MLPhYs 学術変革領域研究(A) 学習物理学の創成
Foundation of "Machine Learning Physics"

Program for Promoting Researches on the Supercomputer Fugaku

Large-scale lattice QCD simulation and development of AI technology

Self-learning Monte Carlo method with equivariant transformer

CCSE, Japan Atomic Energy Agency Yuki Nagai

Collaborators

CCSE, Japan Atomic Energy Agency

Masahiko Okumura, Keita Kobayashi, Motoyuki Shiga

YN, M. Okumura, K. Kobayashi, and M. Shiga,
"Self-learning Hybrid Monte Carlo: A First-principles Approach",
Phys. Rev. B 102, 041124(R) (2020)

YN, M. Okumura, A. Tanaka
"Self-learning Monte Carlo method with Behler-Parrinello neural networks",
Phys. Rev. B 101, 115111 (2020)

K. Kobayashi, YN, M. Itakura, and M. Shiga, J. Chem. Phys. **155**, 034106 (2021)

Liang Fu's group in MIT

YN, H. Shen, Y. Qi, J. Liu, and L. Fu
"Self-learning Monte Carlo method: Continuous-time algorithm",
Physical Review B 96, 161102(R) (2017)
Editors' Suggestion

I was a visiting researcher in MIT from 2016 to 2017

Application to Lattice QCD

Akinori Tanaka — RIKEN AIP, RIKEN ITHEMS

Akio Tomiya — IPUT Osaka

Yuki Nagai, Akinori Tanaka, Akio Tomiya,
"Self-learning Monte-Carlo for non-abelian gauge theory with dynamical fermions",
Phys. Rev. D **107**, 054501 (2023)

YN and Akio Tomiya,
"Gauge covariant neural network for 4 dimensional non-abelian gauge theory",
arXiv:2103.11965

Application to quasicrystals

YN, Yutaka Iwasaki, Koichi Kitahara, Yoshiki Takagiwa, Kaoru Kimura, Motoyuki Shiga,
"Atomic diffusion due to hyperatomic fluctuation for quasicrystals"
"arXiv:2302.14441

Recent works

Application with Transformers

YN and A. Tomiya, "Self-learning Monte Carlo with equivariant Transformer", arXiv:2306.11527

Lattice QCD code for generic purpose

Open source LQCD code in Julia Language

LatticeQCD.jl

Akio Tomiya and YN

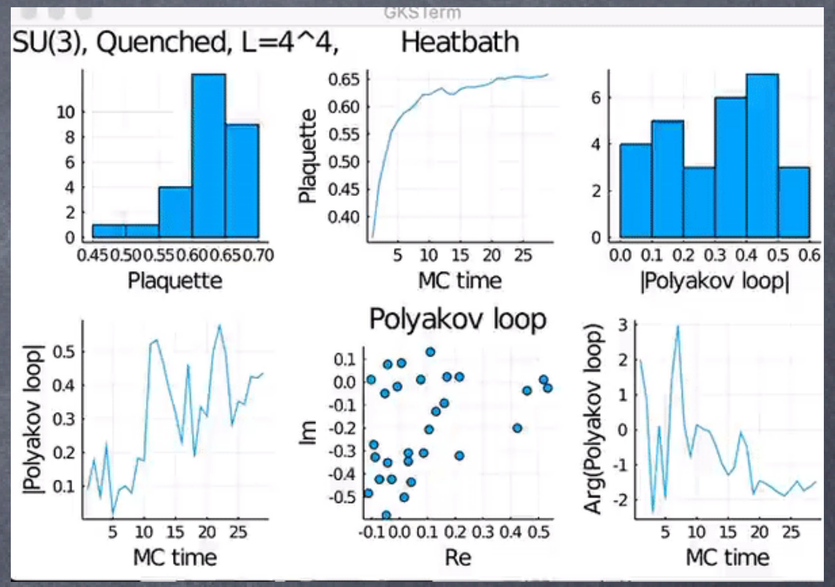
Machines: Laptop/desktop/Jupyter/Supercomputers

Functions: SU(Nc)-heatbath, (R)HMC, Self-learning HMC, SU(Nc) Stout
Dynamical Staggered, Dynamical Wilson, Dynamical Domain-wall
Measurements

Start LQCD in 5 min

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

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



LatticeQCD.jl
QCDMeasurements.jl
LatticeDiracOperators.jl
Gaugefields.jl
Wilsonloop.jl | CLIME_jll

We can use any gauge actions in HMC

Automatic differentiation technique using Wilsonloop.jl

Lattice QCD code for generic purpose

Open source LQCD code in Julia Language

LatticeQCD.jl

Akio Tomiya and YN

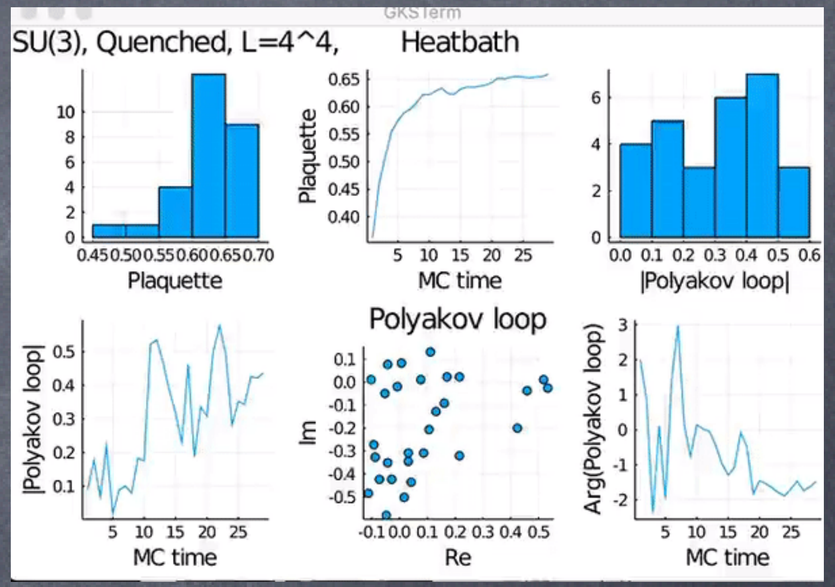
Machines: Laptop/desktop/Jupyter/Supercomputers

Functions: SU(Nc)-heatbath, (R)HMC, Self-learning HMC, SU(Nc) Stout
Dynamical Staggered, Dynamical Wilson, Dynamical Domain-wall
Measurements

Start LQCD in 5 min

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

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



LatticeQCD.jl
QCDMeasurements.jl
LatticeDiracOperators.jl
Gaugefields.jl
Wilsonloop.jl | CLIME_jll

We can use any gauge actions in HMC

Automatic differentiation technique using Wilsonloop.jl

Outline

- Introduction
- Transformer and Attention mechanism
- Equivariant Transformer
- Results

YN and A. Tomiya, "Self-learning Monte Carlo with equivariant Transformer", arXiv:2306.11527

Introduction

Self-learning Monte Carlo

We calculate a partition function $Z = \int \exp(-S)$ or $\Sigma \exp(-\beta H)$

With the use of Monte Carlo method, we can calculate physical variables

Sometimes, the computational cost is heavy.

Configurations → **Heavy tasks** → Boltzmann weight

Configurations → **effective model** → Boltzmann weight

Spins **Electrons** **Atoms, molecules** **Lattice QCD**

To propose a new configuration, we use the effective model

SLMC and SLHMC

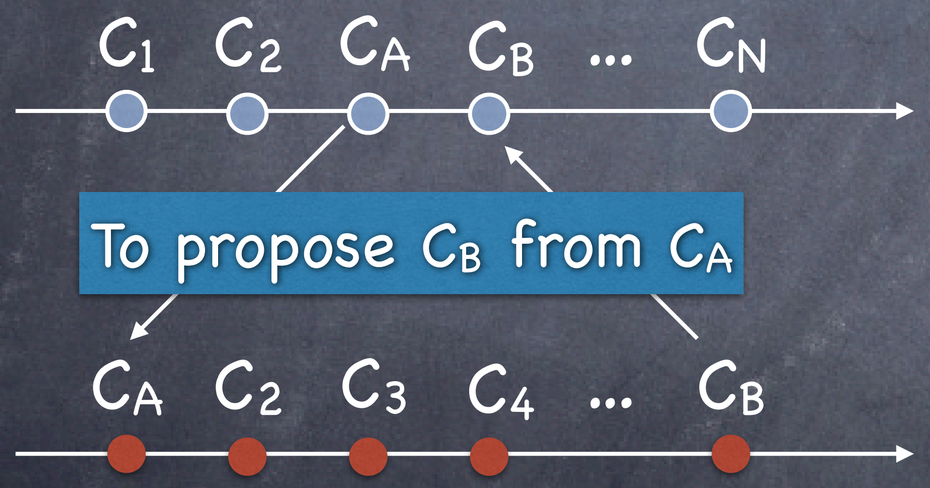
Self-learning Monte Carlo method (SLMC)

Self-learning Hybrid Monte Carlo method (SLHMC)

To speed up the Markov Chain Monte Carlo (MCMC) simulations

SLMC

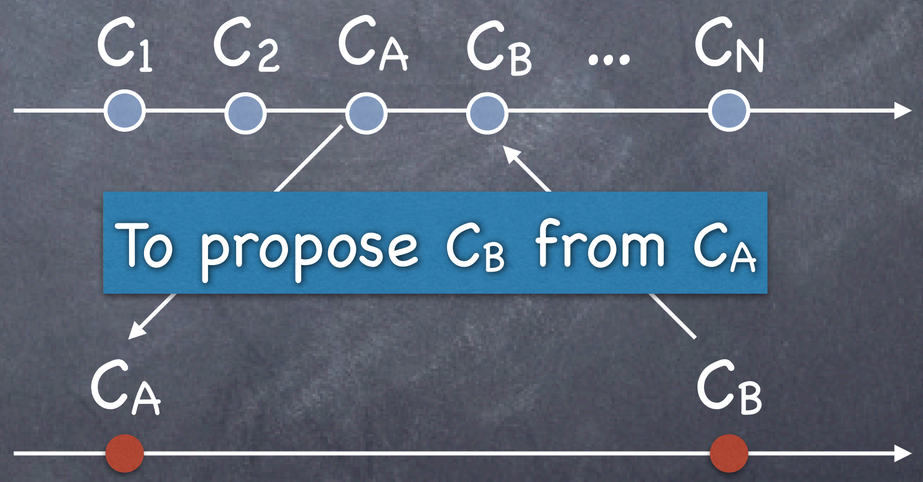
Markov chain with the probability W(C)



Another Markov chain with the probability W'(C)

SLHMC

Markov chain with the probability W(C)



Machine learning molecular dynamics

Machine learning techniques are used for proposing new configuration!

Self-learning Monte Carlo

Spin systems

J. Liu, Y. Qi, Z. Y. Meng, and L. Fu, Phys. Rev. B 95, 041101(R) (2017)

H. Kohshiro and YN,,
"Effective Ruderman–Kittel–Kasuya–Yosida-like Interaction in Diluted Double-exchange Model: Self-learning Monte Carlo Approach",
J. Phys. Soc. Jpn. 90, 034711 (2021)

YN and A. Tomiya, "Self-learning Monte Carlo with equivariant Transformer", arXiv:2306.11527

Fermion+classical spins

Electrons

YN, H. Shen, Y. Qi, J. Liu, and L. Fu
"Self-learning Monte Carlo method: Continuous-time algorithm",
Physical Review B 96, 161102(R) (2017) ***Editors' Suggestion***

YN, M. Okumura, A. Tanaka
"Self-learning Monte Carlo method with Behler-Parrinello neural networks",
Phys. Rev. B **101**, 115111 (2020)

Continuous time Quantum Monte Carlo

Atoms/molecules

Machine-learning MD

YN, M. Okumura, K. Kobayashi, and M. Shiga,
"Self-learning Hybrid Monte Carlo: A First-principles Approach",
Phys. Rev. B **102**, 041124(R) (2020)

K. Kobayashi, YN, M. Itakura, and M. Shiga,
"Self-learning hybrid Monte Carlo method for isothermal–isobaric ensemble: Application to liquid silica",
J. Chem. Phys. **155**, 034106 (2021)

YN, Yutaka Iwasaki, Koichi Kitahara, Yoshiki Takagiwa, Kaoru Kimura, Motoyuki Shiga, "Atomic diffusion due to hyperatomic fluctuation for quasicrystals"
"arXiv:2302.14441

Lattice QCD

SU(N) Gauge theory on the lattice

YN, Akinori Tanaka, Akio Tomiya,
"Self-learning Monte-Carlo for non-abelian gauge theory with dynamical fermions",
Phys. Rev. D 107, 054501 (2023)

YN and Akio Tomiya,
"Gauge covariant neural network for 4 dimensional non-abelian gauge theory",
arXiv:2103.11965

SLMC and SLHMC in lattice QCD

YN, Akinori Tanaka, Akio Tomiya,
"Self-learning Monte-Carlo for non-abelian gauge theory with dynamical fermions",
Phys. Rev. D 107, 054501 (2023)

SU(2) gauge fields with staggered fermions with 4 flavors in 4D

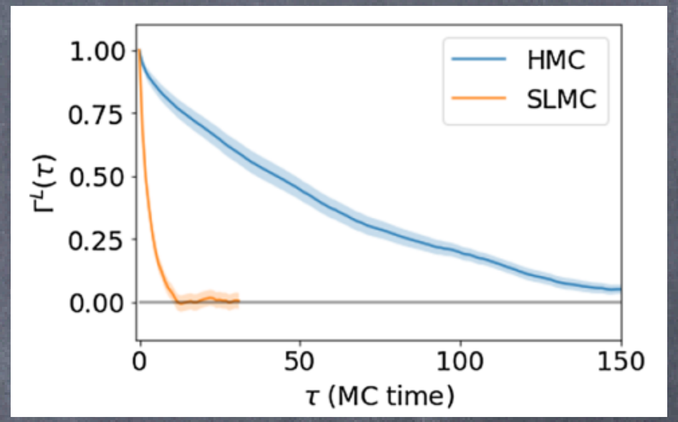
$$S[U] = S_g[U] + S_f[U],$$

$$S_f[U] = -\log \det M^\dagger M,$$

integrated fermion action

effective model without fermion actions

$$S^\theta_{\rm eff}[U] = \sum_n \left[\beta_{\rm plaq} \sum_{\mu=1}^{4} \sum_{\nu>\mu} \left(1 - \frac{1}{2}{\rm tr} U_{\mu\nu}(n)\right) + \beta_{\rm rect} \sum_{\mu=1}^{4} \sum_{\nu\neq\mu} \left(1 - \frac{1}{2}{\rm tr} R_{\mu\nu}(n)\right)\right]$$
$$+ \beta^{\mu=1}_{\rm Pol} \sum_{n_2,n_3,n_4} {\rm tr}\left[\prod_{n_1=0}^{N_1-1} U_1(\vec{n}, n_4)\right] + \beta^{\mu=2}_{\rm Pol} \sum_{n_1,n_3,n_4} {\rm tr}\left[\prod_{n_2=0}^{N_2-1} U_2(\vec{n}, n_4)\right]$$
$$+ \beta^{\mu=3}_{\rm Pol} \sum_{n_1,n_2,n_4} {\rm tr}\left[\prod_{n_3=0}^{N_3-1} U_3(\vec{n}, n_4)\right] + \beta^{\mu=4}_{\rm Pol} \sum_{n_1,n_2,n_3} {\rm tr}\left[\prod_{n_4=0}^{N_4-1} U_4(\vec{n}, n_4)\right] + \beta_{\rm const},$$

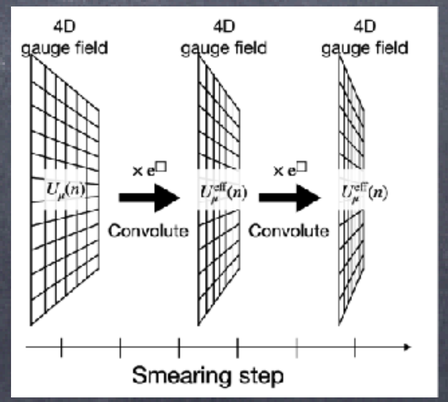


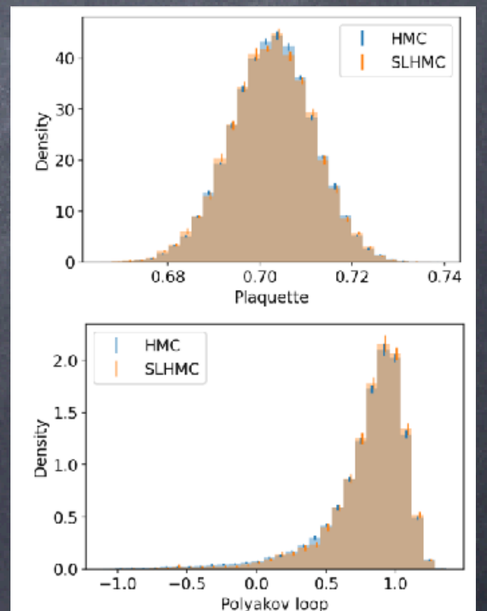
YN and Akio Tomiya,
"Gauge covariant neural network for 4 dimensional non-abelian gauge theory",
arXiv:2103.11965

effective model with different mass

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

$$S[U] = S_{\rm g}[U] + S_{\rm f}[\phi, U; m_{\rm l}],$$





$U^{\rm NN}$: trainable stout smearing

Problems of SLMC

Configurations → **Heavy tasks** → Boltzmann weight

Configurations → **effective model** → Boltzmann weight

How to construct effective models?

Quality of the effective model is very important

In previous studies,

the linear regression is used to construct the effective model inspired by the physical insight

Use Transformer!!

Transformer and Attention mechanism

Generative AIs

These AI have same architecture called Transformer

Transformer

AI Chat, Visualization, language translation

protein foldings etc.

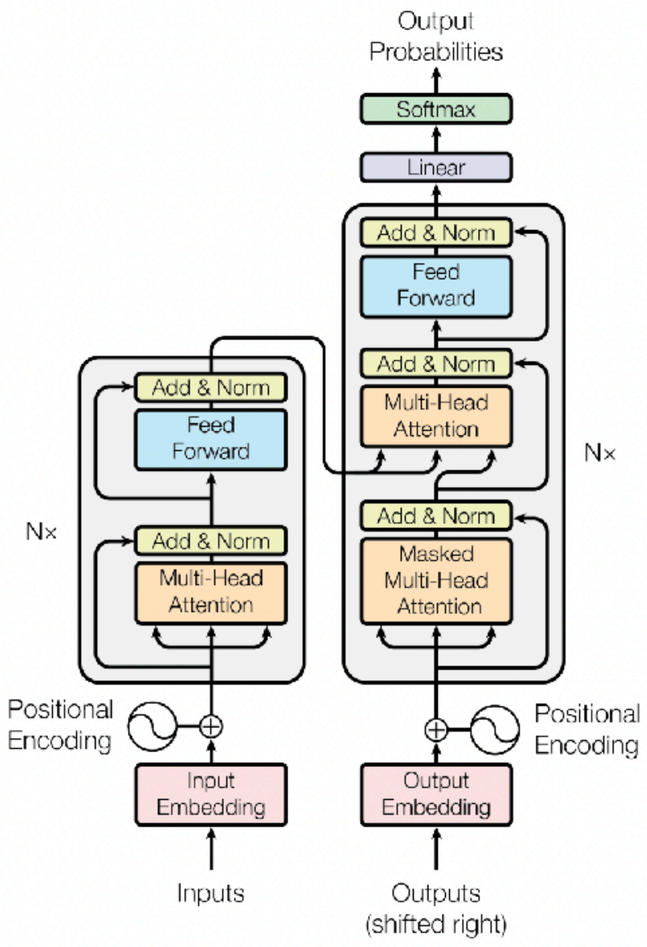


Figure 1: The Transformer - model architecture.

Scaling lows of Transformer

https://arxiv.org/abs/2001.08361

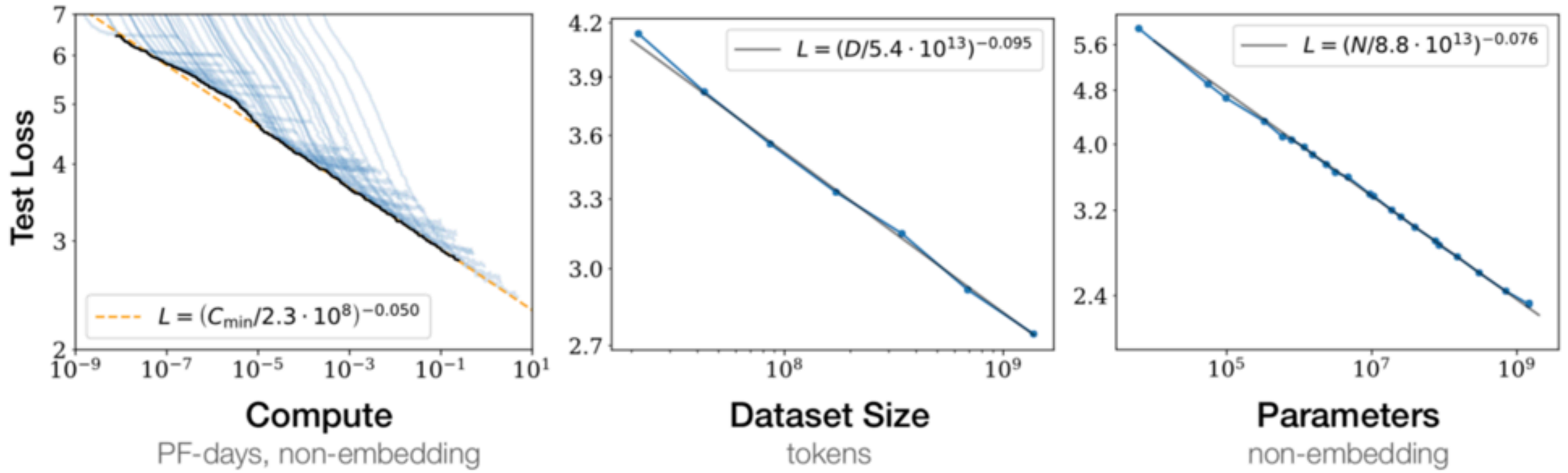


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when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

It requires huge data (e.g. GPT uses all electric books in the world)
= weak inductive bias, large data makes prediction better

Transformer and Attention

When we translate a sentence, we pay "attention" to words:

English: I am Yuki Nagai, who studies machine learning and physics

German: Ich bin Yuki Nagai, der Maschinenlernen und Physik studiert

translated by DeepL

Non-local dependencies can be treated by "Attention layer"

What are most important relations in words?

"Attention" layer can capture these relations

In physics terminology, this is **non local correlation.**
The attention layer enables us to treat it with a neural net!

What is the attention mechanism?

There are many websites to explain the transformer and attention mechanism, in terms of language processing...

I try to explain the attention in terms of simple mathematics

This came from discussions with Dr. Tomiya

What is the attention mechanism?

There are many websites to explain the transformer and attention mechanism, in terms of language processing...

I try to explain the attention in terms of simple mathematics

This came from discussions with Dr. Tomiya

1. We consider a vector/matrix/tensor **A** Ai or Aij or Aijk
2. We make three variables **K,Q,V** from **A**

   K = W^KA, Q = W^QA, V = W^VA **W^k,W^Q,W^V:trainable parameters**
3. We generate new vector/matrix/tensor **B**

$$B_l = A_l + \sum_i P_i^l V_i \qquad P = \sigma(QK^T)$$

correlation between Q and K

l=i or ij or ijk

What is the attention mechanism?

$K = W^K A$, $Q = W^Q A$, $V = W^V A$ W^k, W^Q, W^V: **trainable parameters**

3. We generate new vector/matrix/tensor **B**

$$B_l = A_l + \sum_i P_i^l V_i$$

weighted sum

$$P = \sigma(QK^T)$$

σ :nonlinear funciton

correlation between Q and K

self-attention mechanism

This is most simplest architecture

In generative AIs, they use the multi-head attention

Simple mechanism but very effective!

How can we use this in physics?

Equvariant transformer

Fermion and spin model

We want to focus on a simple lattice model

fermions and classical spins

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

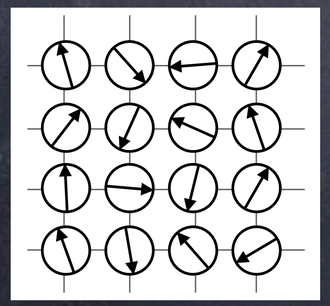
called double exchange model in condensed matter physics

Partition function:

$$Z = \sum_{\{\mathbf{S}\}} \prod_n (1 + e^{-\beta(\mu - E_n(\{\mathbf{S}\}))})$$

Configurations: classical spins $\{S_i\}$

S_i: i-th three dimensional vector in spin space



Input: spin configurations {S}

diagonalization

Output: Boltzmann weight

We want to replace the diagonalization

Fermion and spin model

fermions and classical spins

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

Simple effective model

J. Liu, H. Shen, Y. Qi, Z. Y. Meng, and L. Fu, Phys. Rev. B 95, 241104(R)(2017)

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

J_n^{eff}: n-th nearest neighbor interaction

This is a linear model

by integrating out fermion degrees of freedom

There are only few parameters J_n^{eff}

Fermion and spin model

fermions and classical spins

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

Simple effective model

J. Liu, H. Shen, Y. Qi, Z. Y. Meng, and L. Fu, Phys. Rev. B 95, 241104(R)(2017)

J_n^{eff}: n-th nearest neighbor interaction

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

This is a linear model

by integrating out fermion degrees of freedom

There are only few parameters J_n^{eff}

Effective model with a transformer

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

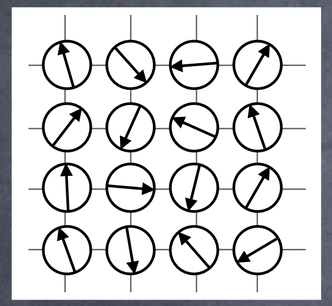
$$\mathbf{S}_i^{\mathrm{NN}} = f^{\mathrm{transformer}}(\{\mathbf{S}_i\})$$

We replace the spins with “translated” spin with a transformer

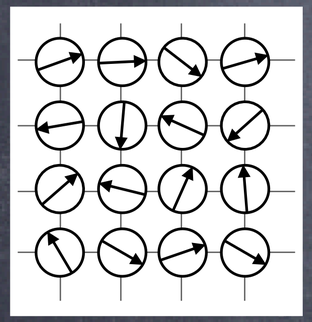
Invariance and equivariance

Hamiltonian has a symmetry ->invariant with the symmetry operation T

S



T[S]



→ H(S) = H(T[S])

<u>symmetry invariant</u>

<u>We can consider two kinds of networks</u>

1. make invariant input and put it into neural networks

S -> C

T[S] -> C

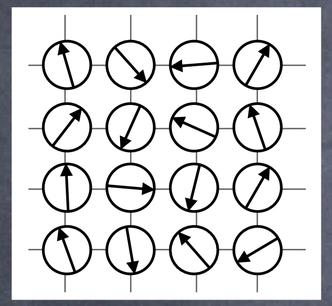
→ H = f(C)

Conventional architecture can be used

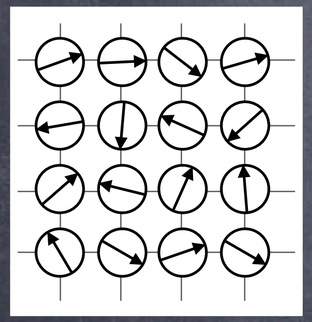
Invariance and equivariance

Hamiltonian has a symmetry ->invariant with the symmetry operation T

S



T[S]



→ H(S) = H(T[S])

symmetry invariant

We can consider two kinds of networks

1. make invariant input and put it into neural networks

S -> C
T[S] -> C
→ H = f(C)

Conventional architecture can be used

2. make equivariant networks and make the output invariant

T'[g(S)] = g(T[S])

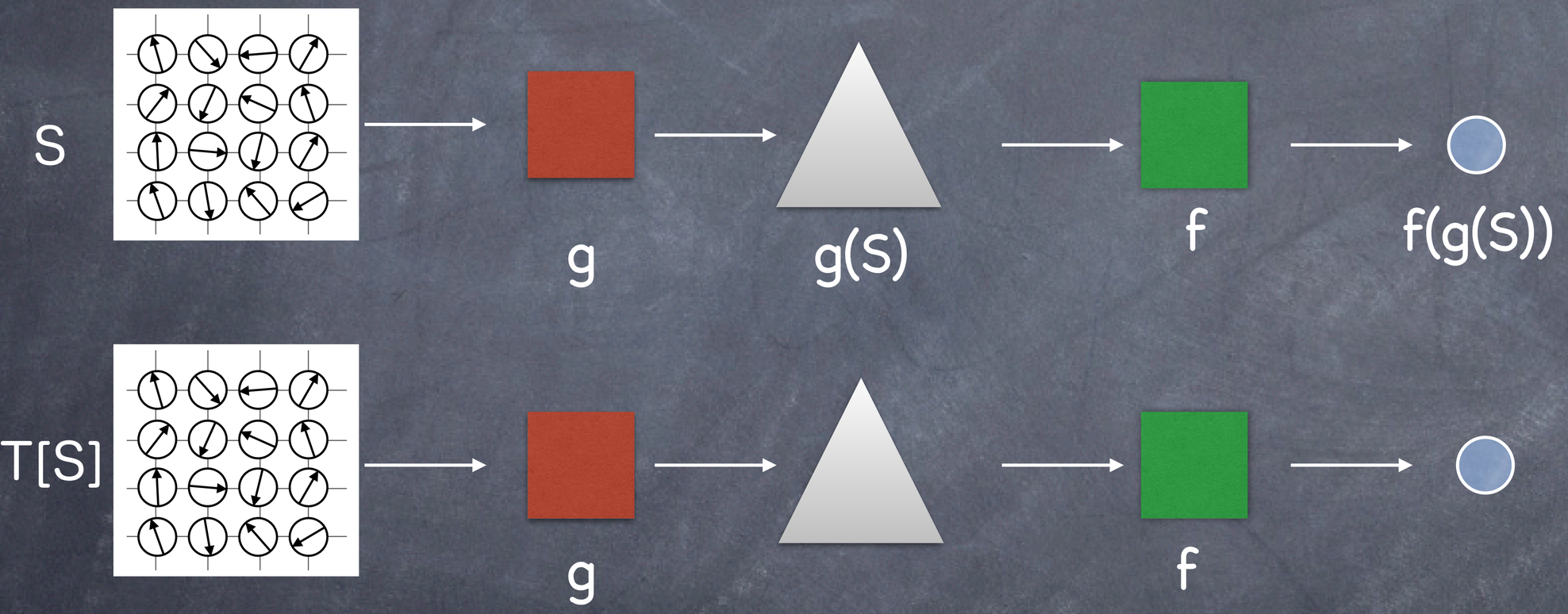
Equivariance

C = g(S) → H = f(C)

This network can keep a symmetry

Invariance and equivariance

2. make equivariant networks and make the output invariant



$f(T'[S]) = f(S)$

Invariance

$T'[g(S)] = g(T[S])$

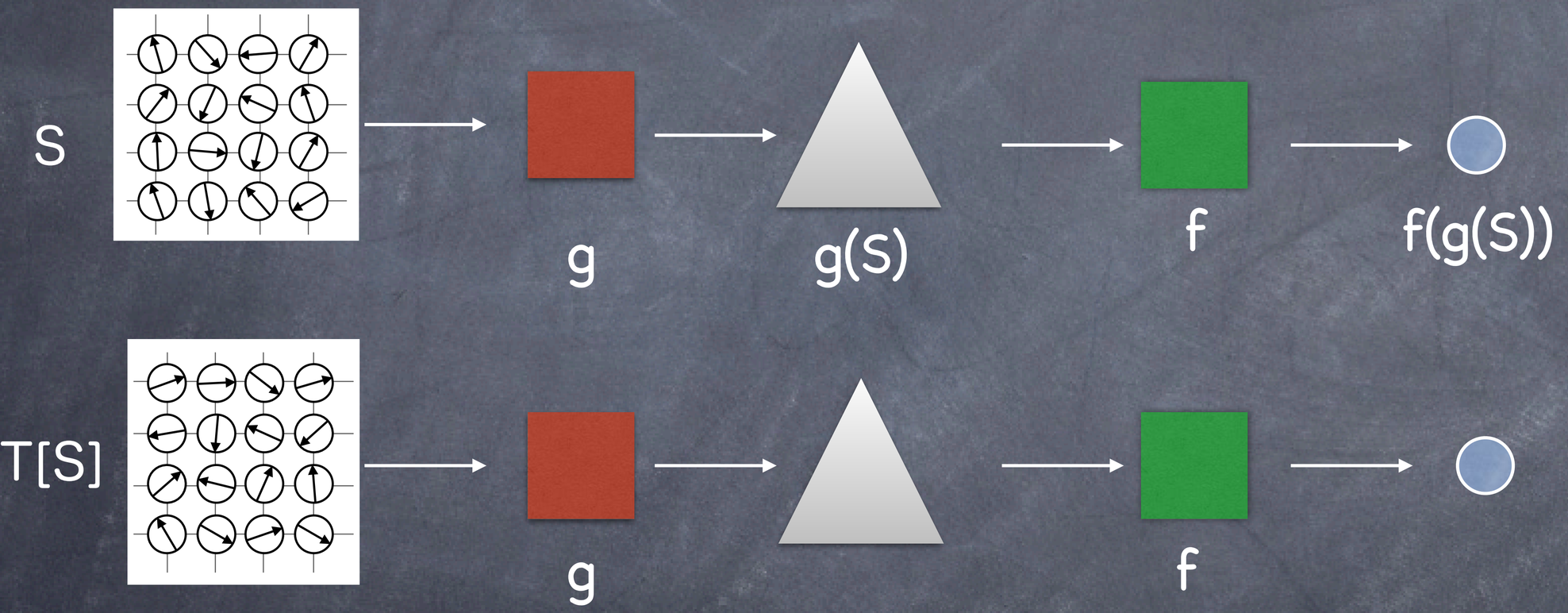
Equivariance

Outputs are same

CNN uses equivariance

Invariance and equivariance

2. make equivariant networks and make the output invariant



$f(T'[S]) = f(S)$

Invariance

$T'[g(S)] = g(T[S])$

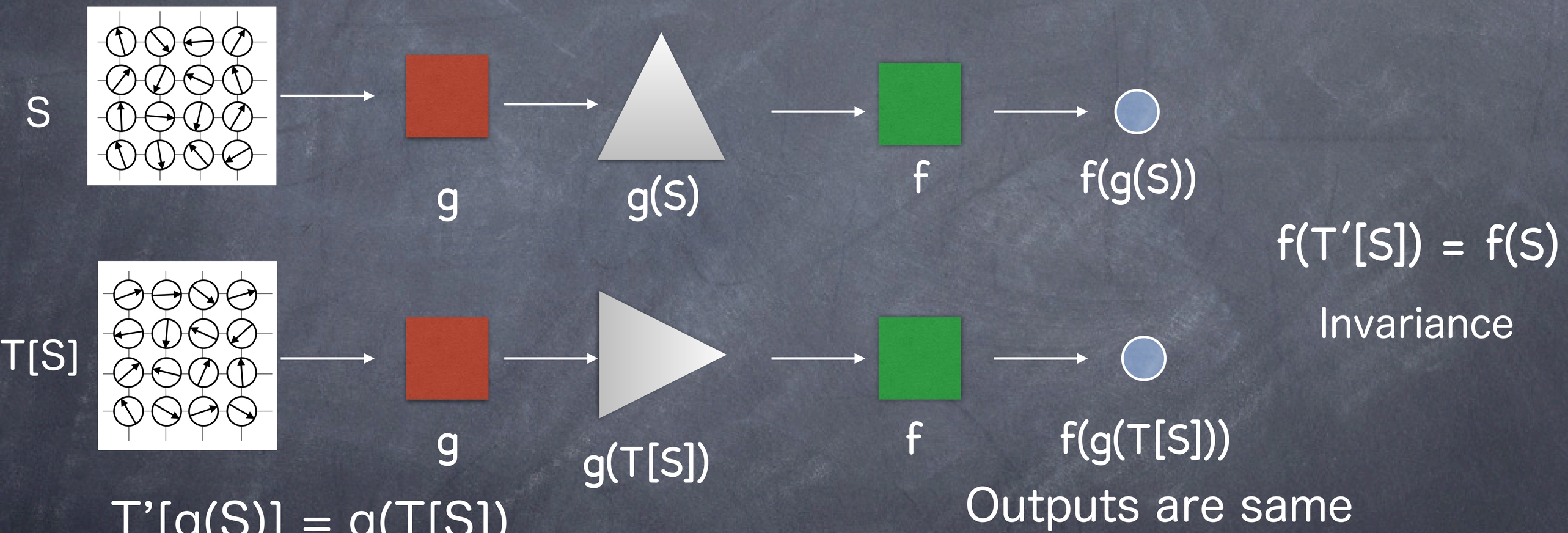
Equivariance

Outputs are same

CNN uses equivariance

Invariance and equivariance

2. make equivariant networks and make the output invariant



$f(T'[S]) = f(S)$

Invariance

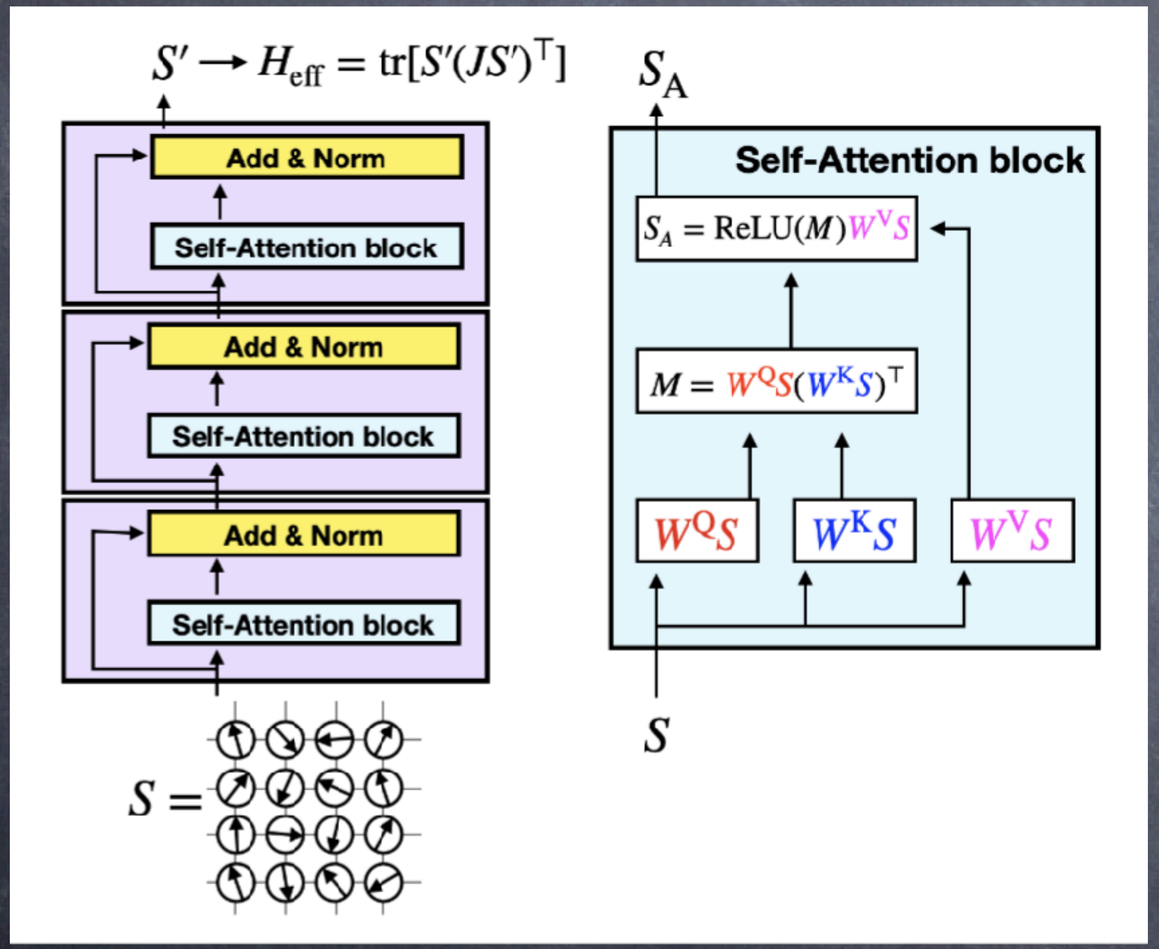
$T'[g(S)] = g(T[S])$

Equivariance

Outputs are same

CNN uses equivariance

Equivariant Transformer for spin systems



$S \rightarrow K = W^K S$

$S \rightarrow Q = W^Q S$

$S \rightarrow V = W^V S$

$$K_i = \sum_l W_l S_{i+l}$$

W only mixes neighbor spins
(short range interaction)

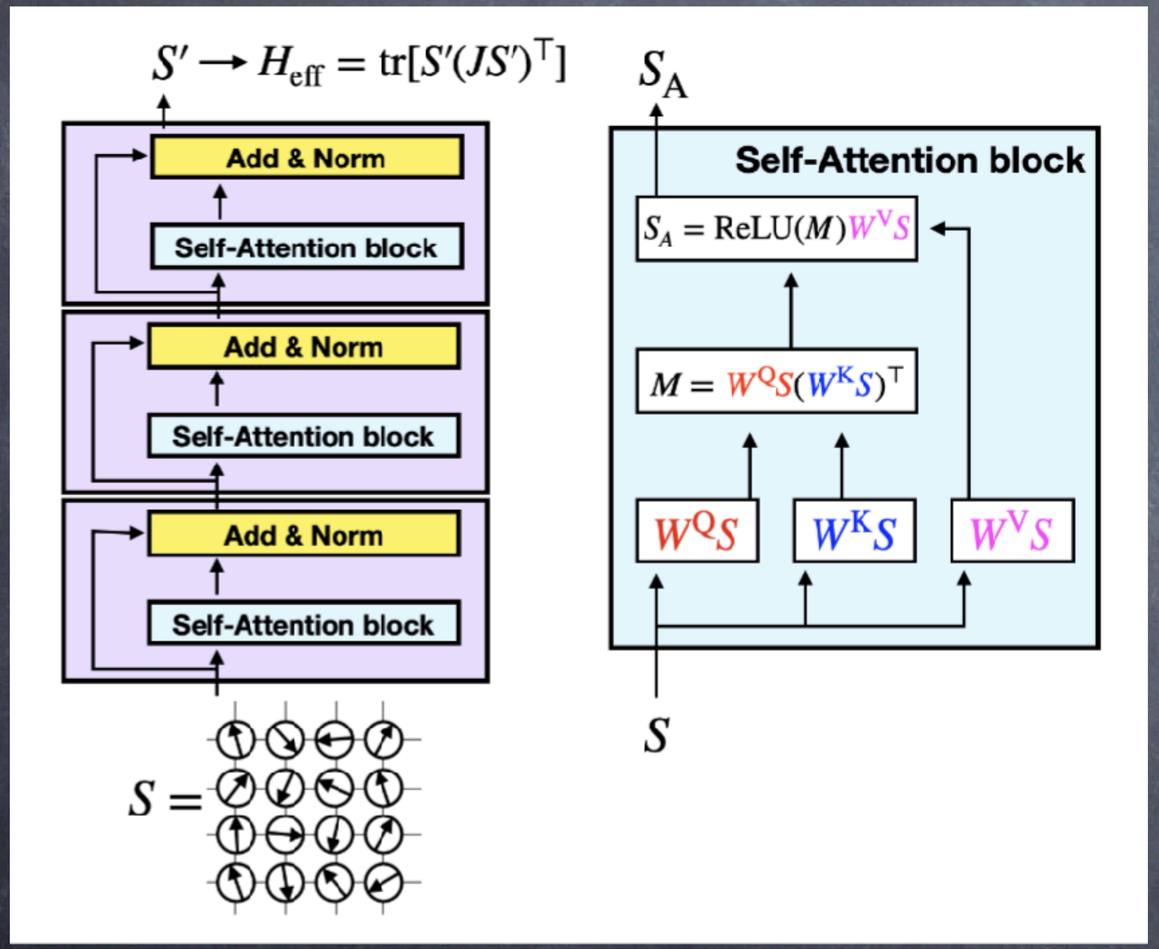
$M = W^Q S S + W^K$

Rotational and translational invariant

$S' = S + \mathrm{ReLU}(M)\, W^V S$

Long range correlation is included

Equivariant Transformer for spin systems



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

Layer 1

$$S_1 = \mathcal{N}(S + {\rm ReLU}(M^1(S))W^{V1}S)$$

Layer 2

$$S_2 = \mathcal{N}(S_1 + {\rm ReLU}(M^2(S_1))W^{V2}S_1)$$

Layer 3

$$S_3 = \mathcal{N}(S_2 + {\rm ReLU}(M^3(S_2))W^{V3}S_2)$$

Last Heisenberg model with effective spins

$$E = \sum_i \sum_l J_l \vec{S}_{3i} \cdot \vec{S}_{3i+l} + E_0$$

If the second term is zero

$$E = \sum_i \sum_l J_l \vec{S}_i \cdot \vec{S}_{i+l} + E_0$$ we get linearized model

Results

Result

Original model: fermions and classical spins

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

Partition function:

$$Z = \sum_{\{\mathbf{S}\}} \prod_n (1 + e^{-\beta(\mu - E_n(\{\mathbf{S}\}))})$$

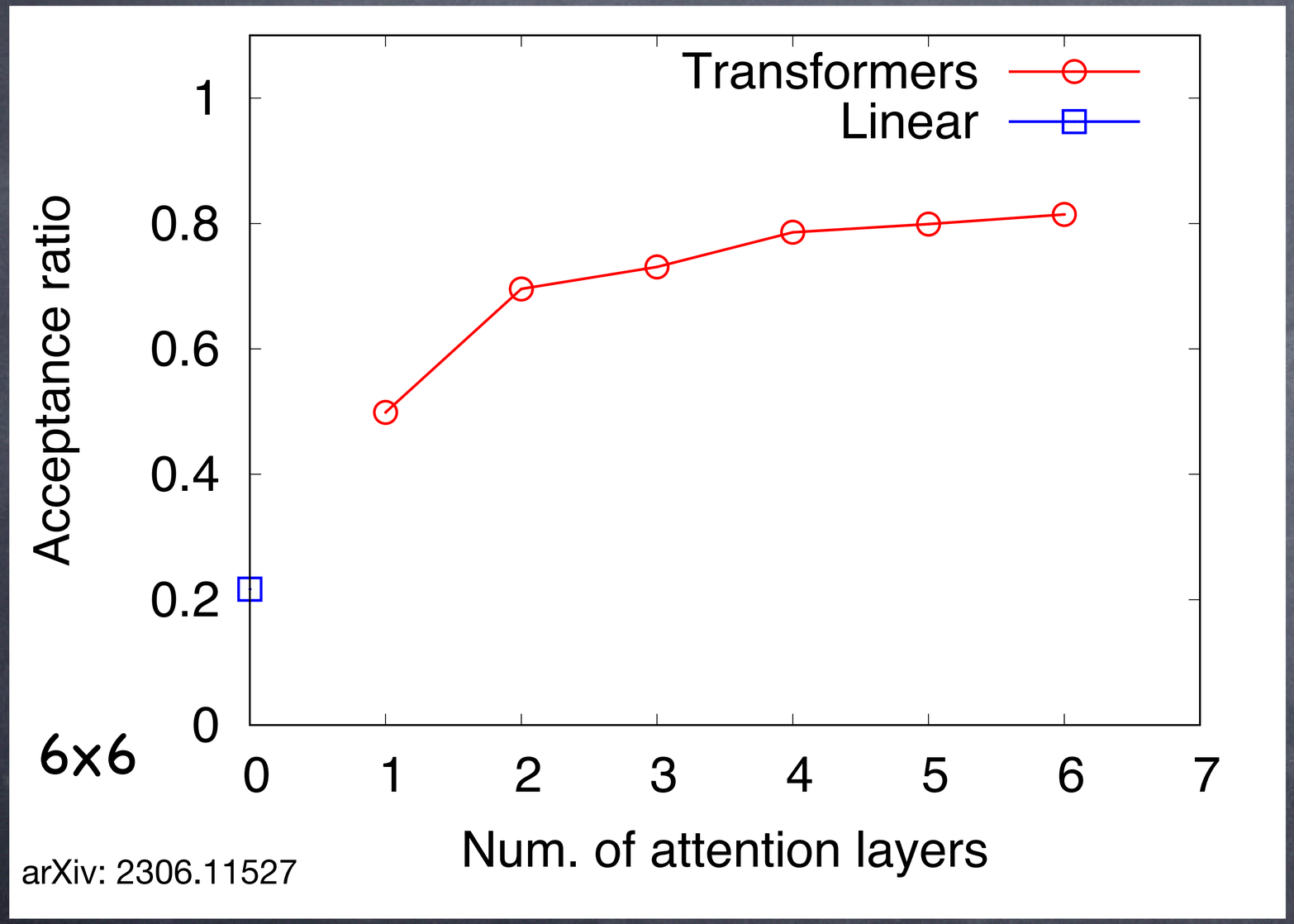
We make the effective model with Transformer

We generate target data and train the network in SLMC, simultaneously

We use Flux.jl, a machine learning framework in Julia language

Results

N=6



6-th nearest neighbors

$$K_i = \sum_l W_l S_{i+l}$$

Num. of parameters per layer

7+7+7 = 21

Last layer: nearest neighbors

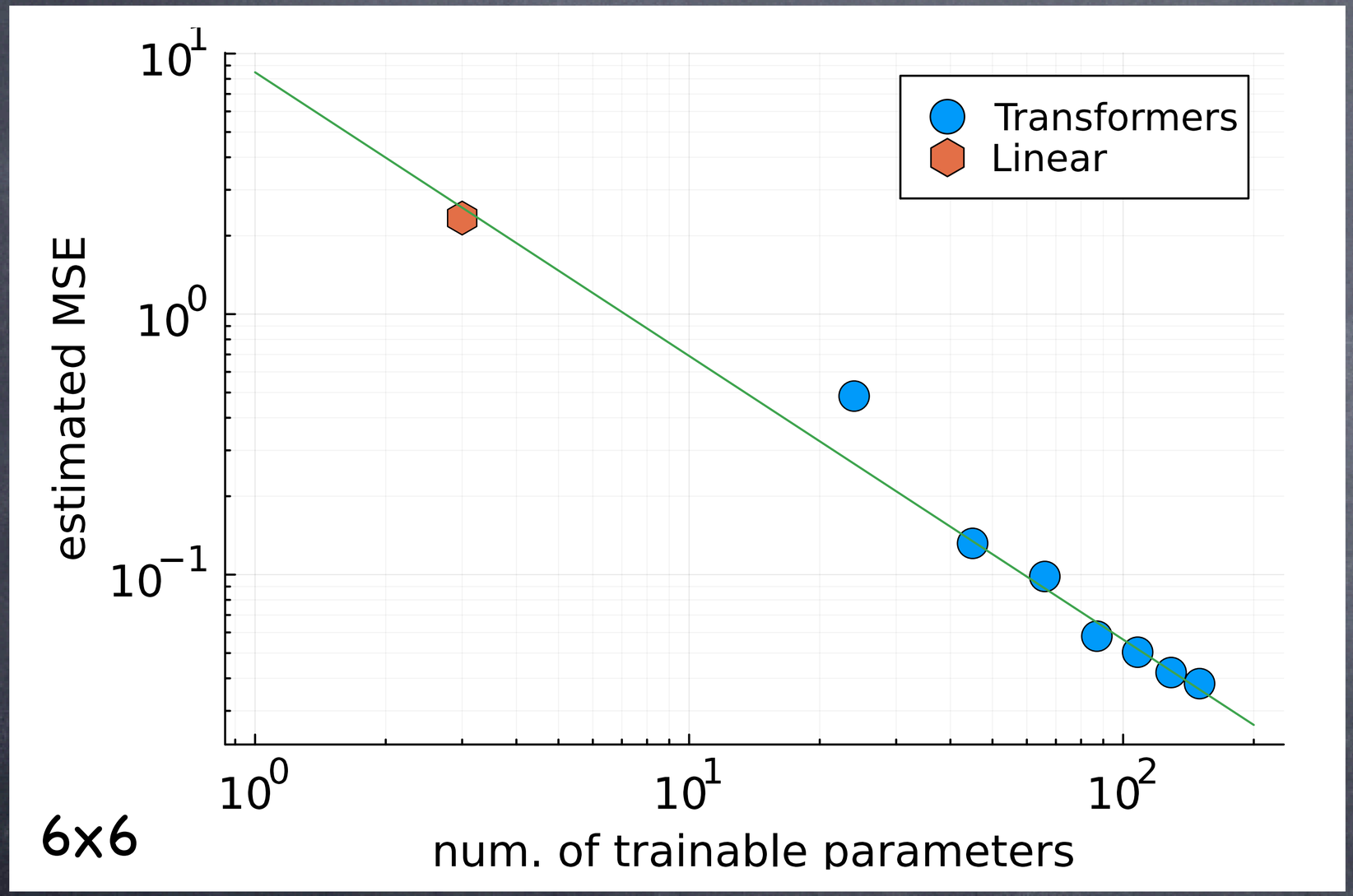
$$E = \sum_i \sum_l J_l \vec{S}_{3i} \cdot \vec{S}_{3i+l}$$

Num. of parameters is small

High acceptance ratio!

Results

arXiv: 2306.11527



6-th nearest neighbors

$$K_i = \sum_l W_l S_{i+l}$$

Num. of parameters per layer

7+7+7 = 21

Scaling low?

This is like the scaling lows in Large Language Models

This is MC simulation

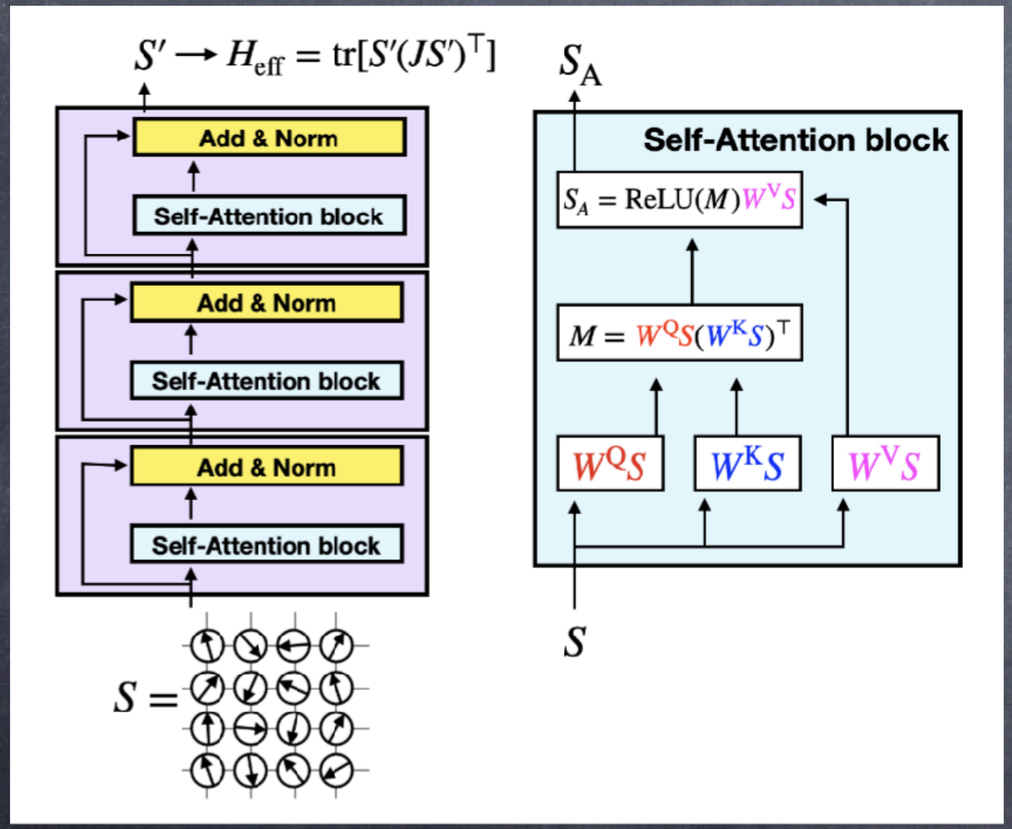
We generate data as we want

Summary

Summary

YN and A. Tomiya, "Self-learning Monte Carlo with equivariant Transformer", arXiv:2306.11527

Equivariant Transformer in spin systems



Equivariant with respect to spin-rotational and translational symmetries

We found the scaling low!

We can improve models with increasing num. of layers

"Transformer and Attention" is very useful!