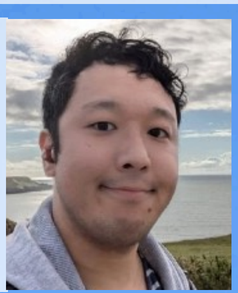
https://conference.ippp.dur.ac.uk/event/1265/contributions/7581/

LATTICE 2024
LIVERPOOL

Developments of CASK: Gauge Symmetric Transformer

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




H. Ohno


Y. Nagai

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

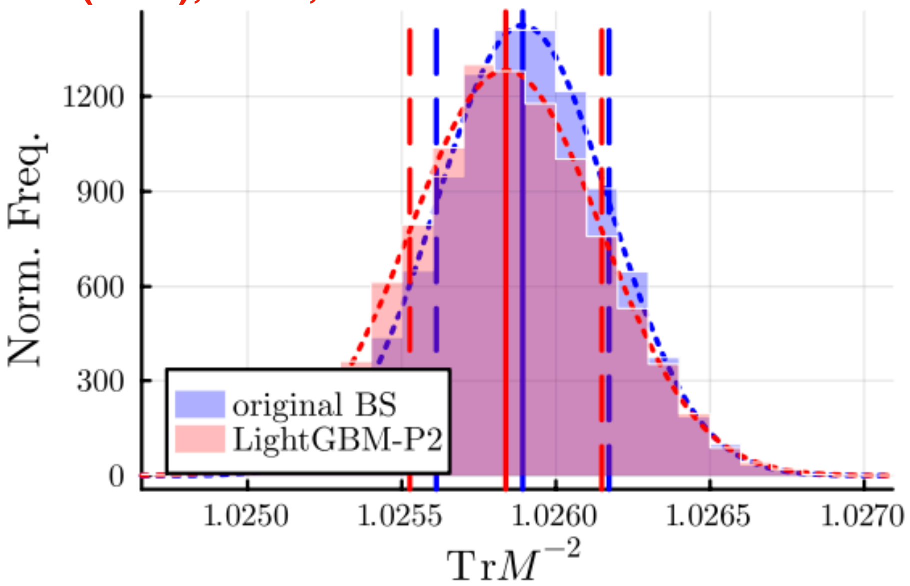
富岳

akio_at_yukawa.kyoto-u.ac.jp

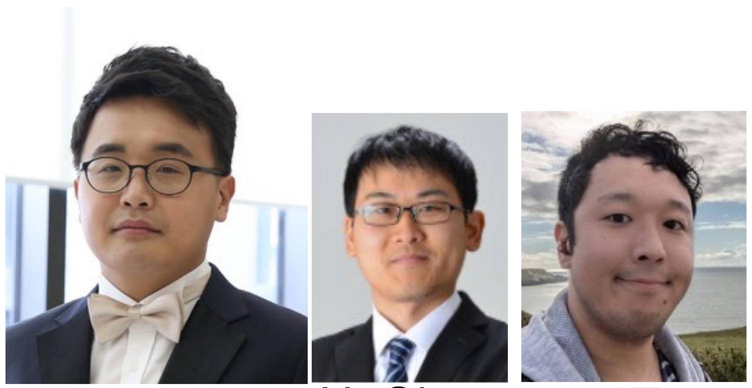
Two related talks in Lattice2024

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

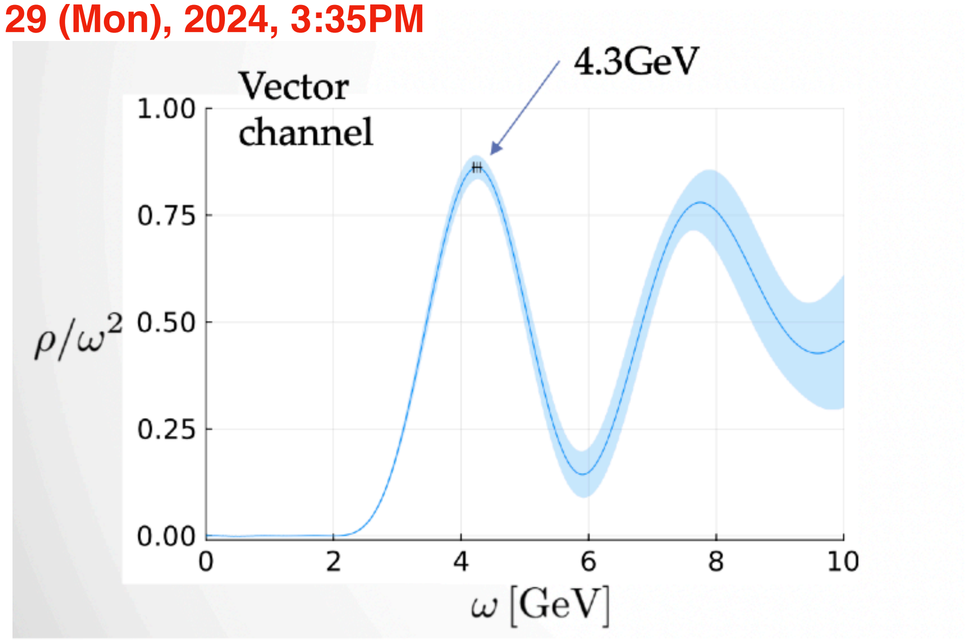


B. J. Choi
U. of Tsukuba

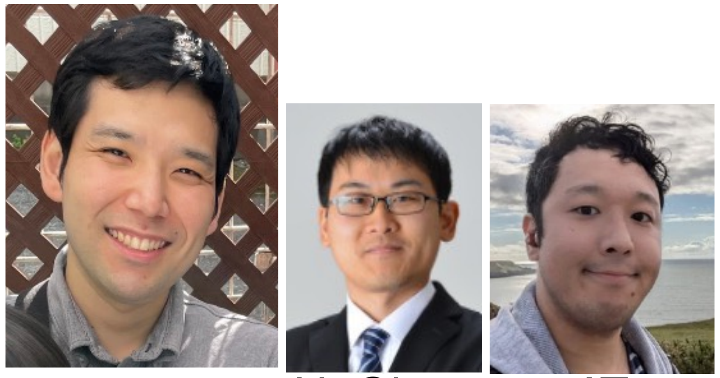
H. Ohno

AT

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



Reconstruction of spectral function using machine learning (sparse modeling)



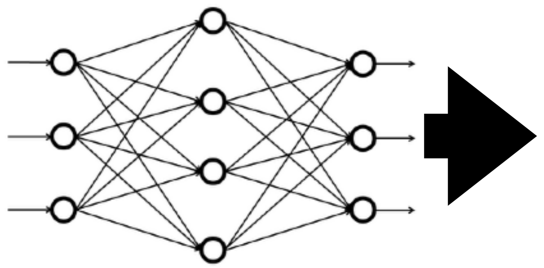
J. Takahashi
Meteorological College

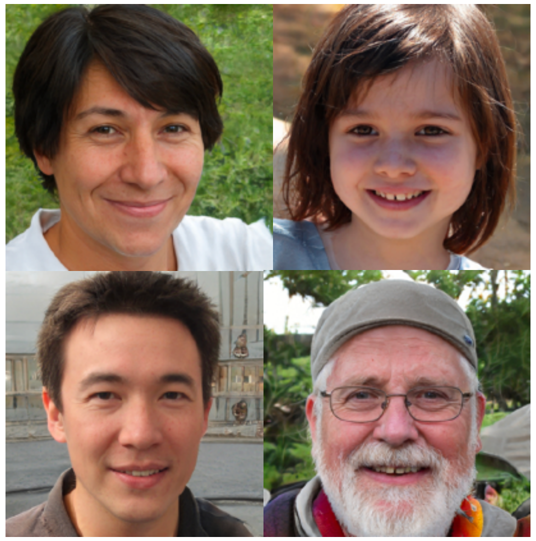
H. Ohno

AT

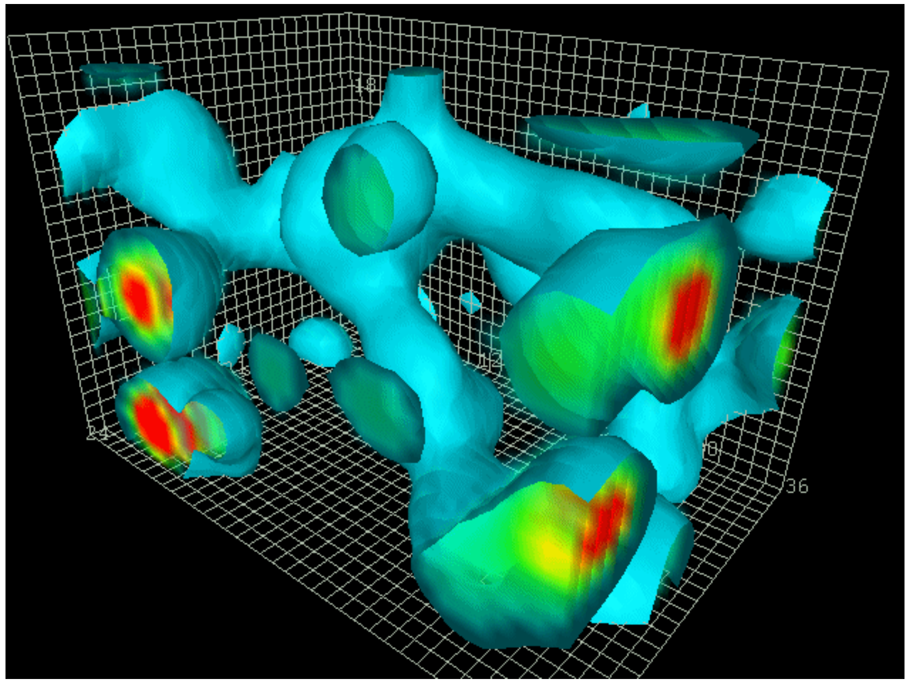
ML for LQCD is needed

- 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?

Attempts to gauge symmetry and fermions

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
TLDR; Tuning of coupling (linear model), and accept/reject

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

2310.13222, AT+: Global symmetric transformer for fermion-spin system
TLDR; Transformer for spin-fermion system, global symmetry

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)

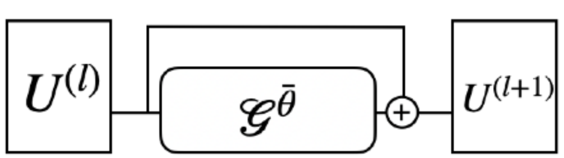
Non-locality from
pseudo-fermions
(1/D) ~ non-local

(I want to mimic
this by NN)

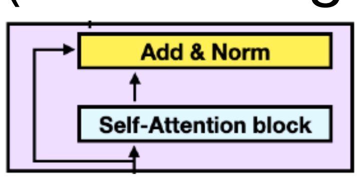
Solutions in neural net:

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

Overview/outline

Akio Tomiya

Gauge covariant transformer for LQCD

Two conditions/restrictions in LQCD:

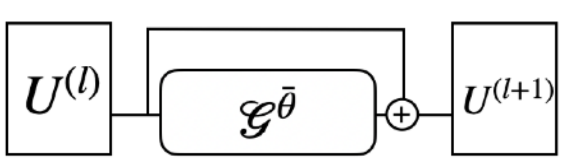
Gauge symmetry
U(x, x+μ)

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

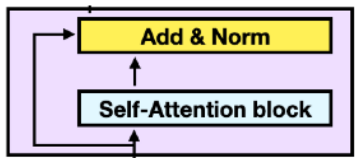
Solutions in neural net:

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 neural network
= trainable smearing (= residual flow)

Akio Tomiya

AT Y. Nagai arXiv: 2103.11965
R Abbott+ 2401.10874

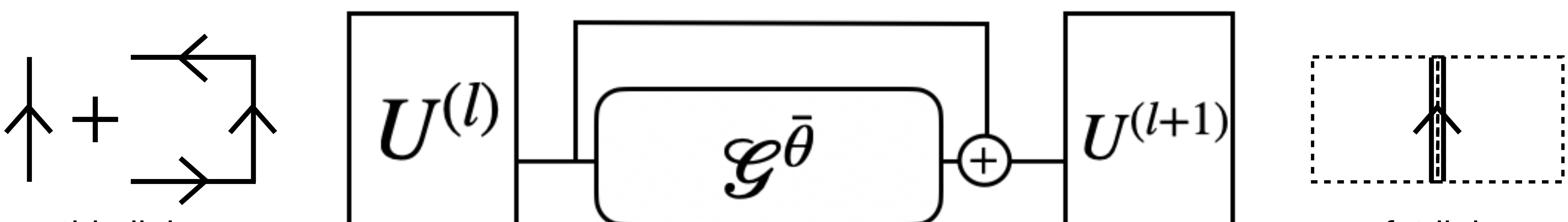
<u>Stout-type covariant neural network</u>

Loops projected on Lie algebra

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

Trainable parameters

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



thin link

Stout kernel
(with trainable parameters)

fat link

This neural network layer makes maps between gauge configurations with covariance! (trainable stout for various purpose)

[1] C. Morningster+ 2003

Overview/outline

Akio Tomiya

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

Equivariance and convolution

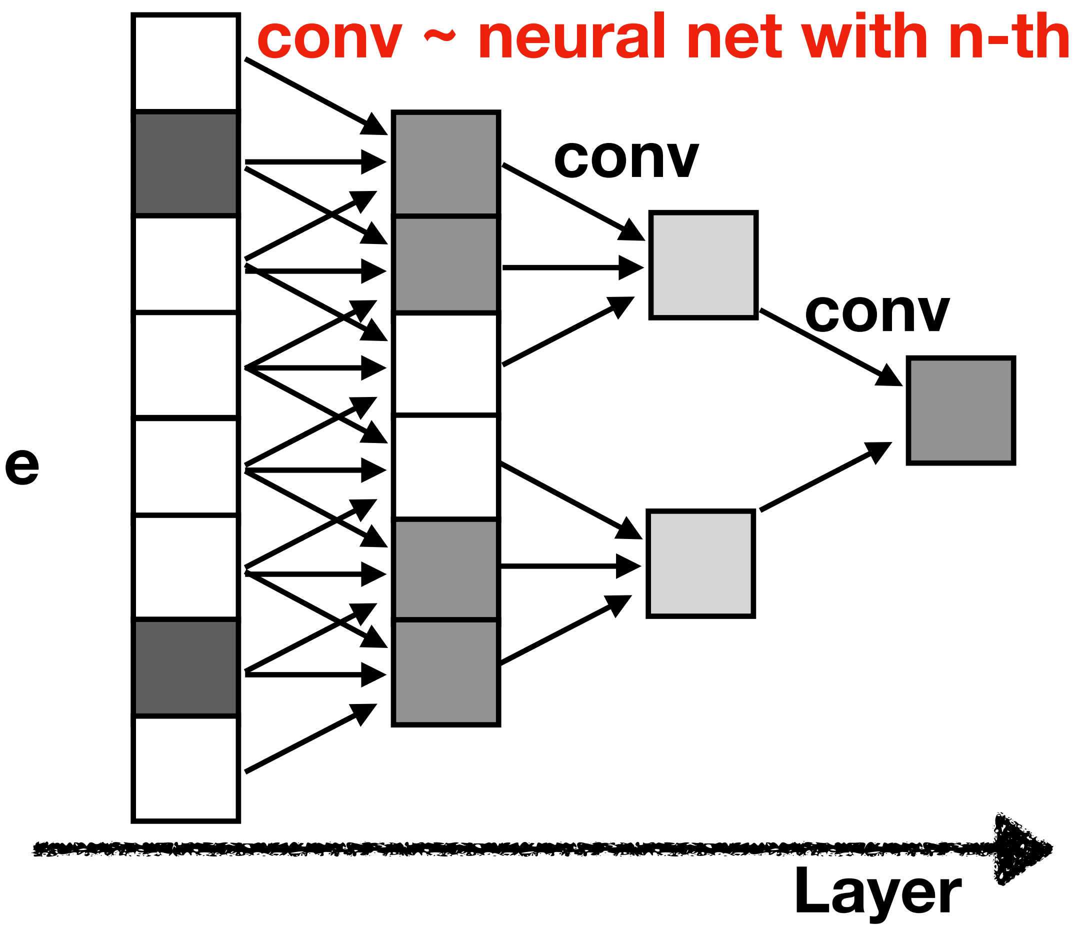
Akio Tomiya

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.

conv ~ neural net with n-th nearest neighbor connections (local)

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 important.
How can we overcome these difficulties?

Transformer and Attention

Attention layer used in Transformers (GPT, Bard)

Akio Tomiya

arXiv:1706.03762

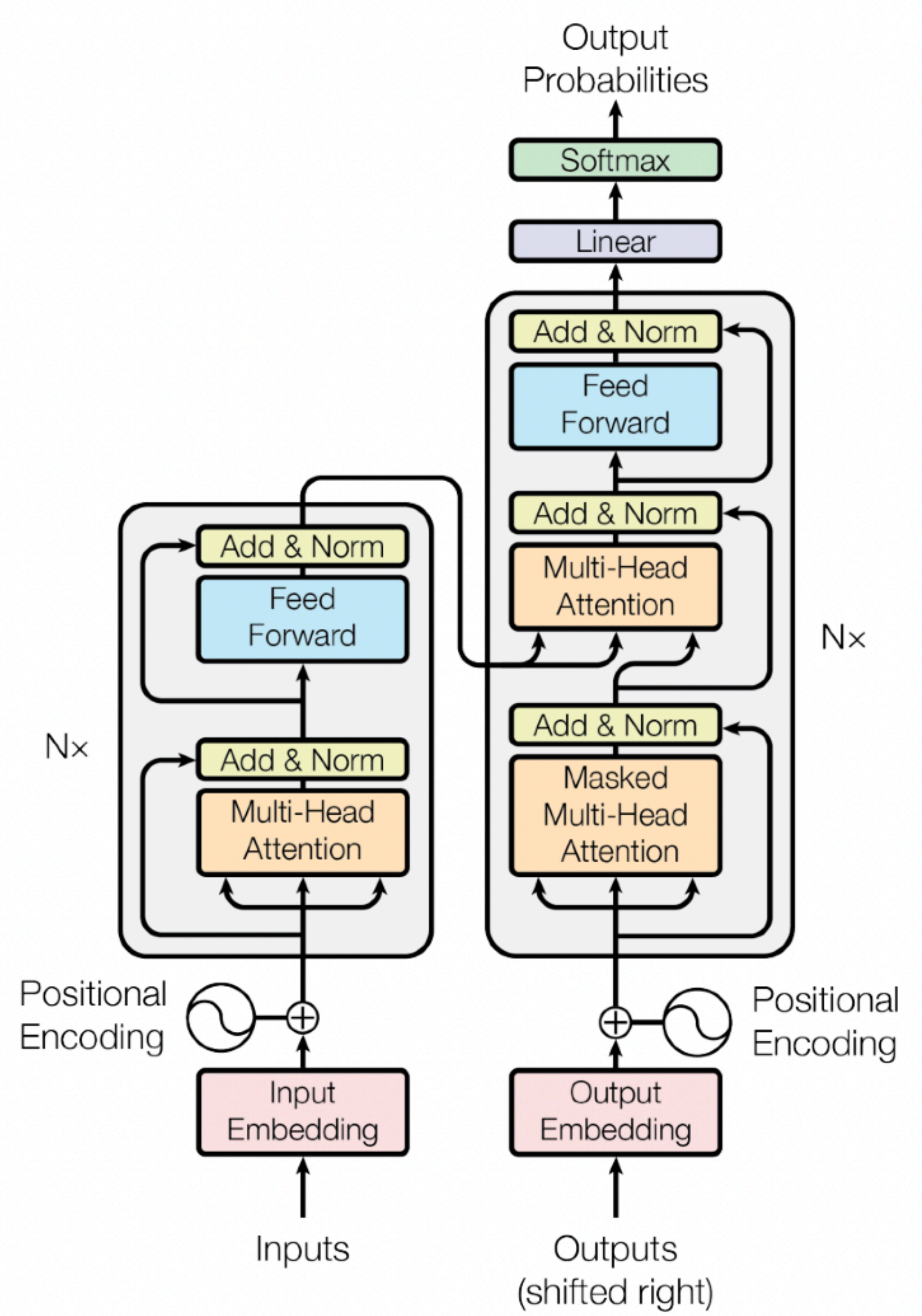


Figure 1: The Transformer - model architecture.

OpenAI ChatGPT

Gemini

DeepL

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

Akio Tomiya

Attention layer can capture non-local correlations

arXiv:1706.03762

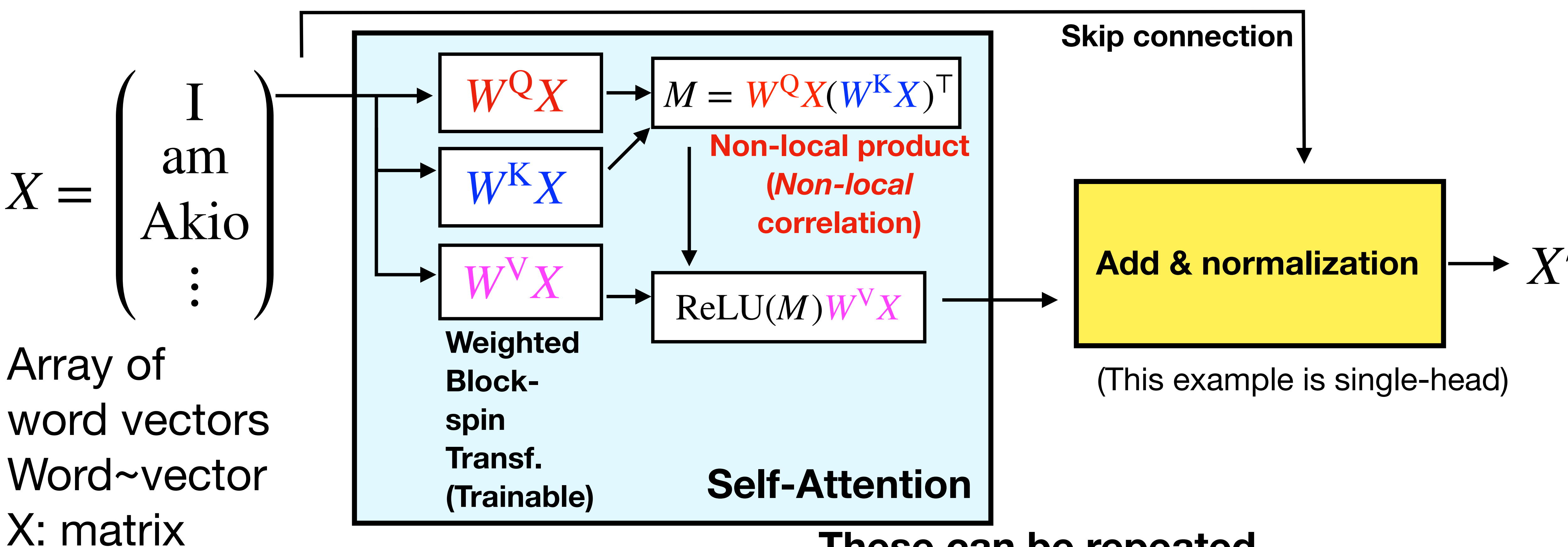
Modifier in language can be non-local

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 for spin system

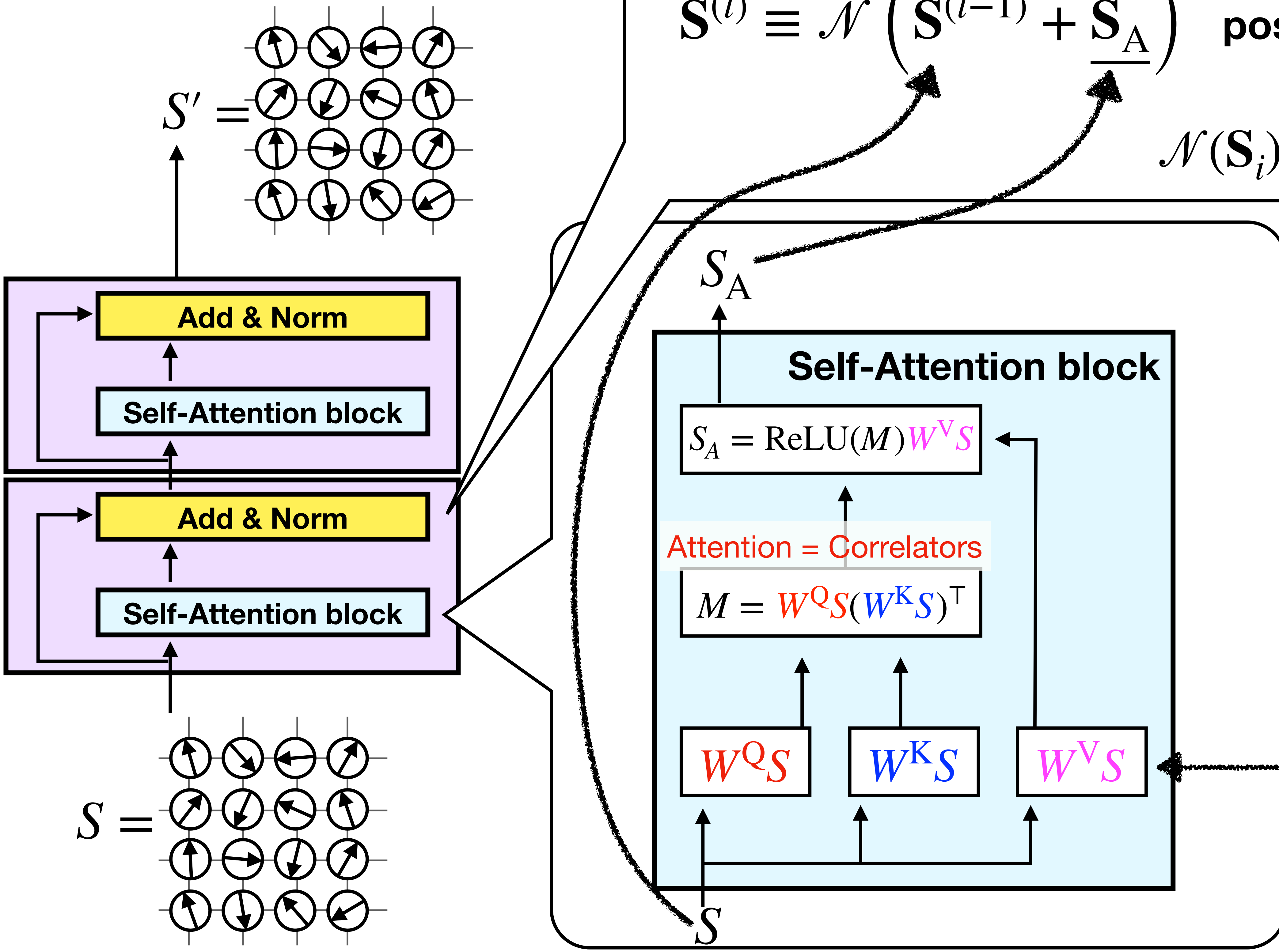
Akio Tomiya

arXiv: 2306.11527.

Symmetric Attention layers (parametrized block spin trf)

$$\mathbf{S}^{(l)} \equiv \mathcal{N}\left(\mathbf{S}^{(l-1)} + \underline{\mathbf{S}_{\mathrm{A}}}\right)$$ position-wise

$$\mathcal{N}(\mathbf{S}_i) = \mathbf{S}_i / \|\mathbf{S}_i\|$$



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

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

How can we make this gauge covariant?

Gauge covariant transformer (*CASK*)

Work in progress



A. Tomiya, H. Ohno, Y. Nagai

Overview/outline

Gauge covariant transformer for LQCD

Two conditions/restrictions in LQCD:

Gauge symmetry
U(x, x+μ)

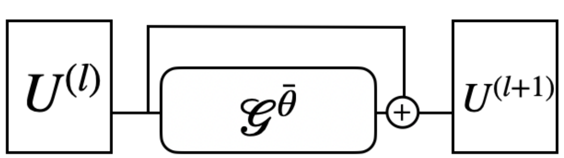
Non-locality from
pseudo-fermions
(1/D) ~ non-local

(I want to mimic
this by NN)

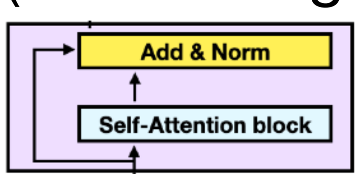
Solutions in neural net:

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

Akio Tomiya

CASK?

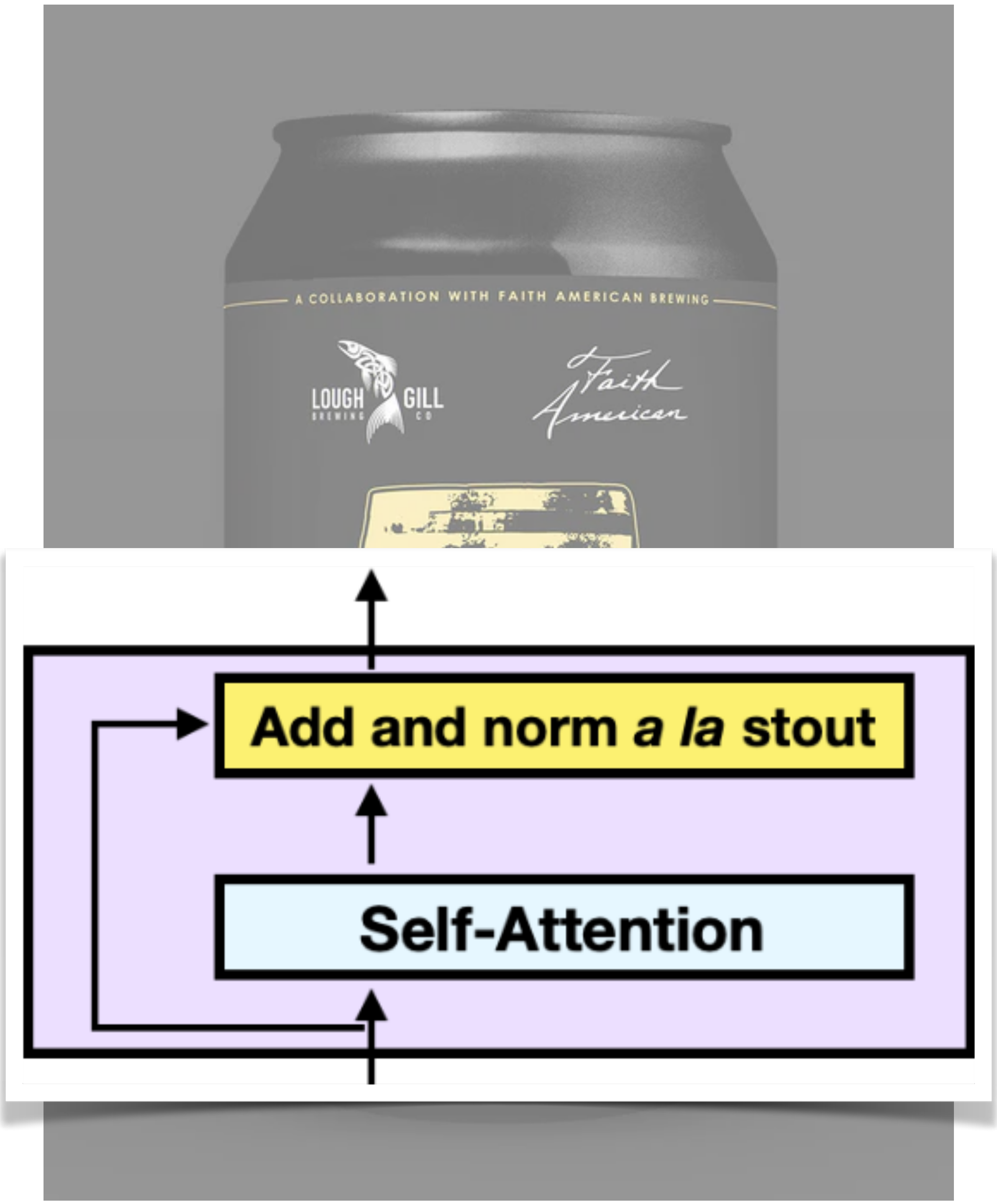
Cask stout
(Whisky Barrel-Aged Stout beer)
= stout beer in a cask

https://loughgillbrewery.com/products/cask-irish-coffee-stout

Gauge covariant transformer: CASK

Akio Tomiya

Stout kernel



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

Covariant attention block
CASK = <u>C</u>ovariant <u>A</u>ttention with <u>S</u>tout <u>K</u>ernel

It is named in an obvious reason😜

Gauge covariant transformer: CASK

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)

https://loughgillbrewery.com/products/cask-irish-coffee-stout

Gauge covariant transformer: CASK

Akio Tomiya

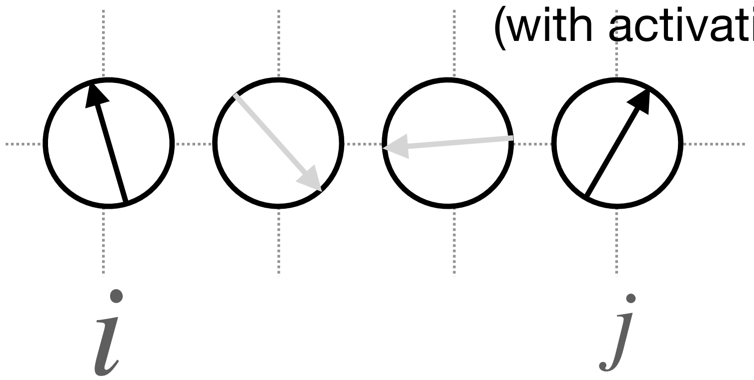
Idea: Attention must be invariant

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)



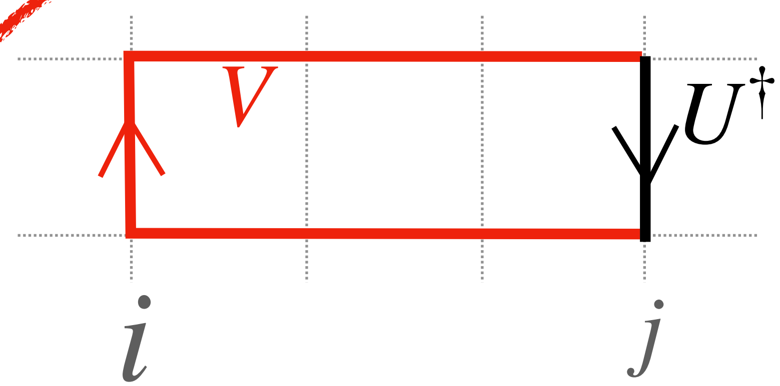
invariant
under global O(3)

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

In total, output is covariant

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

not invariant
(cannot be used)



invariant under
local SU(N)

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

(with activation)

In total, output is covariant

Gauge covariant transformer: CASK

Structure of gauge symmetric attention using stout

[1] 2021 AT+

Procedure in three steps:

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

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

Loop operator projected on Lie algebra

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

weights

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

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

cf. sparse attention, star attention

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

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

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

Loop operator projected on Lie algebra

Gauge covariant transformer: CASK

Akio Tomiya

Physically symmetric Attention layer for LQCD

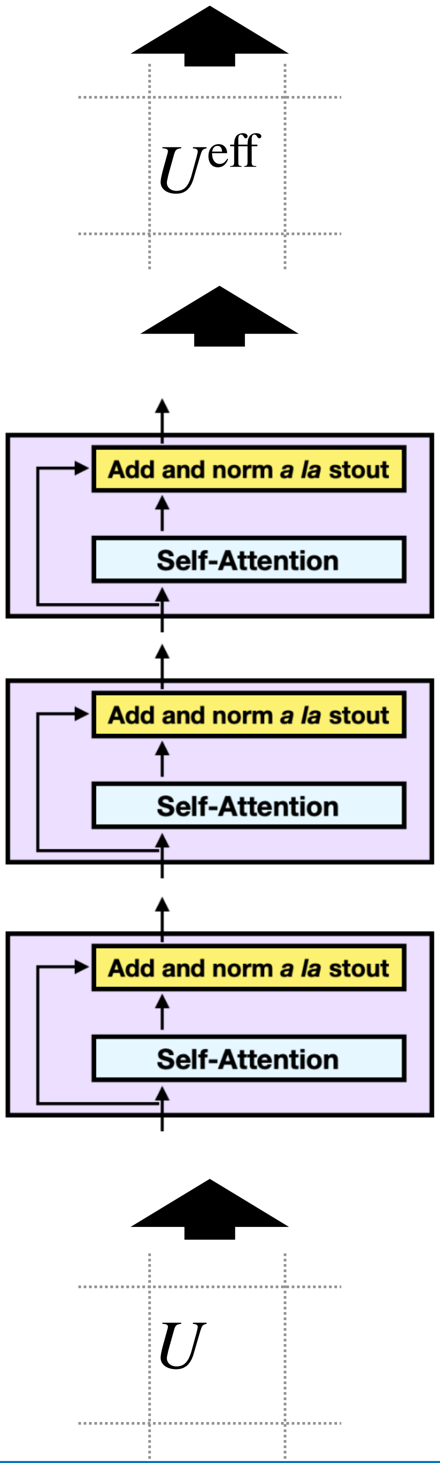
Attention layer can capture global correlation
Equivariance reduces data demands for training

	Equivariance	Gauge?	Capturable correlation	Data demmands	Applications
Convolution ($\in$ equivariant layers)	Yes 👍	Yes 👍	Local 😲	Low 👍	VAE, GAN Normalizing flow SLHMC 2103.11965 AT+
Standard Attention layer arXiv:1706.03762	No 😲	No 😲	Global 👍	Huge 😲	ChatGPT GEMINI Vision Transformer
***Equivariant* attention for spin**	Yes 👍	No 😲	Global 👍	?	Kondo system (2310.13222 AT+ 2306.11527 AT+)
***Equivariant* attention for gauge**	Yes 👍	Yes 👍	Global 👍	?	This work

Gauge covariant transformer: CASK

Simulation parameter

Construct effective action using operators with U^{eff}



LatticeQCD.jl

- 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
  - CASK with plaquette covariant kernel
    - Attention = 7-links rect staple (=3 plaquette)
  - U links are replaced by U^{eff} in D_{stag}
- “Adaptively reweighted HMC”

Gauge covariant transformer: CASK

Loss = difference of action

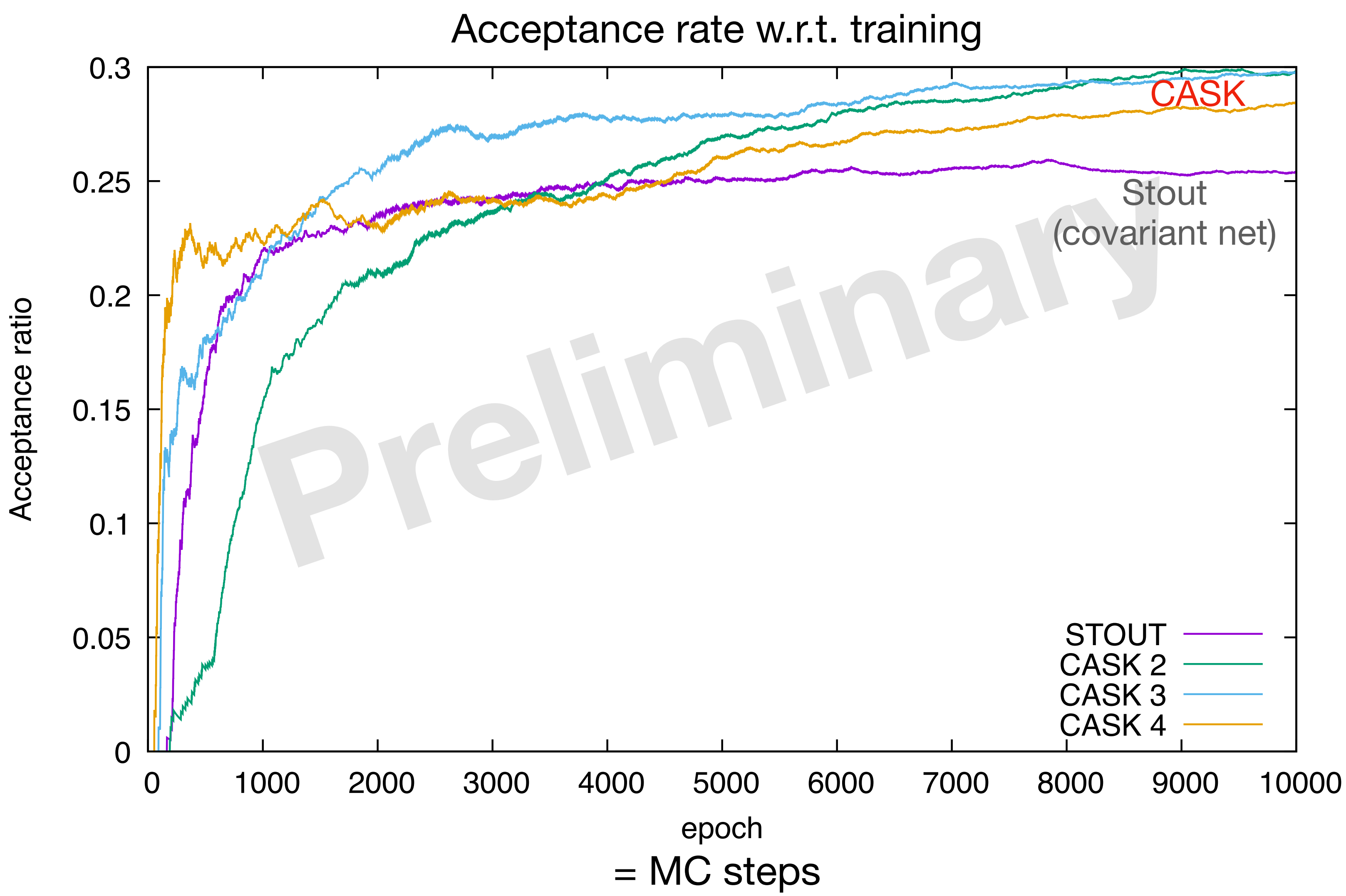
= MC steps

- Loss decreases along with the training steps
- it works as same as the stout (covariant net)
- Gain?

Gauge covariant transformer: CASK

Akio Tomiya

Attention blocks improve acceptance



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

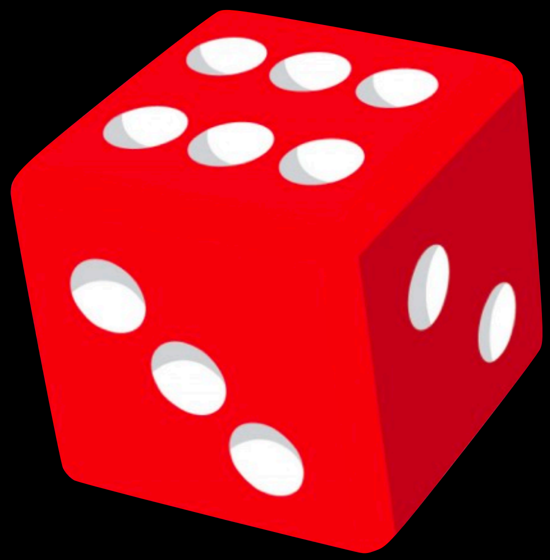
Summary

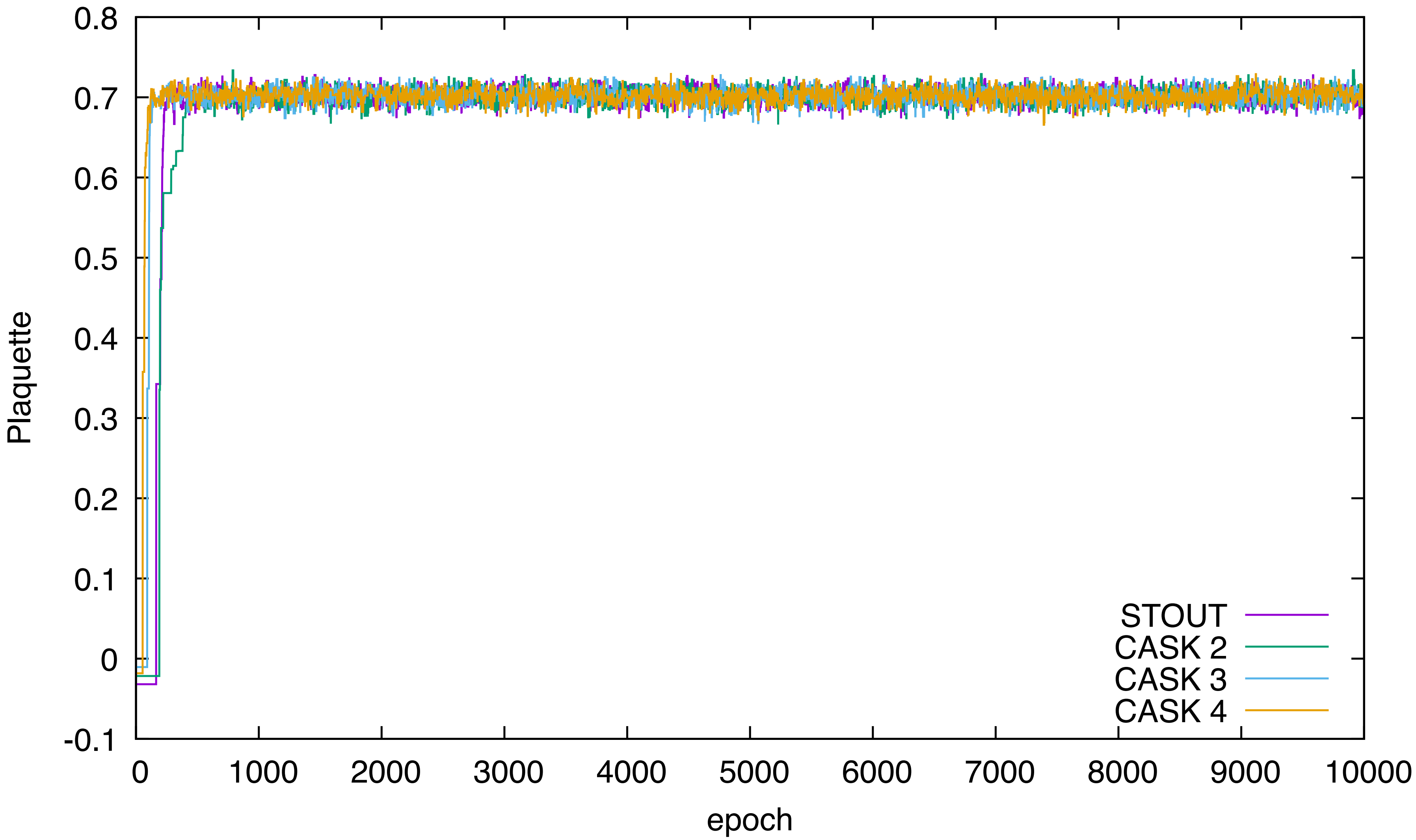
Transformer NN for Lattice QCD

- 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 Neural network and it has 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 (contour deform, flow, control variates)?

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)

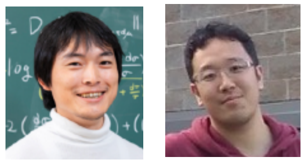
Applications

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

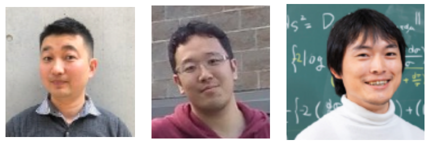
Mimicking a trvializing map using a neural net which is reversible and has tractable Jacobian.
Exact algorithm, no dynamical fermions. SU(N) is treated with diagonalization.

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

Dynamical fermions, 4 Dimension

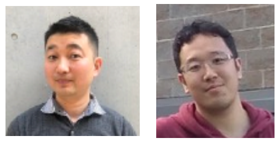
Self-learning Monte Carlo (SLMC) for lattice QCD — arxiv 2010.11900 Y. Nagai, AT, A. Tanaka

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

Akio Tomiya

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)

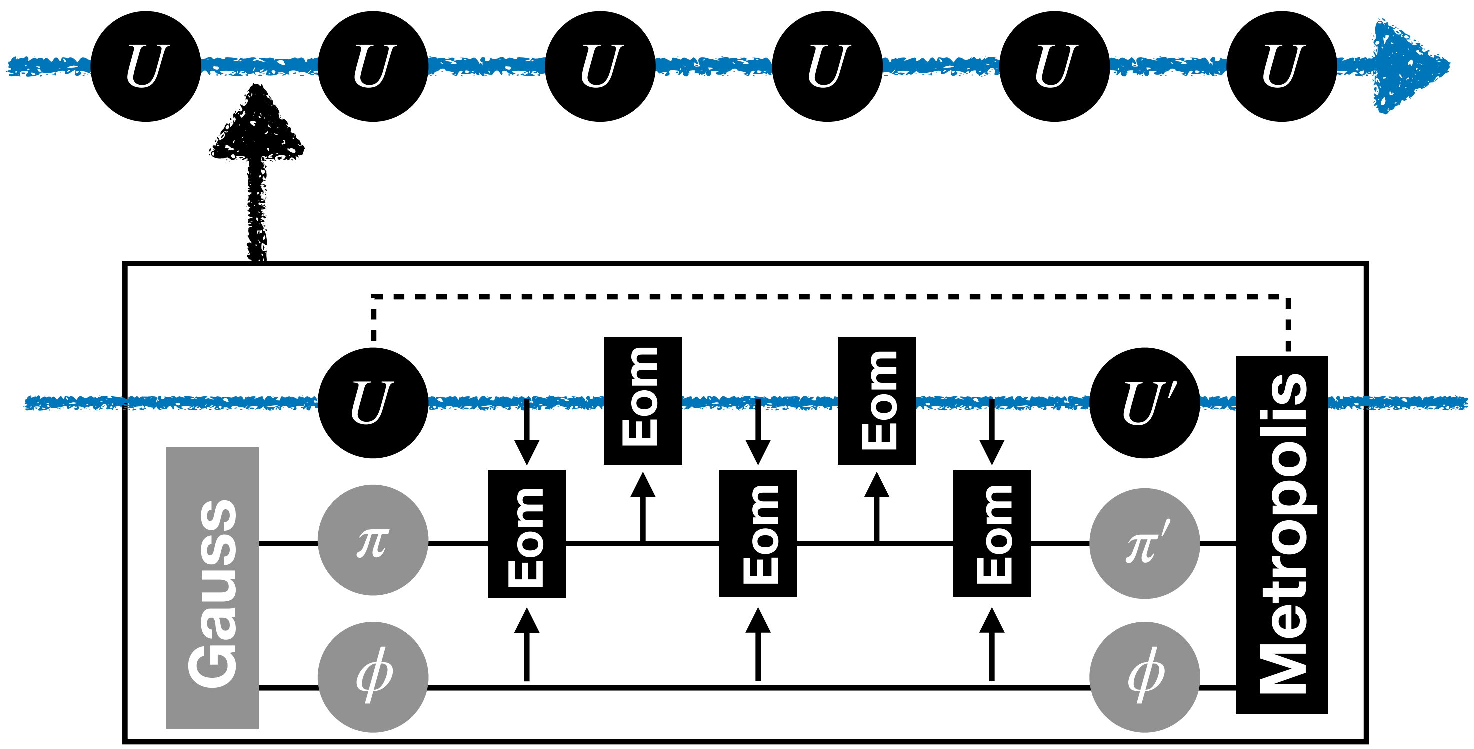
Akio Tomiya

SLHMC = Exact algorithm with ML

SLHMC for gauge system with dynamical fermions

arXiv: 2103.11965 and reference therein

<u>HMC</u>



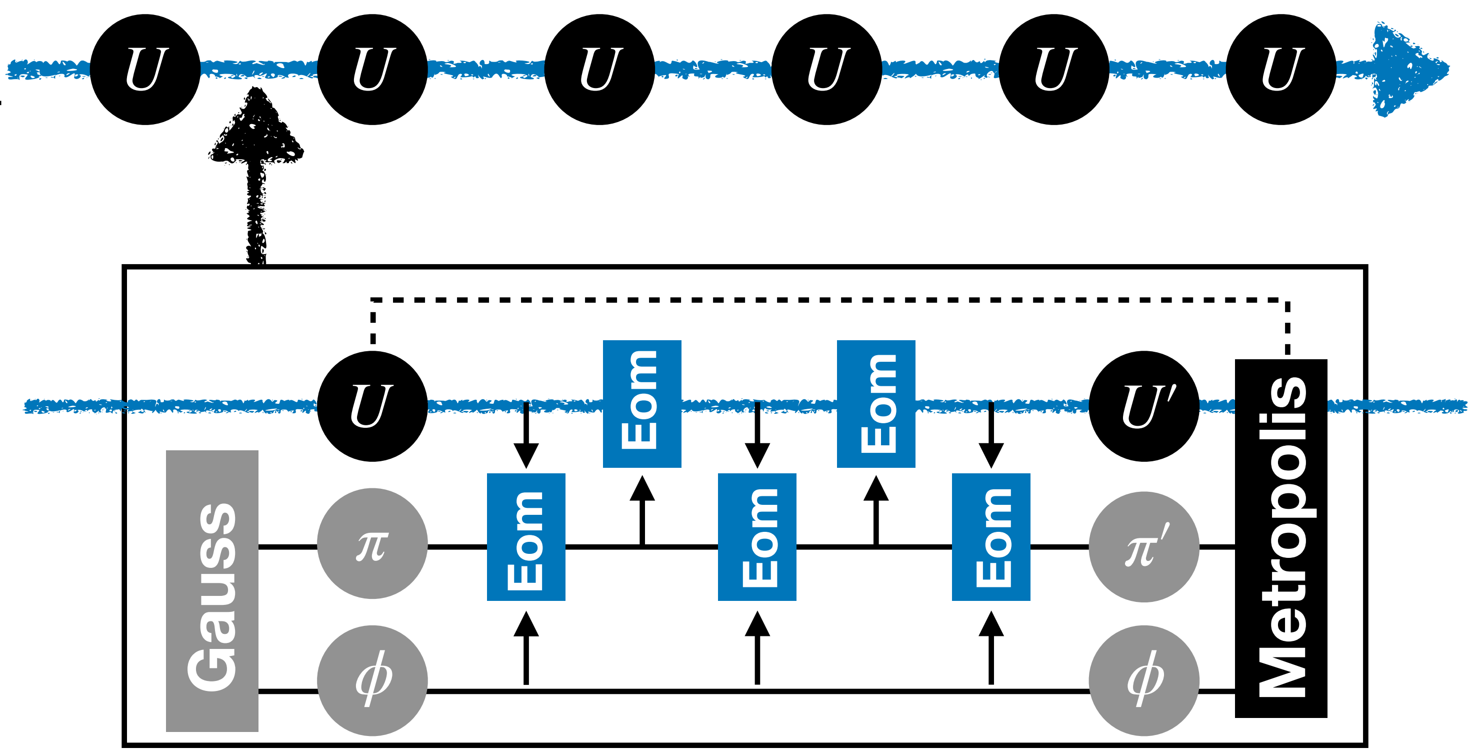
Eom **Metropolis**

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

<u>SLHMC</u>



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>

Application for the staggered in 4d

Akio Tomiya

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

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

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

Code Full scratch,
fully written in Julia lang.

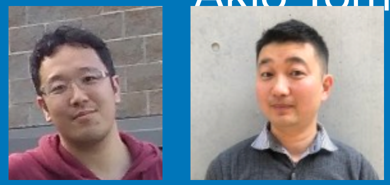
LatticeQCD.jl AT+ (in prep)

(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

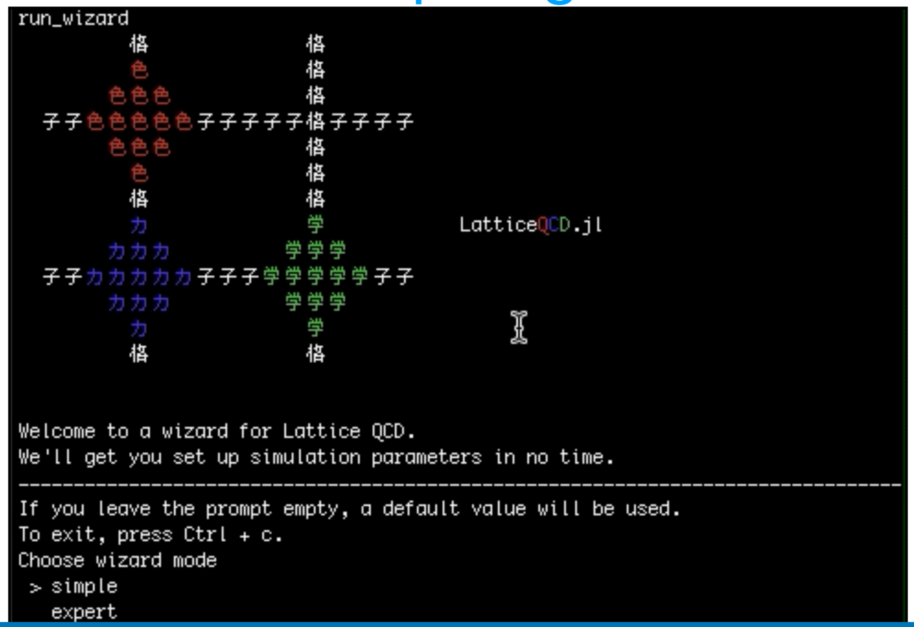
LatticeQCD.jl (Official package) : Laptop/desktop/PC-cluster/Jupyter (Google colab)

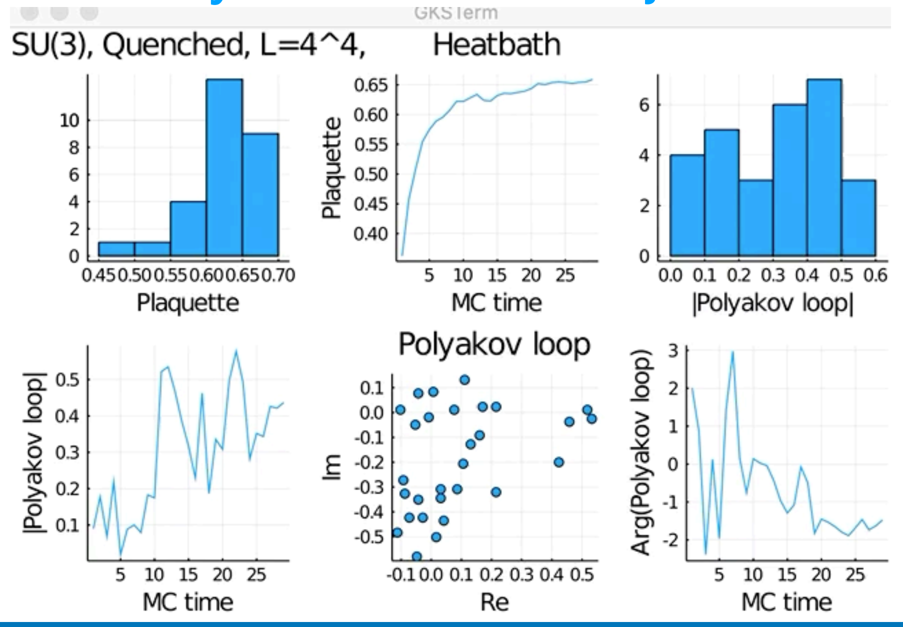
SU(Nc)-heatbath/SLHMC/SU(Nc) Stout/(R)HMC/staggered/Wilson-Clover
Domain-wall (experimental) + Measurements

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)

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

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)

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

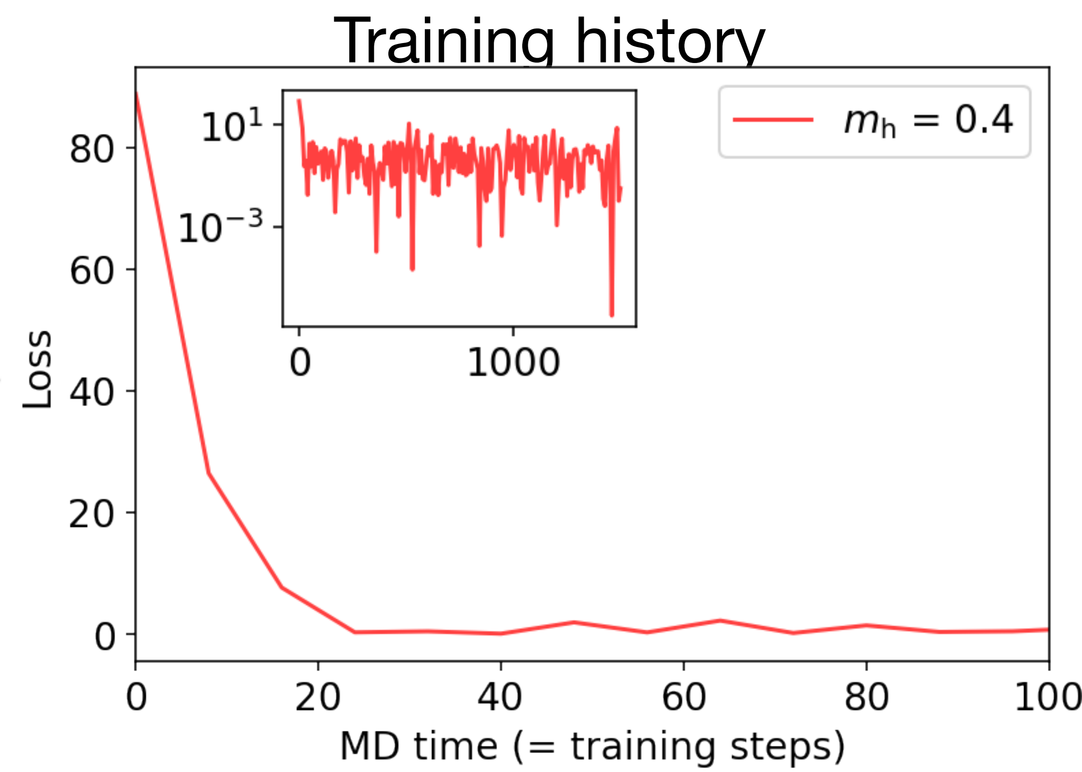
$$\theta \leftarrow \theta - \eta\frac{\partial L_\theta(\mathcal{D})}{\partial \theta},$$

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

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!

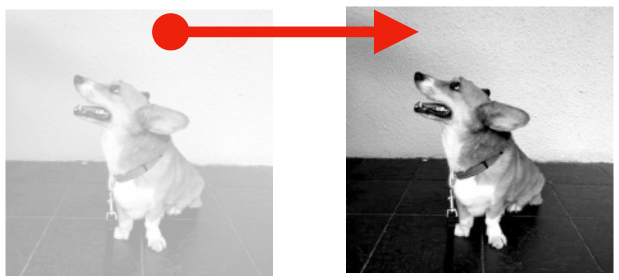
We perform SLHMC with these values!

Equivariance and convolution

Akio Tomiya

Knowledge ∋ Convolution layer = trainable filter, Equivariant

Filter on image



shift to right

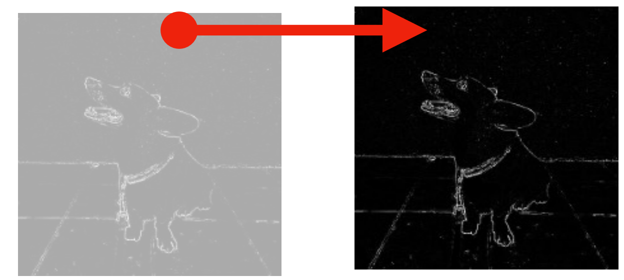
$*$

Laplacian filter

0	1	0
1	-2	1
0	1	0

(Discretization of ∂^2)

$=$

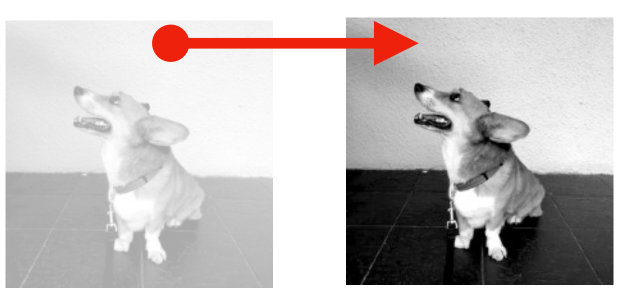


shift to right

Edge detection

Translational operation is *commutable* with filtering (equivariant)

Convolution layer



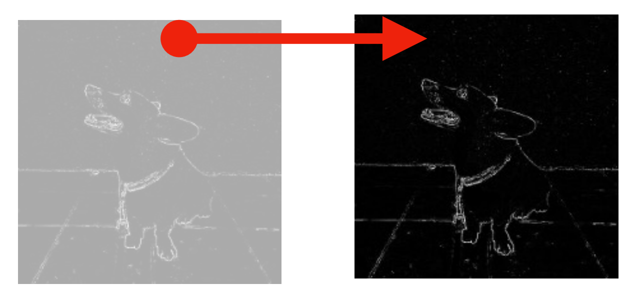
shift to right

$*$

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

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)

Organizing "Deep Learning and physics"

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

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

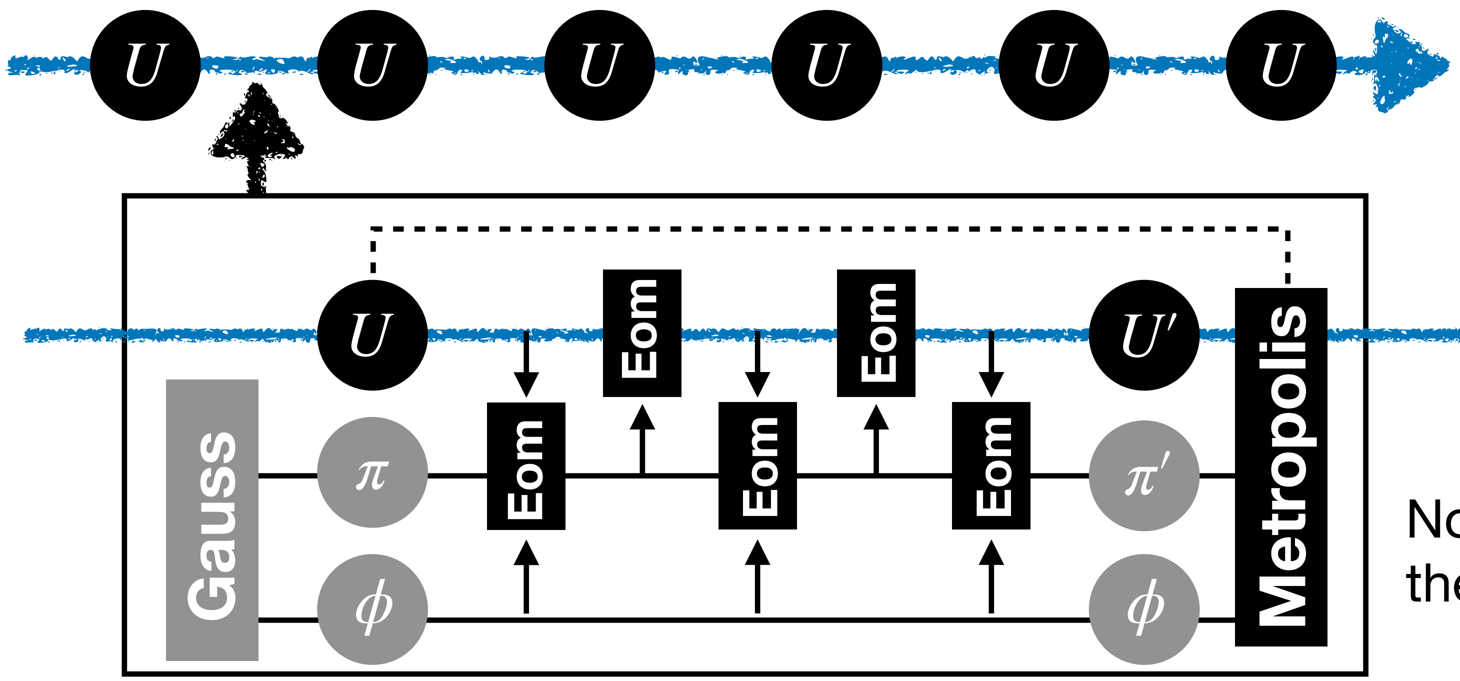
SLHMC = Exact algorithm with ML

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

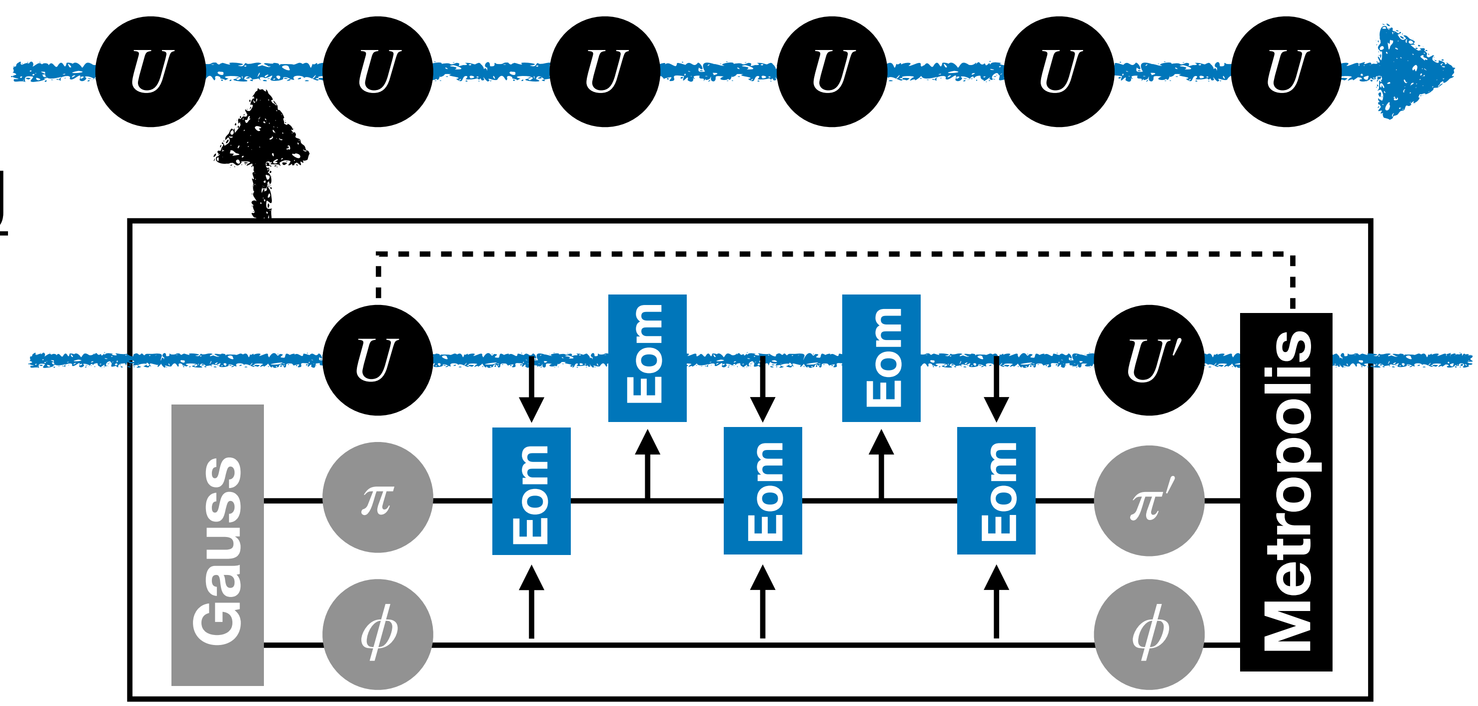


Eom **Metropolis**
Both use

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

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

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 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:

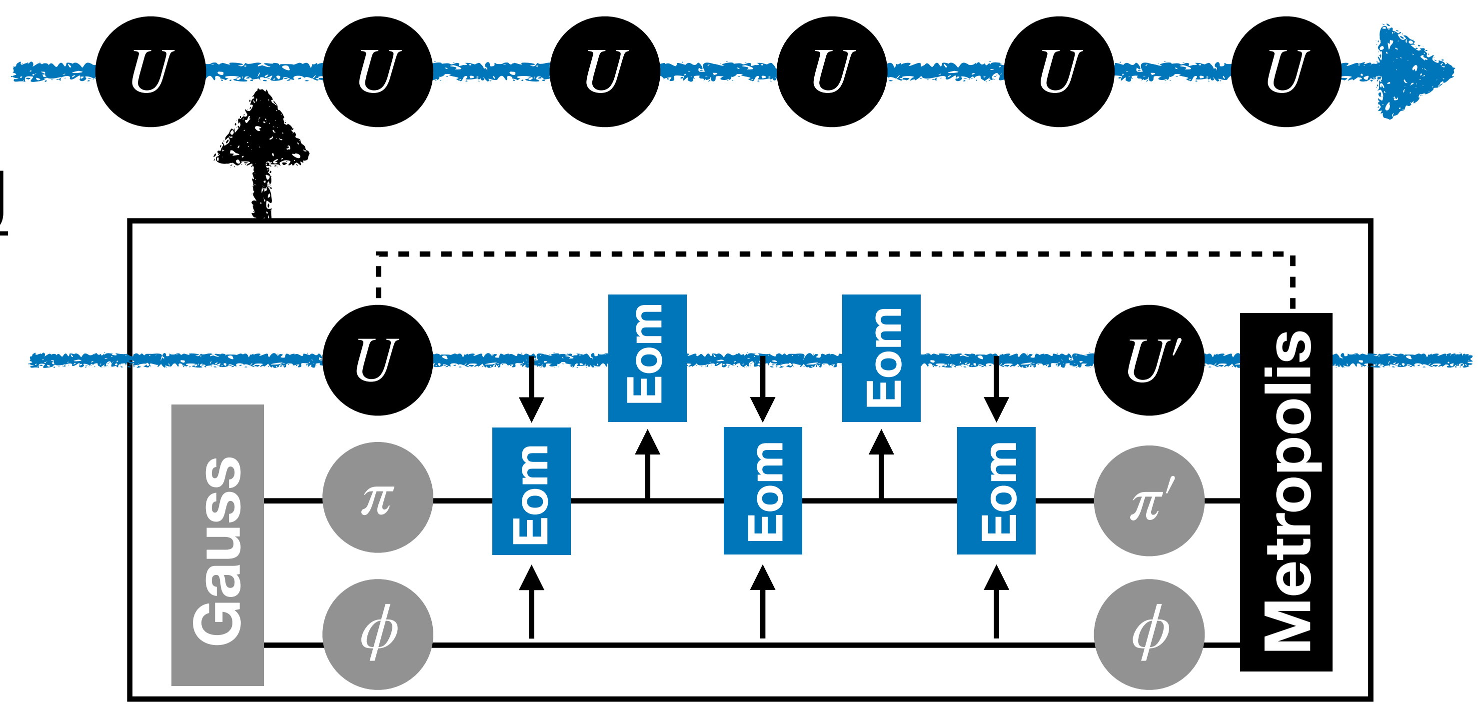
(Final target: Domain-wall vs overlap)

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

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>

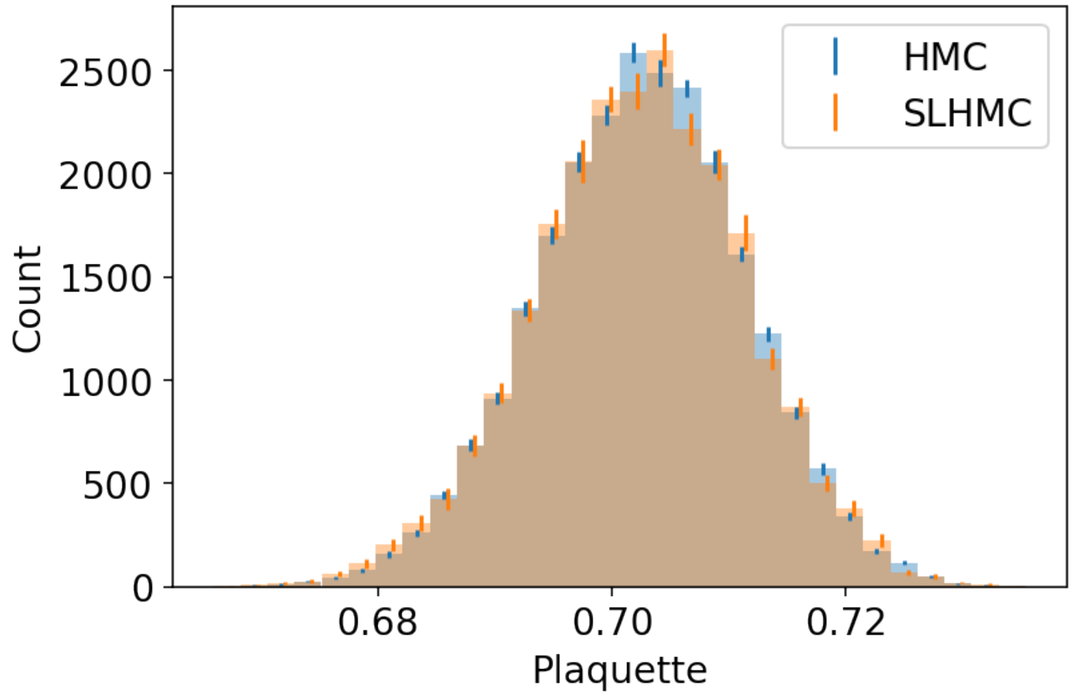
SLHMC works as an adaptive reweighting!

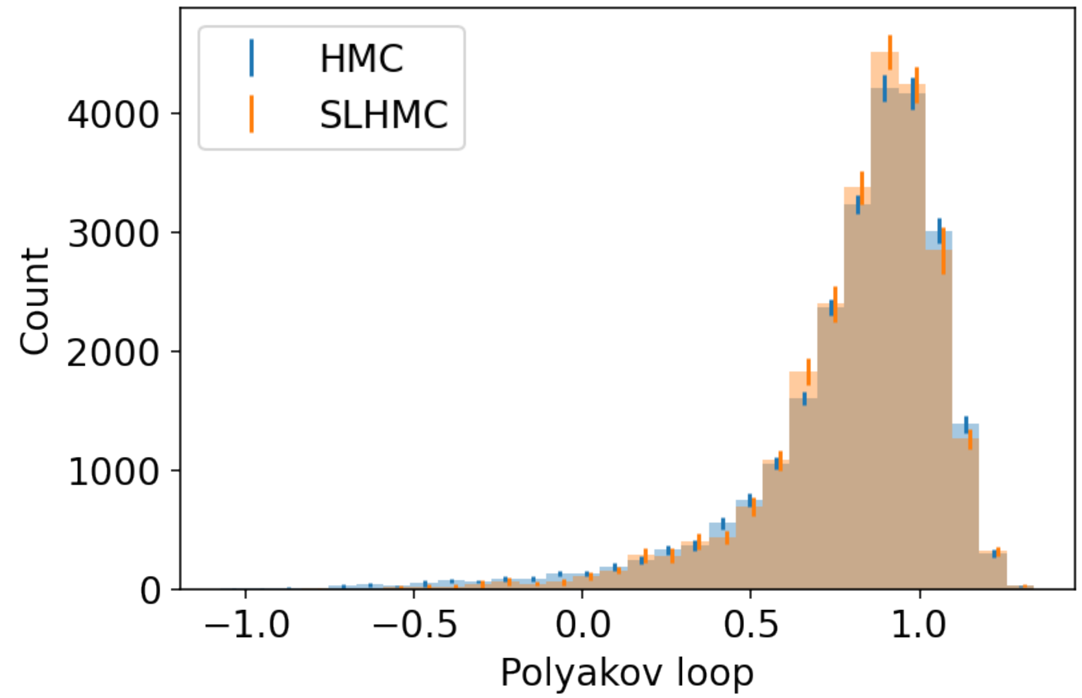
Application for the staggered in 4d

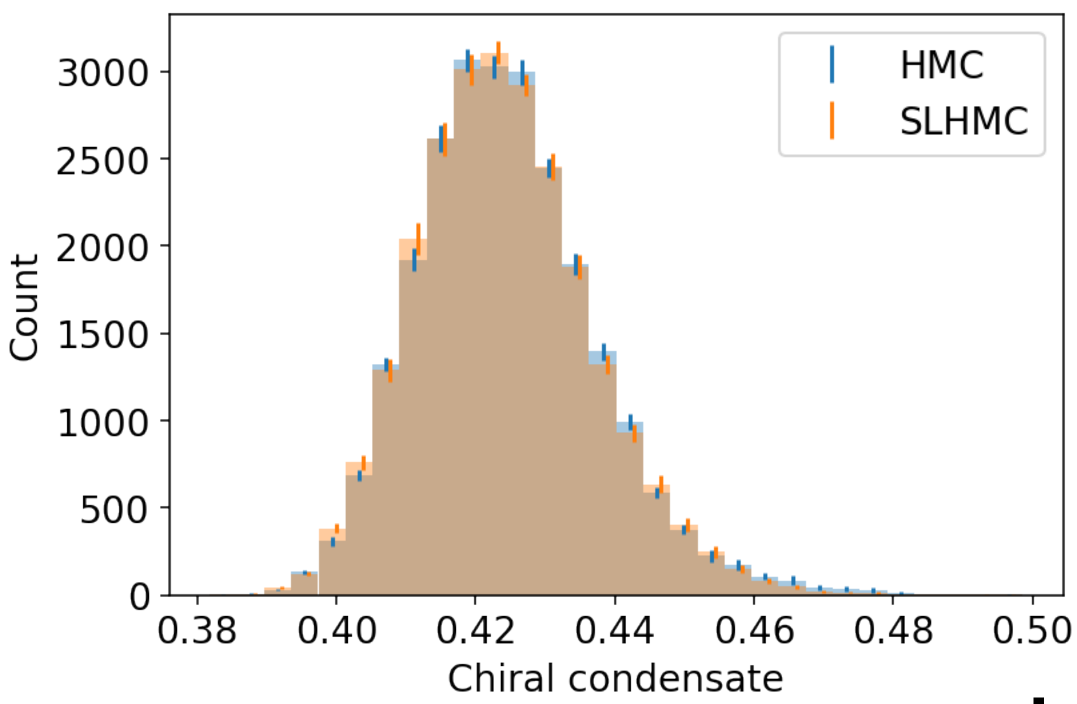
Akio Tomiya

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

Introduction

Configuration generation with machine learning is developing

Year	Group	ML	Dim.	Theory	Gauge sym	Exact?	Fermion?	Lattice2021/ref
2017	**AT+**	RBM + HMC	2d	Scalar	-	No	No	arXiv: 1712.03893
2018	K. Zhou+	GAN	2d	Scalar	-	No	No	arXiv: 1810.12879
2018	J. Pawlowski +	GAN +HMC	2d	Scalar	-	Yes?	No	arXiv: 1811.03533
2019	MIT+	Flow	2d	Scalar	-	Yes	No	arXiv: 1904.12072
2020	MIT+	Flow	2d	U(1)	Equivariant	Yes	No	arXiv: 2003.06413
2020	MIT+	Flow	2d	SU(N)	Equivariant	Yes	No	arXiv: 2008.05456
2020	**AT+**	SLMC	**4d**	SU(N)	Invariant	Yes	Partially	arXiv: 2010.11900
2021	M. Medvidović+	A-NICE	2d	Scalar	-	No	No	arXiv: 2012.01442
2021	S. Foreman	L2HMC	2d	U(1)	Yes	Yes	No	
2021	**AT+**	SLHMC	**4d**	QCD	Covariant	Yes	YES!	
2021	L. Del Debbio+	Flow	2d	Scalar, O(N)	-	Yes	No	
2021	MIT+	Flow	2d	Yukawa	-	Yes	Yes	
2021	**S. Foreman, AT+**	Flowed HMC	2d	U(1)	Equivariant	Yes	No but compatible	arXiv: 2112.01586
2021	XY Jing	Neural net	2d	U(1)	Equivariant	Yes	No	
2022	J. Finkenrath	Flow	2d	U(1)	Equivariant	Yes	Yes (diagonalization)	arxiv: 2201.02216
2022	MIT+	Flow	2d	U(1)	Equivariant	Yes	Yes (diagonalization)	arXiv:2202.11712

+…

This is not complete list. Related to lattice field theory and biased

Transformer and Attention

Attention layer can capture non-local correlations

arXiv:1706.03762

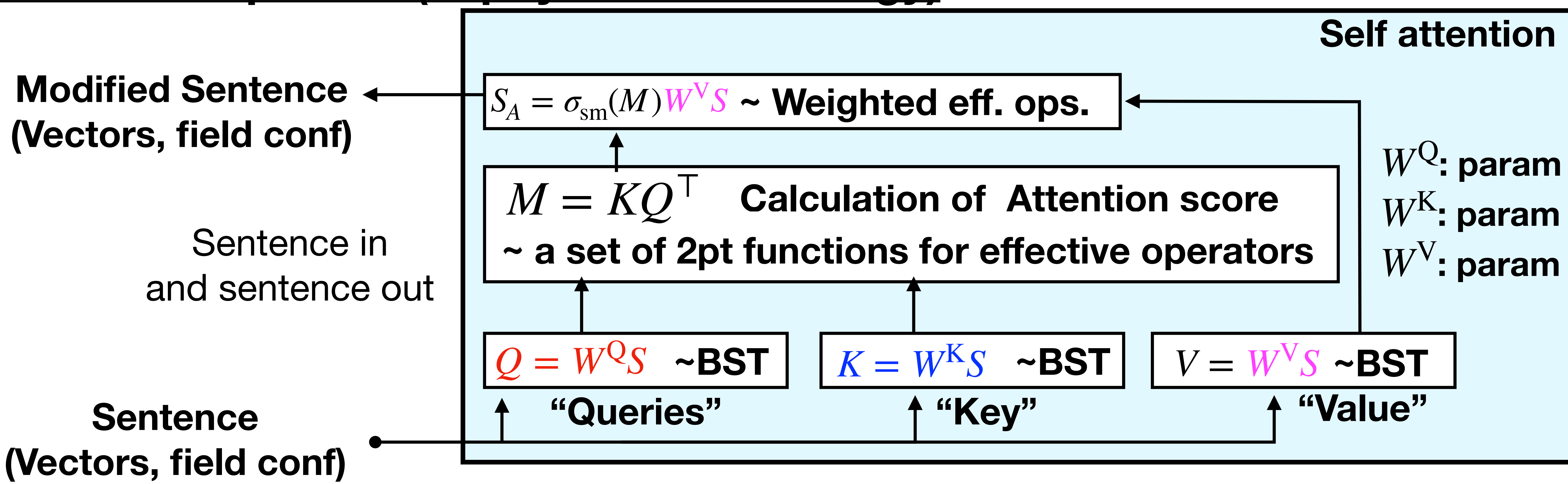
Modifier in language can be non-local

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!

Schematic picture (in physics terminology)

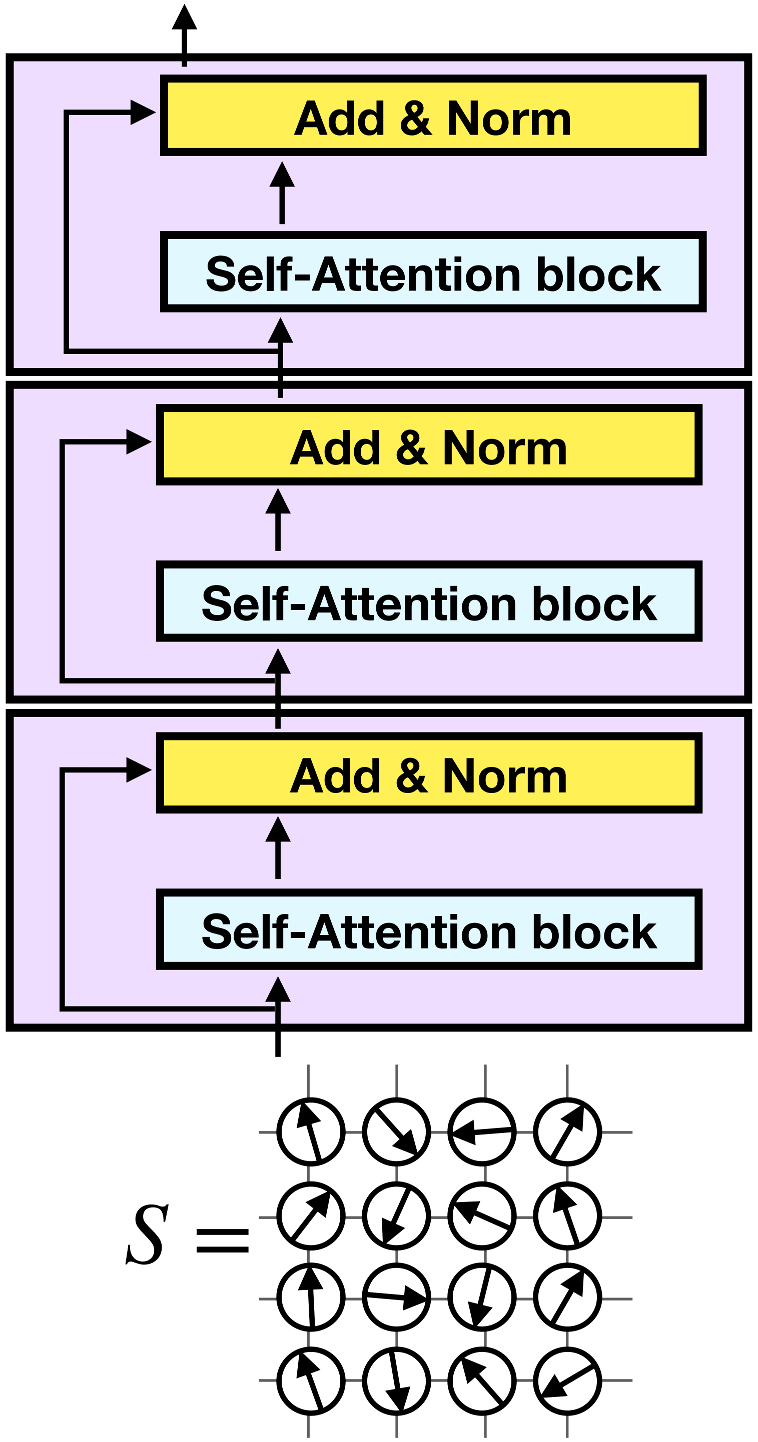


Self-learning Monte-Carlo

Attention layer makes effective spin field

arXiv: 2306.11527.

$$S' \longrightarrow H_{\text{eff}} = \text{tr}[S'(JS')^\top]$$



Metropolis with Heff → Metropolis *-Hastings* with H& Heff →

SLMC

$$\{\mathbf{S}\} \longrightarrow \{\mathbf{S}\} \longrightarrow \{\mathbf{S}\} \longrightarrow \{\mathbf{S}\} \longrightarrow \{\mathbf{S}\} \longrightarrow$$

Next, we explain
how can we realize the effective model
with Attention layers

Self-learning Monte-Carlo

Equivariant Attention layer

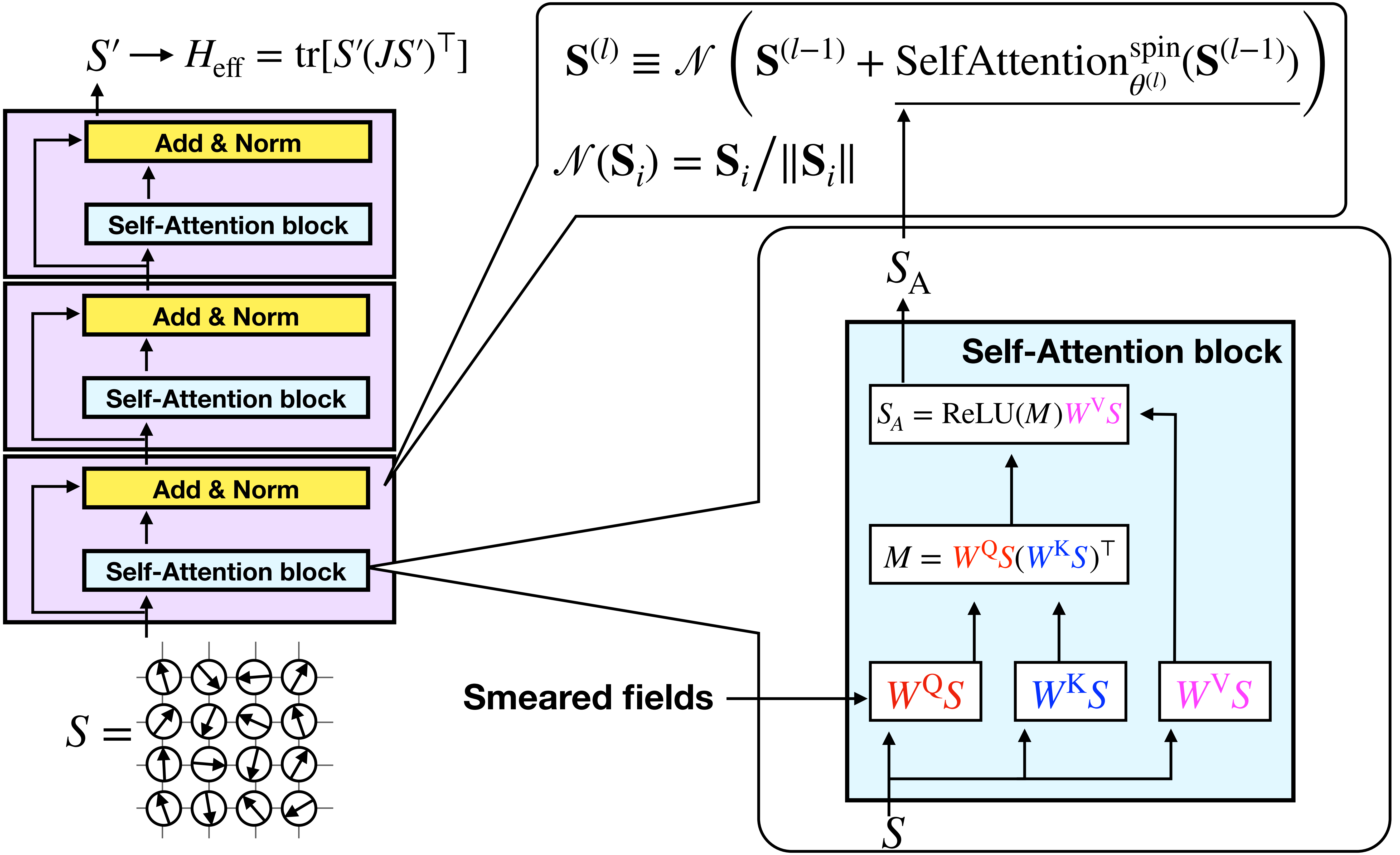
Akio Tomiya

arXiv: 2306.11527.

$$\mathbf{S}^{(l)} \equiv \mathcal{N}\left(\mathbf{S}^{(l-1)} + \mathrm{SelfAttention}^{\mathrm{spin}}_{\theta^{(l)}}(\mathbf{S}^{(l-1)})\right)$$

$$\mathcal{N}(\mathbf{S}_i) = \mathbf{S}_i / \|\mathbf{S}_i\|$$

$$S' \longrightarrow H_{\mathrm{eff}} = \mathrm{tr}[S'(JS')^{\top}]$$



Self-learning Monte-Carlo

Previous work

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

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

Brute force effective model:
n nearest neighbor

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

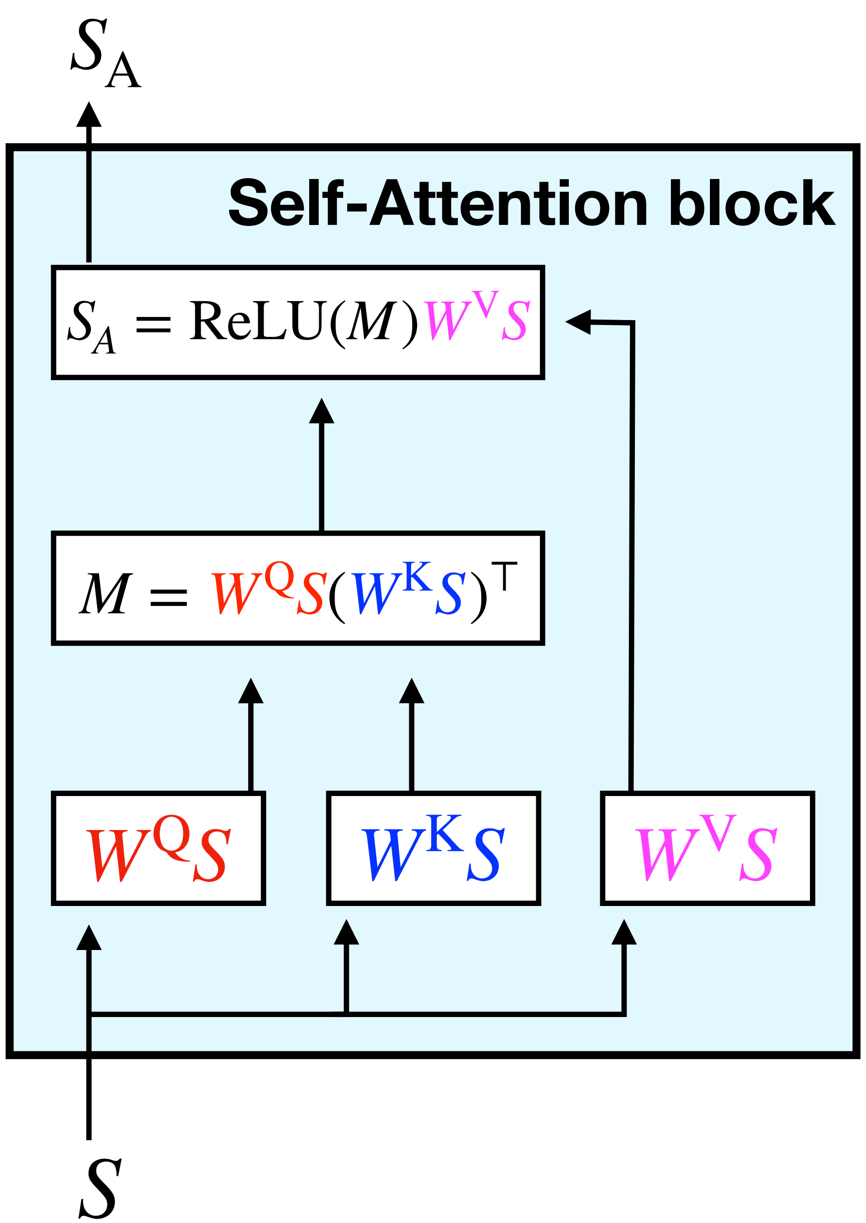
$$\longrightarrow H_{\mathrm{eff}} = -\sum_{\langle i,j\rangle_n} J^{\mathrm{eff}}_n \mathbf{S}^{\mathrm{NN}}_i \cdot \mathbf{S}^{\mathrm{NN}}_j + E_0$$

Self-learning Monte-Carlo

Akio Tomiya

Equivariant under spin-rotation & translation

arXiv: 2306.11527.



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

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

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

Gram matrix

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

Spin rotation for Si keeps G invariant.
G is a matrix for coordinate but not for spin.

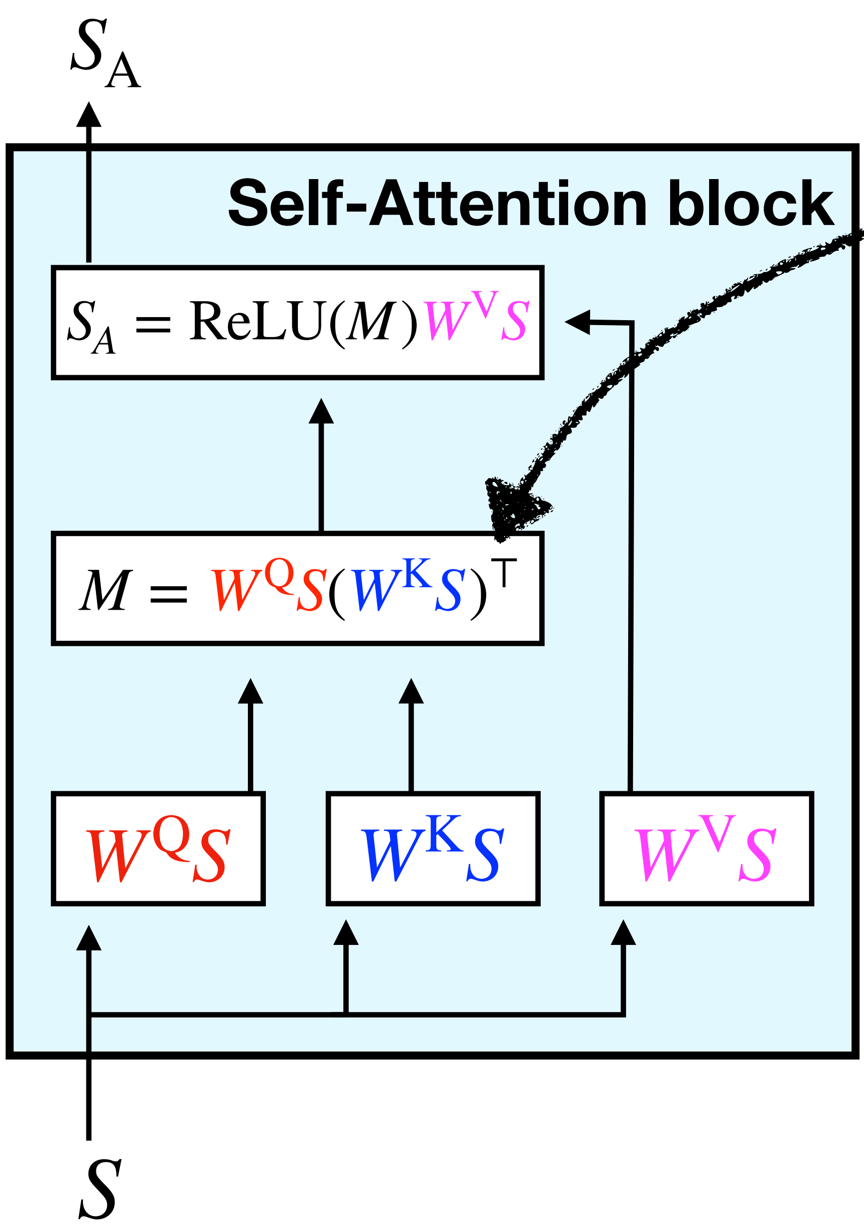
If an effective hamiltonian is a function Gram matrix, it has rotational symmetry

Akio Tomiya

Self-learning Monte-Carlo

Equivariant under spin-rotation & translation

arXiv: 2306.11527.



Weight matrix (fit parameters)

$$M = W^{\mathrm{Q}} \underbrace{(SS^{\top})}_{\text{Rotationally invariant}} (W^{\mathrm{K}})^{\top}$$

W^{Q}, W^{K} keep translational equivariance by weight sharing

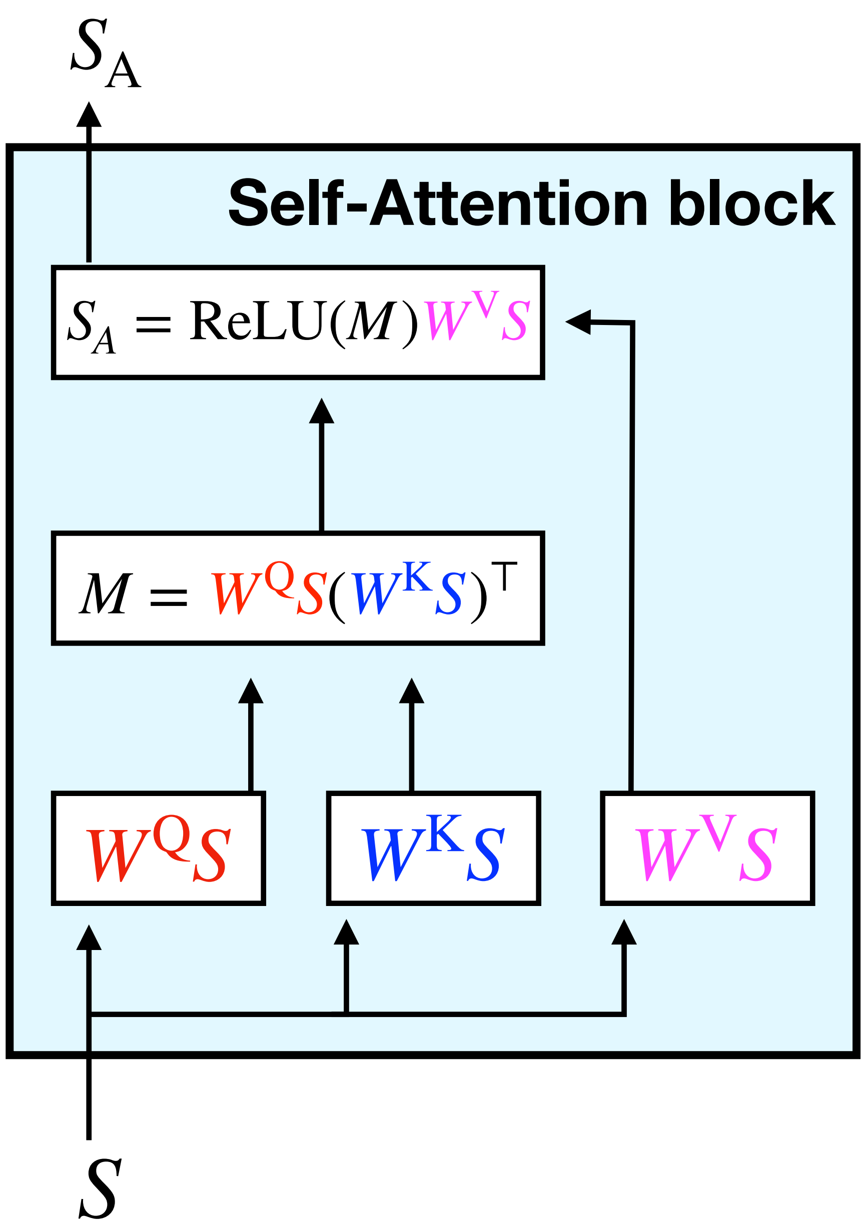
$$SS^{\top} = \begin{pmatrix} \vec{s}_1 \cdot \vec{s}_1 & \vec{s}_1 \cdot \vec{s}_2 & \vec{s}_1 \cdot \vec{s}_3 & \vec{s}_1 \cdot \vec{s}_4 \\ \vec{s}_2 \cdot \vec{s}_1 & \vec{s}_2 \cdot \vec{s}_2 & \vec{s}_2 \cdot \vec{s}_3 & \vec{s}_2 \cdot \vec{s}_4 \\ \vec{s}_3 \cdot \vec{s}_1 & \vec{s}_3 \cdot \vec{s}_2 & \vec{s}_3 \cdot \vec{s}_3 & \vec{s}_3 \cdot \vec{s}_4 \\ \vec{s}_4 \cdot \vec{s}_1 & \vec{s}_4 \cdot \vec{s}_2 & \vec{s}_4 \cdot \vec{s}_3 & \vec{s}_4 \cdot \vec{s}_4 \end{pmatrix}$$

Self-learning Monte-Carlo

Equivariant under spin-rotation & translation

Akio Tomiya

arXiv: 2306.11527.



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

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

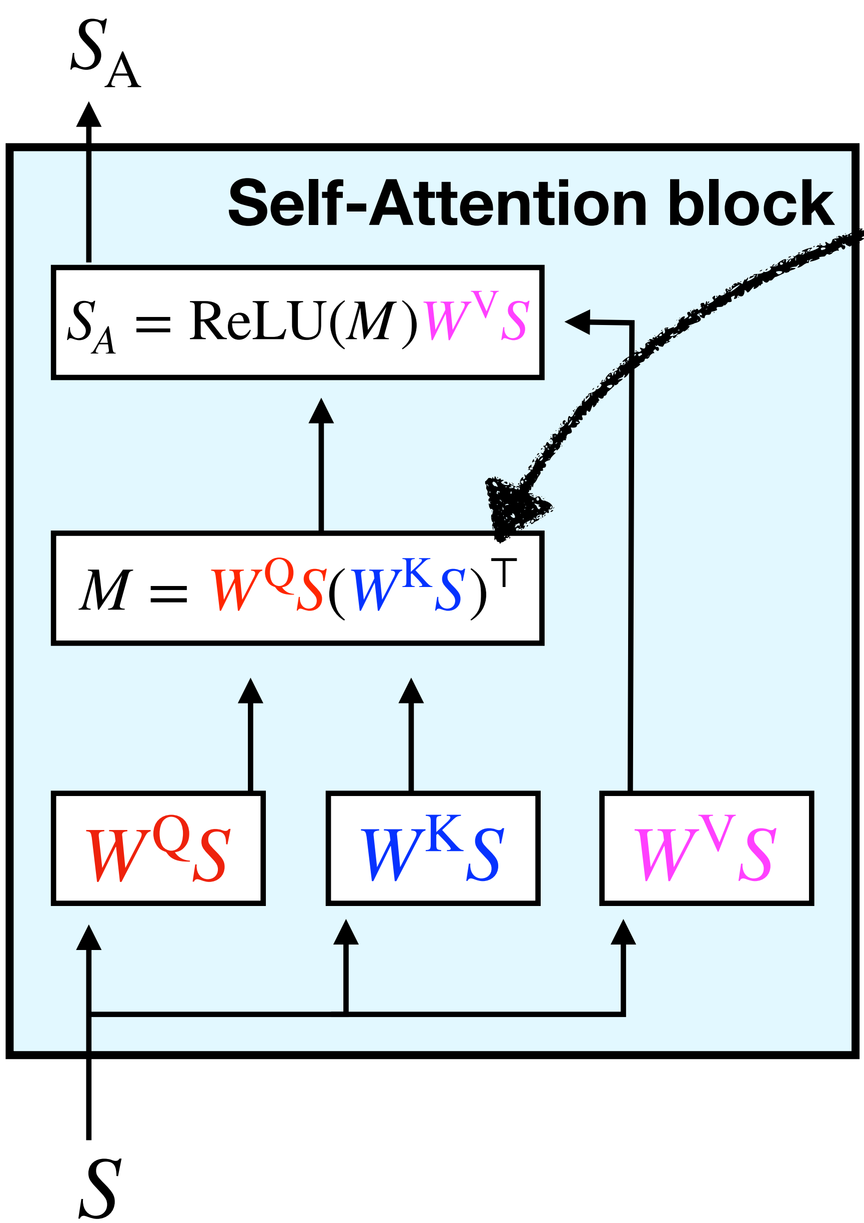
Spin rotation for Si keeps G invariant.
G is a matrix for coordinate but not for spin.

Self-learning Monte-Carlo

Equivariant under spin-rotation & translation

Akio Tomiya

arXiv: 2306.11527.



Weight matrix (fit parameters)

$$M = W^Q \underbrace{(SS^\top)}(W^K)^\top$$

$$= S_Q S_K{}^\top$$

Rotationally invariant
Transl. equivariant

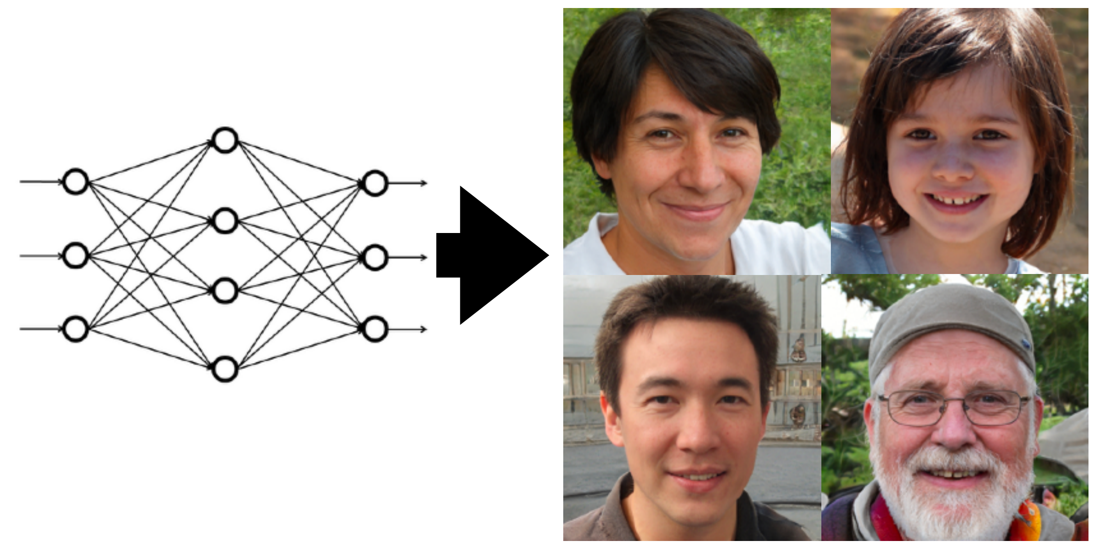
$$S_\alpha = \begin{pmatrix} \vec{s}_1 \cdot \vec{s}_1 & \vec{s}_1 \cdot \vec{s}_2 & \vec{s}_1 \cdot \vec{s}_3 & \vec{s}_1 \cdot \vec{s}_4 \\ \vec{s}_2 \cdot \vec{s}_1 & \vec{s}_2 \cdot \vec{s}_2 & \vec{s}_2 \cdot \vec{s}_3 & \vec{s}_2 \cdot \vec{s}_4 \\ \vec{s}_3 \cdot \vec{s}_1 & \vec{s}_3 \cdot \vec{s}_2 & \vec{s}_3 \cdot \vec{s}_3 & \vec{s}_3 \cdot \vec{s}_4 \\ \vec{s}_4 \cdot \vec{s}_1 & \vec{s}_4 \cdot \vec{s}_2 & \vec{s}_4 \cdot \vec{s}_3 & \vec{s}_4 \cdot \vec{s}_4 \end{pmatrix}$$

How to treat gauge fields with neural networks?

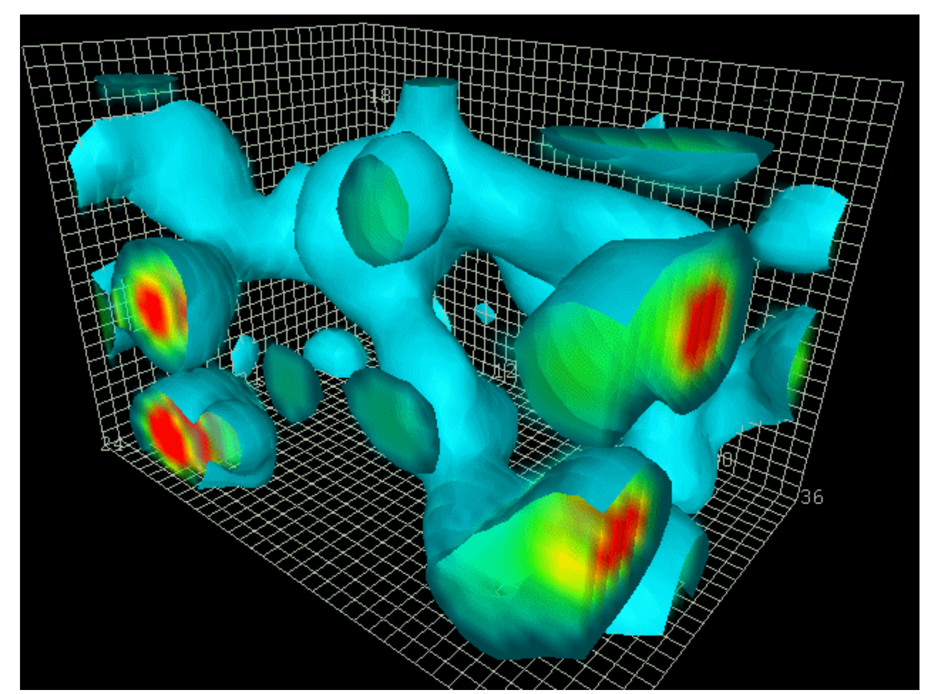
(maybe skip)

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. AlphaFold2
- 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/

Introduction

Akio Tomiya

Configuration generation with machine learning is developing

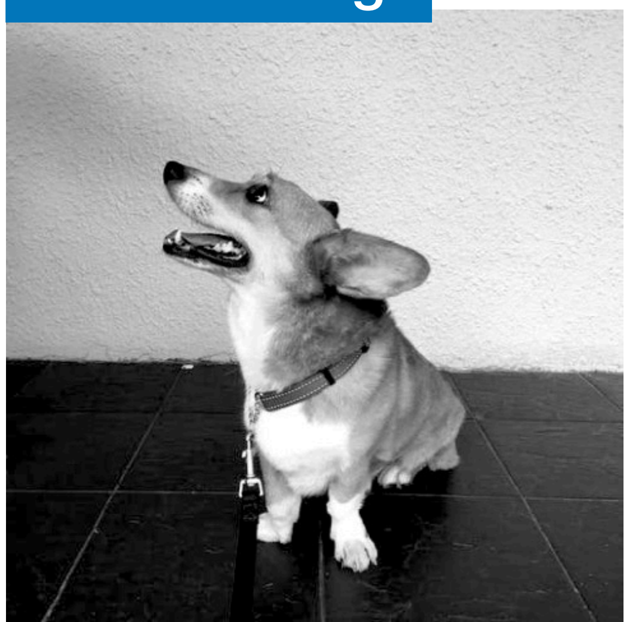
Year	Group	ML	Dim.	Theory	Gauge sym	Exact?	Fermion?	Reference
2017	**AT, Akinori Tanaka**	RBM + HMC	2d	Scalar	-	No	No	arXiv: 1712.03893
2018	K. Zhou+	GAN	2d	Scalar	-	No	No	arXiv: 1810.12879
2018	J. Pawlowski +	GAN +HMC	2d	Scalar	-	Yes?	No	arXiv: 1811.03533
2019	MIT+	Flow	2d	Scalar	-	Yes	No	arXiv: 1904.12072
2020	MIT+	Flow	2d	U(1)	Equivariant	Yes	No	arXiv: 2003.06413
2020	MIT+	Flow	2d	SU(N)	Equivariant	Yes	No	arXiv: 2008.05456
2020	**AT, Akinori Tanaka +**	SLMC	**4d**	SU(N)	Invariant	Yes	Partially	arXiv: 2010.11900
2021	M. Medvidovic´+	A-NICE	2d	Scalar	-	No	No	arXiv: 2012.01442
2021	S. Foreman	L2HMC	2d	U(1)	Yes	Yes	No	
2021	**AT+**	SLHMC	**4d**	QCD	Covariant	Yes	YES!	This talk
2021	L. Del Debbio+	Flow	2d	Scalar, O(N)	-	Yes	No	
2021	MIT+	Flow	2d	Yukawa	-	Yes	Yes	
2021	**S. Foreman, AT+**	Flowed HMC	2d	U(1)	Equivariant	Yes	No but compatible	arXiv: 2112.01586
2021	XY Jing	Neural net	2d	U(1)	Equivariant	Yes	No	
2022	J. Finkenrath	Flow	2d	U(1)	Equivariant	Yes	Yes (diagonalization)	arxiv: 2201.02216
2022	MIT+	Flow	2d, 4d	U(1), QCD	Equivariant	Yes	Yes	arXiv:2202.11712 +
2022	AT+	Flow	2d, 3d	Scalar		Yrs		

up to 2022

What is conv. neural networks?

The convolution layer can treat a translation transformation

Filter on image



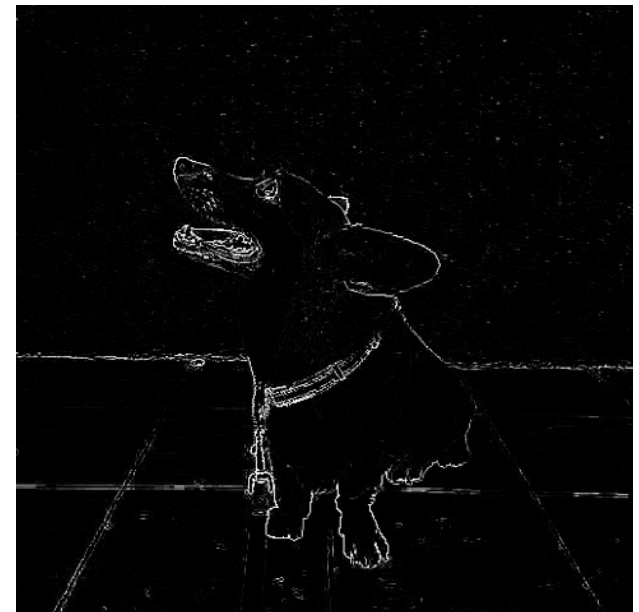
$*$

Laplacian filter

0	1	0
1	-2	1
0	1	0

(Discretization of ∂^2)

$=$



Edge detection

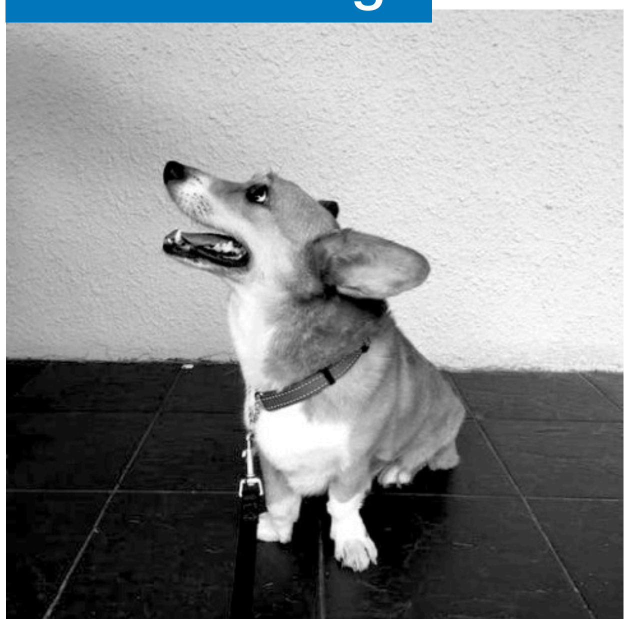
IMPORTANT: If inputs are shifted to right, outputs are shifted to right

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

What is conv. neural networks?

Convolution layer = trainable filter

Filter on image



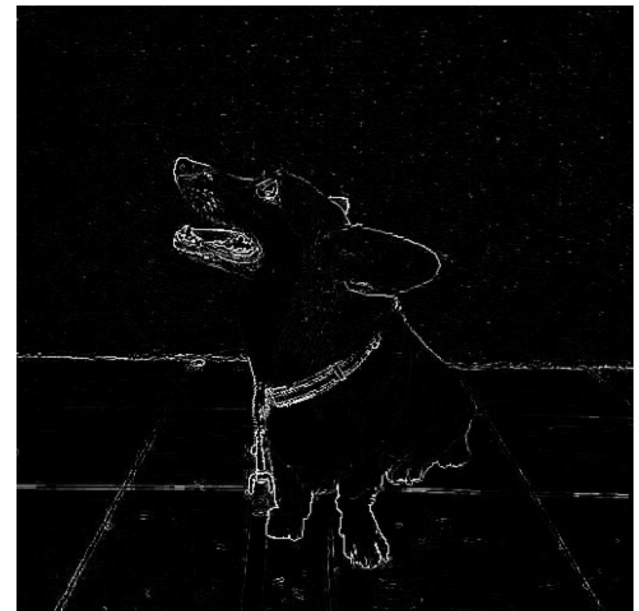
$*$

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!

Convolution neural network

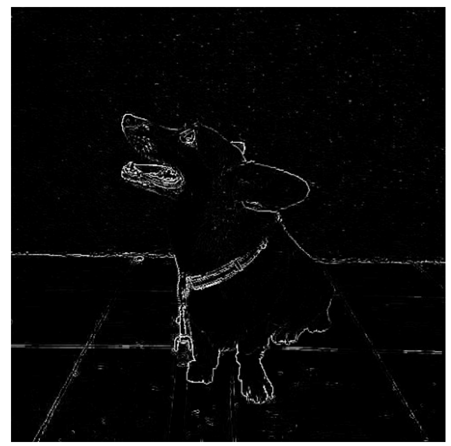
Akio Tomiya

Training can be done with back propagation

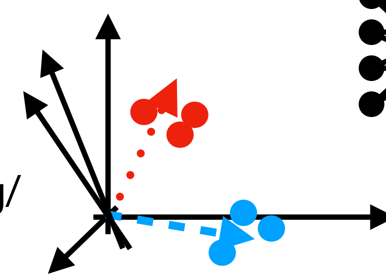
loss function quantifies error of output

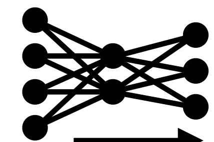
w_{11}	w_{12}	w_{13}
w_{21}	w_{22}	w_{23}
w_{31}	w_{32}	w_{33}

Translation *equivariant* map with trainable parameters

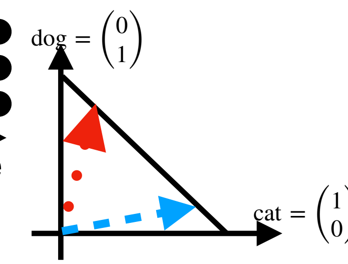


G.A. Pooling/ flatten





Dense net



feed L

Feedback = training (Steepest descent)

Smearing

Smoothing improves global properties

Eg.

Coarse image

Numerical derivative is unstable

Gaussian filter

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 1 \\ 1 & 2 & 1 \end{bmatrix} \longrightarrow$$

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

Smoothing with gauge symmetry, APE type

Akio Tomiya

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

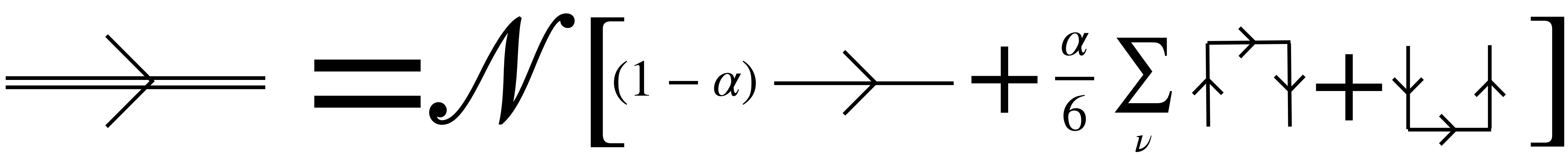
$$\mathcal{N}[M] = \frac{M}{\sqrt{M^\dagger M}}$$

Or projection

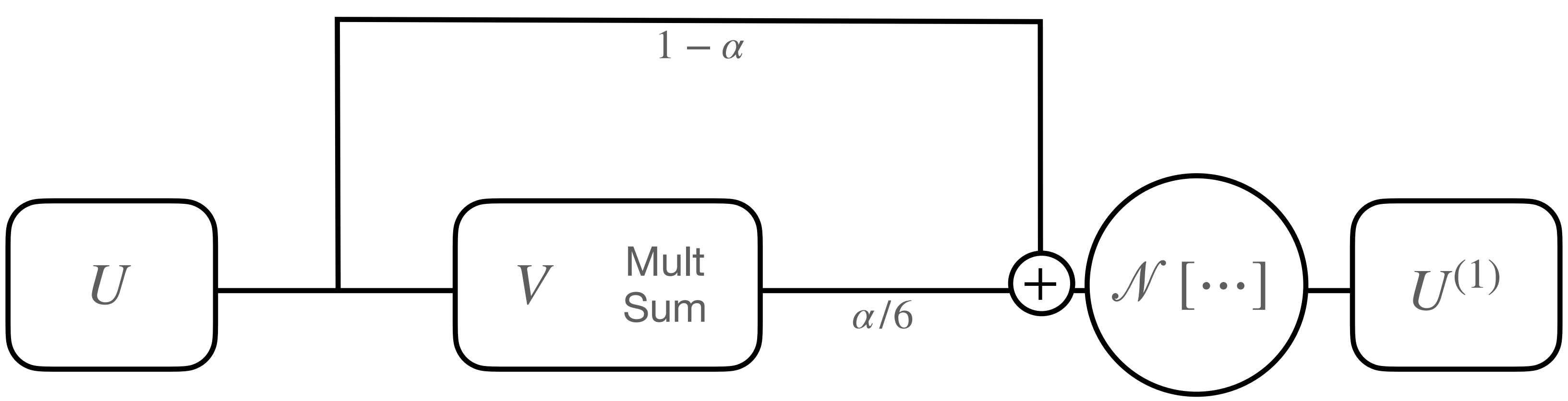
$$V_\mu^\dagger[U](n) = \sum_{\mu\neq\nu} U_\nu(n)U_\mu(n+\hat{\nu})U_\nu^\dagger(n+\hat{\mu}) + \cdots$$

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

Schematically,



In the calculation graph,



Smearing is a gauge covariant map

Gauge covariant neural network
= trainable smearing

Akio Tomiya

AT Y. Nagai arXiv: 2103.11965

<u>Smearing</u> = gauge covariant way of transform gauge configurations

Covariant sum

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

staple

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

Normalization

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

Or projection

<u>Gauge covariant neural network</u> = general smearing with <u>tunable parameters</u> w

$$\begin{cases} z_\mu^{(l)}(n) = w_1^{(l)} U_\mu^{(l-1)}(n) + w_2^{(l)} \mathcal{G}_{\bar{\theta}}^{(l)}[U] \\ \mathcal{N}(z_\mu^{(l)}(n)) \end{cases}$$

Train (tune, fitting)

point-wise (local)

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

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

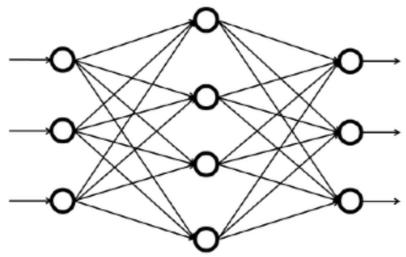
There are several realization of gauge covariant maps https://arxiv.org/abs/2305.02402

Gauge covariant neural network

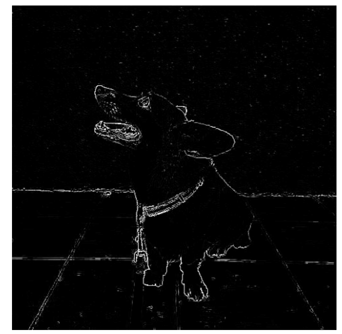
Schematic illustrations for neural networks (NN)

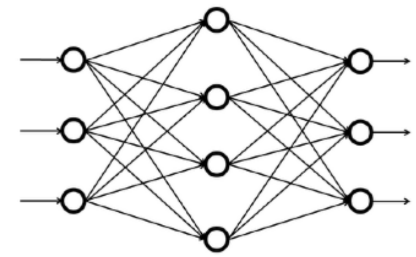
Neural networks for images

Parametrized function



Convolution
Parameters θ





Dense NN
Parameters θ

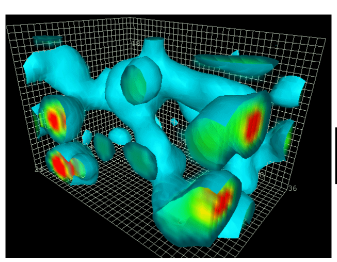
Cat $\longleftrightarrow$ **Ans Dog**

Tune by backprop (train)

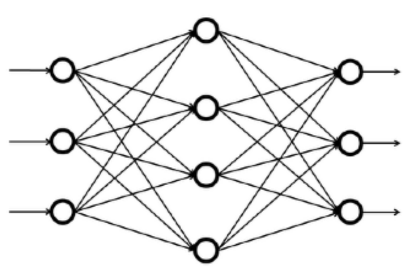
Neural networks for gauge configurations

Parametrized function

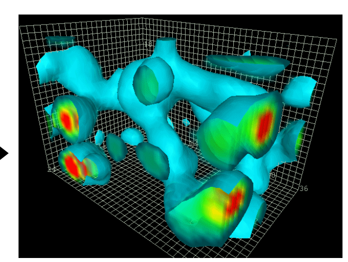
AT Nagai 2103.11965



U



Covariant NN
(Tunable smearing)
Parameters θ



$U^{\mathrm{NN}}[U]$

Wilson loop
Dirac op.
(Functional
of configurations)

$S_{\overline{D}}[U^{\mathrm{NN}}[U]] \longleftrightarrow$ **"Ans"** $S_D[U]$

Tune by backprop (train)

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

Akio Tomiya

Gauge covariant neural network

= trainable smearing

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)

Application for the staggered in 4d

Toy application

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

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

Construction of target action

$$U \mapsto D[U] \mapsto \underline{S_{\mathrm{f}}} = \phi^\dagger \left(D^\dagger D(m)\right)^{-1} \phi \Big|_{m=0.3}$$

Compare

Construction of Action in MD

$$U \mapsto U_\theta^{\mathrm{NN}}[U] \mapsto D\left[U_\theta^{\mathrm{NN}}[U]\right] \mapsto \underline{S_\theta} = \phi^\dagger \left(D^\dagger D(m)\right)^{-1} \phi \Big|_{m=0.4}$$

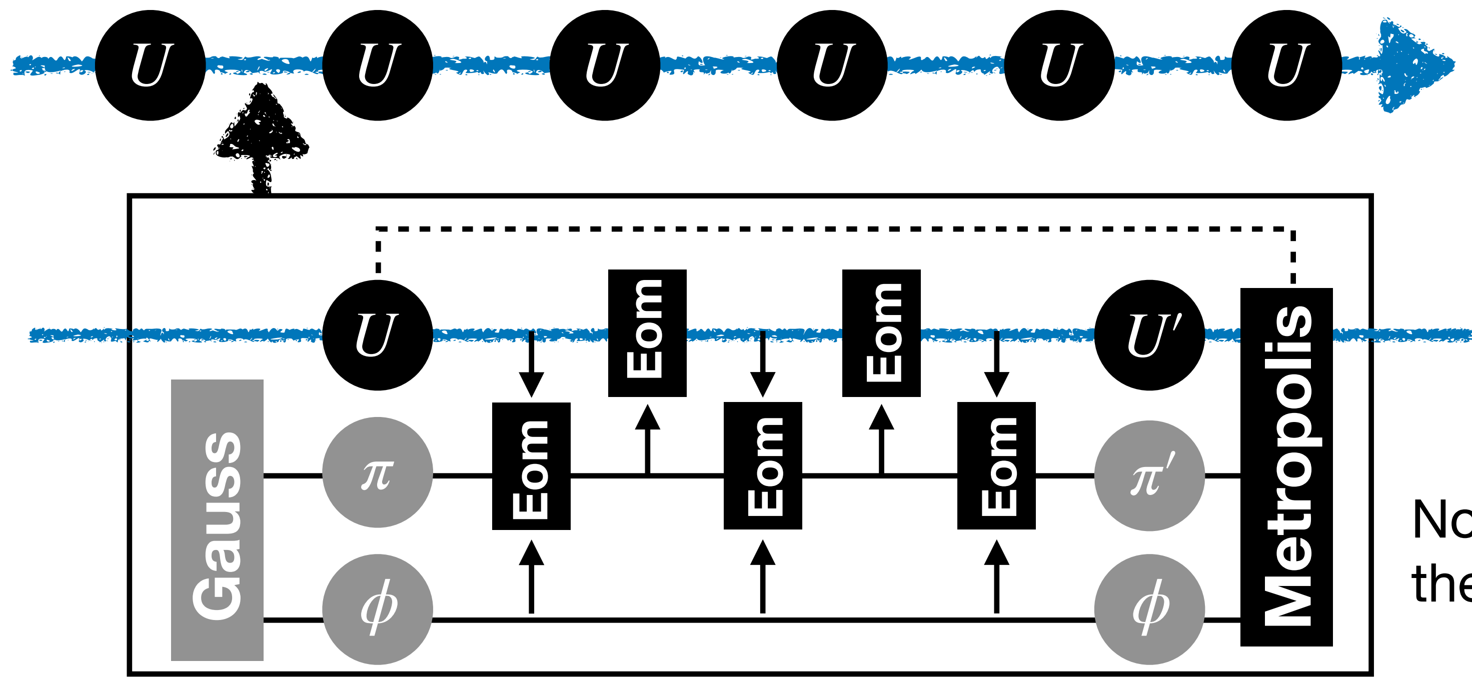
SLHMC = Exact algorithm with ML

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

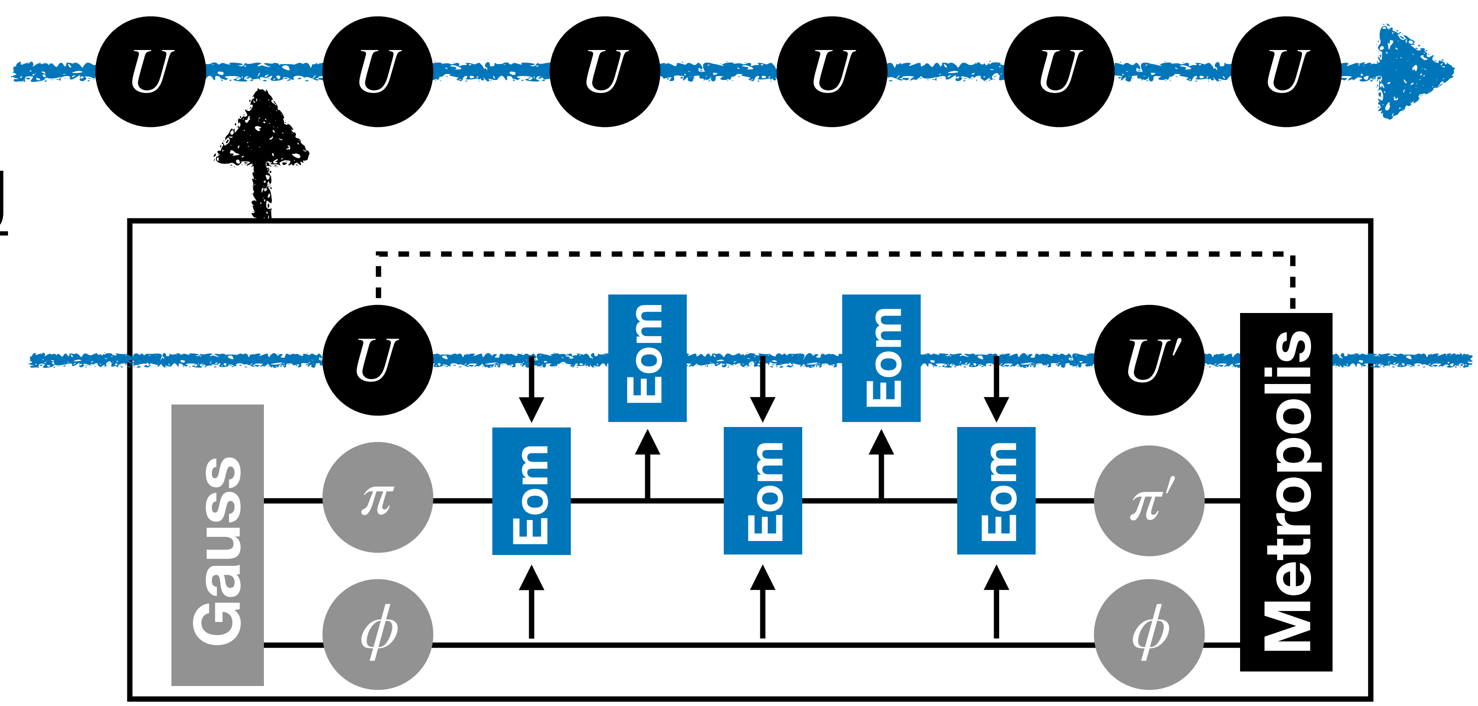


Eom **Metropolis**
Both use

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

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

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 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:

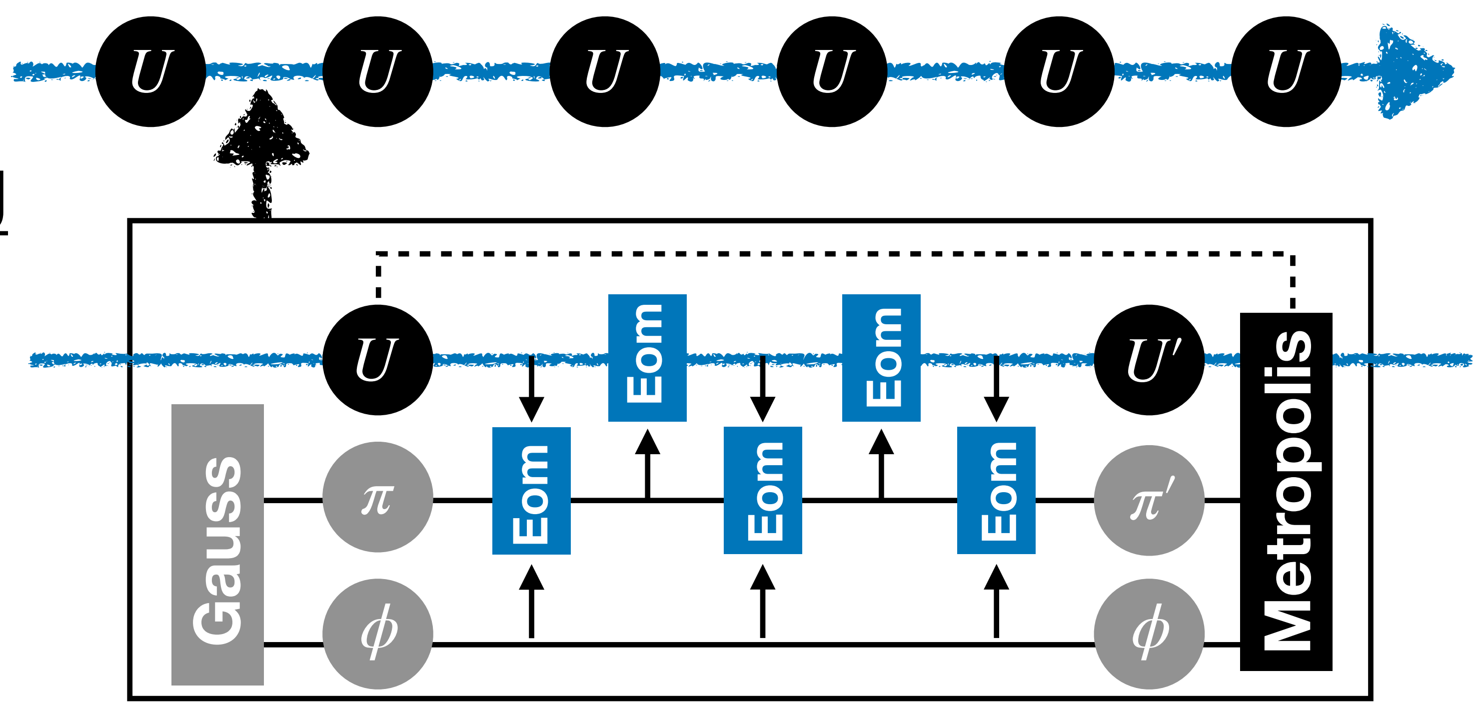
(Final target: Domain-wall vs overlap)

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

<u>**Self Learning HMC**</u>



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>

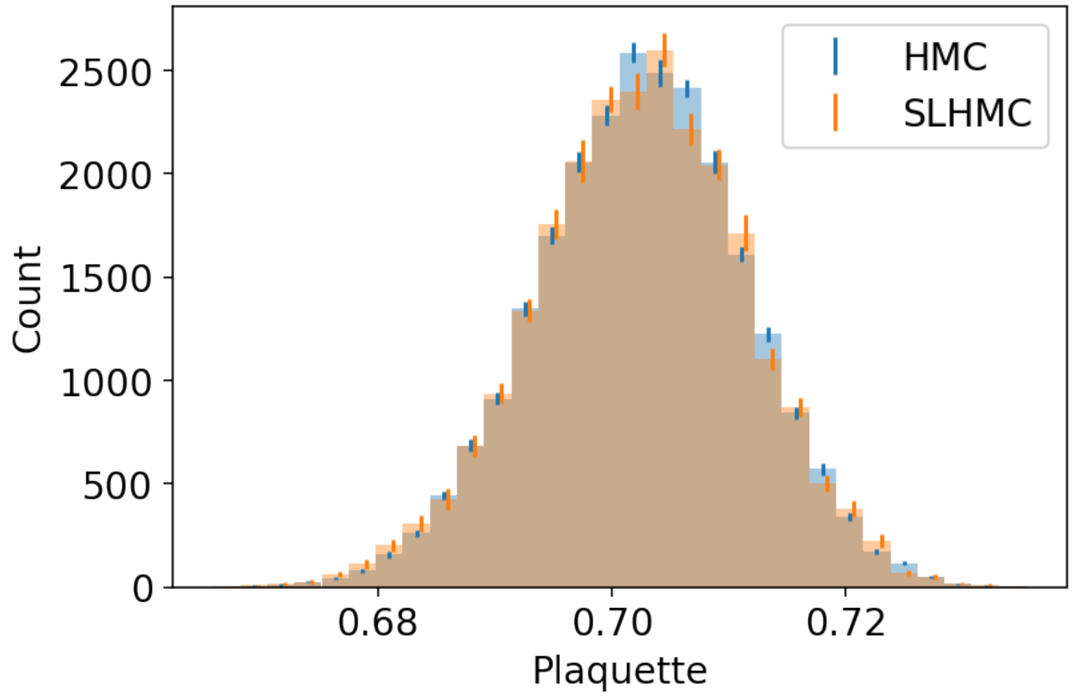
SLHMC works as an adaptive reweighting!

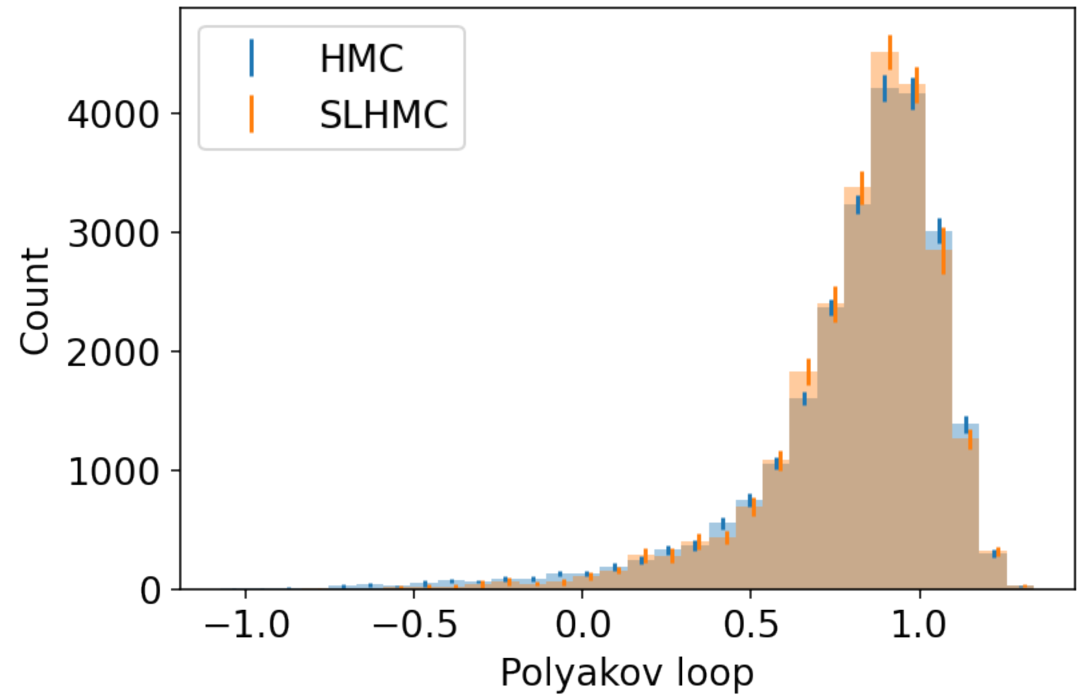
Akio Tomiya

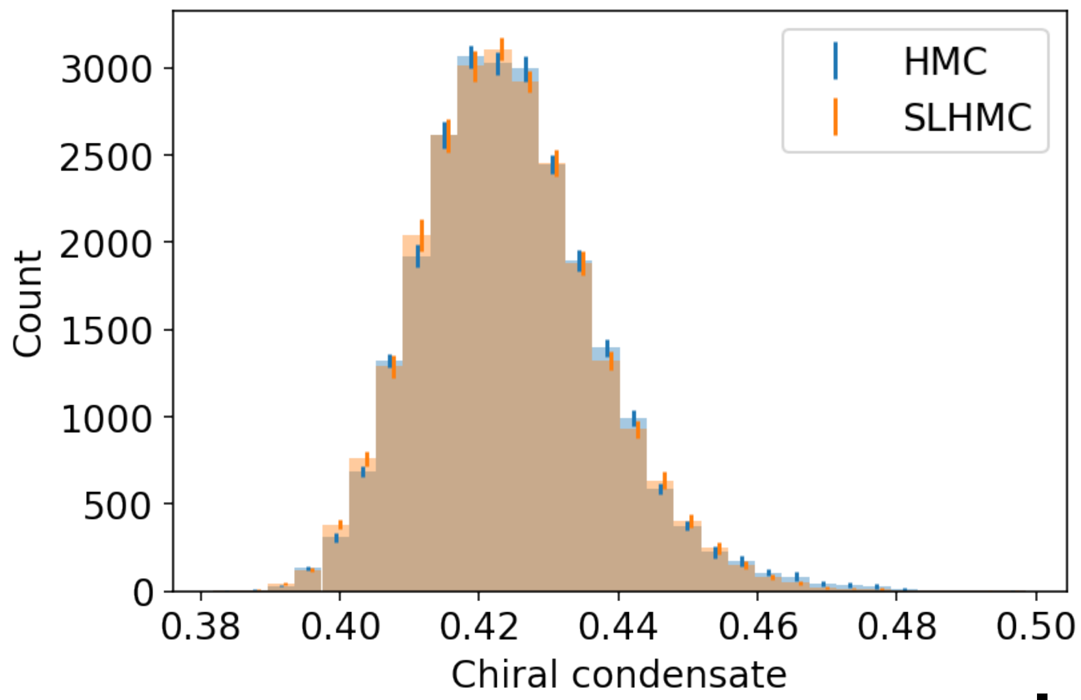
Application for the staggered in 4d

Results are consistent with each other

arXiv: 2103.11965







Expectation value

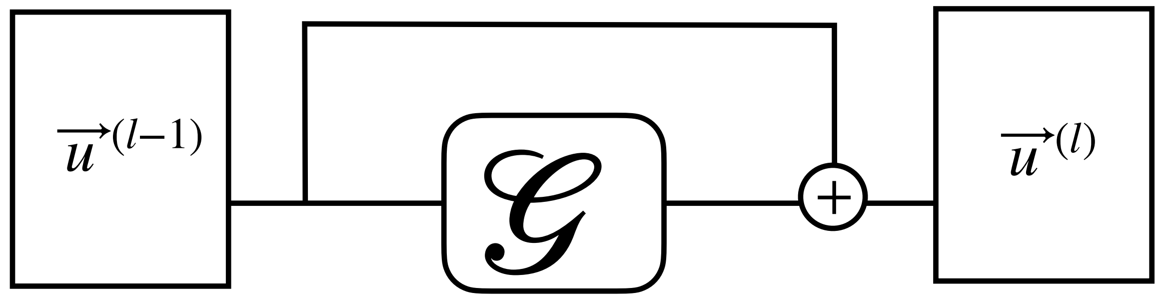
Algorithm	Observable	Value
HMC	Plaquette	0.7025(1)
SLHMC	Plaquette	0.7023(2)
HMC	|Polyakov loop|	0.82(1)
SLHMC	|Polyakov loop|	0.83(1)
HMC	Chiral condensate	0.4245(5)
SLHMC	Chiral condensate	0.4241(5)

Implemented by LatticeQCD.jl julia

Gauge covariant neural network

Neural ODE of Cov-Net = “gradient flow”

ResNet



arXiv: 1512.03385

Continuum
Layer
Limit

Neural ODE

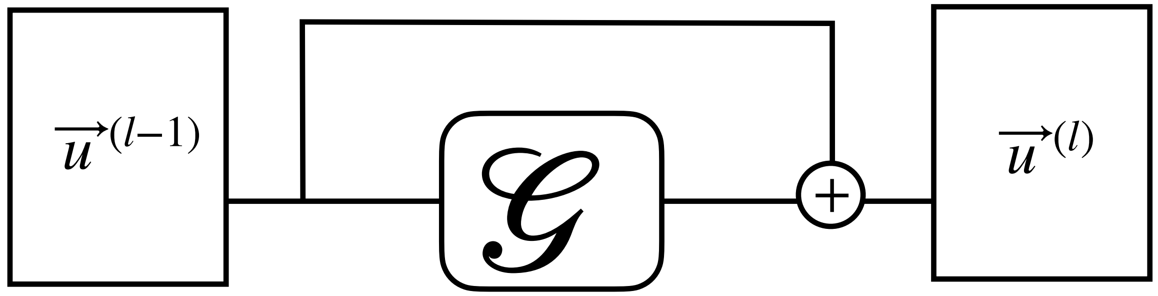
$$\frac{d\vec{u}^{(t)}}{dt} = \mathcal{G}(\vec{u}^{(t)})$$

arXiv: 1806.07366
(Neural IPS 2018 best paper)

Gauge covariant neural network

Neural ODE of Cov-Net = “gradient flow”

ResNet



arXiv: 1512.03385

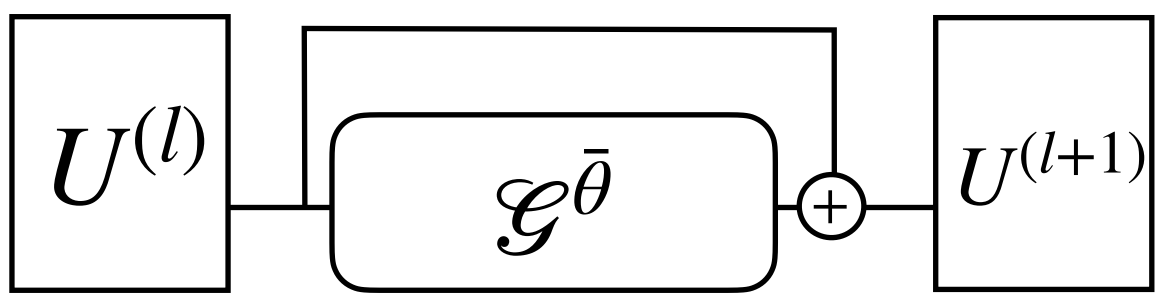
Continuum
Layer
Limit

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

Package structure

Akio Tomiya

Our lattice QCD codes are constructed by following repositories

Dependency (Automatically solved)

LatticeQCD.jl

QCDMeasurements.jl

LatticeDiracOperators.jl

Gaugefields.jl

Wilsonloop.jl | CLIME_jll

Wrapper for LatticeDiracOperators.jl & Gaugefields.jl, QCDMeasurements.jl
- Wizard for parameter files
- HMC/RHMC for SU(Nc)
  - Stout + Wilson/Staggered/DW
- Heatbath for SU(Nc)
- Measurements
- Jupyter, Colab/**PC/Supercomputers**

etc

Measurements in LQCD (Correlator, Flow, Qtop, etc)

Fermions (+HMC), Wilson, KS, DW, MPI
PC/Supercomputers

Gauge fields (+HMC/Heatbath), MPI
PC/Supercomputers

ILDG I/O

Symbolic operations of Wilson/Polyakov loops

See https://github.com/akio-tomiya/LatticeQCD.jl in detail

Benchmark of Julia + QCD

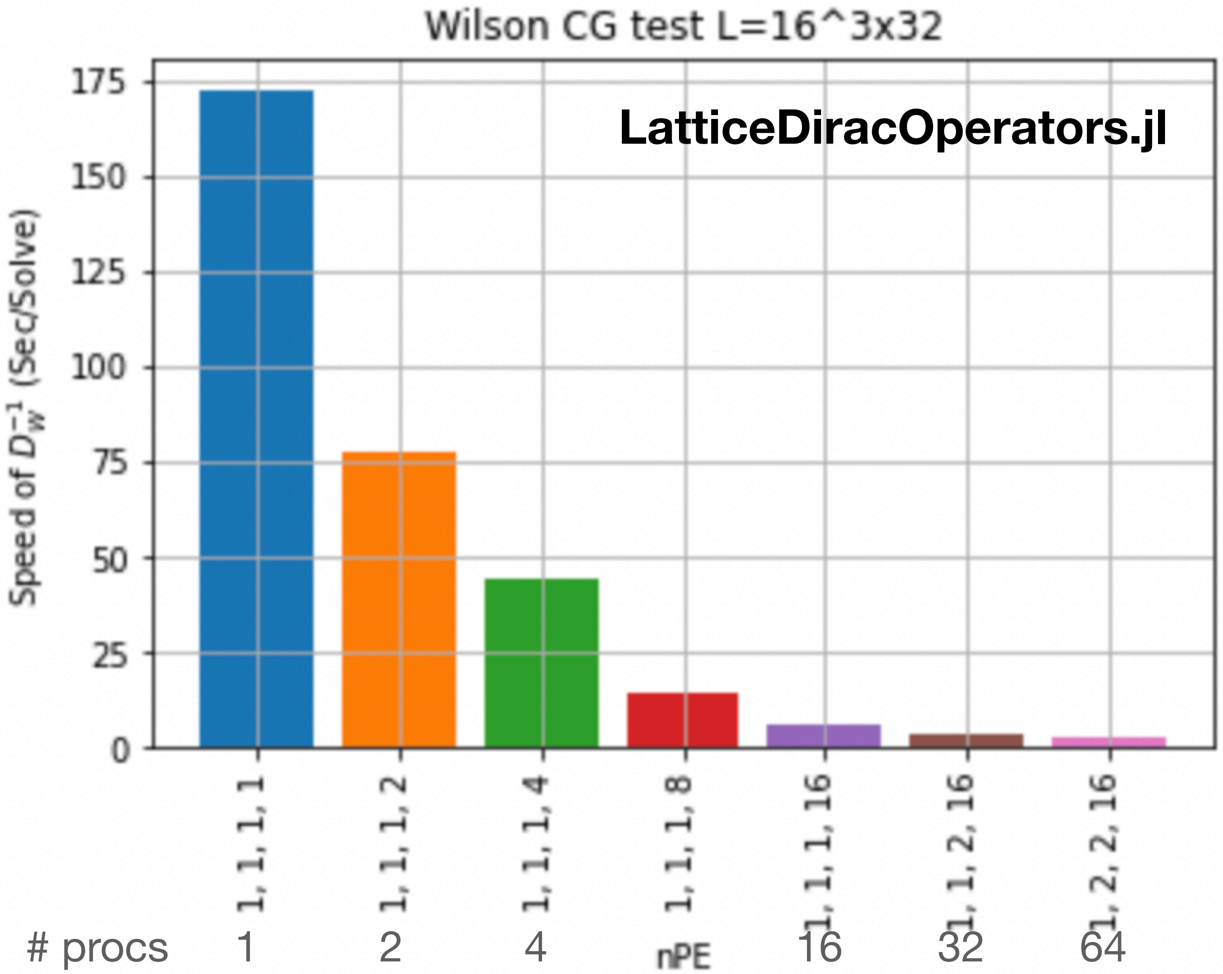
Wilson inversion / MPI parallel, Strong Scaling

Akio Tomiya

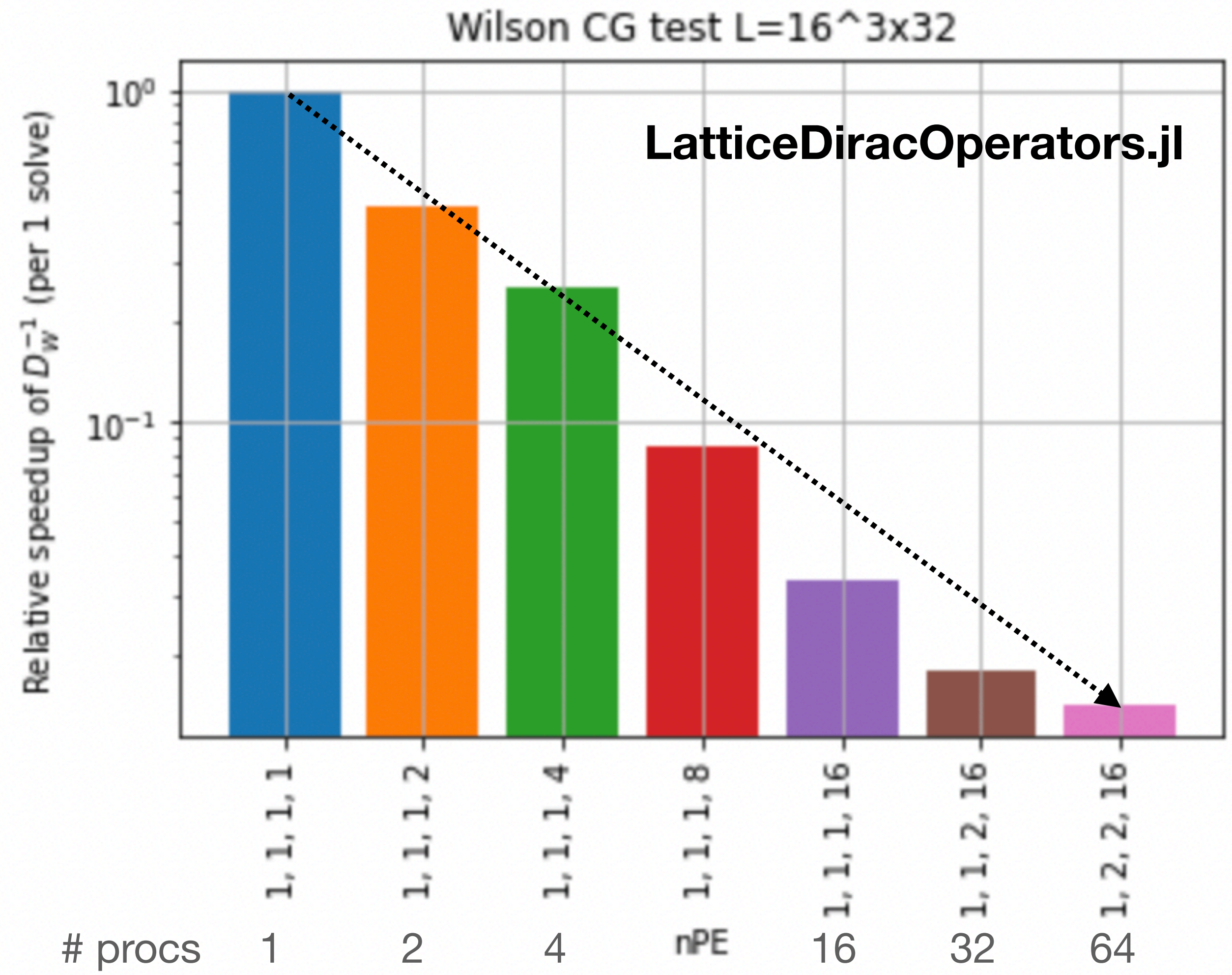
AT & Y. Nagai in prep

Tested on Yukawa-21@YITP

Absolute execution time



Relative speed up



It looks scaling well

We need more contributors!
Please help us

We thank to H. Ohno & Issaku Kanamori

Benchmark

Code comparison

Akio Tomiya

```julia
using Random

function main()
T = 10
K = 10^4
N = 12
#
A = zeros(ComplexF64, (N,N))
V = zeros(ComplexF64, N)
W = zeros(ComplexF64, N)

function myprod(A,V,W)
    for k = 1:N
        for i = 1:N
            W[i] +=  A[i, k]*V[k]
        end
    end
end
…(cut)…
```

Attached in backup

```c
#include <stdio.h>
#include <complex.h>
#include <math.h>
#include <time.h>
#include <stdlib.h>

#define T 10
#define K 10000
#define N 12

…(cut)…
void myprod(double complex A[N][N], double complex *V,
 double complex *W) {
  for (int k = 0; k < N; k++) {
    for (int i = 0; i < N; i++) {
      W[i] += V[k] * A[k][i];
    }
  }
}
…(cut)…
```

Attached in backup

- Complex matrix (12x12) times complex vector (d=12)
    - One set= 10^4 times, and repeated 10 times and averaged
- Code of Julia looks like Python (short, simple) but fast as C
  Julia: 0.0014 (sec), C: 0.0033 (sec). Single core performance is similar

I thank Taku Izubuchi

Benchmark

Why Julia? (My personal opinion)

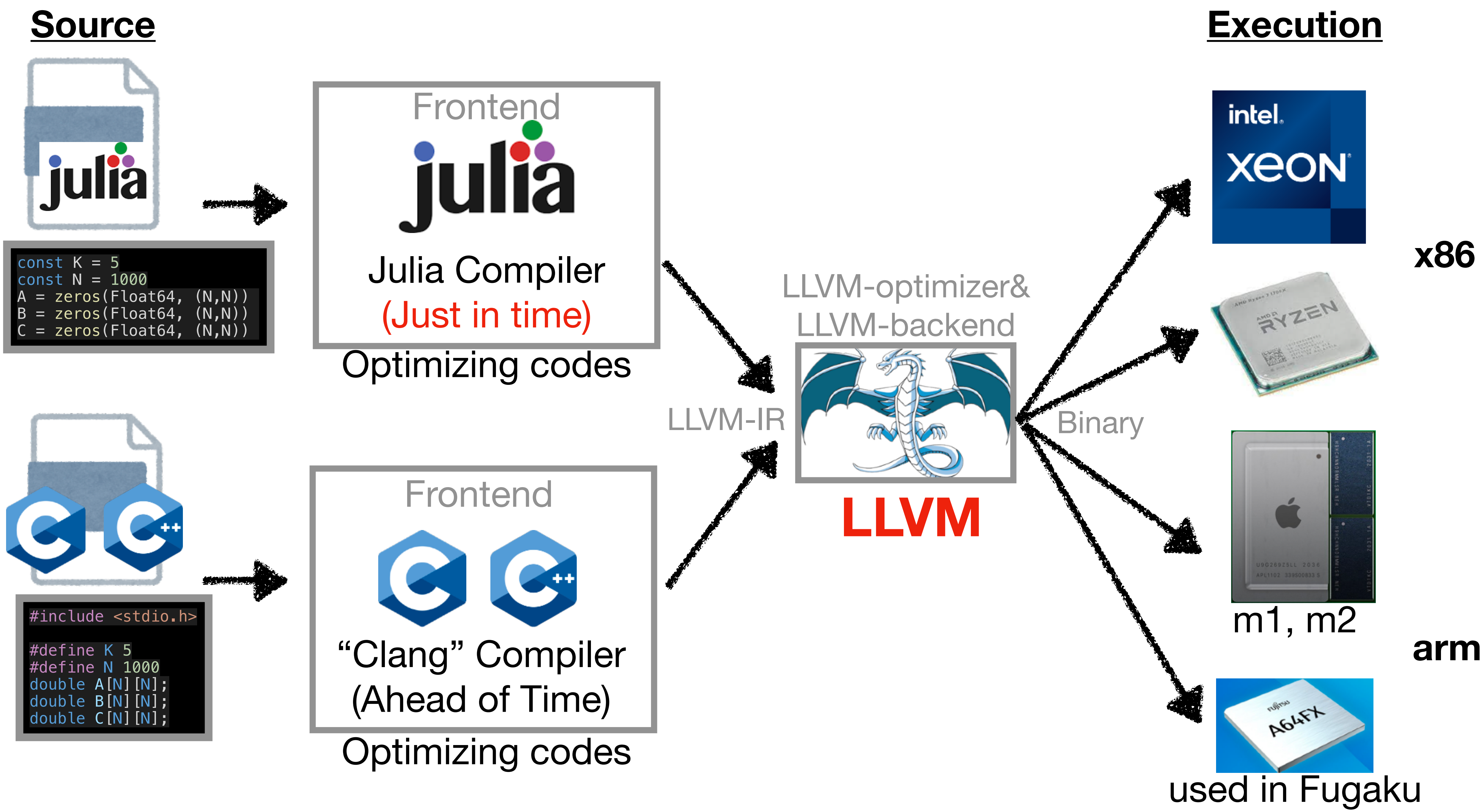
[1] https://akio-tomiya.github.io/julia_in_physics/
[2] https://qr.ae/prgSG5

- Modern scientific programming language
- Easy to make codes. Fast as C/C++ (Julia& C use LLVM)
- Fewer compiling/dependency issues.
- Many people are potentially interested in. (More than 400 people registered to “Julia in physics 2022 online workshop” [1]). 4,923 public repo on Github
- No two Language problem. “The fact that while the users are programming in a high-level language such as R and Python, the performance-critical parts have to be rewritten in C/C++ for performance”. [2]
    - Neural network friendly (Flux.jl). Tensor networks also (iTensor.jl).
- Works on/with
    - Xeon, Radeon/Apple silicon/A64FX
    - MPI, GPU

Akio Tomiya

LLVM?

LLVM = common backend for making binaries on multiple architectures



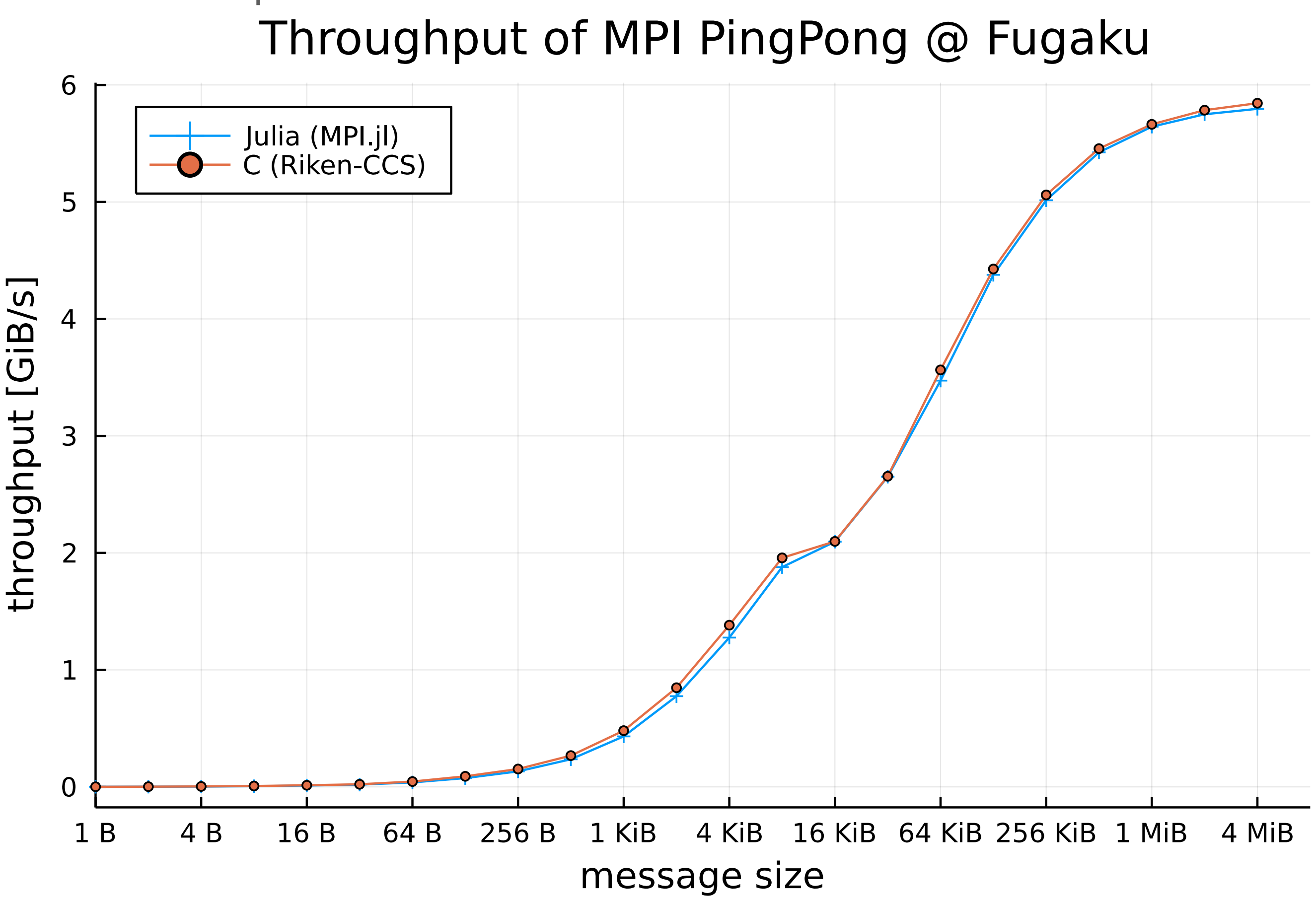
https://www.fujitsu.com/jp/about/businesspolicy/tech/fugaku/
https://ja.wikipedia.org/wiki/Apple_M1
https://ja.wikipedia.org/wiki/Ryzen
https://ja.wikipedia.org/wiki/Xeon
https://gigazine.net/news/20200623-japan-fugaku-fastest-supercomputer/
See: https://en.wikipedia.org/wiki/LLVM and related pages

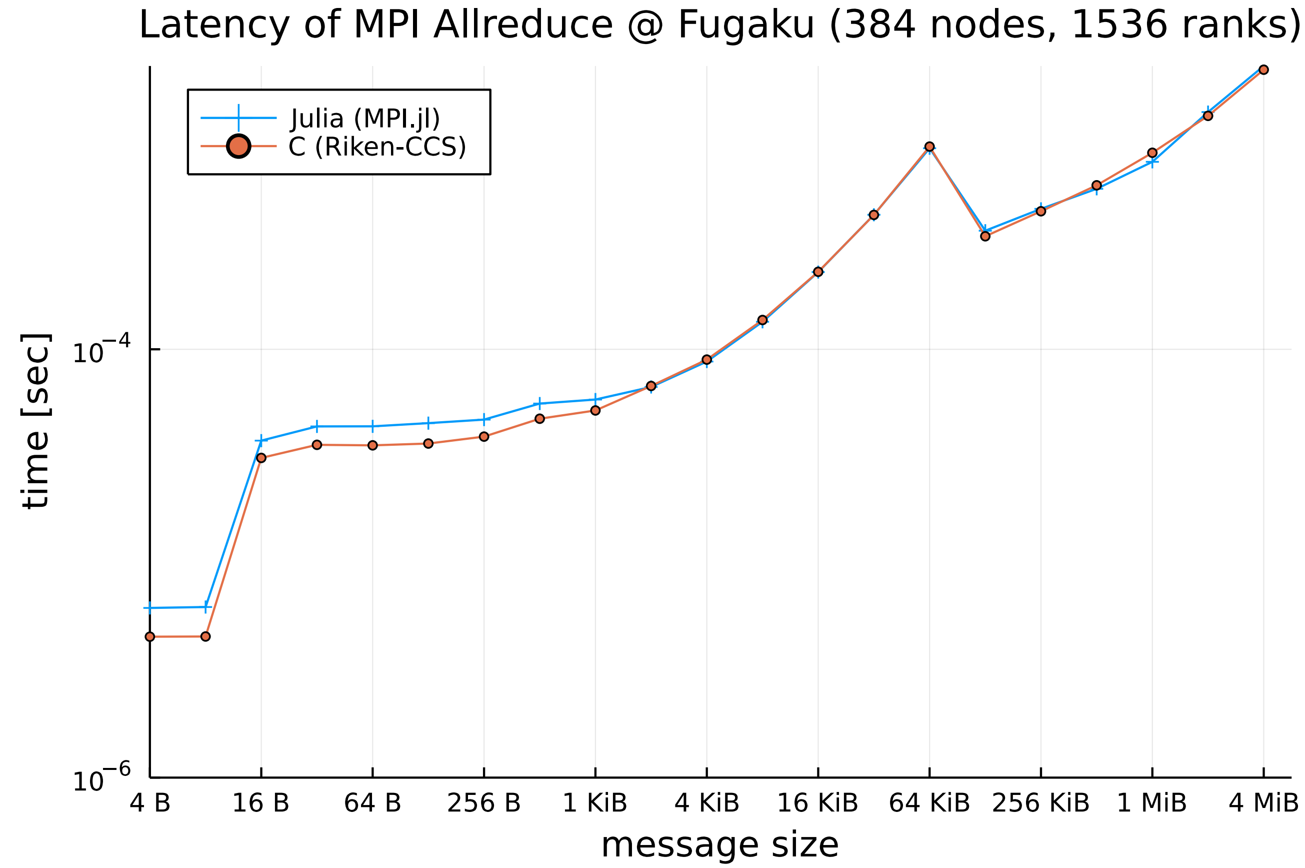
Julia is ready on Fugaku(?)
Parallelization with A64FX/Fugaku

M. Giordano, arXiv:2207.12762v1 [cs.DC] 26 Jul 2022

Tests of MPI + julia on Fugaku

Send-Recv performance





julia has similar scaling of MPI with C
(no obvious overhead)

Machine Learning (ML) is a branch of artificial intelligence (AI) that utilizes algorithms and statistical models to allow computers to perform specific tasks without explicit instructions. In simpler terms, it's teaching machines to learn patterns from data and make intelligent decisions.

ML is typically categorized into three main types: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning is when the algorithm learns from labeled data to predict outcomes for unseen data. Unsupervised learning is when the algorithm identifies patterns in unlabeled data. Reinforcement learning is a process in which the algorithm learns to make decisions by interacting with an environment where it receives rewards or penalties.

At the heart of ML is the concept of learning from experience (E) with respect to some task (T) and performance measure (P), a machine is said to learn if its performance at tasks in T, as measured by P, improves with experience E.

Machine learning's potential applications are vast, including but not limited to natural language processing, image recognition, and predictive analytics. As a physicist, you may find its uses in pattern detection, prediction, and simulation in physical systems particularly intriguing.

Transformer and Attention

Akio Tomiya

Physically symmetric Attention layer

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

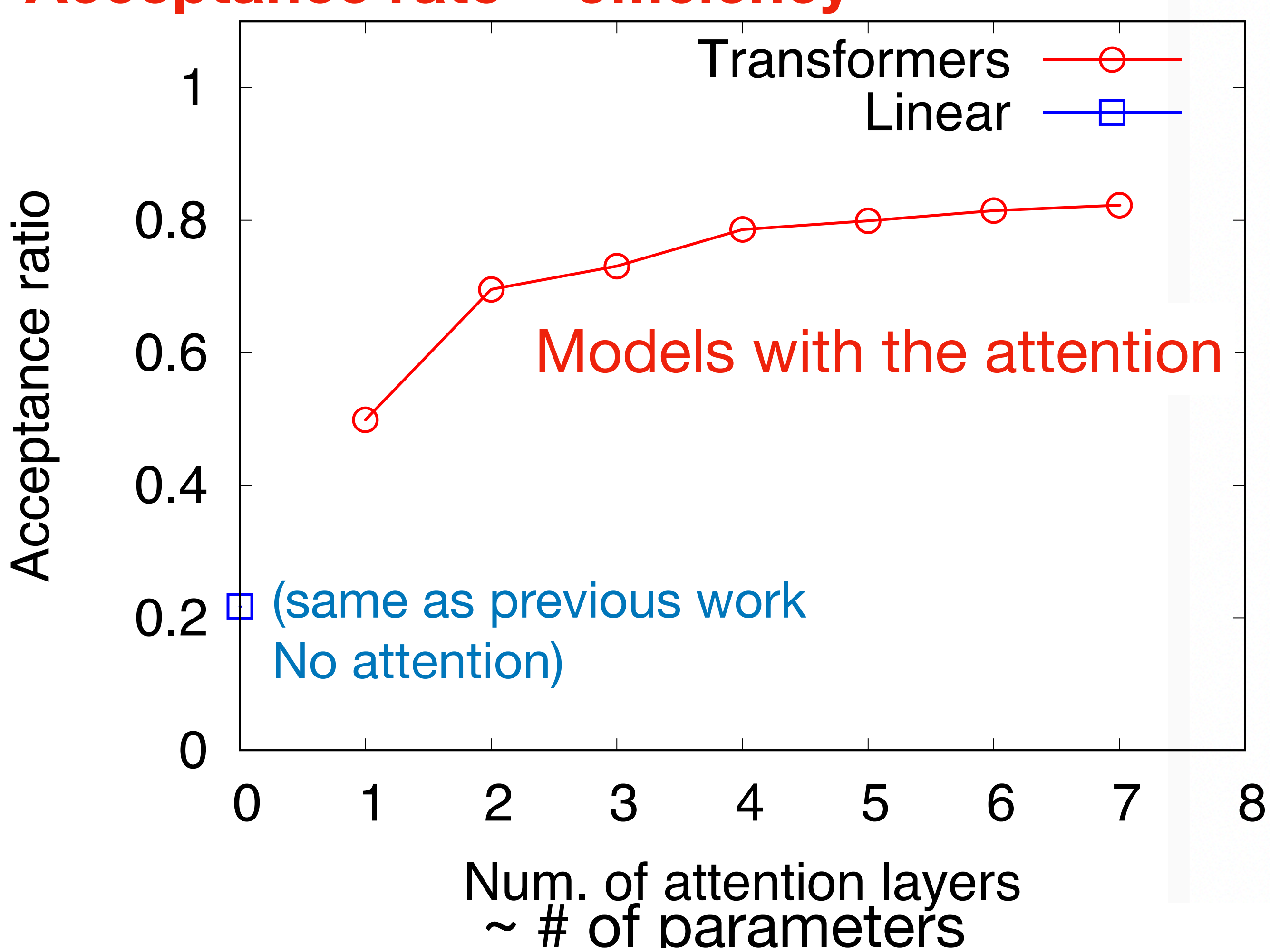
	Equivariance	**Capturable correlation**	**Data demmands**	**Applications**
Convolution ($\in$ equivariant layers)	Yes 👍	Local 😲	Low 👍	Image recognition VAE, GAN Normalizing flow
Standard Attention layer	No 😲	Global 👍	Huge 😲	ChatGPT GEMINI Vision Transformer arXiv:1706.03762
Physically *Equivariant* attention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

Transformer and Attention

Akio Tomiya
arXiv: 2306.11527 + update

Application to O(3) spin model with fermions

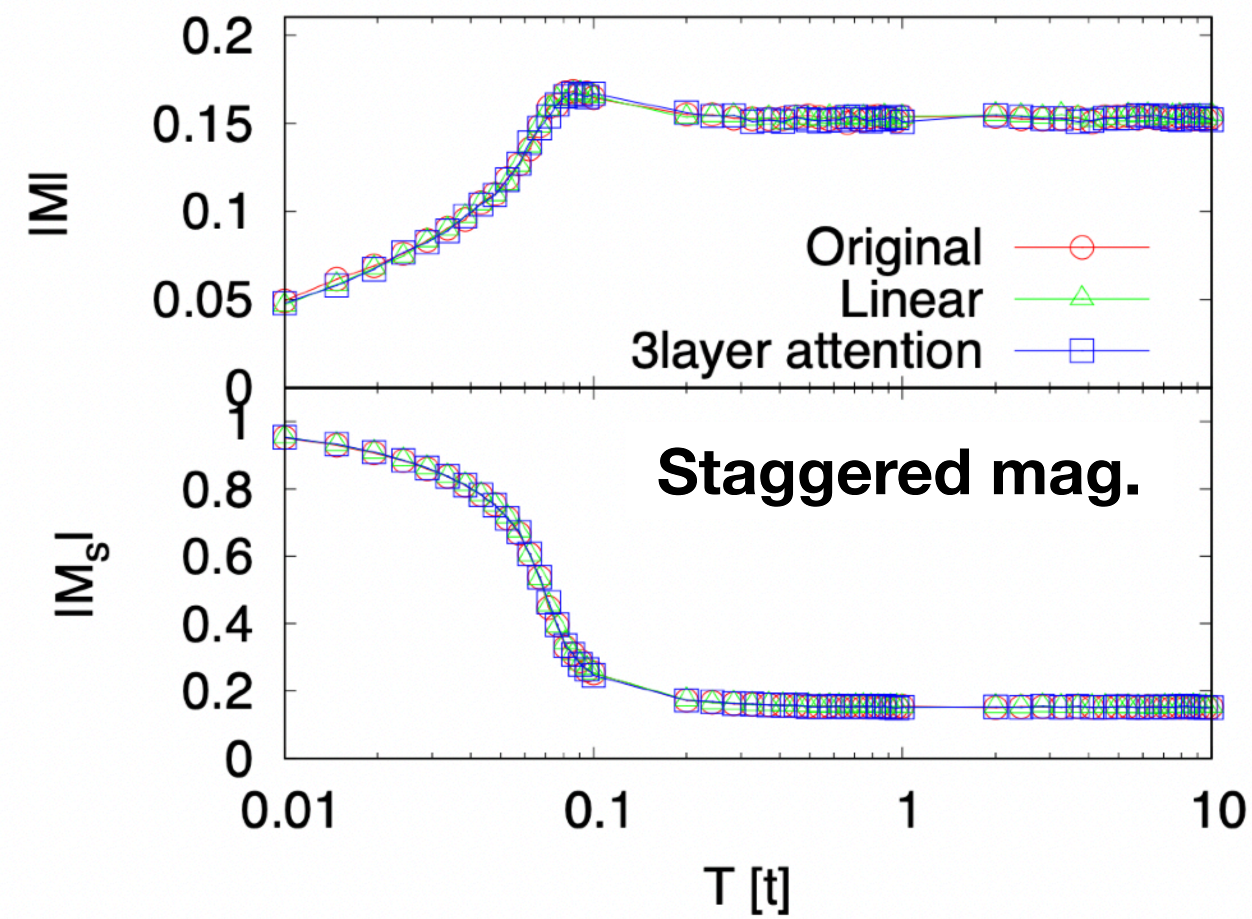
Acceptance rate ~ efficiency



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)

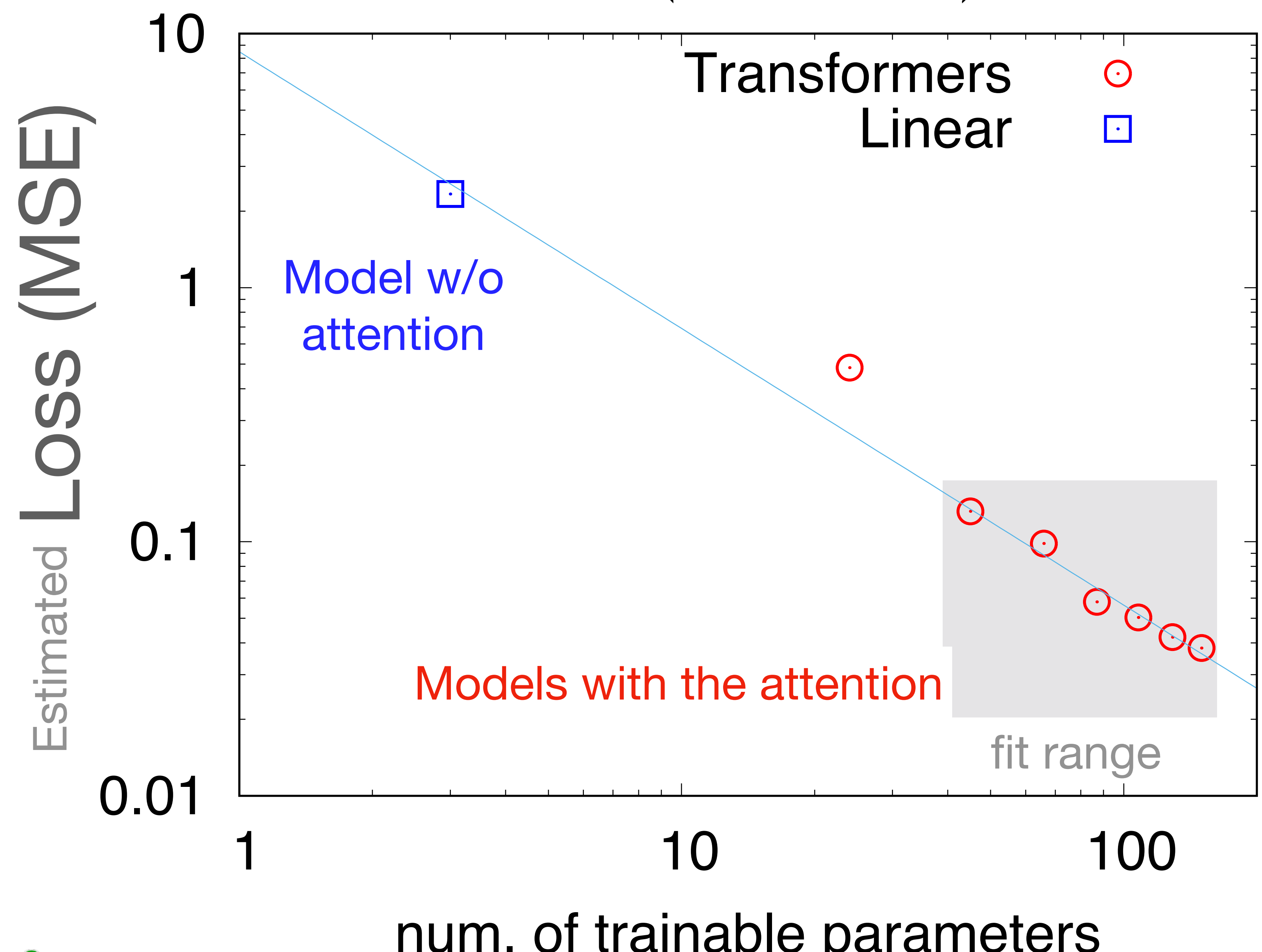
Transformer and Attention

Akio Tomiya

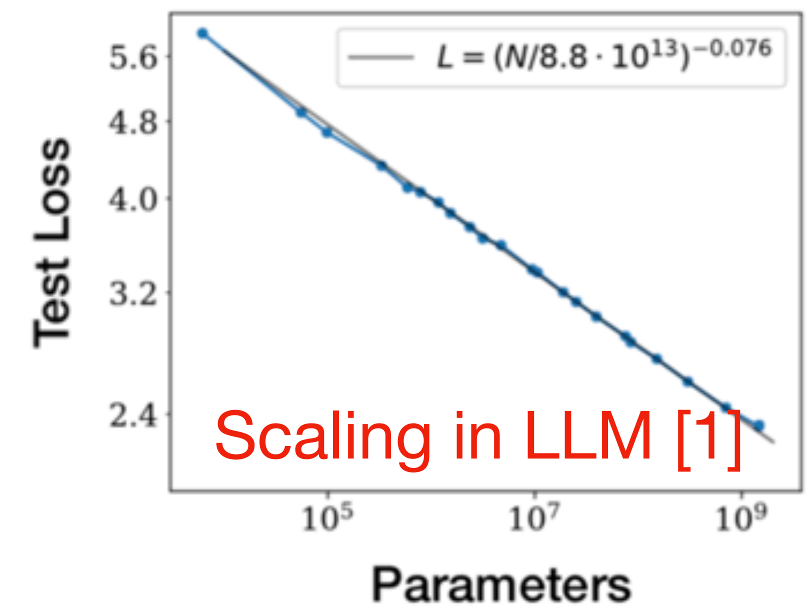
Loss function shows Power-type scaling law as LLM

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)