# Gauge covariant neural network for 4 dimensional non-abelian gauge theory

Y. Nagai
JAEA

RBRC
RIKEN BNL Research Center

Akio Tomiya (RIKEN-BNL → Intrl. professional university of tech. in Osaka as a faculty)
akio.tomiya_AT_riken.jp

https://arxiv.org/abs/2103.11965

# Self-introduction

## Riken/BNL, Lattice QCD & Machine learning

**What am I?**

I am a particle physicist, working on lattice QCD.
I want to apply machine learning on it.

**My papers**

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

Evidence of effective axial $U(1)$ symmetry restoration at high temperature QCD
A Tomiya, G Cossu, S Aoki, H Fukaya, S Hashimoto, T Kaneko, J Noaki, ...
Physical Review D 96 (3), 034509
Axial anomaly with a chiral fermion

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 computer

**Bio**

2010 : University of Hyogo
2015 : PhD in Osaka university
2015 - 2018 : Postdoc in Wuhan
2018 - 2021 : SPDR in Riken/BNL
2021 - : Intrl. Professional Univ. of Tech. in Osaka as a faculty (大阪国際工科専門職大学)

# Outline

**Two topics**

1. Introduction (Lattice QCD)
2. Neural network, filtering and the convolution
3. Smearing
4. Gauge covariant neural network

**Neural network and symmetry**
**Gauge covariant network**

---

5. Demo: Self-learning HMC
6. Summary

**An application**

# Lattice QCD

## Well-defined quantum field theory

# Lattice QCD

## Electromagnetism = U(1) gauge theory

### Electromagnetism (in relativistic notation)

$$S = \int d^4x \left[ -\frac{1}{4} F_{\mu\nu} F^{\mu\nu} + \bar{\psi} \left( i\partial\!\!\!/ + eA\!\!\!/ - m \right) \psi \right]$$

$$F_{\mu\nu} = \partial_\mu A_\nu - \partial_\nu A_\mu$$

$A_\mu(x) \in \mathbb{R}$

$\mu = 0,1,2,3$

- This describes Electro& magnetic phenomena: $F_{0i}$ is E, $F_{ij}$ is B $(i, j = 1,2,3)$
- U(1) gauge symmetry controls: S is invariant following local transformation,

$$A_\mu(x) \to A_\mu(x) - \partial_\mu \Omega(x) \qquad \Omega(x) \in \mathbb{R}$$

### Quantum Electro-Dynamics (QED)

$$[E(x), A(y)] \sim \delta(x - y)$$ Quantization

$$|\psi(t)\rangle = e^{-iHt} |\psi(0)\rangle$$ $H$: Hamiltonian from S above

- Quantized electromagnetism
- The most precise theory in the world

# Lattice QCD

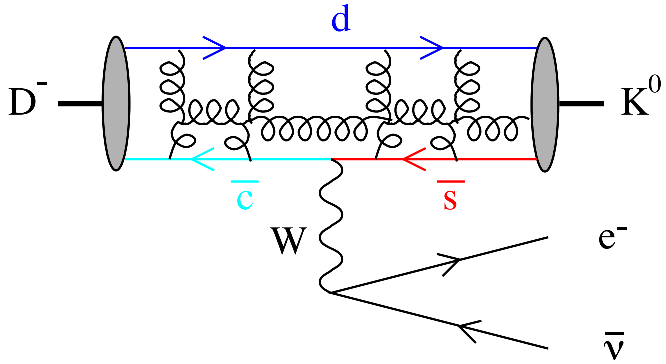
## QCD = Matrix version of quantum electro dynamics

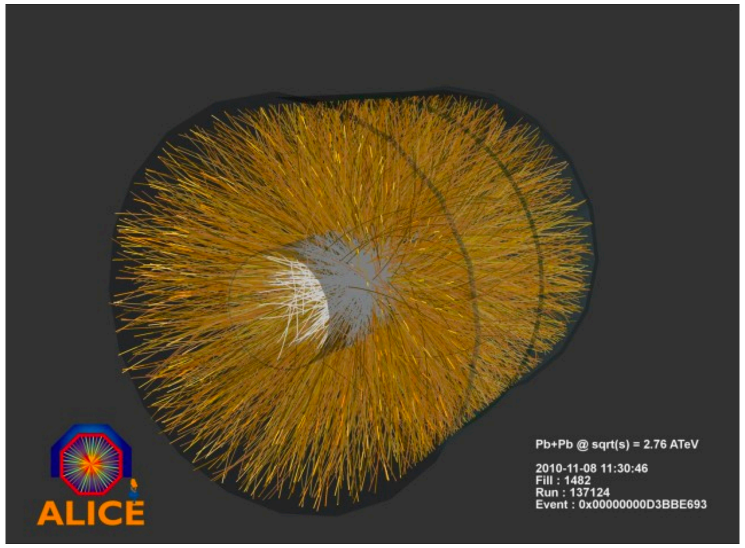
**QCD (Quantum Chromo-dynamics) in 3 + 1 dimension**

$$S = \int d^4x \left[ -\frac{1}{2} \mathrm{tr}\, F_{\mu\nu} F^{\mu\nu} + \bar{\psi}\left( i\partial\!\!\!/ + gA\!\!\!/ - m \right)\psi \right]$$

$$F_{\mu\nu} = \partial_\mu A_\nu - \partial_\nu A_\mu - \mathrm{i}g[A_\mu, A_\nu]$$ $A_\mu(x) \in su(3)$, 3x3 traceless matrix, harmitian

$$|\psi(t)\rangle = \mathrm{e}^{-\mathrm{i}Ht} |\psi(0)\rangle$$ $H$: Hamiltonian from S above




- Generalization of QED, $A_\mu(x)$ is matrix (Yang-Mills-Uchiyama)
- Action above enables us to calculate (in principle) followings:
  - Equation of state of neutron star, Tc
  - Forces between nuclei
  - Scattering of quarks and gluons, Parton distributions
  - Mass of hadrons, etc
- We cannot use perturbation since g >> 1

# Lattice QCD

## Gauge transf can be defined on the lattice

K. Wilson 1974

$$S = \int d^4x \left[ +\frac{1}{2} \mathrm{tr}\, F_{\mu\nu}F_{\mu\nu} + \bar{\psi}\left(\partial\!\!\!/ - \mathrm{i}gA\!\!\!/ + m\right)\psi \right]$$

Lattice regulation $U_\mu = \mathrm{e}^{aigA_\mu}$

$$S[U, \psi, \bar{\psi}] = a^4 \sum_n \left[ -\frac{1}{g^2} \mathrm{Re}\ \mathrm{tr}\ U_{\mu\nu} + \bar{\psi}\left(D\!\!\!/ + m\right)\psi \right]$$

$a$ is lattice spacing(cutoff = $a^{-1}$)

Both gives same expectation value (for long range)

(They are same except for infinitely Irrelevant operators)

$$\mathrm{Re}\ U_{\mu\nu} \sim \frac{-1}{2} g^2 a^4 F_{\mu\nu}^2 + O(a^6)$$

Gauge transformation on the lattice is simpler

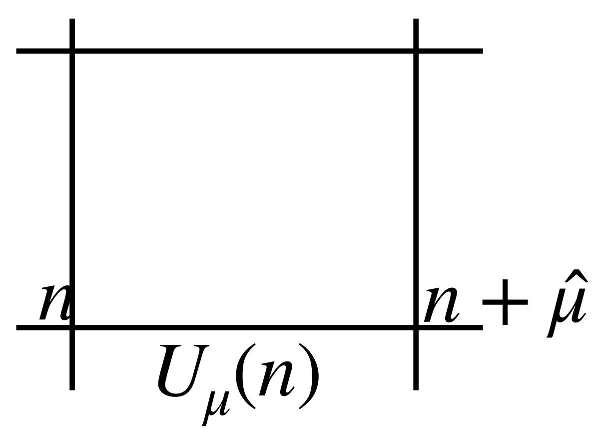
$$A_\mu(x) \to G(x)A_\mu(x)G^{-1}(x) - G(x)\partial_\mu G^{-1}(x)$$

Lattice reg.

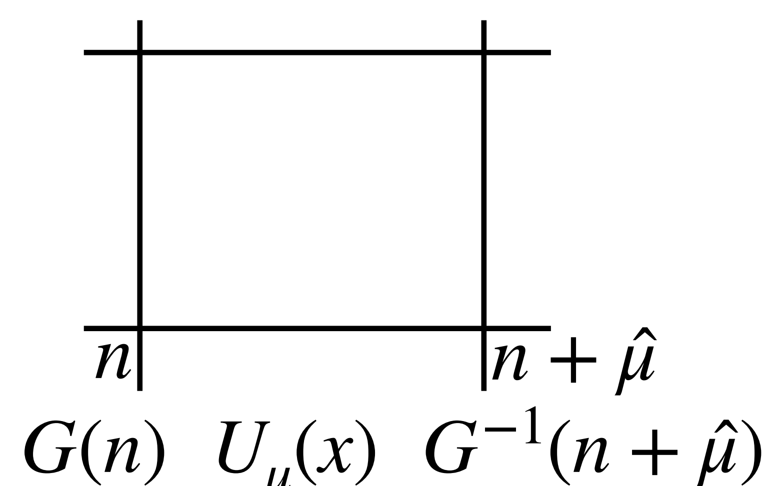
$$U_\mu(x) \to G(n)U_\mu(x)G^{-1}(n+\hat{\mu})$$

Gauge field on the bonds

Gauge trf on the points




Gauge trf →



# Lattice QCD

## Regulated theory for QCD

K. Wilson 1974

$$S = \int d^4x \left[ + \frac{1}{2} \mathrm{tr}\, F_{\mu\nu} F_{\mu\nu} + \bar{\psi} \left( \partial\!\!\!/ - \mathrm{i} g A\!\!\!/ + m \right) \psi \right]$$

Lattice regulation

$U_\mu = \mathrm{e}^{aigA_\mu}$

$$S[U, \psi, \bar{\psi}] = a^4 \sum_n \left[ - \frac{1}{g^2} \mathrm{Re}\ \mathrm{tr}\ U_{\mu\nu} + \bar{\psi} \left( D\!\!\!\!/ + m \right) \psi \right]$$

$a$ is lattice spacing(cutoff = $a^{-1}$)

Both gives same expectation value (for long range)

(They are same except for infinitely Irrelevant operators)

$$\mathrm{Re}\ U_{\mu\nu} \sim \frac{-1}{2} g^2 a^4 F_{\mu\nu}^2 + O(a^6)$$

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}U \mathcal{D}\bar{\psi} \mathcal{D}\psi e^{-S} \mathcal{O}(U) = \frac{1}{Z} \int \mathcal{D}U e^{-S_{\mathrm{gauge}}[U]} \det(D + m) \mathcal{O}(U)$$

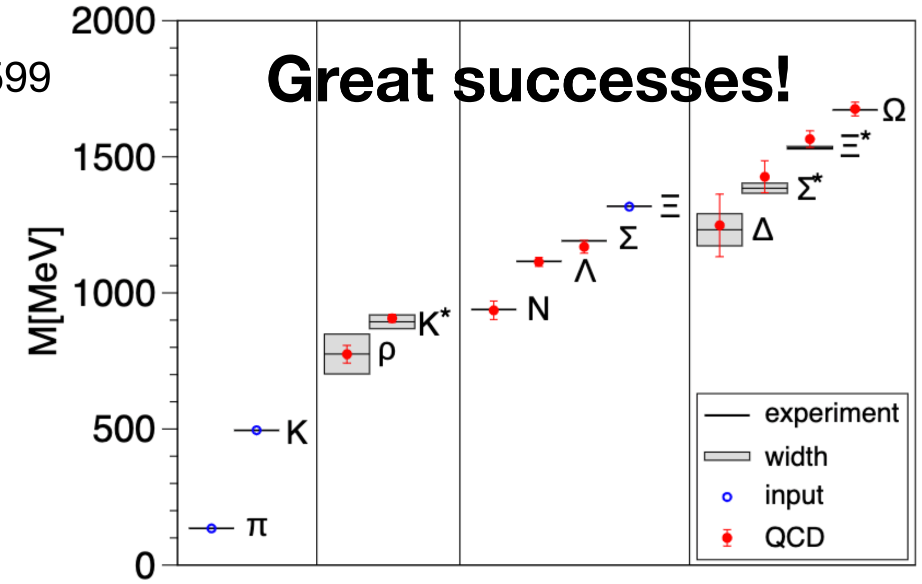
$$= \frac{1}{Z} \int \mathcal{D}U e^{-S_{\mathrm{eff}}[U]} \mathcal{O}(U)$$

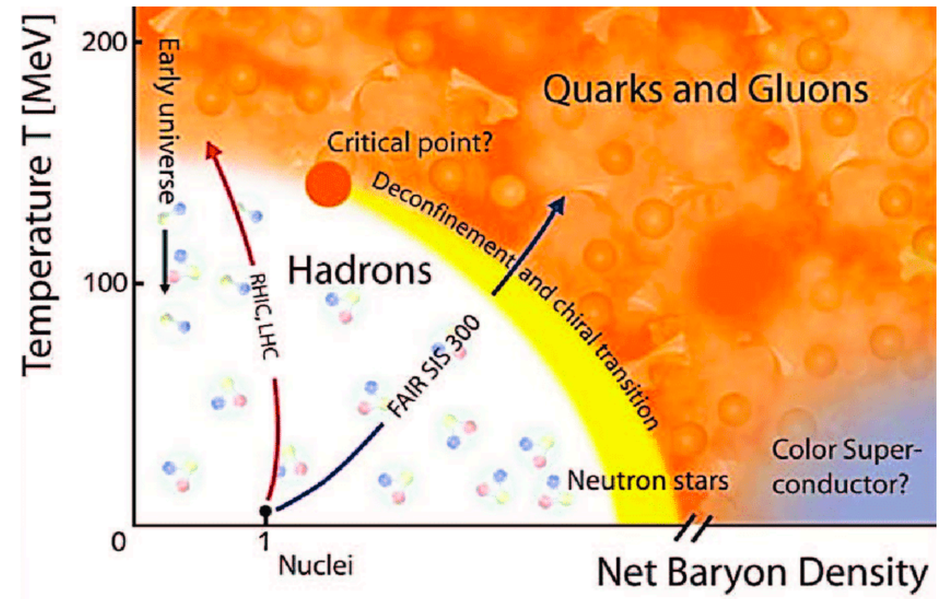
This integral gives expectation values (path integral).

# Lattice QCD

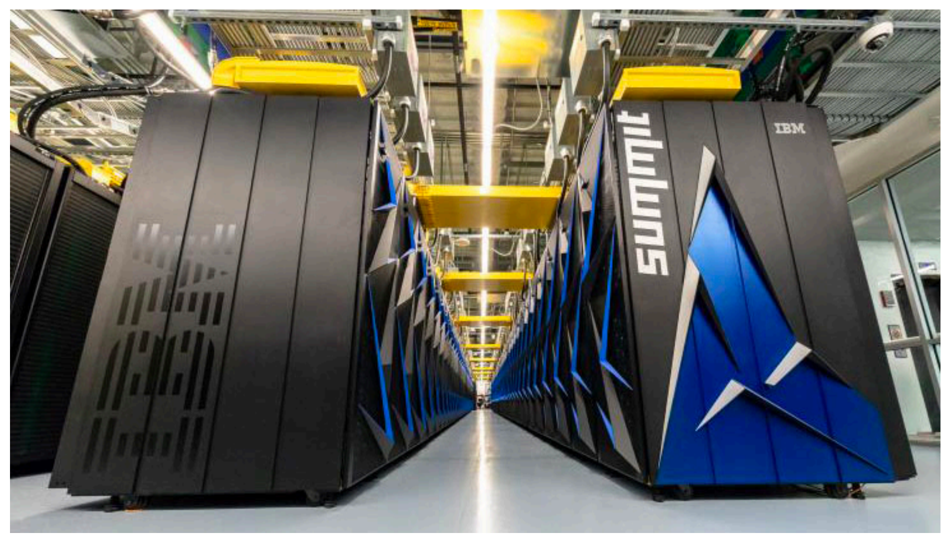
## LQCD makes us quantitative, a tool to investigate QFT

arXiv:0906.3599








Outcome?
- Force between nuclei
- Entanglement
- Form factors
- Parton distribution, etc…

# Lattice QCD
## You can start it in 10 minutes

AT & Y. Nagai in prep

We made a public LQCD code by Julia language:
https://github.com/akio-tomiya/LatticeQCD.jl

Easy and quick start on laptop/desktop
HMC/heatbath/**SLMC** + Measurements
(SU(Nc) Stout, RHMC for staggered, Wilson-Clover)

**Compatible speed with a Fortran code**

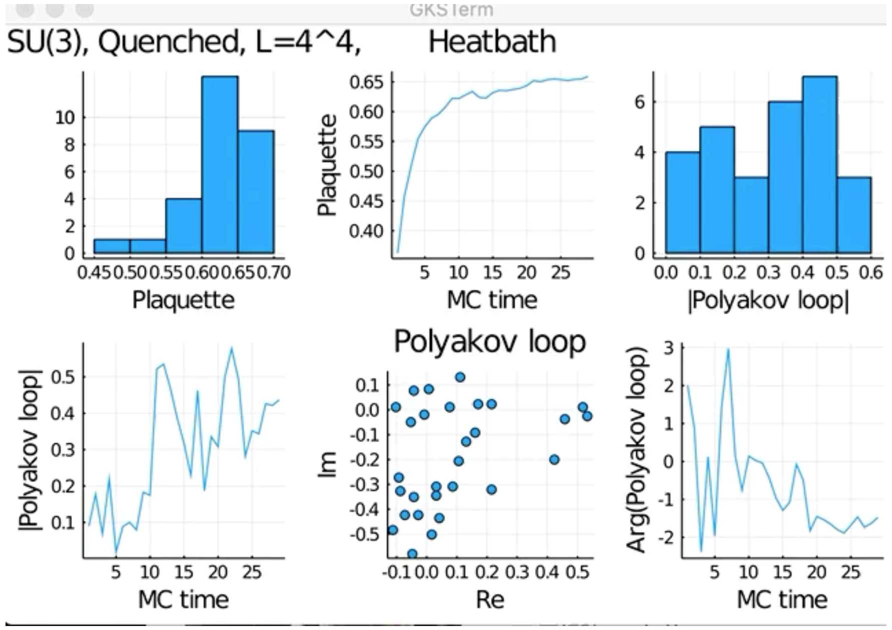
You can start in 3 steps (in 10 min)

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

LatticeQCD.jl


A.T.

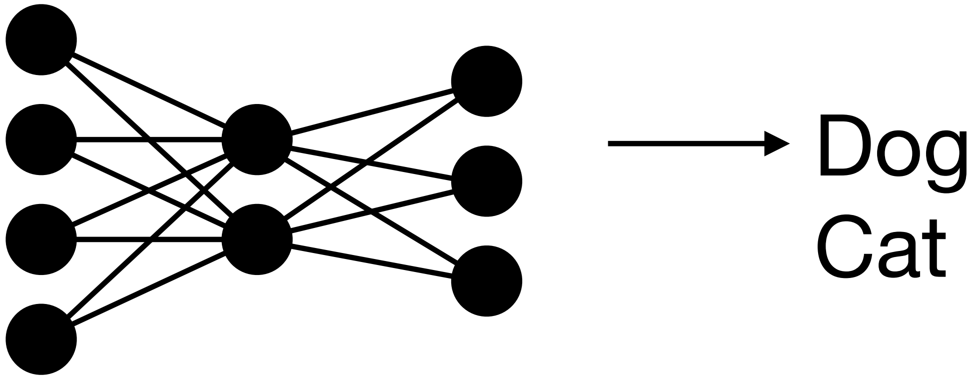

Y. Nagai

# Introduction

## Machine learning makes map between data

For example: image recognition




How can we deal with data with gauge symmetry?
(Can we embed in full HMC?)

cf.
If data with global symmetry (Ising model),
conventional architectures work well

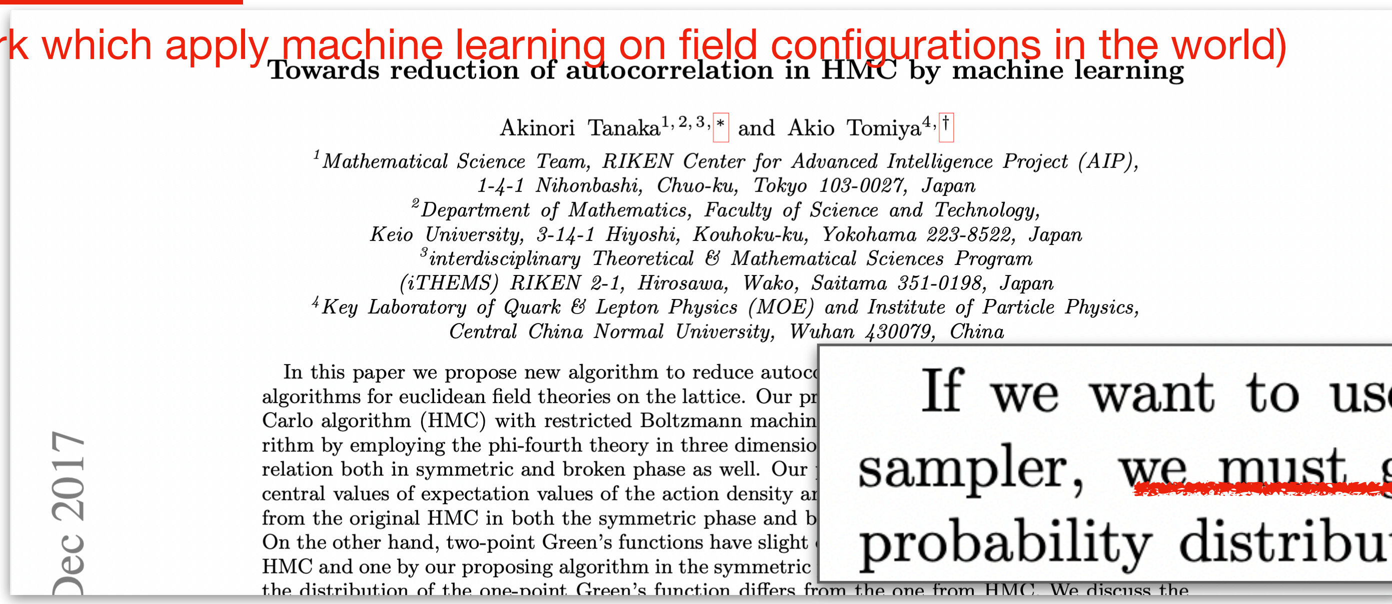
# Motivation and results

## Gauge symmetric neural network

**(Privite) motivation** I had wanted make gauge symmetric neural network since 2017

(The first work which apply machine learning on field configurations in the world)

arXiv: 1712.03893

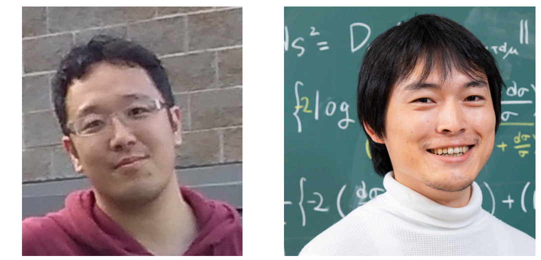

Towards reduction of autocorrelation in HMC by machine learning

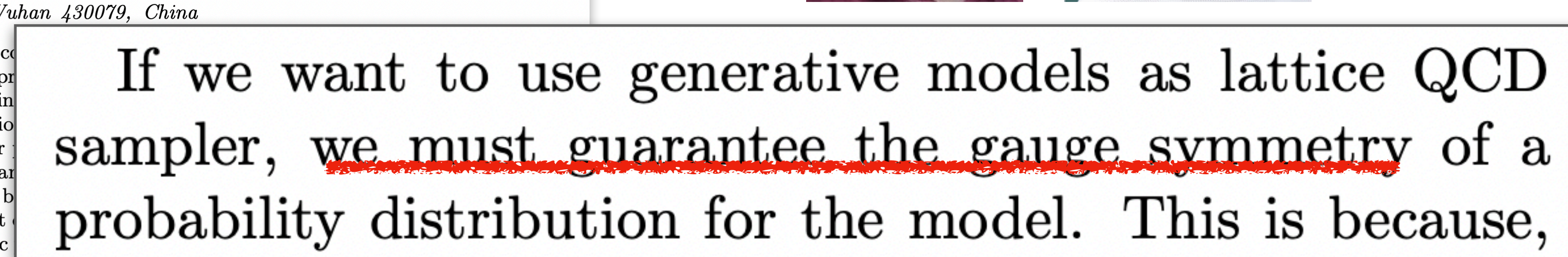
Akinori Tanaka[1,2,3,*] and Akio Tomiya[4,†]

[1]Mathematical Science Team, RIKEN Center for Advanced Intelligence Project (AIP), 1-4-1 Nihonbashi, Chuo-ku, Tokyo 103-0027, Japan
[2]Department of Mathematics, Faculty of Science and Technology, Keio University, 3-14-1 Hiyoshi, Kouhoku-ku, Yokohama 223-8522, Japan
[3]interdisciplinary Theoretical & Mathematical Sciences Program (iTHEMS) RIKEN 2-1, Hirosawa, Wako, Saitama 351-0198, Japan
[4]Key Laboratory of Quark & Lepton Physics (MOE) and Institute of Particle Physics, Central China Normal University, Wuhan 430079, China

In this paper we propose new algorithm to reduce autoc… algorithms for euclidean field theories on the lattice. Our pr… Carlo algorithm (HMC) with restricted Boltzmann machin… rithm by employing the phi-fourth theory in three dimensio… relation both in symmetric and broken phase as well. Our … central values of expectation values of the action density ar… from the original HMC in both the symmetric phase and b… On the other hand, two-point Green's functions have slight … HMC and one by our proposing algorithm in the symmetric … the distribution of the one-point Green's function differs from the one from HMC. We discuss the

Dec 2017

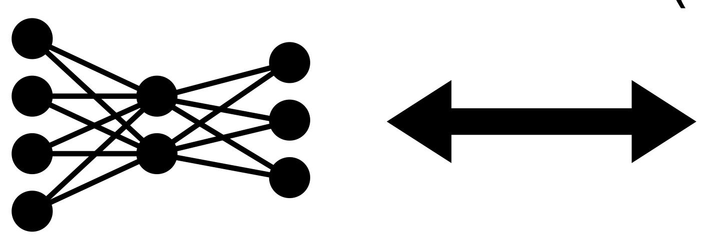



If we want to use generative models as lattice QCD sampler, we must guarantee the gauge symmetry of a probability distribution for the model. This is because,

**What we found**

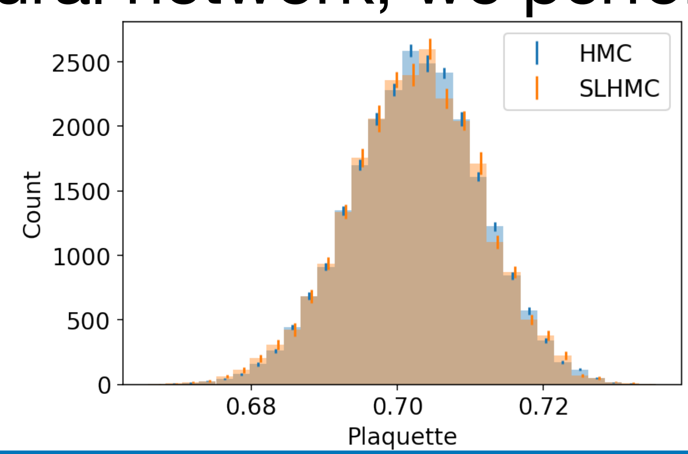
arXiv: 2103.11965

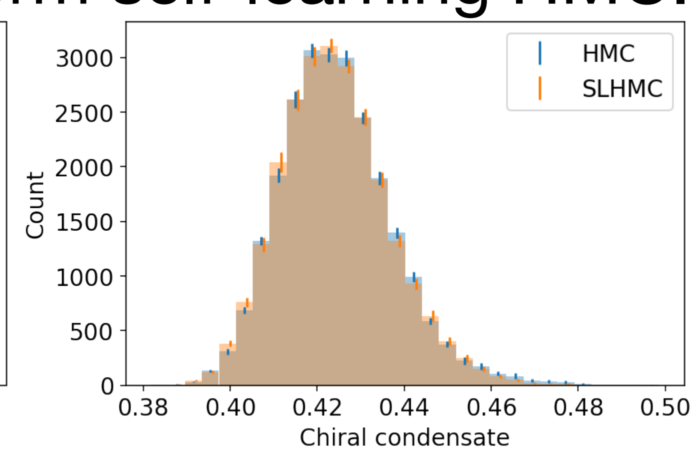
1. "Gauge symmetric neural network" = (trainable ) smearing



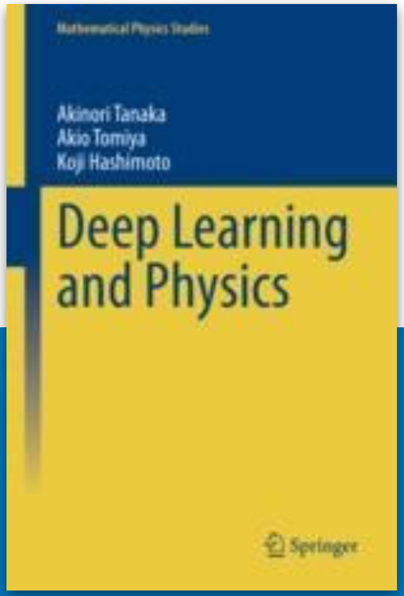
$$U^{\rm NN}_\mu(n)[U] = U^{(3)}_\mu(n)\left[U^{(2)}_\mu(n)\left[U^{(1)}_\mu(n)\left[U_\mu(n)\right]\right]\right]$$

2. Using the neural network, we perform self-learning HMC. Looks good.



# Neural network, filtering and the convolution

# What is the neural networks?



## Affine transformation + element-wise transformation

**Matrix**

$$[W\vec{x}]_i = \sum_j w_{ij} x_j$$

Matrix can "mimic" any linear map

Component of neural net

$$u_i(x_j) = \begin{cases} z_i^{(l)} = \sum_j w_{ij}^{(l)} x_j + b_i^{(l)} \\ u_i = \sigma^{(l)}(z_i^{(l)}) \end{cases}$$

Affine transf.
(b=0 called linear transf.)

element-wise (local)

**Fully connected neural net**

$$f_\theta(\vec{x}) = \sigma^{(l=2)}(W^{(l=2)}\sigma^{(l=1)}(W^{(l=1)}\vec{x} + \vec{b}^{(l=1)}) + \vec{b}^{(l=2)})$$

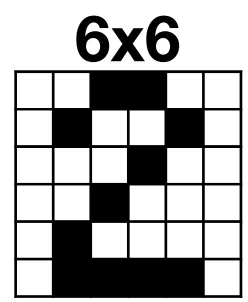
$\theta$ **is a set of parameters:** $w_{ij}^{(l)}, b_i^{(l)}, \cdots$

**Neural network = map between vector to vector**

# What is the neural networks?

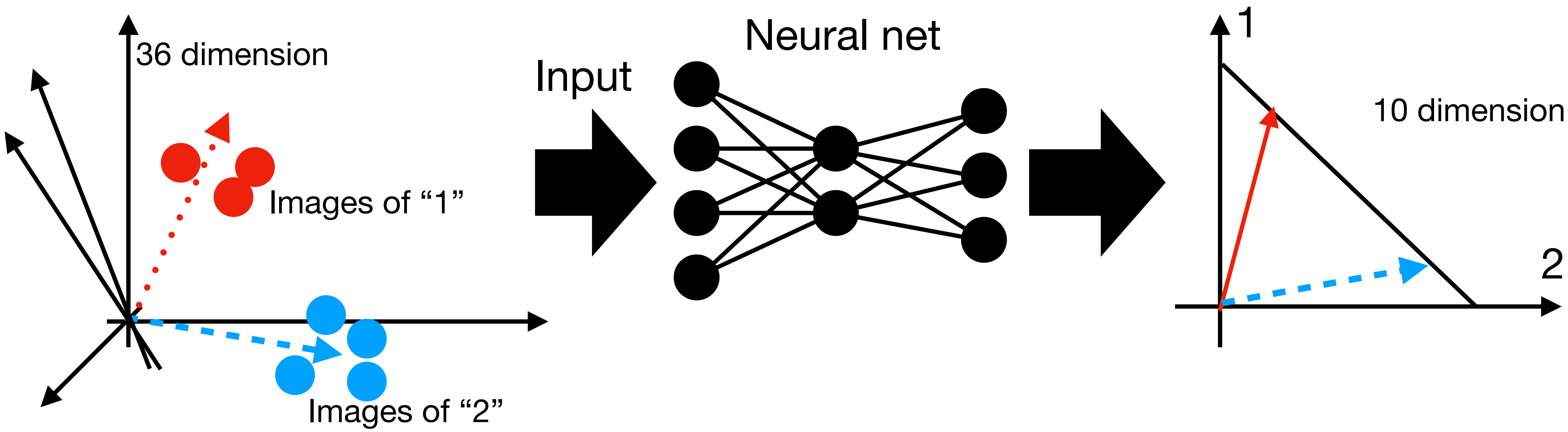
## Neural network is a universal approximator

~




$$= \begin{pmatrix} 0.000 \\ 0.000 \\ 0.8434 \\ 0.756 \\ 0.3456 \\ 0.64 \\ 0.251 \\ 0.11 \\ 0.23 \\ \vdots \end{pmatrix}$$

**Image is a vector**




“0” = (1,0,0,…)
“1” = (0,1,0,…)
“2” = (0,0,1,…)
…
“9” = (0,0,…,1)

**Fact: neural network can mimic any function!**

**(Intuitively、# of unit in neural net ~ basis in the Fourier transformation)**

In this example, NN mimics image (36-dim vector) and label (10-dim vector)

# What is the neural networks?

Akio Tomiya

## Training can be done with a gradient optimizer & delta rule

1d example of training

$$\begin{cases} z^{(l)} = w^{(l)} u^{(l-1)} \\ u^{(l+1)} = \sigma(z^{(l)}) \end{cases} \qquad u^{(l=0)} = x \qquad f_\theta(x) = \sigma(w^{(l=2)}\sigma(w^{(l=1)}\sigma(w^{(l=0)}x)))$$

$$\boxed{x} = \boxed{u^{(l=0)}} \xrightarrow{\times w^{(l=1)}} \boxed{z^{(l=1)}} \xrightarrow{\sigma(\,\cdot\,)} \boxed{u^{(l=1)}} \xrightarrow{\times w^{(l=2)}} \boxed{z^{(l=2)}} \xrightarrow{\sigma(\,\cdot\,)} \boxed{u^{(l=2)}} = \boxed{f_\theta(x)}$$

# What is the neural networks?

Akio Tomiya

## Training can be done with a gradient optimizer & delta rule

1d example of training

$$\begin{cases} z^{(l)} = w^{(l)} u^{(l-1)} \\ u^{(l+1)} = \sigma(z^{(l)}) \end{cases} \qquad u^{(l=0)} = x \qquad f_\theta(x) = \sigma(w^{(l=2)}\sigma(w^{(l=1)}\sigma(w^{(l=0)}x)))$$

$$x = u^{(l=0)} \xrightarrow{\times w^{(l=1)}} z^{(l=1)} \xrightarrow{\sigma(\cdot)} u^{(l=1)} \xrightarrow{\times w^{(l=2)}} z^{(l=2)} \xrightarrow{\sigma(\cdot)} u^{(l=2)} = f_\theta(x)$$

Training:

$$w^{(l)} \leftarrow w^{(l)} - \eta \frac{\partial L_\theta}{\partial w^{(l)}}$$ (Gradient)

$$L_\theta = \sum_{i \in \text{data}} \frac{1}{2} \left| y_i - f_\theta(x_i) \right|^2$$

Chain rule gives a recursive formula of the delta (called the delta rule)

$$\frac{\partial L_\theta}{\partial w^{(l)}} = \frac{\partial L_\theta}{\partial z^{(l)}} \frac{\partial z^{(l)}}{\partial w^{(l)}} = \delta^{(l)} u^{(l-1)} \qquad \delta^{(l)} \equiv \frac{\partial L_\theta}{\partial z^{(l)}}$$

**Delta rule:** $$\delta^{(l)} = \frac{\partial L_\theta}{\partial z^{(l+1)}} \frac{\partial z^{(l+1)}}{\partial z^{(l)}} = \delta^{(l+1)} w^{(l+1)} \sigma'(z^{(l)})$$

**Delta is determined recursively:** $\delta^{(l=2)} \to \delta^{(l=1)}$**, and we get** $\frac{\partial L_\theta}{\partial w^{(l)}} = \delta^{(l)} u^{(l-1)}$

# What is the neural networks?

## Training can be done with a gradient optimizer & delta rule

**Forward process**

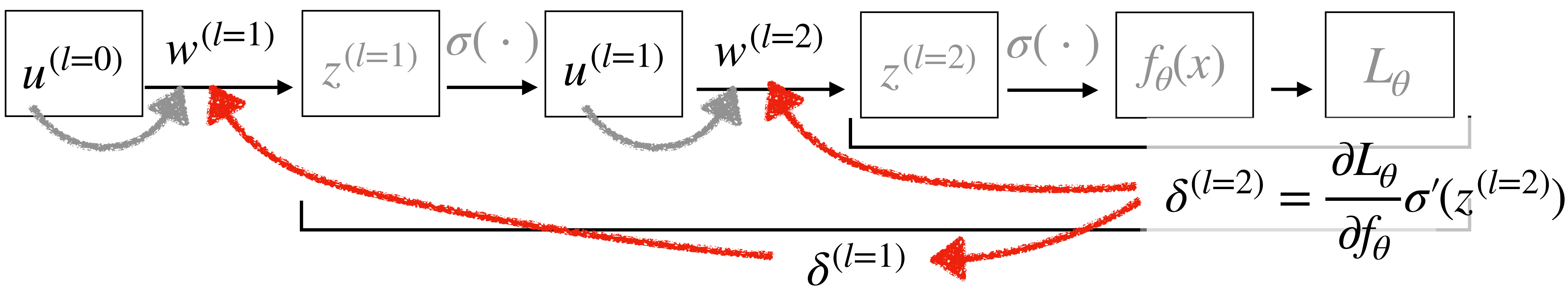
$$f_\theta(x) = \sigma(w^{(l=2)}\sigma(w^{(l=1)}\sigma(w^{(l=0)}x)))$$

$$x = u^{(l=0)} \xrightarrow{\times w^{(l=1)}} z^{(l=1)} \xrightarrow{\sigma(\cdot)} u^{(l=1)} \xrightarrow{\times w^{(l=2)}} z^{(l=2)} \xrightarrow{\sigma(\cdot)} u^{(l=2)} = f_\theta(x)$$

**Training is done with propagating error in backward: “Backprop”**

$$\frac{\partial L_\theta}{\partial w^{(l)}} = \delta^{(l)} u^{(l-1)}$$

Delta rule: (recursion) $\delta^{(l)} = \delta^{(l+1)} w^{(l+1)} \sigma'(z^{(l)})$



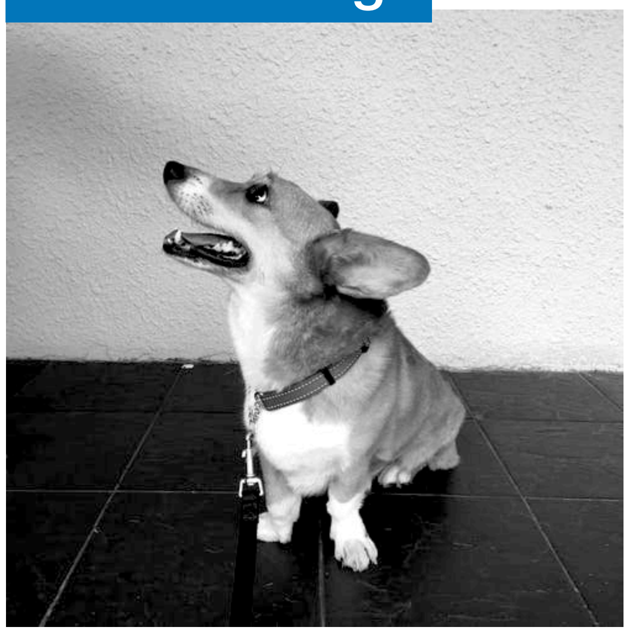

Training:

$$w^{(l)} \leftarrow w^{(l)} - \eta \frac{\partial L_\theta}{\partial w^{(l)}}$$

# What is the neural networks?

## Convolution layer = Trainable filter

### Filter on image



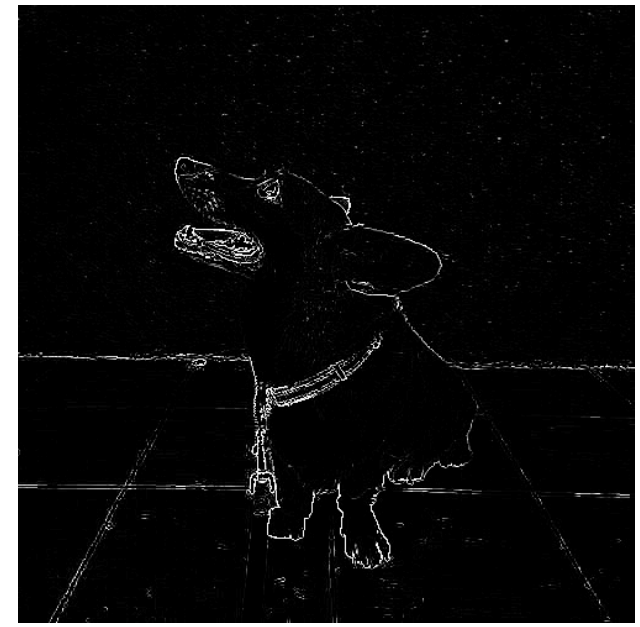
$*$

**Laplacian filter**

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | -2 | 1 |
| 0 | 1 | 0 |

(Discretization of $\partial^2$)

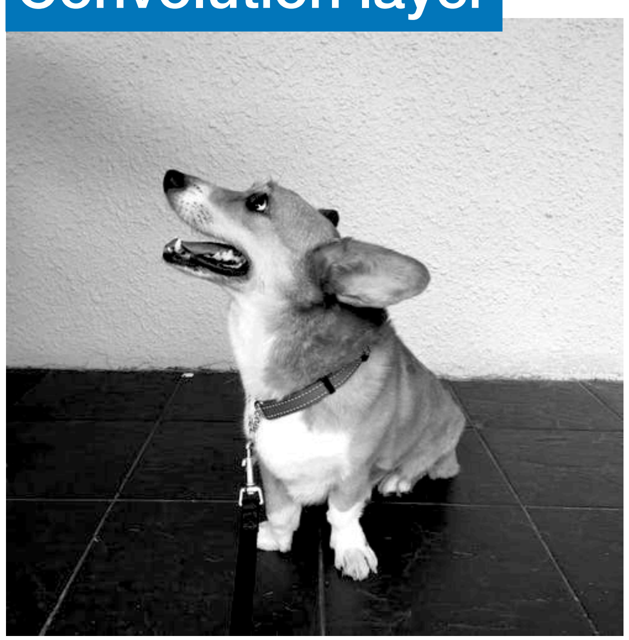
$=$



**Edge detection**

### Convolution layer

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



$*$

**Trainable filter**

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

$=$

**Edge detection**

**Smoothing**

**…**

Gaussian filter

$\frac{1}{16}$

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 4 | 2 |
| 1 | 2 | 1 |

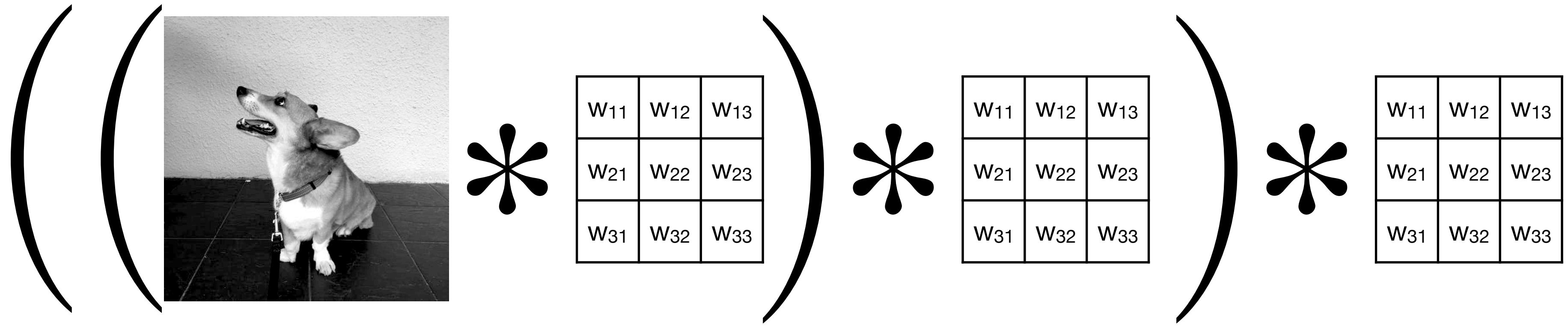
**(Training and data determines what kind of filter is realized)**
**Extract features**

# What is the neural networks?

## Convolution layers can be nested as well as fully connected

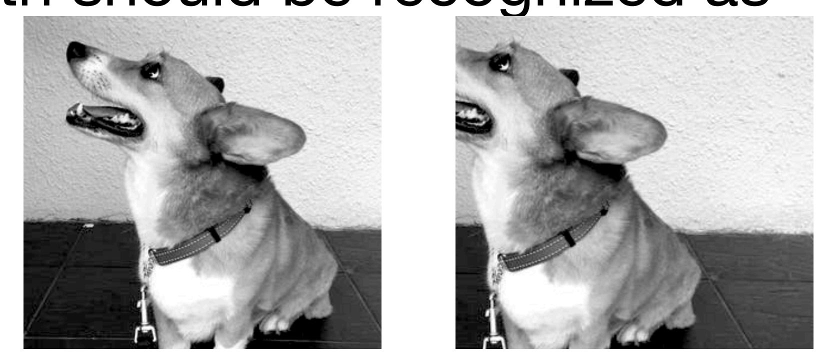
We can make a composite function with the convolutional layers

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




1. The convolution layers improves performance of image recognition
2. Convolutional layer = sparsened version of fully connected with "weight sharing"
3. Filtering operation does not care the absolute coordinate = translation symmetry

Both should be recognized as "dog"



Modern viewpoint:

Symmetry improves performance. (T. Cohen+, group equivariant NN)

(Rotational symmetry: Spherical convolution T. Cohen+). Approximator is guaranteed to respect the sym.

In machine learning context, some data has "gauge symmetry"
Info of gauge symmetry is useful to improve performance (T. Cohen+, gauge equivariant NN)

# What is the neural networks?

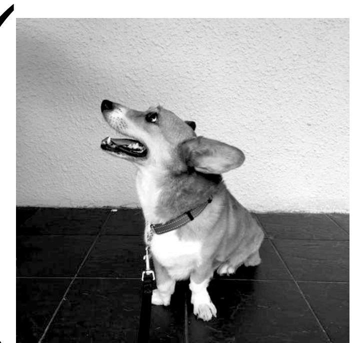
## Convolution + fully connected

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

e.g.:

$$f_\theta(\vec{x}) = \sigma^{(l=2)}(W^{(l=2)}\sigma^{(l=1)}(W^{(l=1)}\vec{v}))$$

$$\vec{v} = \mathrm{Flatten}\Bigg(\Bigg(\Bigg(\Bigg($$



$$* \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}\Bigg) * \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}\Bigg) * \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}\Bigg)$$

Parameters in convolutional layers can be trained as same as fully connected ones.

(For modern implementation, it should be multi-channel & use global average pooling but here we ignore it)

We can extract information of physics from configurations: (AT+ 2016)

# Smearing

Smoothing with gauge symmetry

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

# Smearing

Akio Tomiya

## Smoothing improves global properties

Eg.

Coarse image

Numerical derivative is unstable

Gaussian filter

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

Smoothened image

Numerical derivative is stable

It distort microscopic structure but global structure (topology) get improved

If one wants to study Topology, we can use the Gauss-Bonnet argument

We want to smoothen gauge configuration with keeping gauge symmetry

**Two types:**

**APE-type smearing** M. Albanese+ 1987

**Stout-type smearing** R. Hoffmann+ 2007, C. Morningster+ 2003

# Smearing
## 1st: smoothing with gauge symmetry

M. Albanese+ 1987
R. Hoffmann+ 2007

### APE-type smearing

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

Normalization

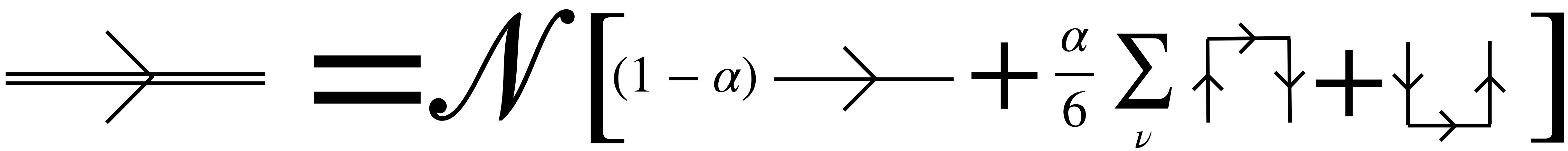
$$\mathcal{N}[M] = \frac{M}{\sqrt{M^\dagger M}}$$

Or projection

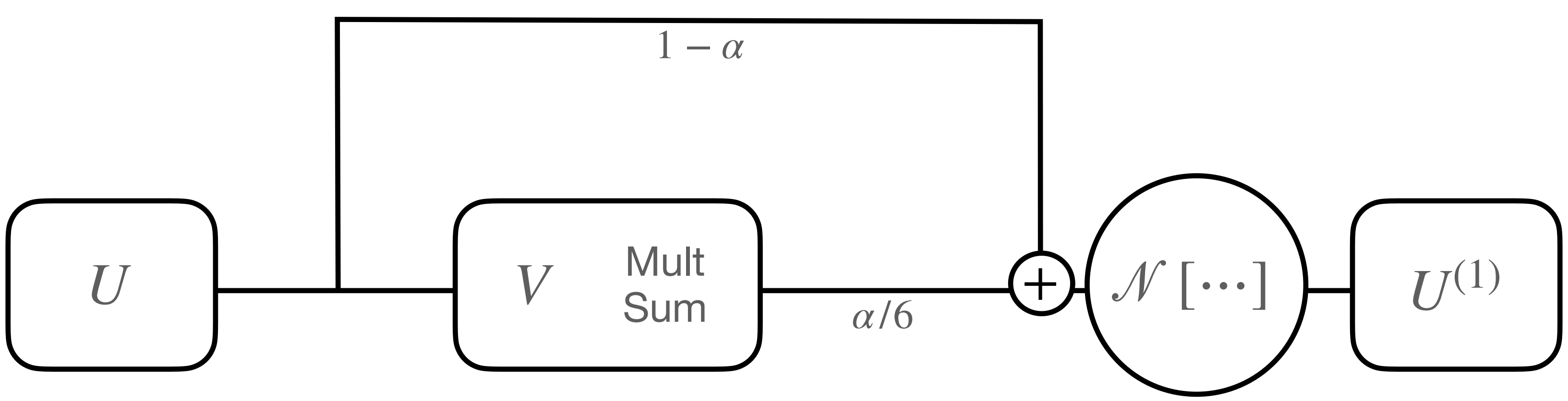
$$V_\mu^\dagger[U](n) = \sum_{\mu\neq\nu} U_\nu(n)U_\mu(n+\hat{\nu})U_\nu^\dagger(n+\hat{\mu}) + \cdots$$

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

### Schematically,




### In the calculation graph,



# Smearing

## 2nd: smoothing with gauge symmetry

**Stout-type smearing**

C. Morningster+ 2003

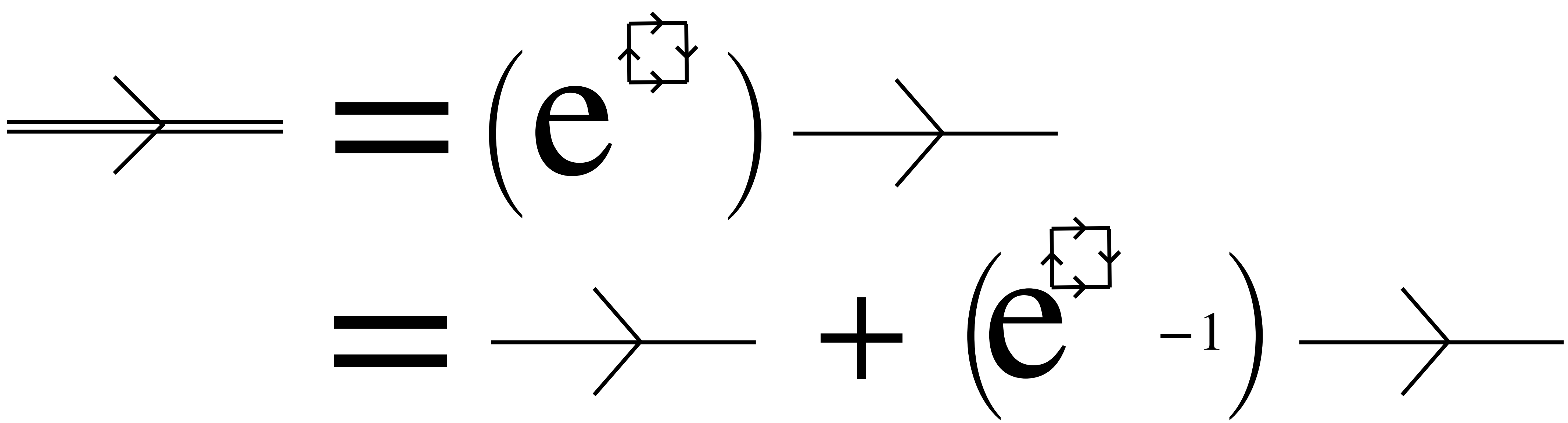
$$U_\mu(n) \to U_\mu^{\text{fat}}(n) = \mathrm{e}^{Q} U_\mu(n)$$

$$= U_\mu(n) + (\mathcal{G} - 1) U_\mu(n) \qquad \mathcal{G} = \exp(Q)$$

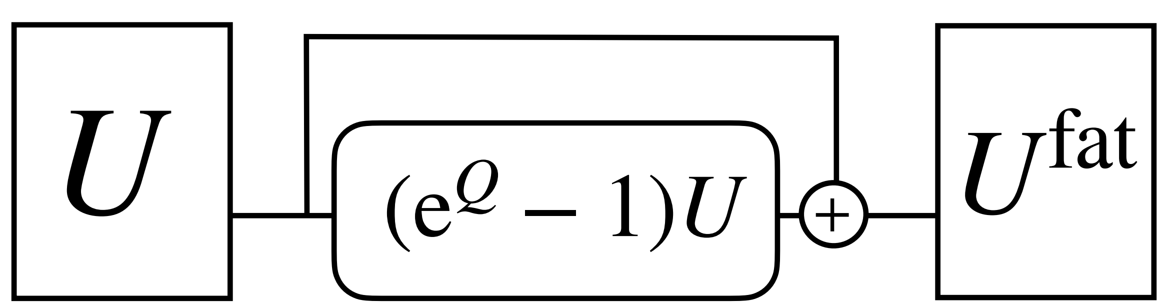
$Q$: anti-hermitian traceless plaquette

This is less obvious but this actually obeys same transformation

**Schematically,**




**In the calculation graph,**



# Smearing

## Smearing decomposes into two parts

**We can generally write smearing as**

$$U_\mu^{\text{fat}}(n) = \begin{cases} z_\mu(n) = w_1 U_\mu(n) + w_2 \mathcal{G}[U] \\ U_\mu^{\text{fat}}(n) = \mathcal{N}(z_\mu(n)) \end{cases}$$

Summation with gauge sym

A local function

# Smearing

## Smearing ~ neural network with fixed parameter!

AT Y. Nagai arXiv: 2103.11965

**We can generally write smearing as**

$$U_\mu^{\text{fat}}(n) = \begin{cases} z_\mu(n) = w_1 U_\mu(n) + w_2 \mathcal{G}[U] \\ U_\mu^{\text{fat}}(n) = \mathcal{N}(z_\mu(n)) \end{cases}$$

Summation with gauge sym

A local function

**It has similar structure with neural networks,**

$$u_i(x_j) = \begin{cases} z_i^{(l)} = \sum_j w_{ij}^{(l)} x_j + b_i^{(l)} \\ u_i = \sigma^{(l)}(z_i^{(l)}) \end{cases}$$

Affine transformation

element-wise (local)

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

**Multi-level smearing = Deep learning (with given parameters)**

**As same as the convolution, we can train weights**

# Gauge covariant neural network

## Trainable smearing

AT Y. Nagai arXiv: 2103.11965

# Gauge covariant neural network
## = trainable smearing

Akio Tomiya

AT Y. Nagai arXiv: 2103.11965

**Gauge covariant neural network = general smearing with trainable parameters**

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

(Behler-Parrinello type neural net)

(Weight "$w$" can be depend on $n$ and $\mu$ = fully connected like. Less symmetric

$$U_\mu^{\mathrm{NN}}(n)[U] = U_\mu^{(3)}(n)\left[U_\mu^{(2)}(n)\left[U_\mu^{(1)}(n)\left[U_\mu(n)\right]\right]\right]$$

Good properties: Obvious gauge symmetry. Translation, rotational symmetries.

(Analogous to convolutional layer, this fully uses information of the symmetries)

$$U_\mu(n) \mapsto U_\mu^{\mathrm{NN}}(n) = U_\mu^{\mathrm{NN}}(n)[U]$$

Gauge covariant composite function. Input = gauge field, Output = gauge field

Trainable (next page)

# Gauge covariant neural network

## Training can be done with (extended) back propagation

AT Y. Nagai arXiv: 2103.11965

**Gauge inv. loss function can be constructed by gauge invariant actions**

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

S: gauge action or fermion action

**Loss function** $$L_\theta[U] = f\left(S^{\mathrm{NN}}[U]\right)$$

$f$: mean-square for example, mini-batch

Training: We can use “gradient descent”. “Adam” (adaptive-momentum) is applicable

$$\theta^{(l)} \leftarrow \theta^{(l)} - \eta \frac{\partial L_\theta[U]}{\partial \theta^{(l)}}$$

$\theta^{(l)}$ is parameters in $l$-th layer

The second term requires the chain rule with matrix functions, we need extended delta rule

$$\frac{\partial L_\theta[U]}{\partial \theta^{(l)}} = \frac{\partial L}{\partial f} \frac{\partial f}{\partial S^{\mathrm{NN}}} \frac{\partial S^{\mathrm{NN}}}{\partial U^{(l+1)}} \frac{\partial U^{(l+1)}}{\partial z^{(l+1)}} \frac{\partial z^{(l+1)}}{\partial \theta^{(l)}}$$

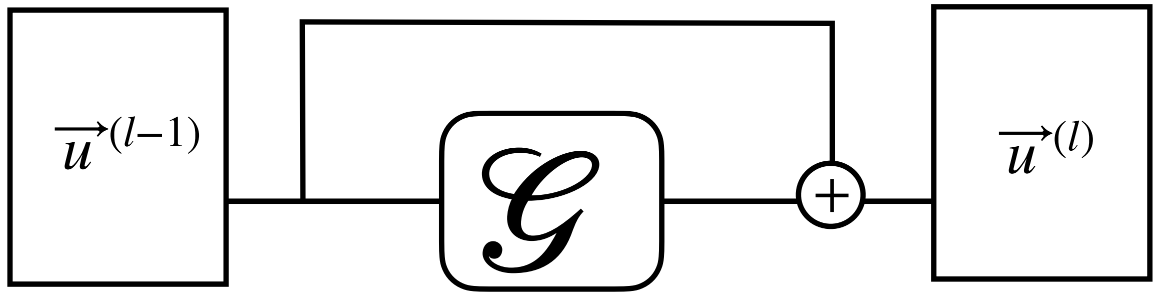
But actually, matrix derivative is common to the HMC force

(-> Extended delta rule, skipped. implementation is almost same as stout force)

# Gauge covariant neural network

## Neural ODE of Cov-Net = “gradient flow”

Res-Net




arXiv: 1512.03385

↓ Continuum Layer Limit

Neural ODE

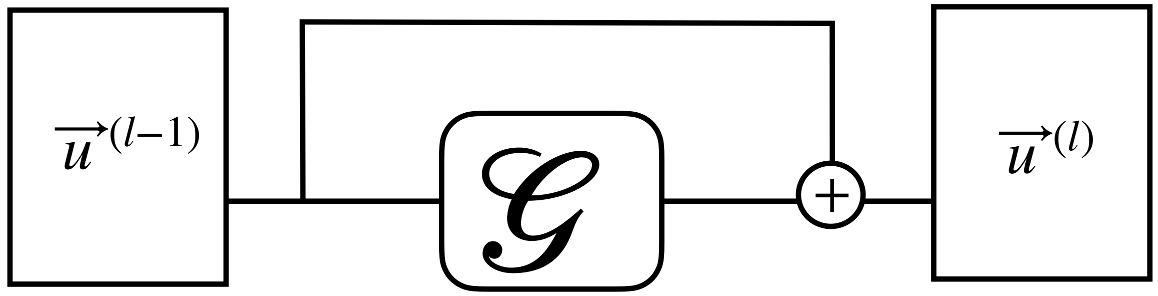
$$\frac{d\vec{u}^{(t)}}{dt} = \mathcal{G}(\vec{u}^{(t)})$$

arXiv: 1806.07366
(Neural IPS 2018 best paper)

# Gauge covariant neural network

## Neural ODE of Cov-Net = “gradient flow”

Res-Net




arXiv: 1512.03385

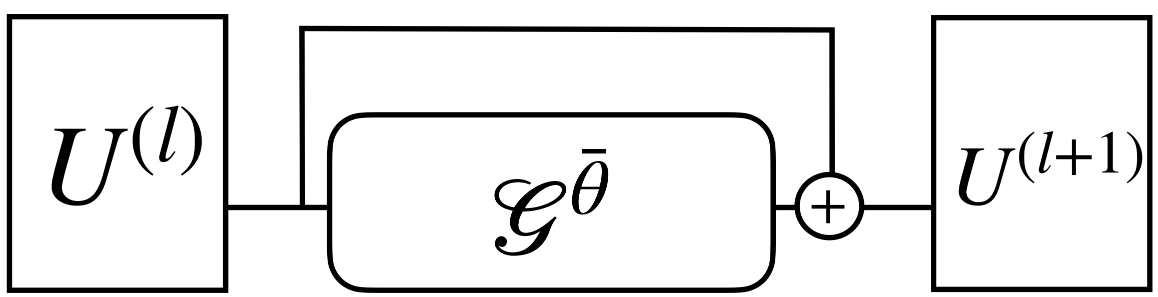
↓ Continuum Layer Limit

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 net

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

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

“Continuous stout smearing is the Gradient flow”

2010 M. Luscher

# Gauge covariant neural network

Akio Tomiya

## Short summary

| | Symmetry | Fixed parameter | Continuum limit of layers | How to Train |
|---|---|---|---|---|
| **Conventional neural network** | Convolution: Translation | Convolution: Filtering | Res-Net: Neural ODE | Delta rule and backprop Gradient opt. |
| **Gauge cov. net** AT Y. Nagai arXiv: 2103.11965 | Gauge symmetry, Translation, Rotation | Smearing | "Gradient" flow | Extended Delta rule and backprop Gradient opt. |

Next, I show a demonstration

(Q. Gauge cov. net works? Useful?)

# Demonstration

## An application for configuration generation

AT Y. Nagai arXiv: 2103.11965

# Demonstration

## Lattice path integral > 1000 dim, Trapezoidal int is impossible

K. Wilson 1974

$$S = \int d^4x \left[ + \frac{1}{2} \mathrm{tr}\, F_{\mu\nu} F_{\mu\nu} + \bar{\psi} \left( \partial\!\!\!/ - ig A\!\!\!/ + m \right) \psi \right]$$

Lattice regulation

$U_\mu = e^{aigA_\mu}$

$$S[U, \psi, \bar{\psi}] = a^4 \sum_n \left[ - \frac{1}{g^2} \mathrm{Re\ tr}\, U_{\mu\nu} + \bar{\psi} \left( D\!\!\!\!/ + m \right) \psi \right]$$

$a$ is lattice spacing(cutoff = $a^{-1}$)

Both gives same expectation value (for long range)

(They are same except for infinitely Irrelevant operators)

$$\mathrm{Re}\ U_{\mu\nu} \sim \frac{-1}{2} g^2 a^4 F_{\mu\nu}^2 + O(a^6)$$

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}U \mathcal{D}\bar{\psi} \mathcal{D}\psi e^{-S} \mathcal{O}(U) = \frac{1}{Z} \int \mathcal{D}U e^{-S_{\mathrm{gauge}}[U]} \det(D+m) \mathcal{O}(U)$$

$$= \frac{1}{Z} \int \mathcal{D}U e^{-S_{\mathrm{eff}}[U]} \mathcal{O}(U)$$

$$\mathcal{D}U = \prod_{n \in \{\mathbb{Z}/L\}^4} \prod_{\mu=1}^{4} dU_\mu(n)$$

>1000 dim. We cannot use Newton–Cotes type integral like Trapezoid, Simpson etc.

(Calculation time is longer Thant the age of the universe if one wants to control the error.)

# Demonstration

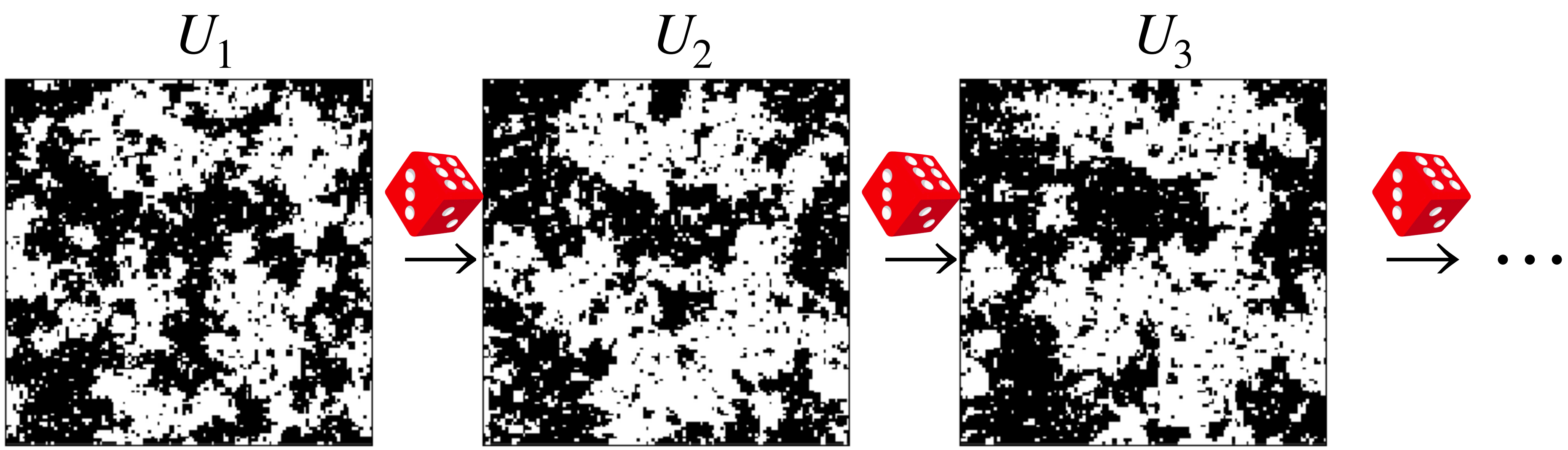
## Monte-Carlo integration is available

Akio Tomiya

M. Creutz 1980

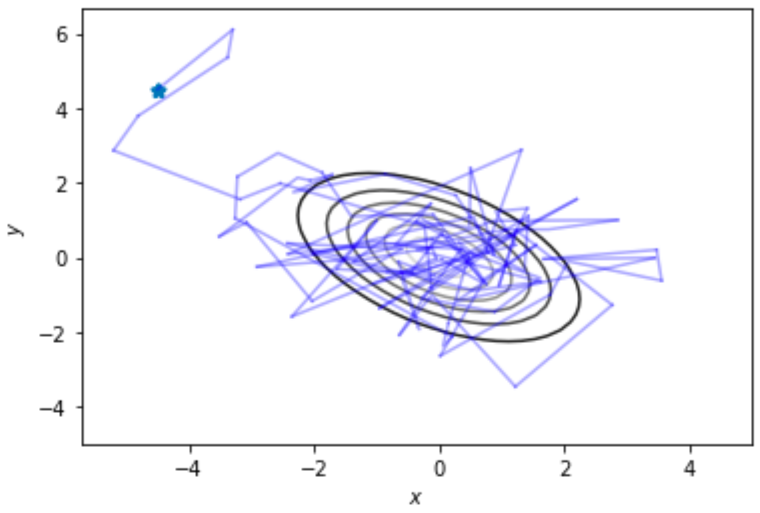
$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}U e^{-S_{\rm eff}[U]} \mathcal{O}(U) \qquad S_{\rm eff}[U] = S_{\rm gauge}[U] - \log\det(\not{D}[U] + m)$$

**Monte-Carlo: Generate field configurations with** "$P[U] = \frac{1}{Z} e^{-S_{\rm eff}[U]}$"**. It gives expectation value**



HMC: Hybrid (Hamiltonian) Monte-Carlo
De-facto standard algorithm

$$S(x, y) = \frac{1}{2}(x^2 + y^2 + xy)$$



# Demonstration

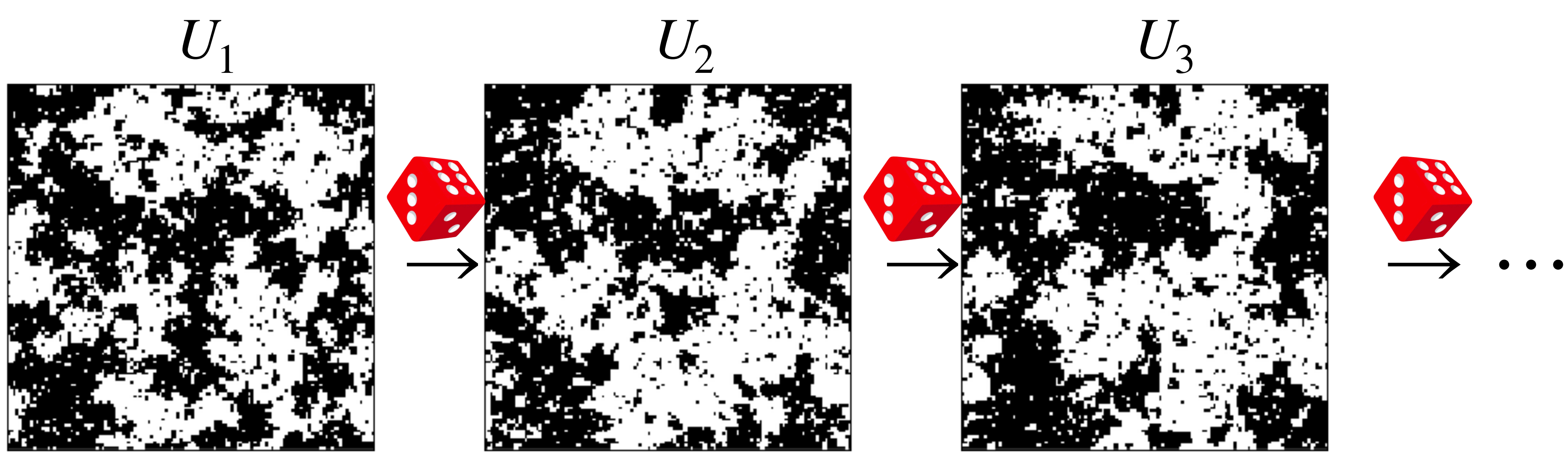
Akio Tomiya

## Monte-Carlo integration is available

M. Creutz 1980

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}U e^{-S_{\text{eff}}[U]} \mathcal{O}(U) \qquad S_{\text{eff}}[U] = S_{\text{gauge}}[U] - \log\det(\not{D}[U] + m)$$

**Monte-Carlo: Generate field configurations with "$P[U] = \frac{1}{Z} e^{-S_{\text{eff}}[U]}$". It gives expectation value**




**Error of integration is determined by the number of sampling**

$$\langle \mathcal{O} \rangle = \frac{1}{N_{\text{sample}}} \sum_{k}^{N_{\text{sample}}} \mathcal{O}[U_k] \pm O\left(\frac{1}{\sqrt{N_{\text{sample}}}}\right)$$

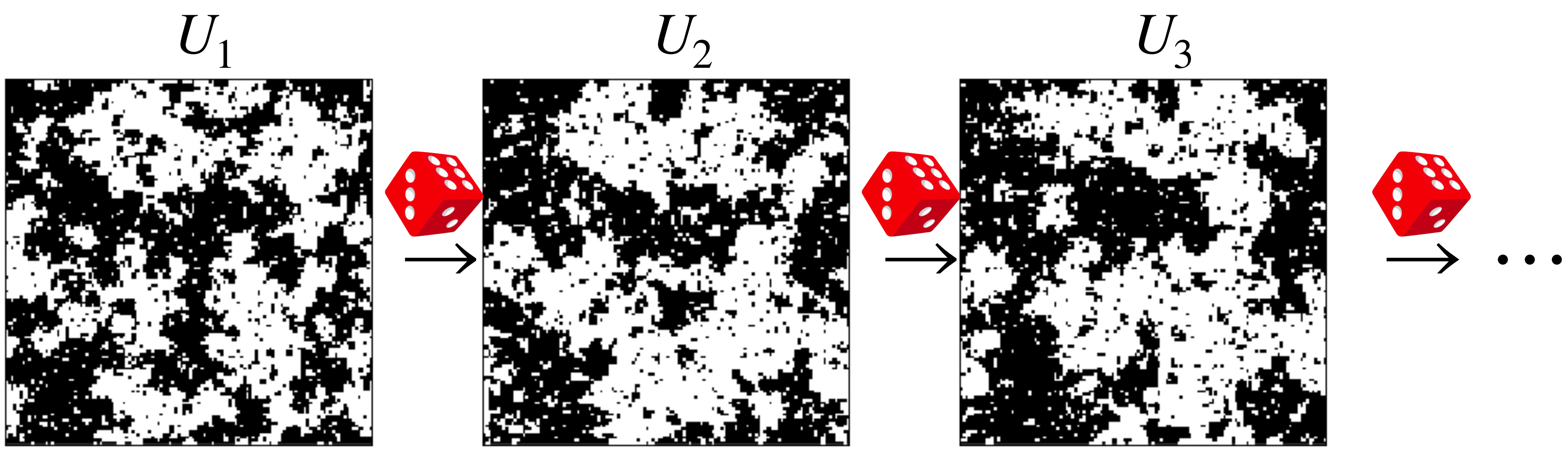
# Demonstration

## Monte-Carlo integration is available

M. Creutz 1980

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}U e^{-S_{\text{eff}}[U]} \mathcal{O}(U) \qquad S_{\text{eff}}[U] = S_{\text{gauge}}[U] - \log \det(\not{D}[U] + m)$$

**Monte-Carlo: Generate field configurations with** "$P[U] = \frac{1}{Z} e^{-S_{\text{eff}}[U]}$"**. It gives expectation value**



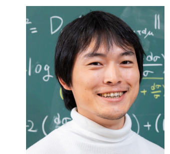

**If an algorithm is not exact (exact = average approaches to the expectation value), we cannot use the results to the other calculation for experiments**

**However, a neural network is an approximator**
**How can it be exact?**

# Demonstration
## Configuration generation with machine learning

## Some history:

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




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

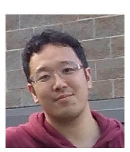
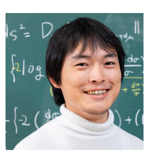
**GAN (Generative adversarial network ): 2d scalar** J. Pawlowski+ 2018 G. Endrodi+ 2018

Results look OK. No proof of exactness (impossible?)

**Flow based model: 2d scalar, pure U(1), pure SU(N)** MIT+ Google Brain 2019 …

Mimicking a trvializing map using a neural net which is reversible and has tractable Jacobian.
Exact algorithm, no dynamical fermions. Gauge equivariant layers. SU(N) is treated with diagonalization. All in 2d.

**Self-learning Monte Carlo 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 (next page)** 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. Cheaper than the previous SLMC work

# Demonstration

## Problems to solve

arXiv: 2103.11965

**Target** **Two color QCD (plaquette + staggered(not rooted) )**

**(Artificial) Q1:**
**Can we perform simulation of QCD using different action form the target (but variational parametrized)?**

**Q2: To get non-zero acceptance, the training must be done successfully. It is possible?**

**HMC: Molecular dynamics + Metropolis test**
**SLHMC: Molecular dynamics (parametrized action) + Metropolis test**

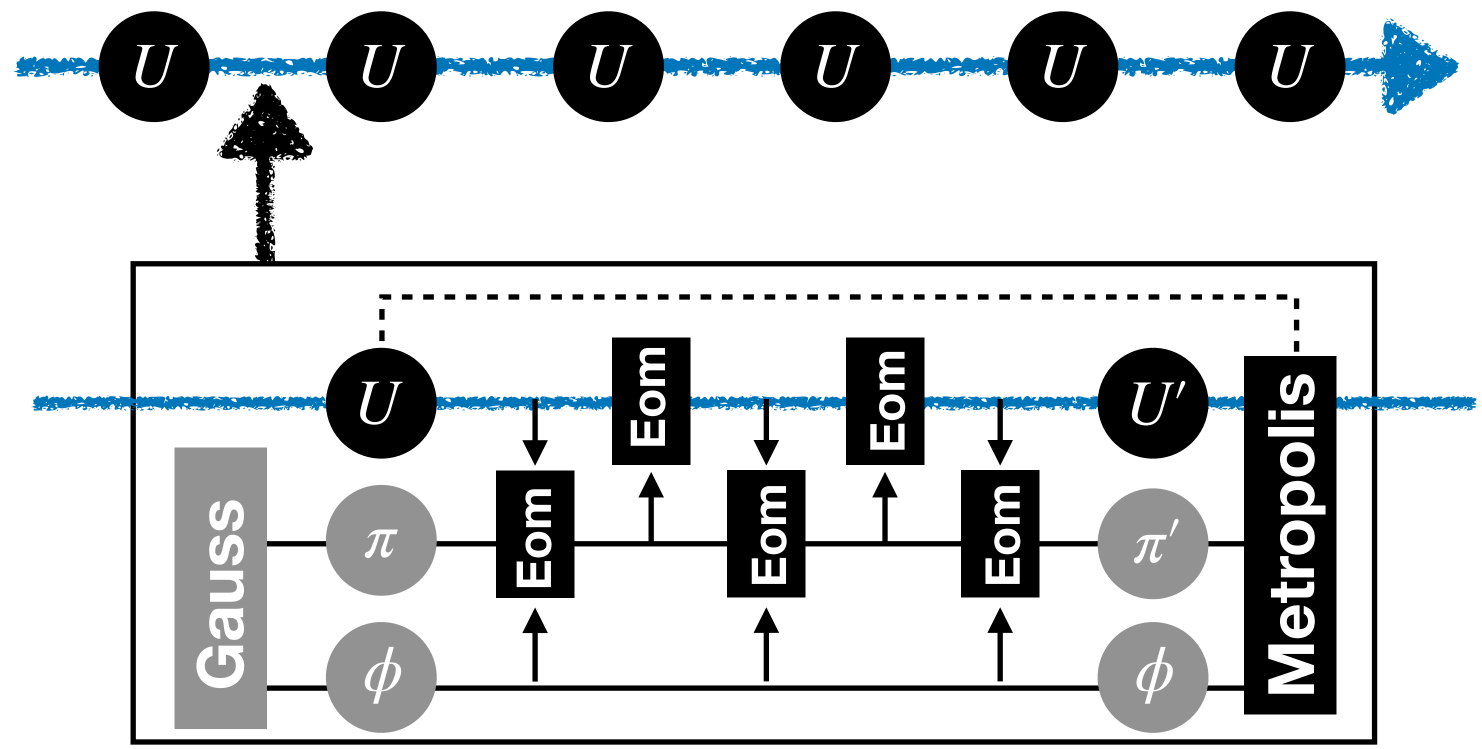
Akio Tomiya

# Gauge covariant net& SLHMC

## SLHMC for gauge system with dynamical fermions

arXiv: 2103.11965 and reference therein

**HMC**



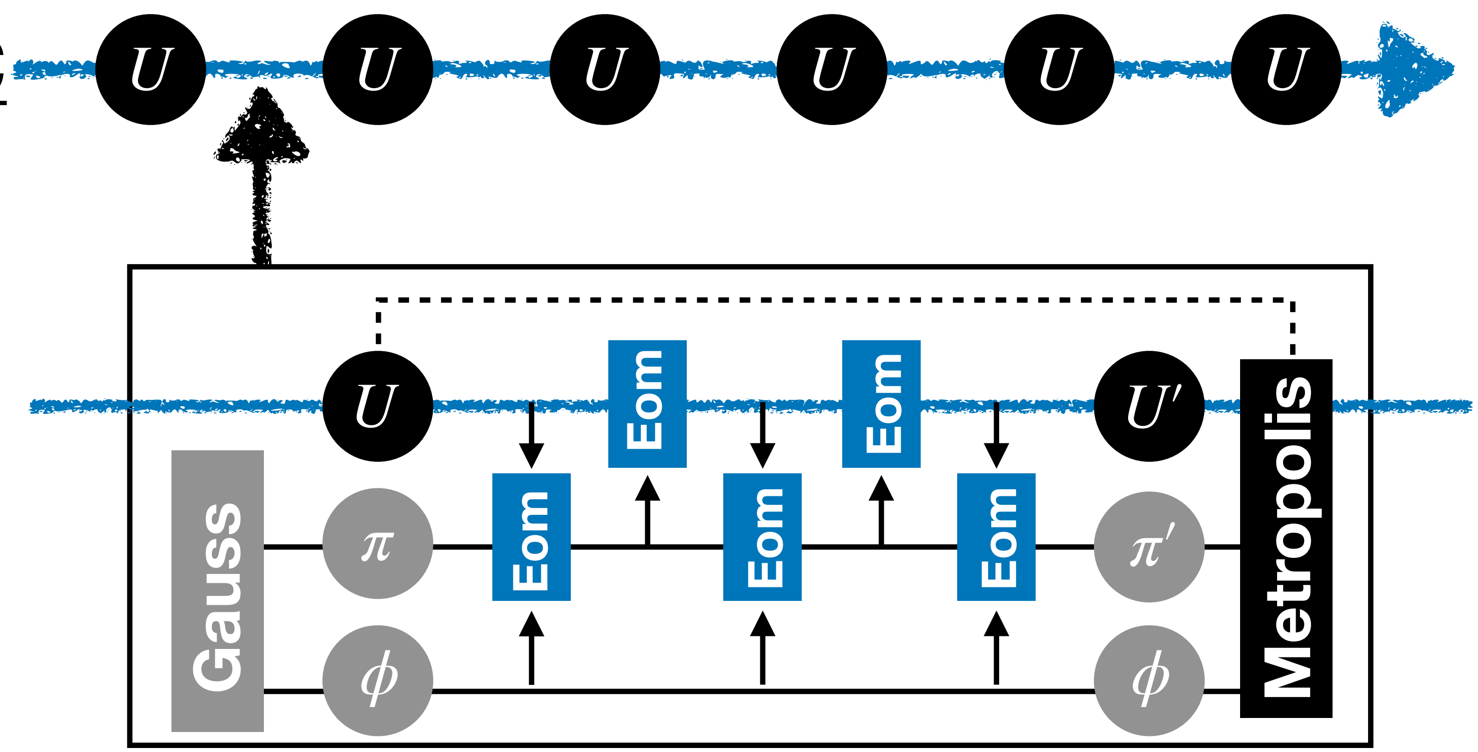

**Eom** **Metropolis**

**Both use**

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

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

**SLHMC**




**Metropolis**

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

**Eom**

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

Neural net approximated fermion action but exact

# Demonstration

## Lattice setup and question

arXiv: 2103.11965

**Target** Two color QCD (plaquette + staggered(not rooted) )

**Algorithms** SLHMC, HMC (comparison)

**Parameter** L=4, m = 0.3, beta = 2.7

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

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

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

**Code** Fully written in Julia LatticeQCD.jl AT+ (in prep)

(But we added some function on the public version)

# Demonstration

## Network: trainable stout (plaq+poly)

arXiv: 2103.11965

**Structure of NN**
**(Polyakov loop+plaq**
**In the stout smearing**
**Reducing rot. sym.)**

We randomly choose this NN.
We can do better.

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

All $\rho$ is weight
$O$ meas an loop operator

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

TA: Traceless, anti-hermitian operation

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

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

2- layered stout
with 6 trainable parameters

**Neural network**
**Parametrized action:**

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

Action is a function of a gauge field
We realize it with NN

**Loss function:**

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

**Training strategy:**

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

# Demonstration

Akio Tomiya

## Results: Loss decreases along with the training

arXiv: 2103.11965

**Loss function:**

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

**Prior HMC run (=training)**

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

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

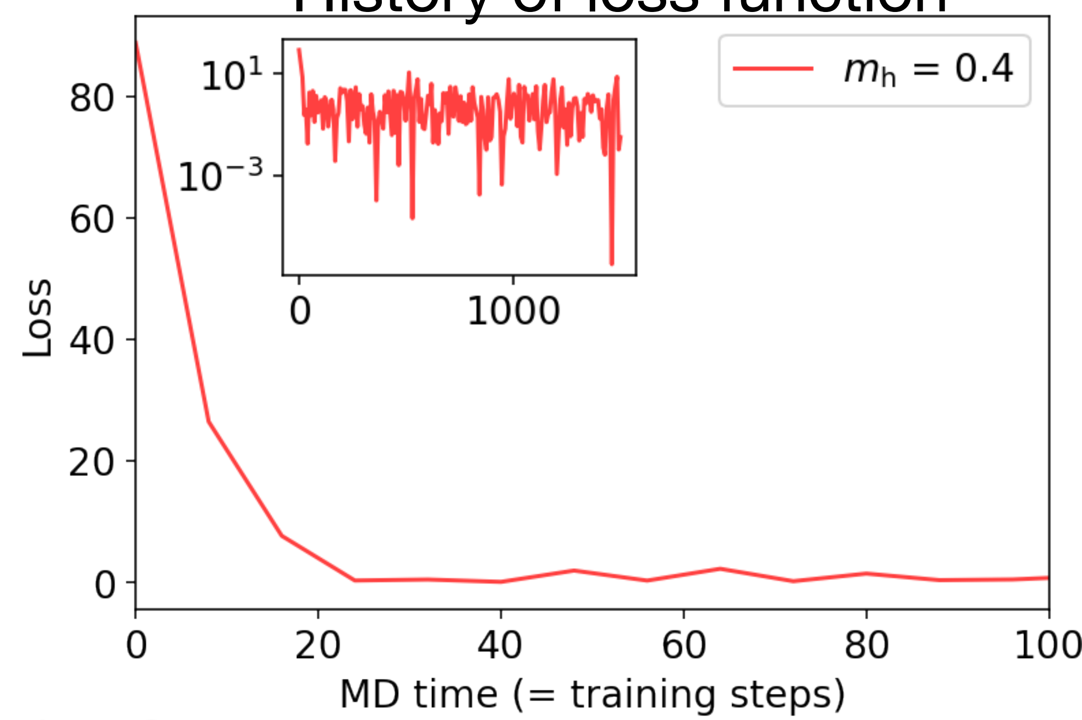
$$\frac{\partial S}{\partial \rho_i^{(l)}} = 2\,\mathrm{Re} \sum_{\mu',m} \mathrm{tr}\left[ U_{\mu'}^{(l)\dagger}(m) \Lambda_{\mu',m} \frac{\partial C}{\partial \rho_i^{(l)}} \right]$$

Ω: sum of un-traced loops

C: one U removed Ω

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

History of loss function



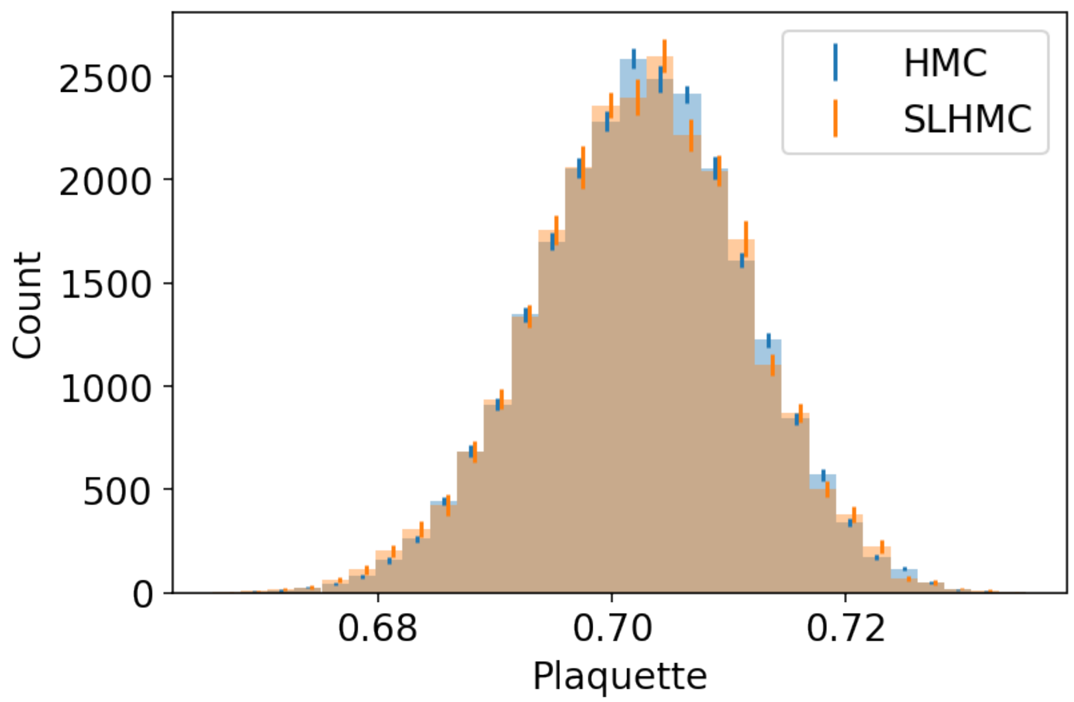

| Layer | Loop | Value of $\rho$ |
|---|---|---|
| 1 | Plaquette | −0.011146476388409423 |
| 2 | Plaquette | −0.011164492428633698 |
| 1 | Spatial Polyakov loop | −0.0030283193221172216 |
| 2 | Spatial Polyakov loop | −0.0029984533773388094 |
| 1 | Temporal Polyakov loop | 0.004248021727233112 |
| 2 | Temporal Polyakov loop | 0.004195253380373369 |

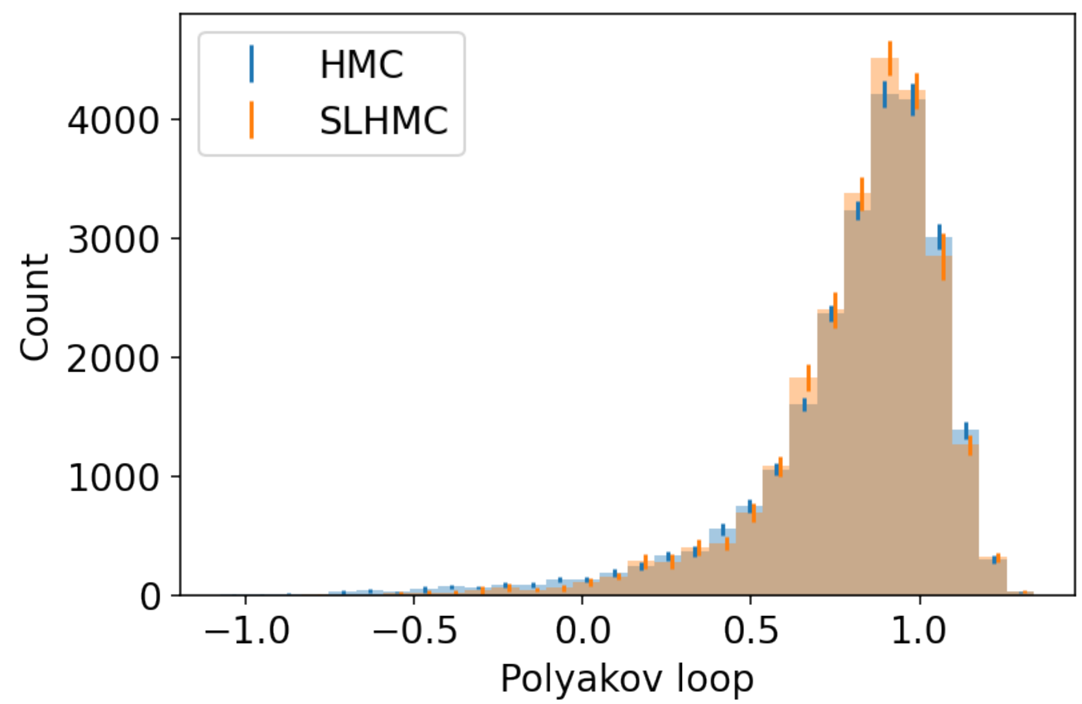
We perform SLHMC with these values!

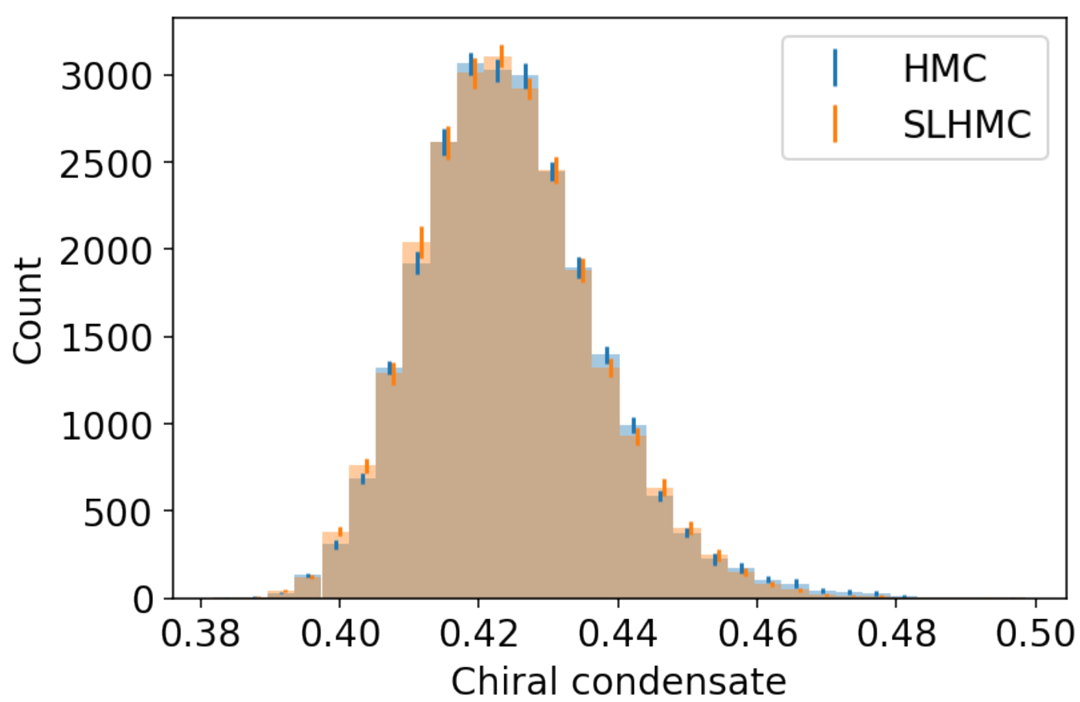
# Demonstration

## 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) |

Acceptance = 40%

# Summary and future work 1/2

## We construct and use gauge covariant neural net

Akio Tomiya

arXiv: 2103.11965

- Convolutional layers = Trainable filters
- Covariant neural network = Trainable smearing
  - We develop the delta rule for rank-2 variables(skipped)。One can implement this on a code with smeared HMC (training part is mostly common to the stout force)
  - Gauge invariant loss function
  - If we choose U(1), ape-type net, expand in $a$, stop weight sharing
    →It becomes fully connected neural net (skipped).
  - Neural ODE for covariant net = “gradient flow” (but it does not have to be a gradient)
- Self-learning HMC = HMC+ neural network parametrized molecular dynamics, exact
- Training: it has only 6parameters  but loss decreases to O(1).
- Results of SLHMC consistent with HMC. We successfully generated configurations with 4 dimensional non-abelian gauge theory with dynamical fermions

# Summary and future work 2/2

Akio Tomiya

## Future works

arXiv: 2103.11965

- Cov-net: What kind of function can it approximate? Does it have universality for deep limit?
- Cov-net: Application for machine learning? (c.f. T. Cohen et al uses data with discrete gauge sym.)
- Cov-net: Can we convert coarse configurations to finer ones? We can do same thing for images with neural nets
- Cov-net: As in (A. Tanaka AT 2016), can we define or find a new order parameter for confinement? How about topological charge estimation (Kitazawa+ 2020)？
- Cov-net: Can we construct GAN？ RBM with it?, combining flow based algortihm?
- Cov-net: Does it have interpretation like AdS/DL (K. Hashimoto 2020)？
- Cov-net: Can we construct  better1st level smearing than HISQ(Highly improved staggered quark, level-2)?
- Cov-net: neural net ~ Gradient flow. Can we use QFT techniques to neural net as  (J. Halverson+ 2020？)。
- SLHMC: S = overlap, S^NN = domain-wall fermion with neural net? It could be better than the reweighing.
- SLHMC: Improves acceptance with complicated neural network
- SLHMC: Measure topological charge in larger system. Topology changing action with neural net?