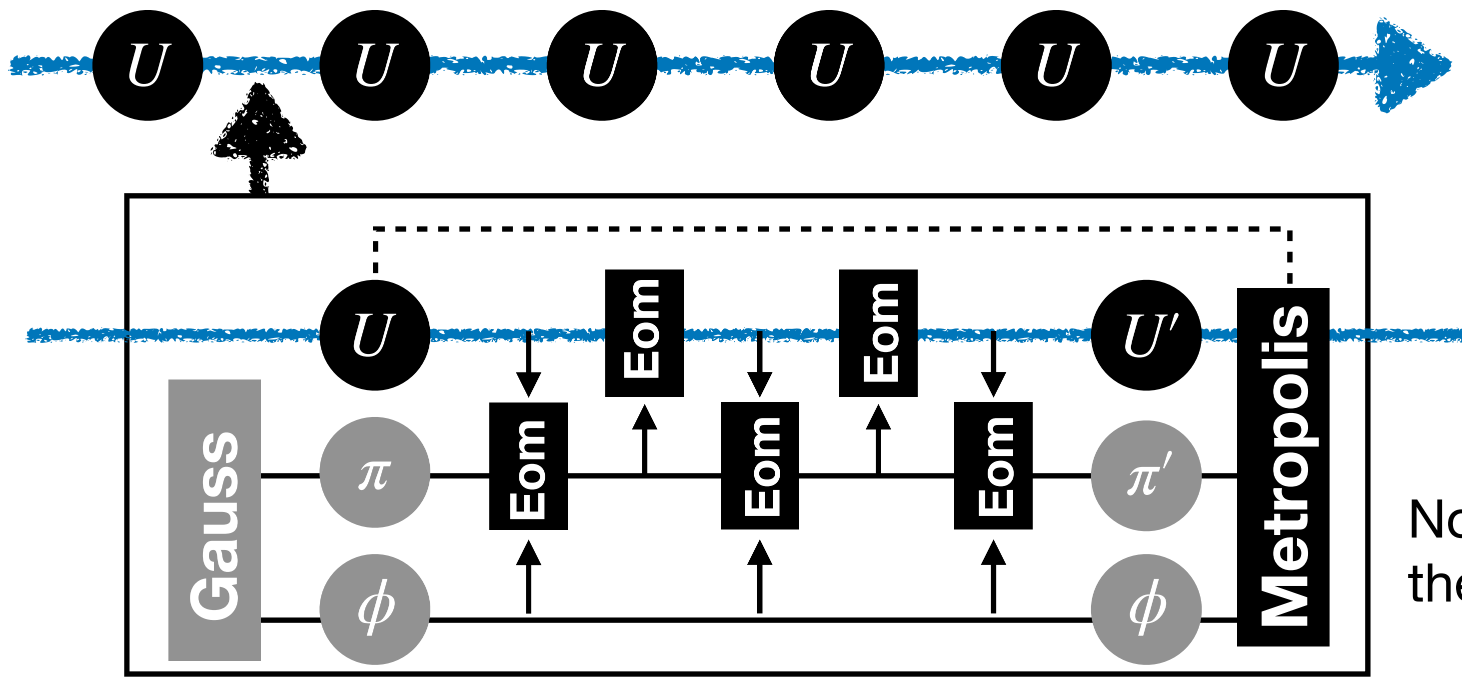
SLHMC = Exact algorithm with ML

Akio Tomiya

SLHMC for gauge system with dynamical fermions

Gauge covariant neural network can mimics gauge invariant functions
-> It can be used in simulation? -> Self learning HMC!

HMC

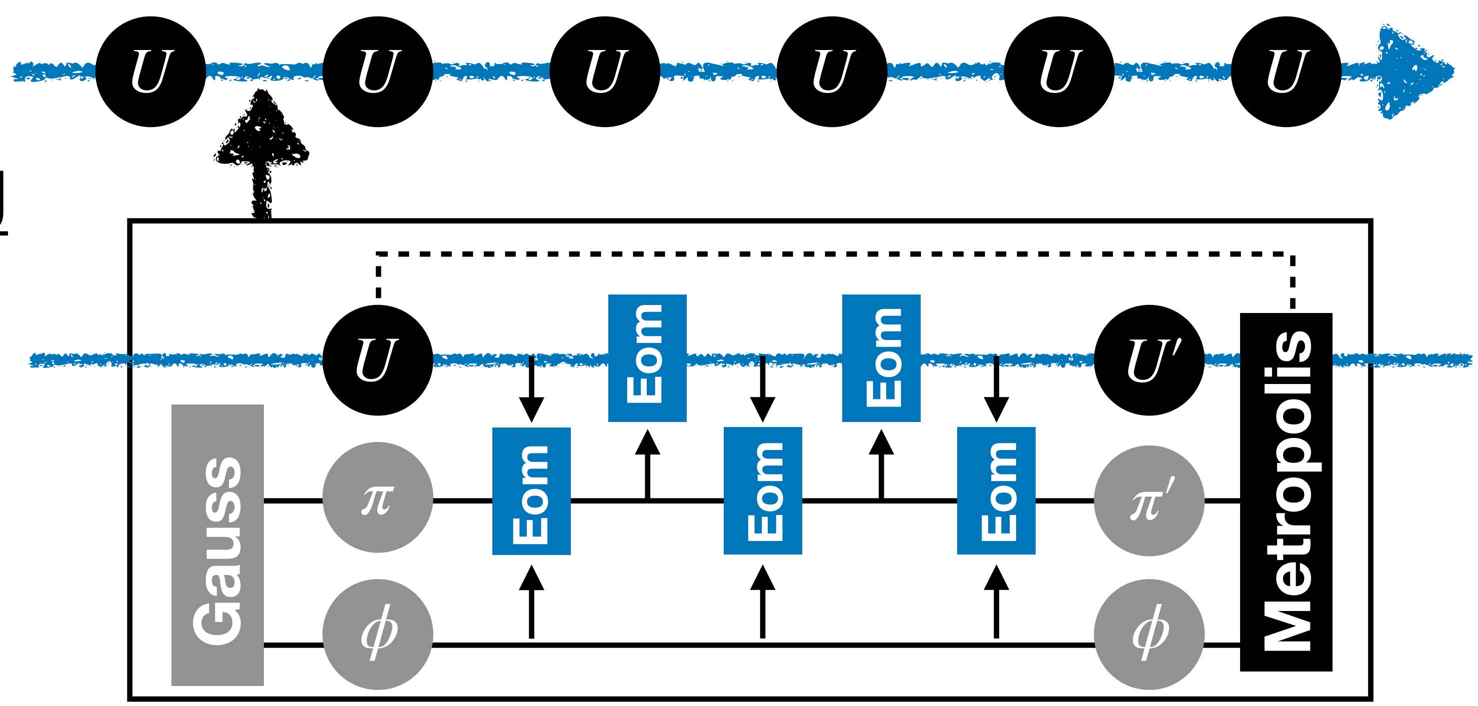


Eom **Metropolis**
Both use

$$H_{\text{HMC}} = \frac{1}{2}\sum \pi^2 + S_{\text{g}} + S_{\text{f}}$$

Non-conservation of H cancels since the molecular dynamics is reversible

Self Learning HMC



Metropolis

$$H = \frac{1}{2}\sum \pi^2 + S_{\text{g}} + S_{\text{f}}[U]$$

Eom

$$H = \frac{1}{2}\sum \pi^2 + S_{\text{g}} + S_{\text{f}}[U^{\text{NN}}[U]]$$

Neural net approximated fermion action but exact

arXiv: 2103.11965
and reference therein

SLHMC works as an adaptive reweighting!

Application for the staggered in 4d

Problems to solve

Akio Tomiya

arXiv: 2103.11965

Mimic different actions:

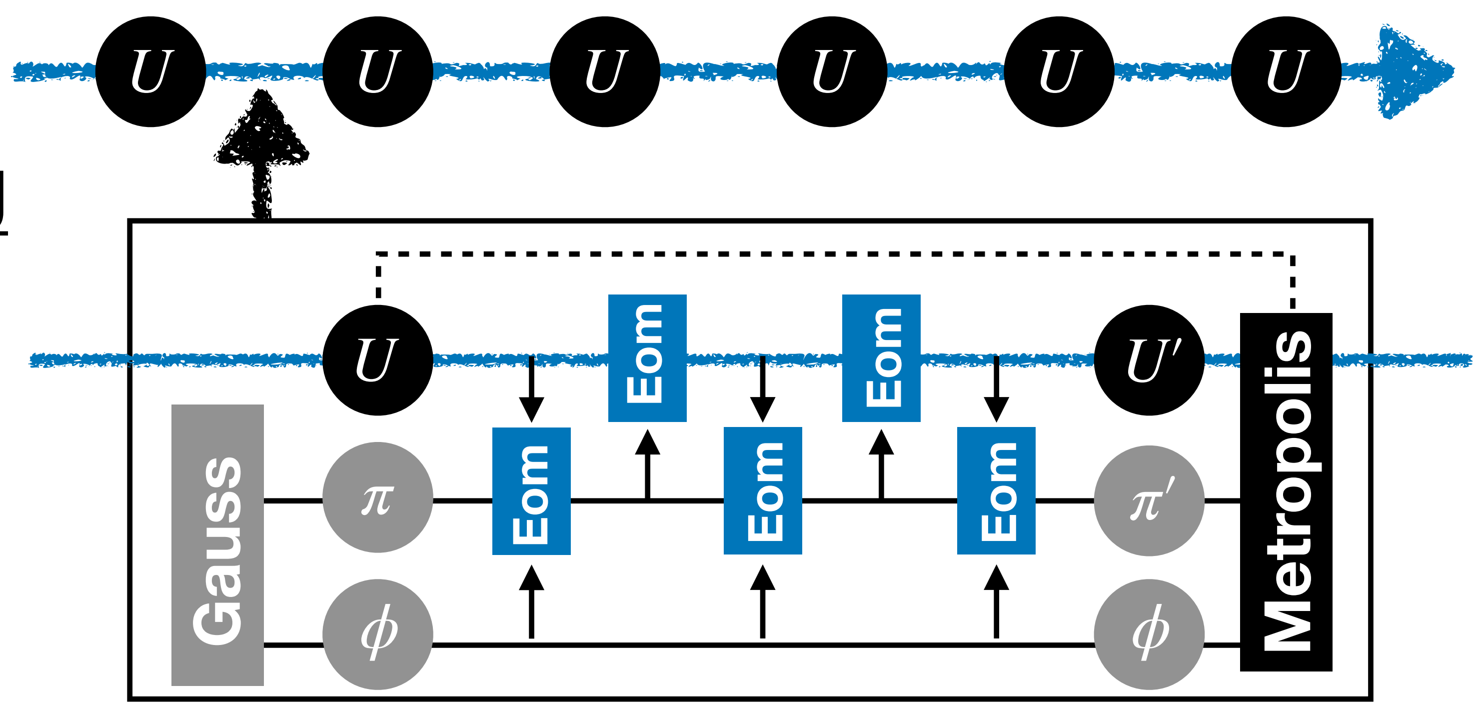
(Final target: Domain-wall vs overlap)

A toy problem: Staggered (heavy) vs Staggered (light)

Target action (Metropolis) $S[U] = S_{\mathrm{g}}[U] + S_{\mathrm{f}}[\phi, U; m = 0.3],$

Action in MD $S_\theta[U] = S_{\mathrm{g}}[U] + S_{\mathrm{f}}[\phi, U_\theta^{\mathrm{NN}}[U]; m_{\mathrm{h}} = 0.4],$

Self Learning HMC



Metropolis

$$H = \frac{1}{2}\sum \pi^2 + S_{\mathrm{g}} + S_{\mathrm{f}}[U]$$

Eom

$$H = \frac{1}{2}\sum \pi^2 + S_{\mathrm{g}} + S_{\mathrm{f}}[U^{\mathrm{NN}}[U]]$$

Neural net approximated fermion action but <u>exact</u>

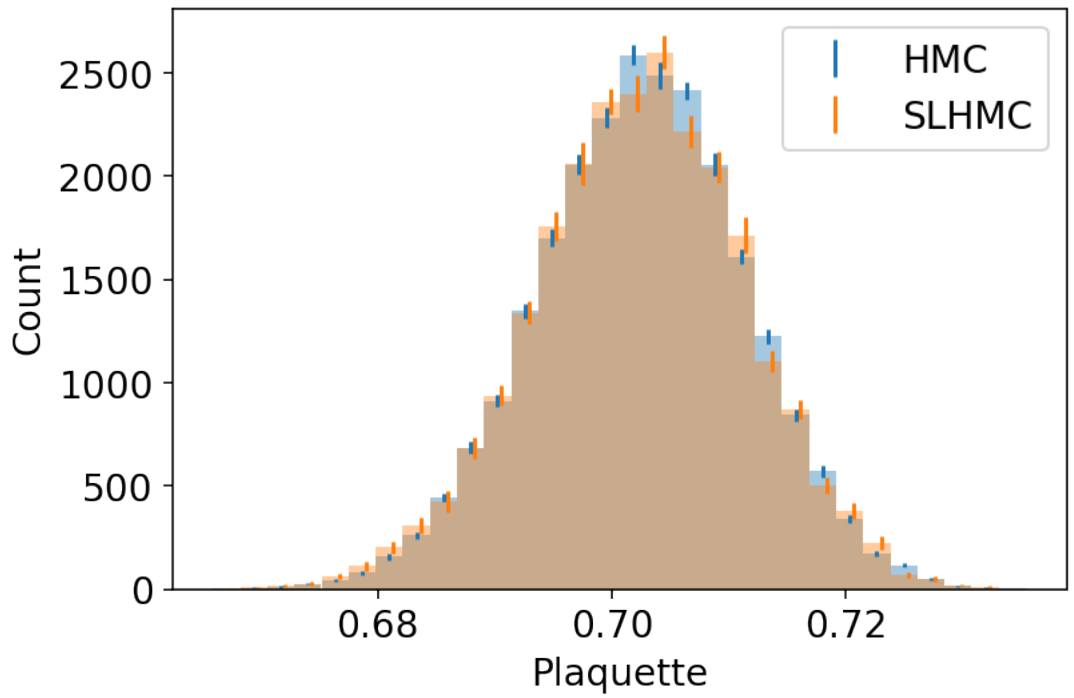
SLHMC works as an adaptive reweighting!

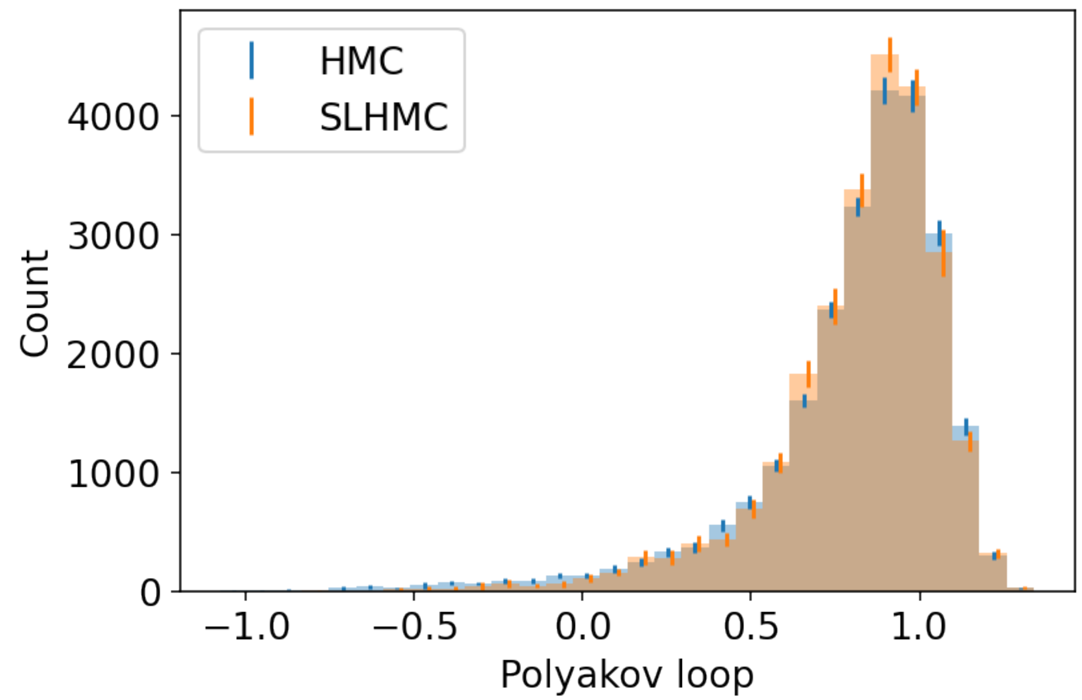
Akio Tomiya

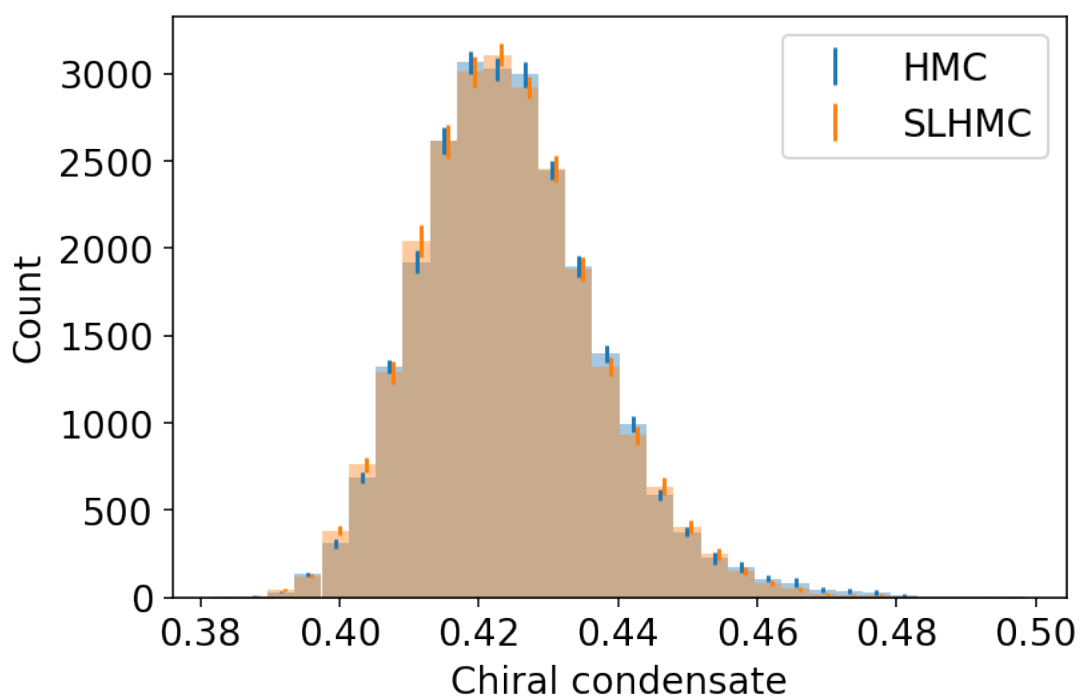
Application for the staggered in 4d

Results are consistent with each other

arXiv: 2103.11965







Expectation value

Algorithm	Observable	Value
HMC	Plaquette	0.7025(1)
SLHMC	Plaquette	0.7023(2)
HMC	|Polyakov loop|	0.82(1)
SLHMC	|Polyakov loop|	0.83(1)
HMC	Chiral condensate	0.4245(5)
SLHMC	Chiral condensate	0.4241(5)

Implemented by LatticeQCD.jl julia