27 Jul 2021, 13:00 - 13:15 (EDT), Algorithms, 2am in JST
https://indico.cern.ch/event/1006302/contributions/4380702/

Smearing is a neural network

Akio Tomiya (RIKEN-BNL → Tenured assistant professor, Intrl. professional university of tech. in Osaka)
akio_AT_yukawa.kyoto-u.ac.jp

Yuki Nagai (Japan Atomic Energy Agency, Senior Scientist, and RIKEN AIP)

Smearing is a neural network

↑

I try to convince you this

Outline

Two topics:

2 topics in this talk

1. Introduction

---

2. Neural network, filtering and the convolution
3. Smearing
4. Gauge covariant neural network

---

5. Demo: Self-learning HMC
6. Summary

Gauge covariant network

An application

Introduction

Lattice QCD

QCD = Matrix version of quantum electro dynamics

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

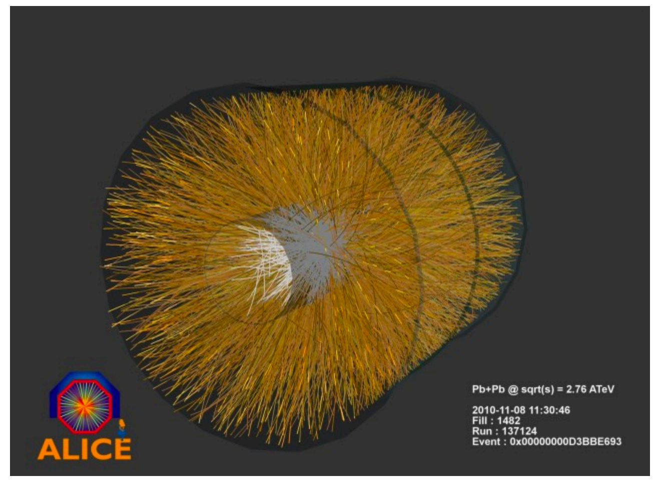
$$S = \int d^4x \left[-\frac{1}{2} \mathrm{tr}\, F_{\mu\nu}F^{\mu\nu} + \bar{\psi}\left(\mathrm{i}\partial\!\!\!/ + gA\!\!\!/ - m\right)\psi \right]$$

$$F_{\mu\nu} = \partial_\mu A_\nu - \partial_\nu A_\mu - \mathrm{i}g[A_\mu, A_\nu]$$

$A_\mu(x) \in su(3)$, 3x3traceless, harmitian

$$|\psi(t)\rangle = \mathrm{e}^{-\mathrm{i}Ht}|\psi(0)\rangle$$

H: Hamiltonian from S



- Generalization of QED, $A_\mu(x)$ is a matrix (Yang-Mills-Uchiyama)
- Action above enables us to calculate followings:
  - Tc of Quark-Hadron, Matrix elements of QCD
  - Forces between nuclei … etc!

Lattice QCD in 4 dimension

F. Wegner 1971
K. Wilson 1974

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

Lattice regulation

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

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}U\mathcal{D}\bar{\psi}\mathcal{D}\psi e^{-S} \mathcal{O}(U)$$

$$|\psi(t)\rangle = \mathrm{e}^{-H\tau}|\psi(0)\rangle$$

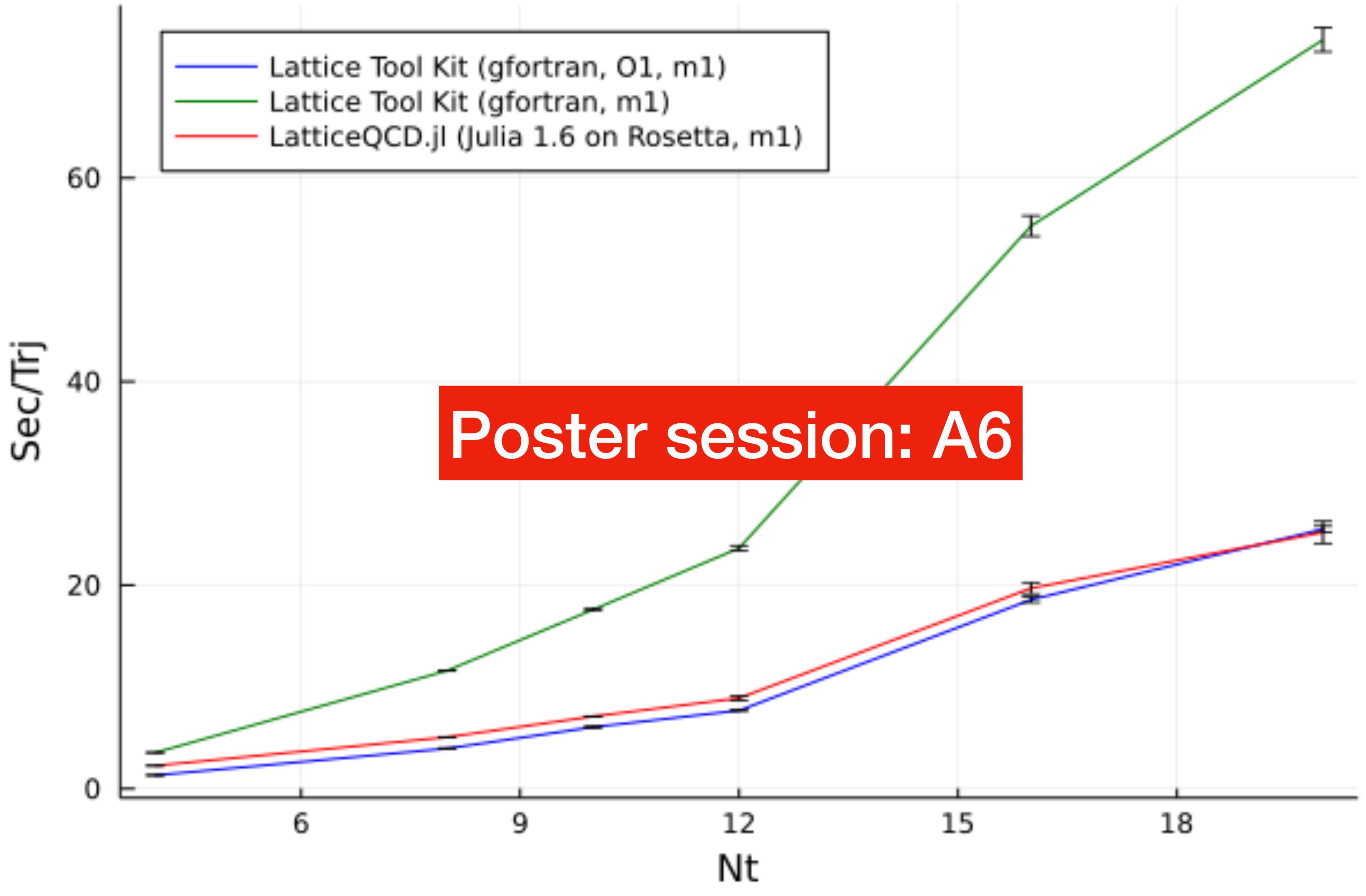
- Lattice QCD has same long-distance physics with continuum QCD
- Euclidean signature, statistic physics

My related talks

U(1)A at fin. temp by Yu Zhang, 28 Jul 2021, 05:45(EDT)
QCD + magnetic field by Xiaodang Wang, 28 Jul 2021, 22:00 (EDT)

Lattice QCD

We made a public code in Julia Language

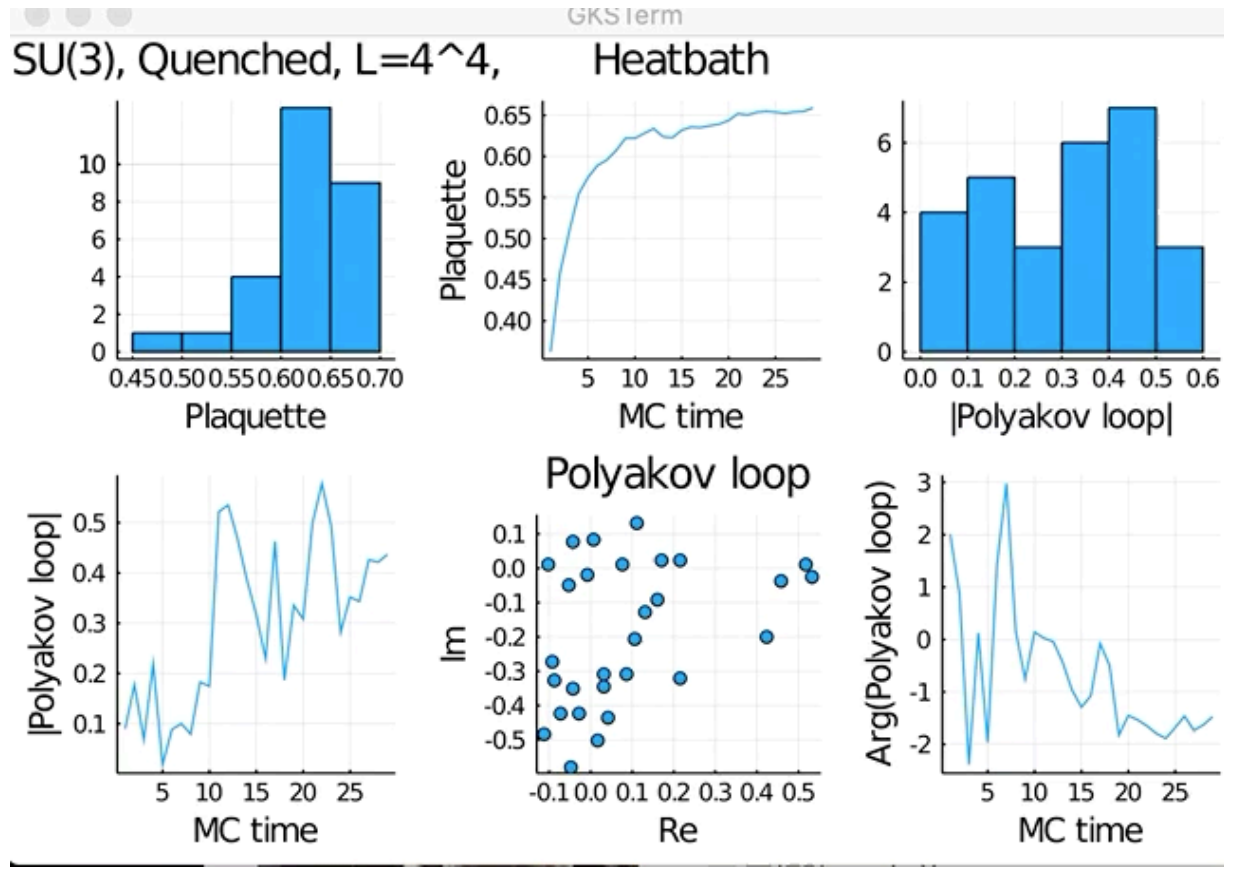


Start lattice QCD calculation?
use LatticeQCD.jl!
https://github.com/akio-tomiya/LatticeQCD.jl

- Fortran compatible speed
- Machine learning friendly architecture/Language
- Working on supercomputer/laptop/Google-colab


AT & Y. Nagai in prep

Start in 3 steps in 10 min



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

Machine learning?

Affine transformation + element-wise transformation

(Supervised) Machine learning
= A framework, which enables us to determine a function between two vector spaces in ansatz using data (= Fancy fit + statistical theory)

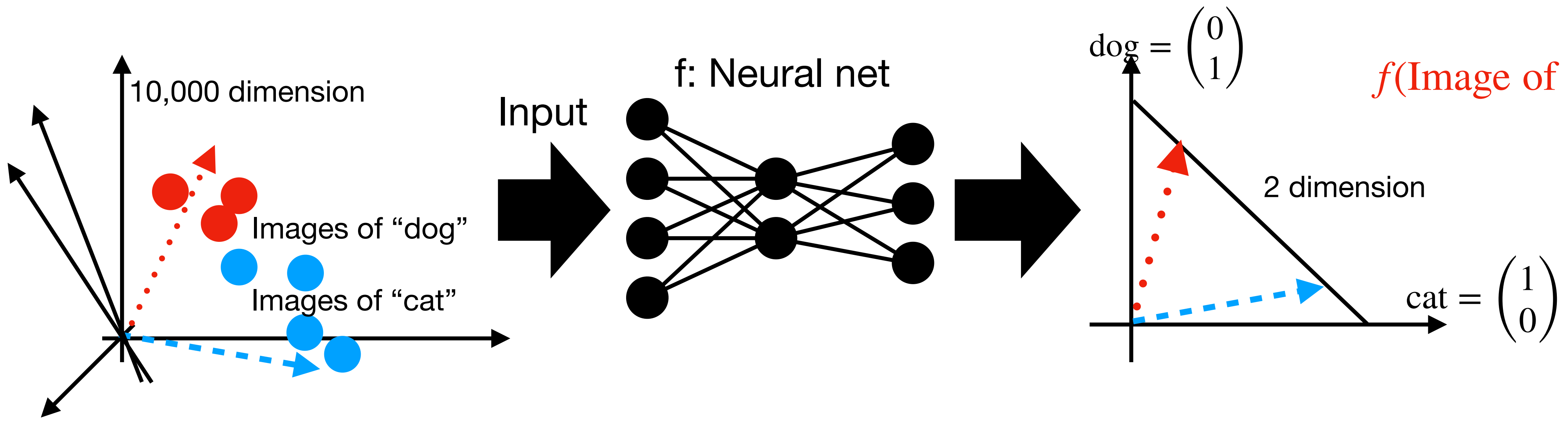
100x100

$$\xRightarrow{\text{Flatten}} \begin{pmatrix} 0.000 \\ 0.000 \\ 0.8434 \\ 0.756 \\ 0.3456 \\ \vdots \end{pmatrix}$$

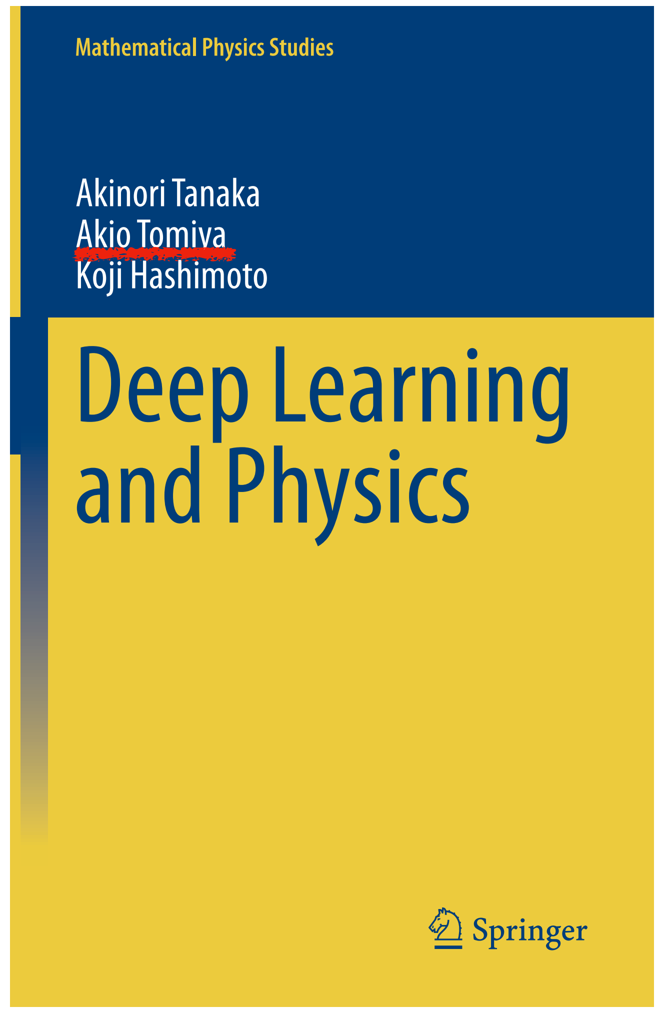
Image is a vector
(this is 10,000 dimension)

$$\text{dog} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Label is 2 dim vector
(cat = $(1, 0)^t$)



$$f(\text{Image of dog}) = \begin{pmatrix} 0.1 \\ 0.9 \end{pmatrix}$$



My related machine learning talks:

- **Poster:** Flowed HMC. Neural net represented trivializing map with HMC **Poster session: F4**
- **Poster:** 3pt function. Error evaluation of measurement with ML **Poster session: B6**

What is the neural networks?

Affine transformation + element-wise transformation

Component of neural net

$$u_i^{(l=1)} = x_i$$

$$u_i^{(l+1)}(u_i^{(l)}) = \begin{cases} z_i^{(l)} = \sum_j w_{ij}^{(l)} u_j^{(l)} + b_i^{(l)} \\ u_i^{(l+1)} = \sigma^{(l)}(z_i^{(l)}) \end{cases}$$

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

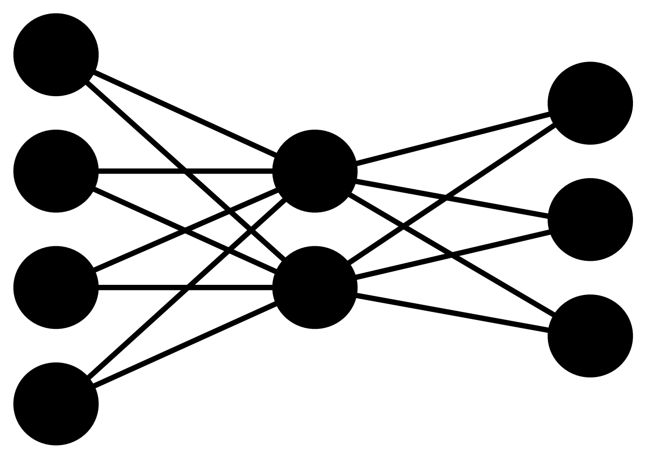
element-wise (local)
activation function ~ tanh

We can construct a neural network via stacking this.

Fully connected neural net

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

θ represents a set of parameters: eg $w_{ij}^{(l)}, b_i^{(l)}, \cdots$, (throughout this talk!)

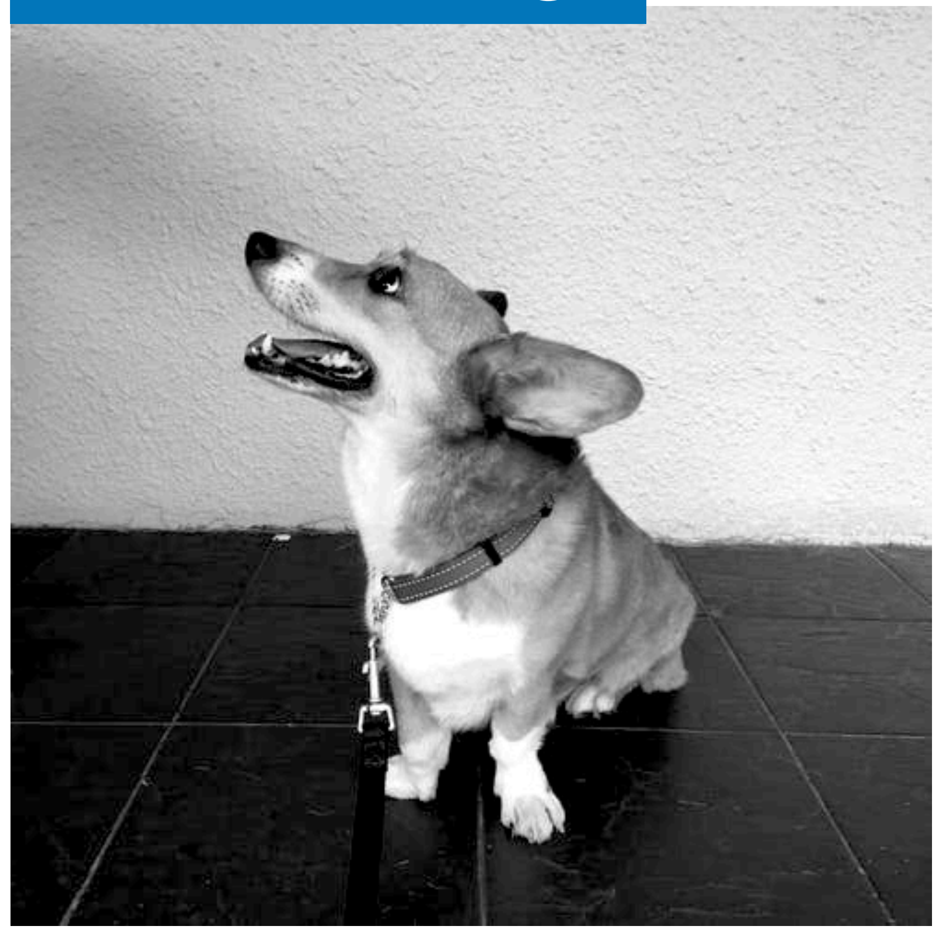


Neural network = Variational map between vector to vector

What is the neural networks?

Convolution layer = trainable filter

Filter on image



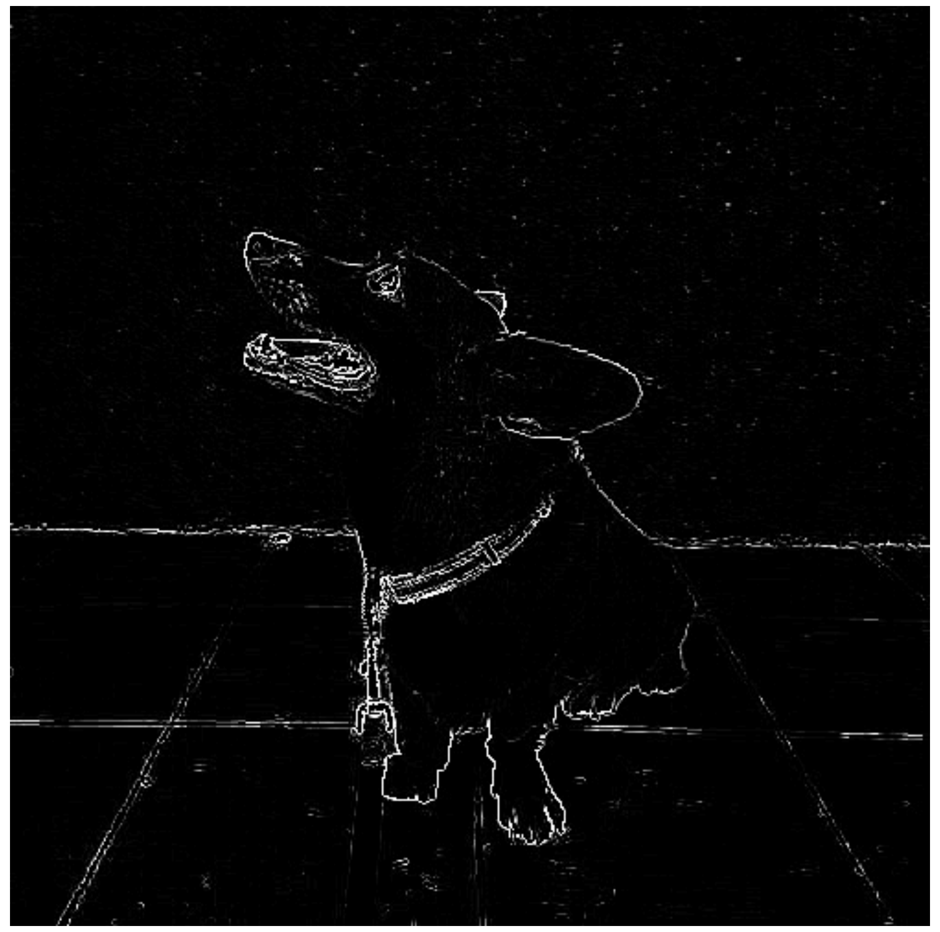
$*$

Laplacian filter

0	1	0
1	-2	1
0	1	0

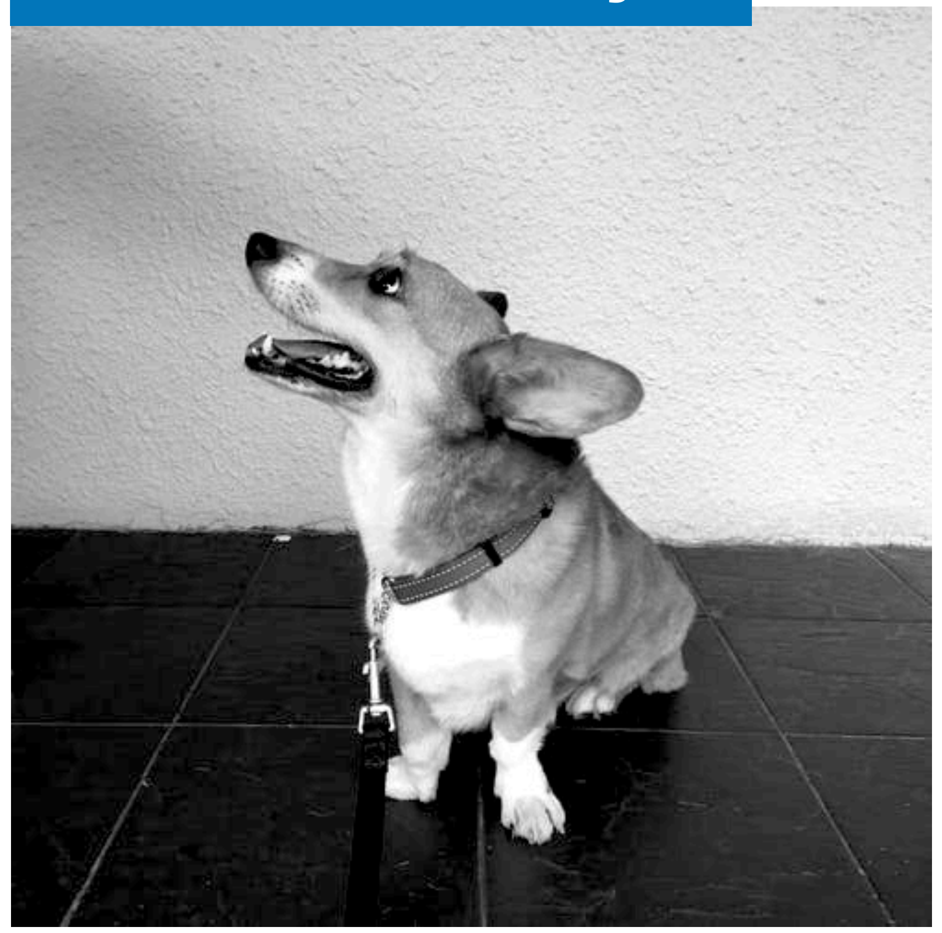
(Discretization of ∂^2)

$=$



Edge detection

Convolution layer



$*$

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)

…

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

Gaussian filter

$\frac{1}{16}$

1	2	1
2	4	1
1	2	1

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

Introduction

Machine learning makes map between data

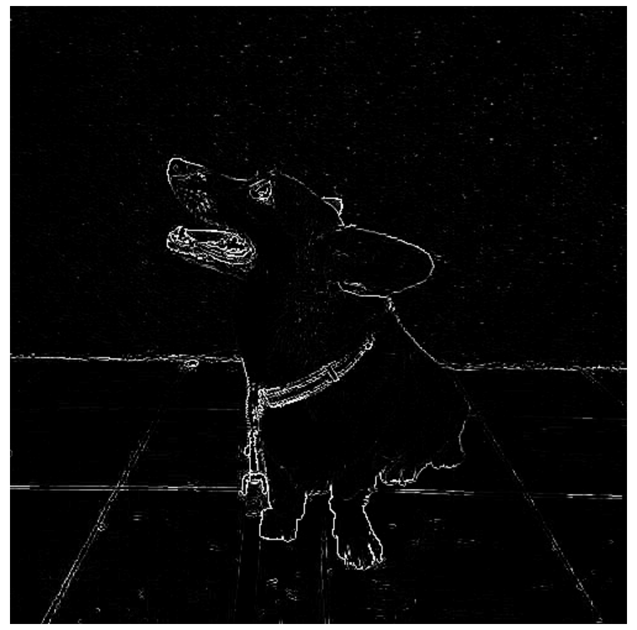
Neural network, which connects gauge field is possible?
- Keeping gauge symmetry, translation, rotation
- Compatible with Dirac operators

Convolutional layer

Trainable filter

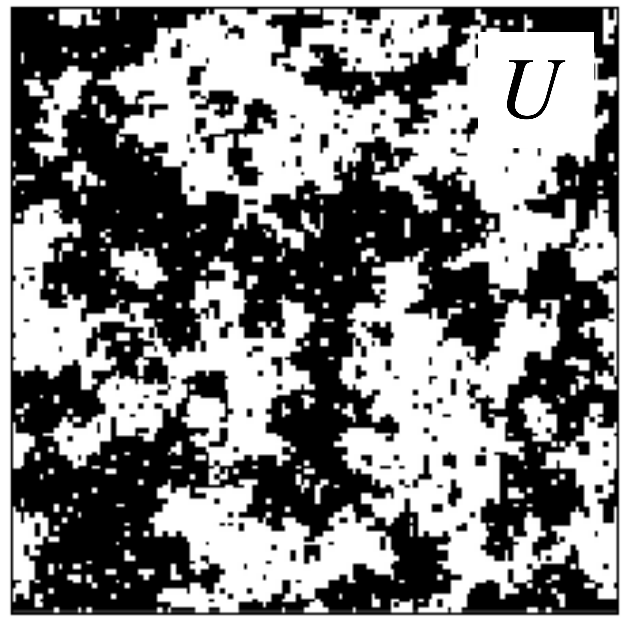
w_{11}	w_{12}	w_{13}
w_{21}	w_{22}	w_{23}
w_{31}	w_{32}	w_{33}

Parametrized map



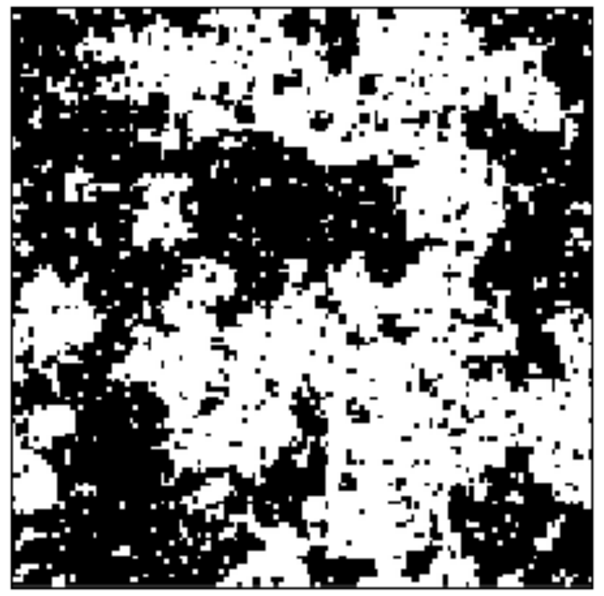
Filter is trained by backpopagation (Chain rule)

Layer for gauge field



?

Parametrized map?



How?

If we find such map, we can parametrize,
Dirac operator, Wilson loop operators (Action, topological charge) .

Smearing
Smoothing with gauge symmetry

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

Smearing

Smoothing improves global properties

Eg.

Coarse image

Numerical derivative is unstable

Gaussian filter

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

Smoothened image

Numerical derivative is stable

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

Gauge symmetric smoothing = smearing

Two types:

APE-type smearing — M. Albanese+ 1987

Stout-type smearing (next slide) — R. Hoffmann+ 2007, C. Morningster+ 2003

Smearing

Smoothing with gauge symmetry, stout type

C. Morninaster+ 2003

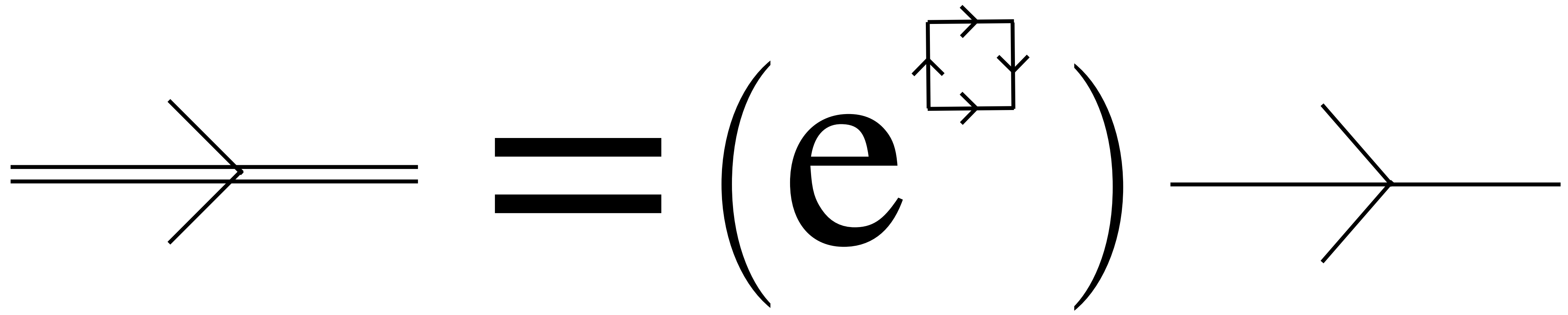
Stout-type smearing

$$U_\mu(n) \to U_\mu^{\text{fat}}(n) = \mathrm{e}^{Q} U_\mu(n)$$
$$= U_\mu(n) + (\mathrm{e}^{Q} - 1) U_\mu(n)$$

Q: anti-hermitian traceless plaquette

This is less obvious but this actually obeys same transformation

Schematically,



To improve

Dirac operators

Gluonic observables

Smearing

Smearing ~ neural network with fixed parameter!

AT Y. Nagai arXiv: 2103.11965

General form of smearing

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

Summation with gauge symmetry
(w1 & w2 are smearing parameters)
Stout case, N is trivial but g is non-trivial

A local function (projection)

Smearing

Smearing ~ neural network with fixed parameter!

AT Y. Nagai arXiv: 2103.11965

General form of smearing

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

Summation with gauge symmetry
(w1 & w2 are smearing parameters)
Stout case, N is trivial but g is non-trivial

A local function (projection)

It has similar structure with conventional neural networks!

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

Affine transformation

element-wise (local)

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

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

As same as the convolution, we can train weights (How?)

Gauge covariant neural network

Trainable smearing

AT Y. Nagai arXiv: 2103.11965

Gauge covariant neural network

Trainable Smearing = gauge covariant neural network

Gauge covariant neural network = general smearing with trainable parameters

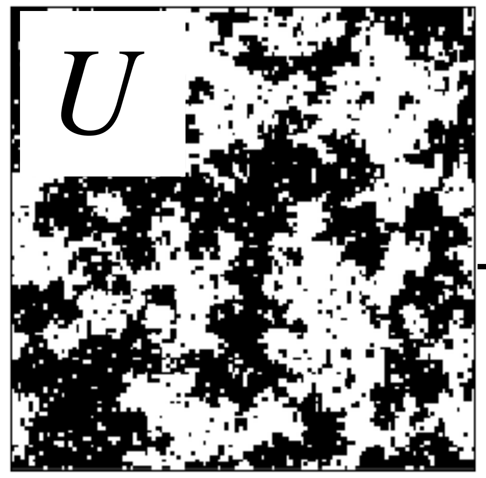
$$U_\mu^{(l+1)}(n)\left[U^{(l)}\right] = \begin{cases} z_\mu^{(l+1)}(n) = w_1^{(l)} U_\mu^{(l)}(n) + w_2^{(l)} \mathscr{G}_{\bar\theta}^{(l)}[U] \\ \mathcal{N}(z_\mu^{(l+1)}(n)) \end{cases}$$

(Weight "w" can be depend on n and μ = fully connected like. Less symmetric, more parameters)

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

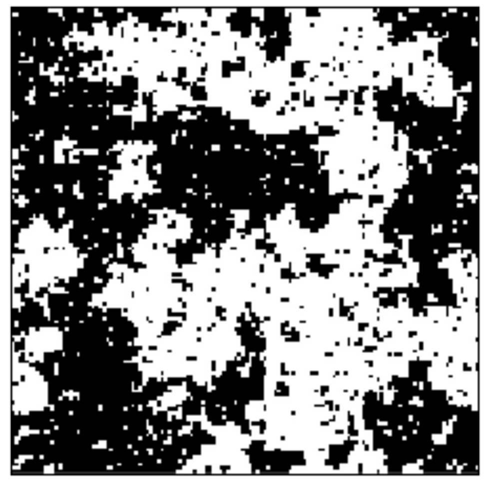
This is trainable!
(Later)

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



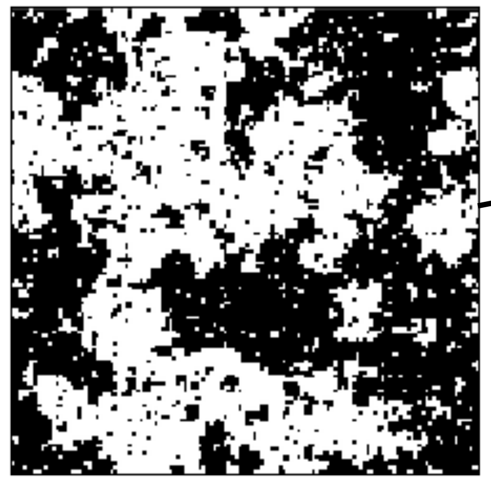
$U_\mu^{(1)}(n)[U]$

Parametrized map



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

Parametrized map



Construct loops $W\left[U_\mu^{\mathrm{NN}}(n)[U]\right]$ Parametrized Wilson loop ~ variational gauge Action

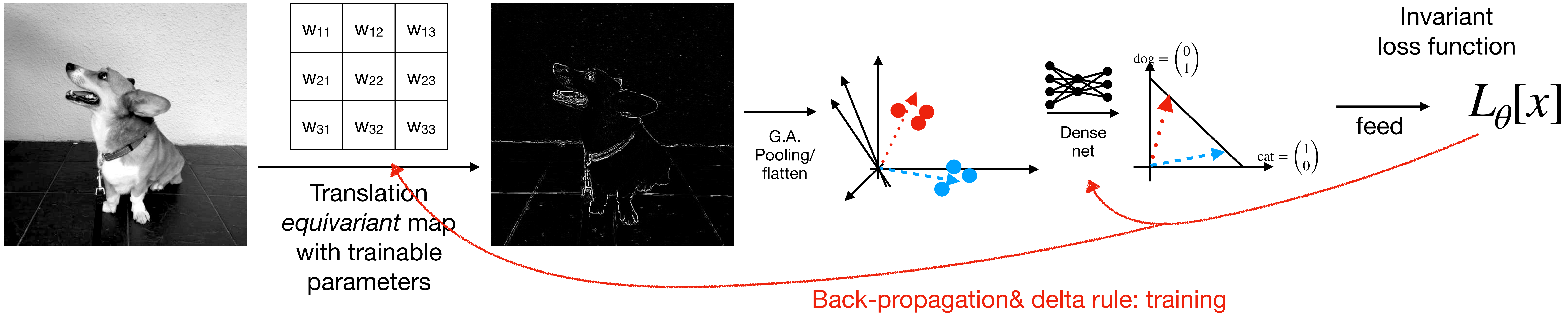
Feed to Dirac op $D\left[U_\mu^{\mathrm{NN}}(n)[U]\right]$ Parametrized Dirac operator ~ variational quark action

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

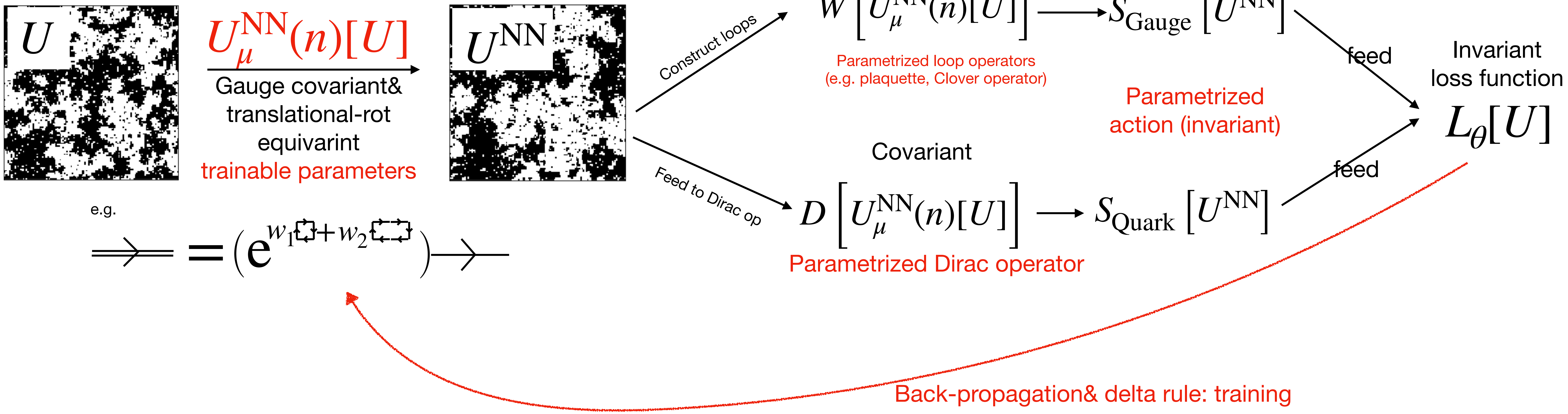
Gauge inv. loss function can be constructed by gauge invariant actions

AT Y. Nagai arXiv: 2103.11965

Convolutional neural network



Covariant neural network layer (A Use case)



cf. Gauge equivariant neural net (M Favoni+)

Gauge covariant neural network

Training can be done with chain rule as stout force

We can train the weight using (extended) delta rule, as same as conventional neural nets!

Loss function $$L_\theta[U] = f(S[U])$$

f: Some well know function (e.g. square difference)

(c.f. Behler-Parrinello type neural net)

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

Training
(until converge)

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

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

The second term requires the chain rule for matrix fields, we need extended delta rule:

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

This matrix derivative is common to the stout force

(-> Extended delta rule, see our paper)

Gauge covariant neural network

Extension of conventional neural network for gauge theory

	Symmetry	Fixed parameter	Continuum limit of layers	How to Train
Usual neural network	Convolution: Translation	Convolution: Filtering (e.g Gaussian/Laplasian)	Res-Net: Neural ODE	Delta rule and backprop Gradient opt.
Gauge covariant net AT Y. Nagai arXiv: 2103.11965	Gauge covariant Translation equivariant 90° rotation equivariant	Smearing	<u>"Gradient" flow</u>	Extended Delta rule and backprop Gradient opt.

Re-usable stout force subroutine
(Implementation is easy & no need to use ML library)

Next, I show a demonstration
(Q. Gauge covariant net works?)

Demonstration of cov. net

A use case

AT Y. Nagai arXiv: 2103.11965

Demonstration of cov. net

A lot of works for machine learning + lattice field theory

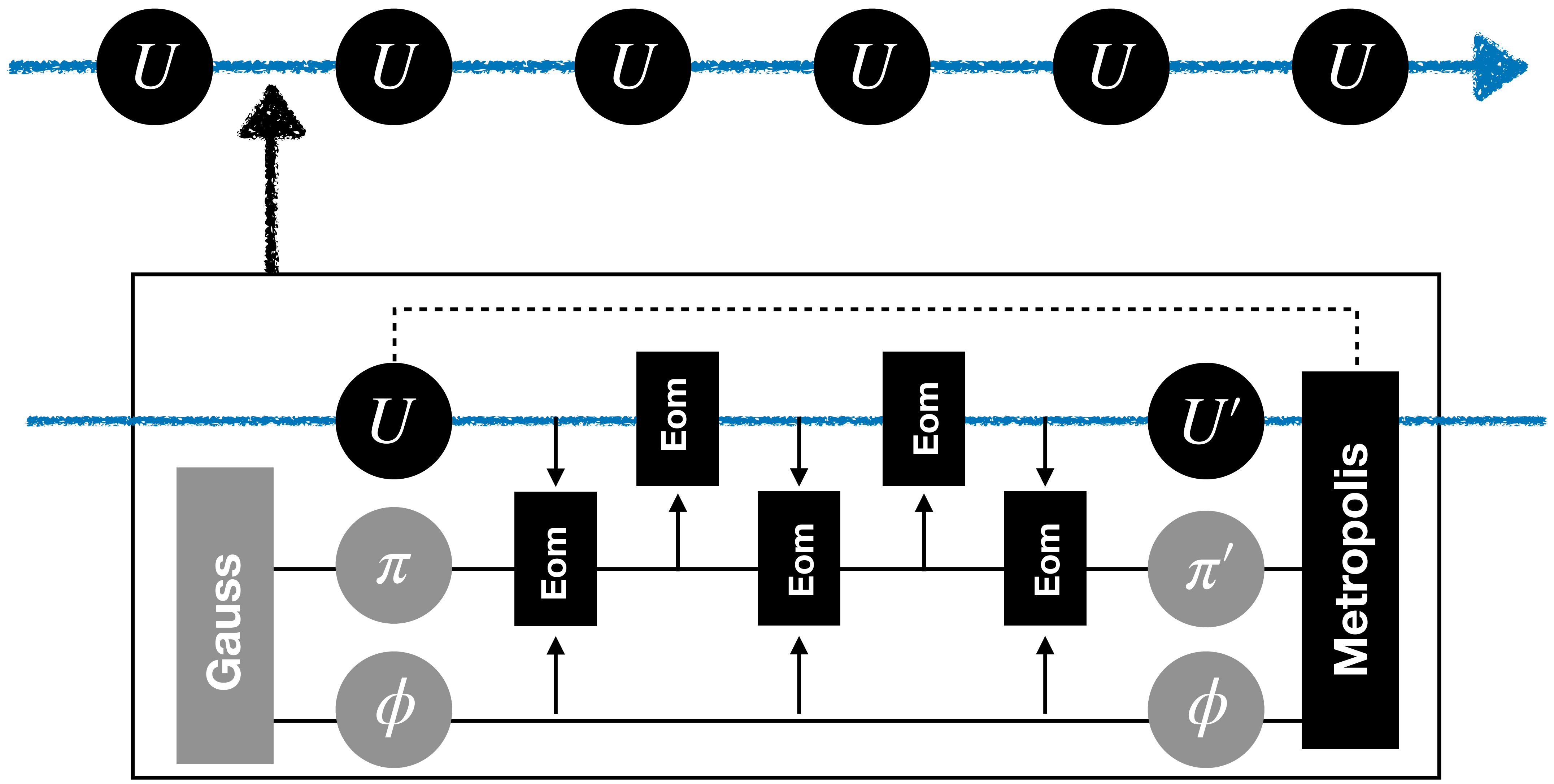
Year	Group	ML	Dim.	Theory	Gauge sym	Exact?	Fermion?	Lattice2021/ref
2017	AT+	RBM + HMC	2d	Scalar	-	No	No	arXiv: 1712.03893
2018	K. Zhou+	GAN	2d	Scalar	-	No	No	arXiv: 1810.12879
2018	J. Pawlowski +	GAN+HMC	2d	Scalar	-	Yes?	No	arXiv: 1811.03533
2019	MIT+	Flow	2d	Scalar	-	Yes	No	arXiv: 1904.12072
2020	MIT+	Flow	2d	2d U(1)	Equivariant	Yes	No	arXiv: 2003.06413
2020	MIT+	Flow	2d	2d SU(N)	Equivariant	Yes	No	arXiv: 2008.05456
2020	AT+	SLMC	4d	SU(N)	Invariant	Yes	Yes	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	Algorithm 29 Jul 2021, 14:45
2021	AT+	SLHMC	4d	QCD	Covariant	Yes	Yes	This talk
2021	L. Del Debbio+	Flow	2d	Scalar, O(N)	-	Yes	No	Postar B (15:00-) 28 Jul 2021
2021	MIT+	Flow	2d	Yukawa	-	Yes	Yes	arXiv:2106.05934
	S. Foreman, AT+	Flowed HMC	2d	U(1)	Equivariant	Yes	No (but	Postar B (15:00–) 28 Jul 2021
	XY Jing	Neural net	2d	U(1)	?	Yes?	No	Algorithm 29 Jul 2021, 13:45
	D. BOYDA	Flow	2d/4d?	?	?	Yes?	No?	Algorithm 29 Jul 2021, 14:00

Demonstration of cov. net

Self-learning HMC

arXiv: 2103.11965 and reference therein

HMC (Exact)



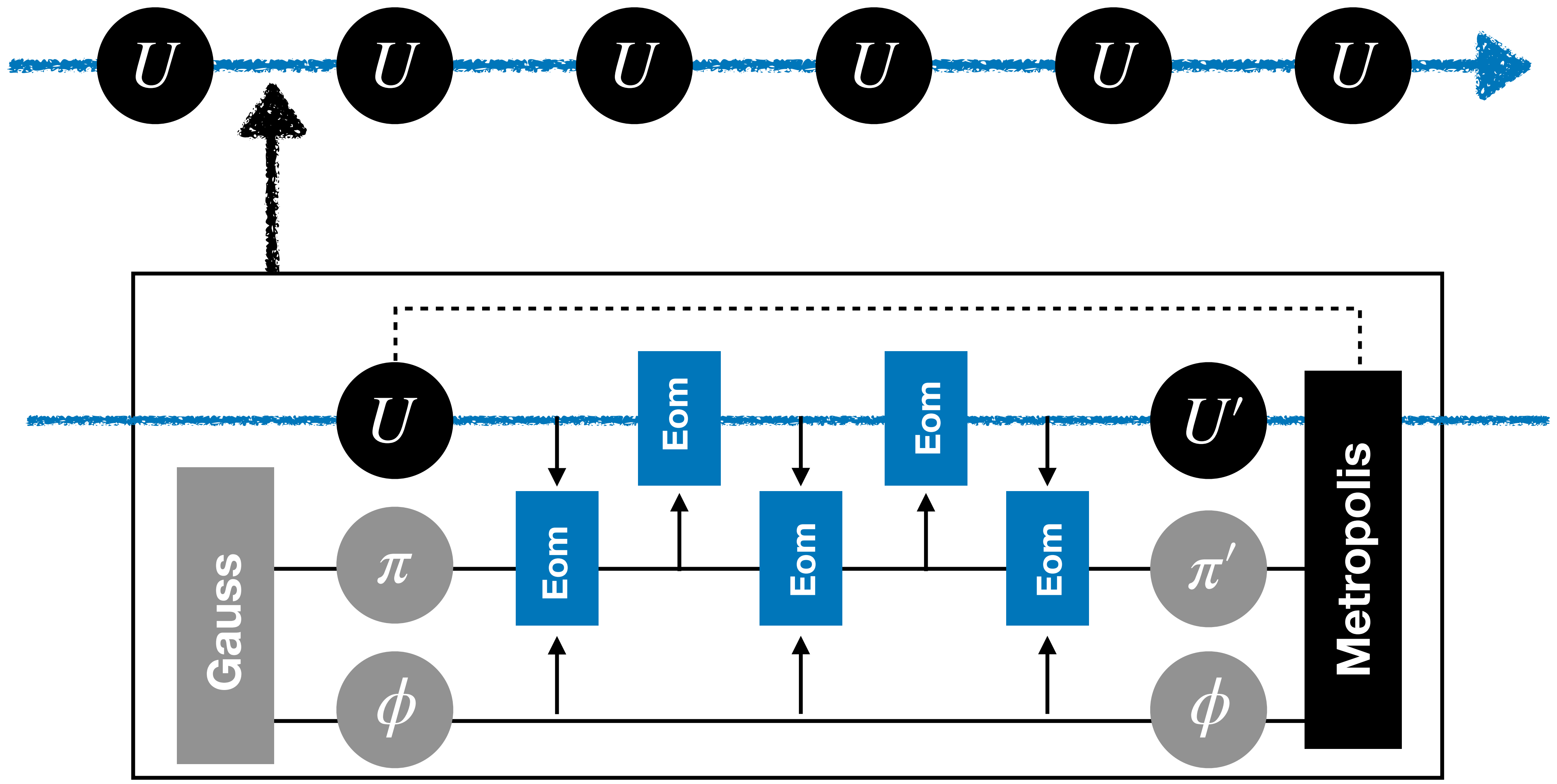
Eom | Metropolis

Both use

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

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

SLHMC (Exact)



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>

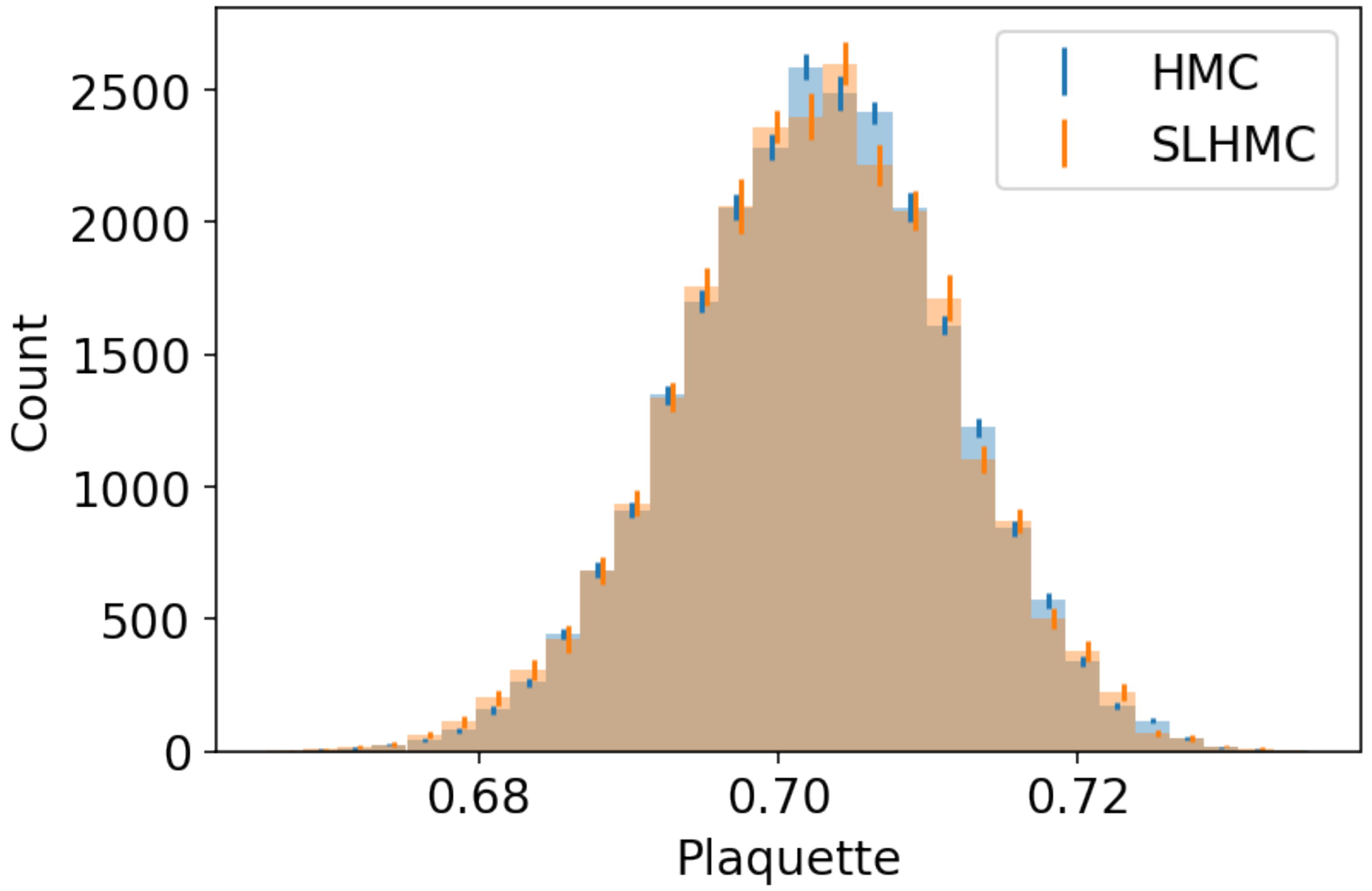
Demonstration of cov. net

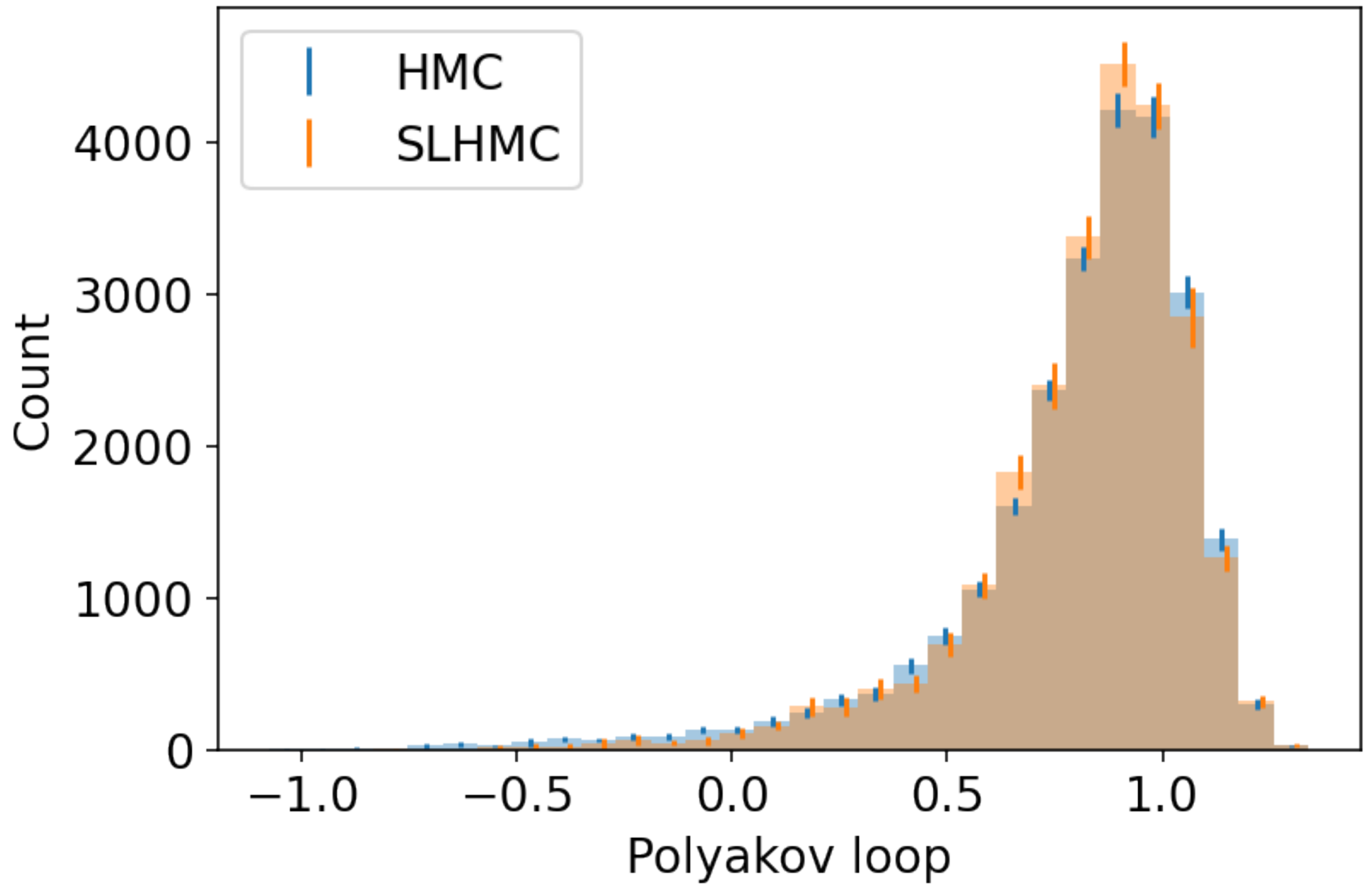
SU(2), Nf=4, 4d

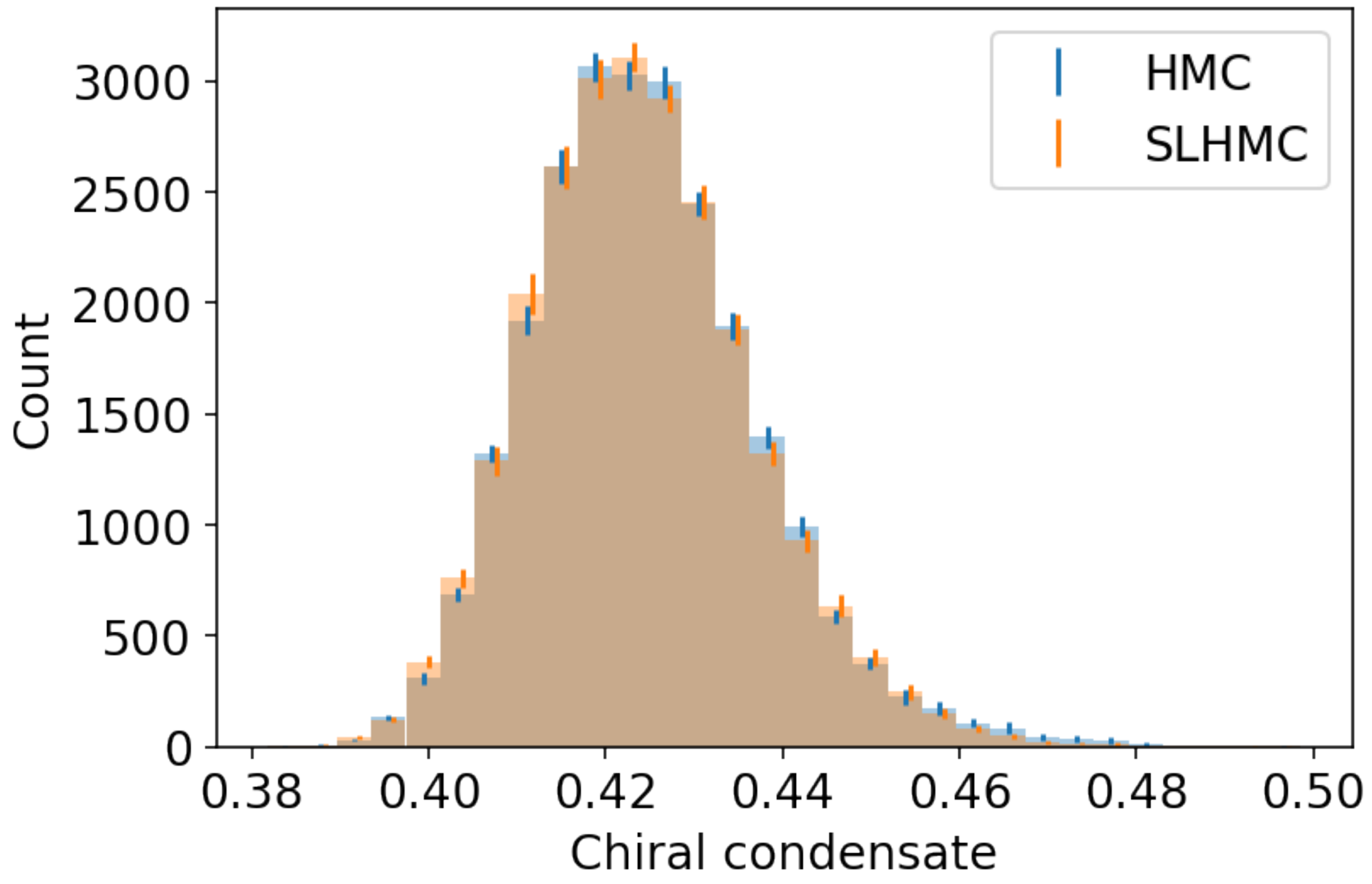
arXiv: 2103.11965

Costly Dirac operator is mimicked by cheaper Dirac operator with covariant net!

SU(2), Nf=4
Target full QC2D m = 0.3 -> Simulated by HMC
Mimicked m = 0.4 + covariant neural net -> by Self-learning HMC







Expectation value

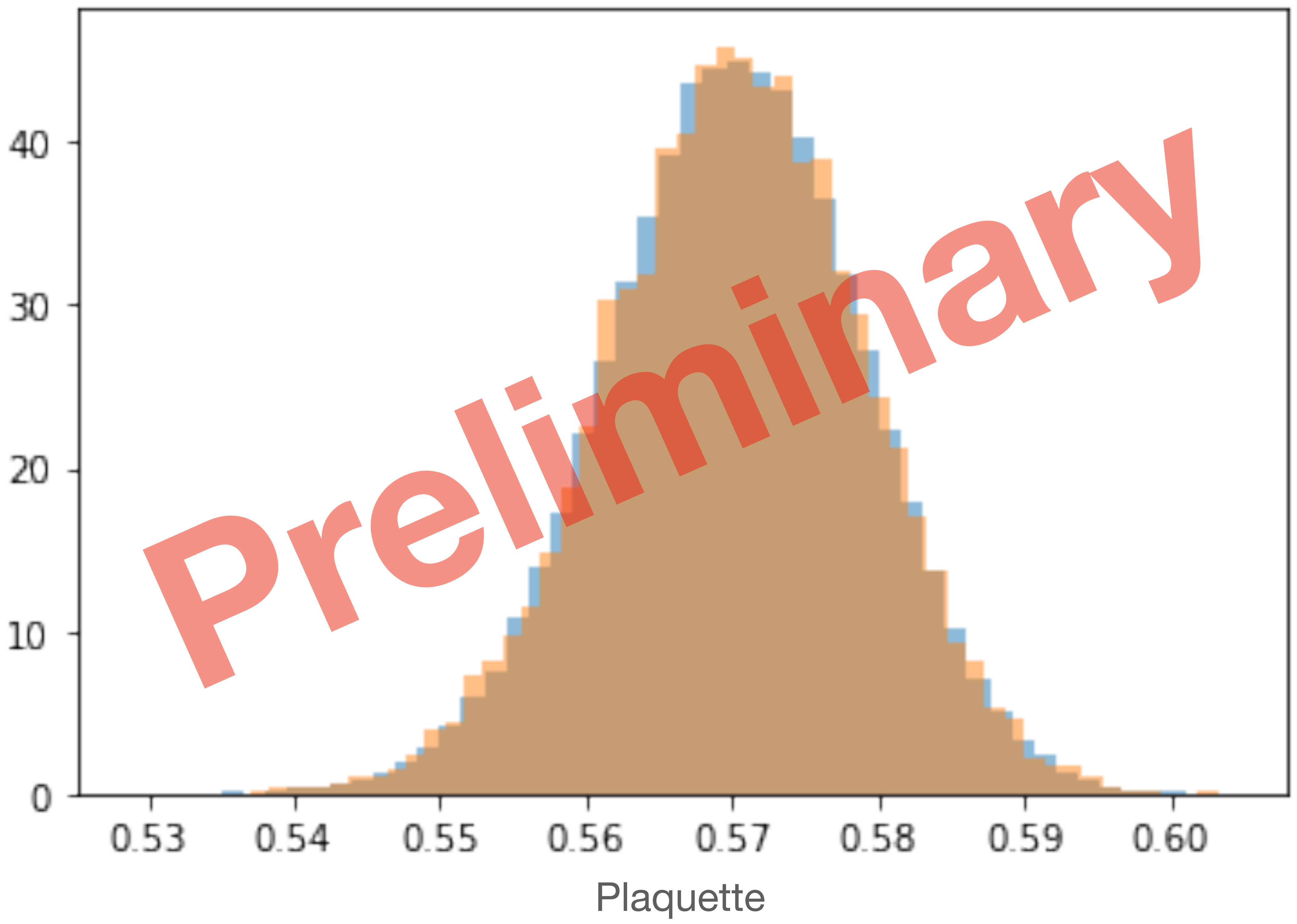
Algorithm	Observable	Value
HMC	Plaquette	0.7025(1)
SLHMC	Plaquette	0.7023(2)
HMC	|Polyakov loop|	0.82(1)
SLHMC	|Polyakov loop|	0.83(1)
HMC	Chiral condensate	0.4245(5)
SLHMC	Chiral condensate	0.4241(5)

Acceptance = 40%

Demonstration of cov. net

Update from the paper! SU(3), Nf=2

SU(3), Nf=2
Target full QCD m = 0.3 -> Simulated by HMC
Mimicked m = 0.4 + covariant neural net -> by Self-learning HMC



SU(3), dynamical fermions in 4d can be deal with machine learning now!

Summary and future work

We propose and use gauge covariant neural net

- Neural network: linear transformation + local non-linear transformations
- Trainable filter on images = convolutional neural network. Translation equivariant
- Trainable smearing on gauge fields = covariant neural network
- We can parametrize Wilson loops and gauge action in 4d
- We can parametrize Dirac operators
- Parameters can be trained by (extended) delta rule ~ stout force calc.
- The training subroutine has the same structure to the smeared force in HMC. Economically implementable
- We performed self-learning HMC for 4d QCD with parametrized Dirac operator and it works!
- Future works: combine to trivializing map? Overlap+parametrized Domain-wall simulation?

LatticeQCD.jl

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

Summary and future work

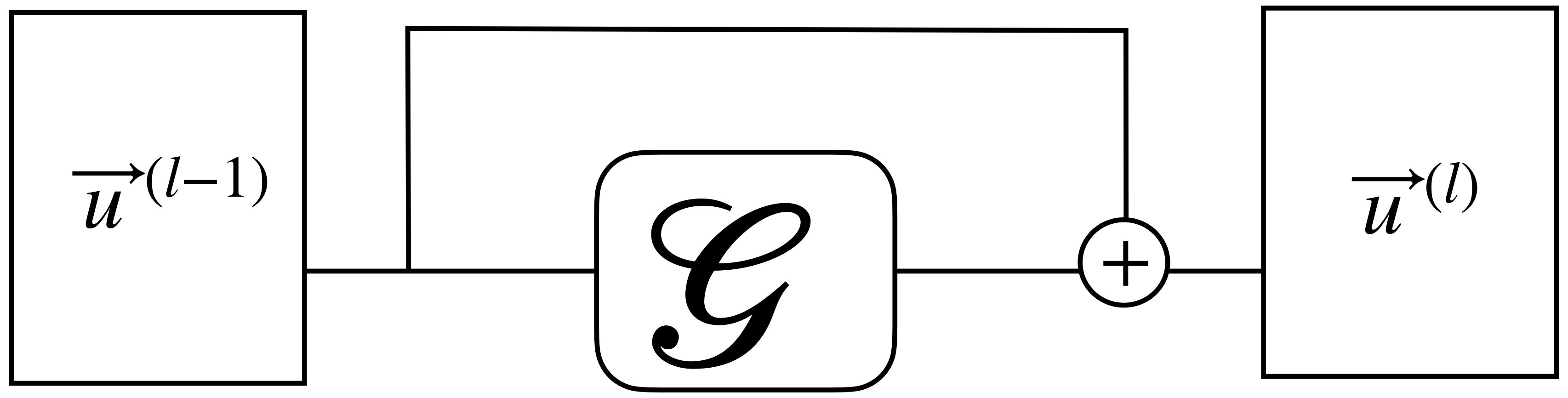
We propose and use gauge covariant neural net

- Covariant neural network = trainable smearing as Convolutional layers = trainable filters
    - We develop the delta rule for SU(N) valued link field variables (skipped). One can implement this on a code with smeared HMC. *Most of necessary subroutines are common to the stout force. No need machine learning library for training*
    - We provide how to construct a gauge invariant loss function
    - We parametrize QCD action in a gauge covariant way
    - To design network, we can use intuition of the system (physicists friendly)
    - 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
- We performed simulations with the covariant neural network parametrized action
    - Training: it has only 6 parameters 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 with parametrized action
- **Future works:** Combine with flow based? Overlap/Domain-wall simulation?

Smearing

Smearing decomposes into two parts

Res-Net



arXiv: 1512.03385

Continuum Layer Limit

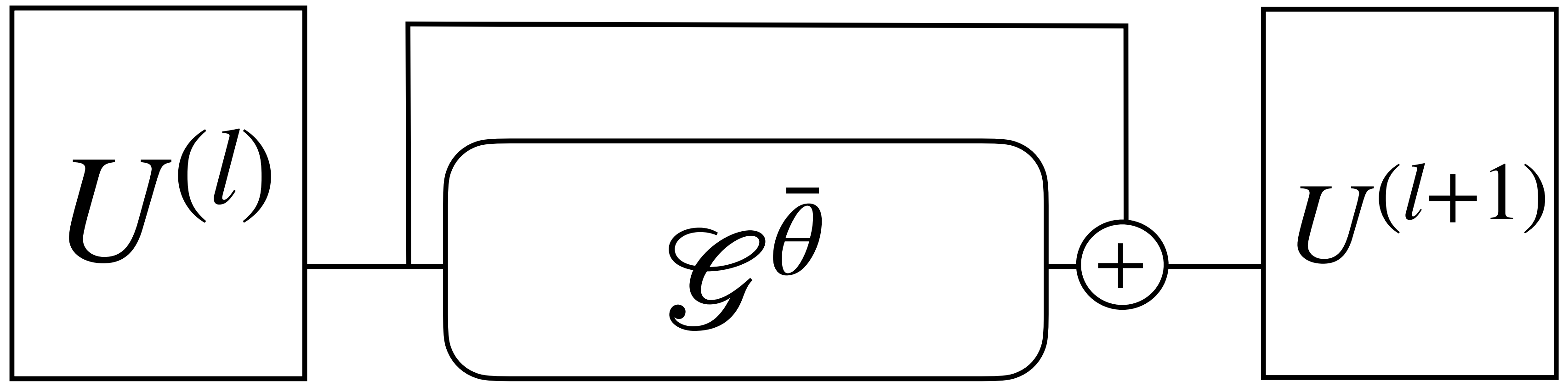
Neural ODE

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

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

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

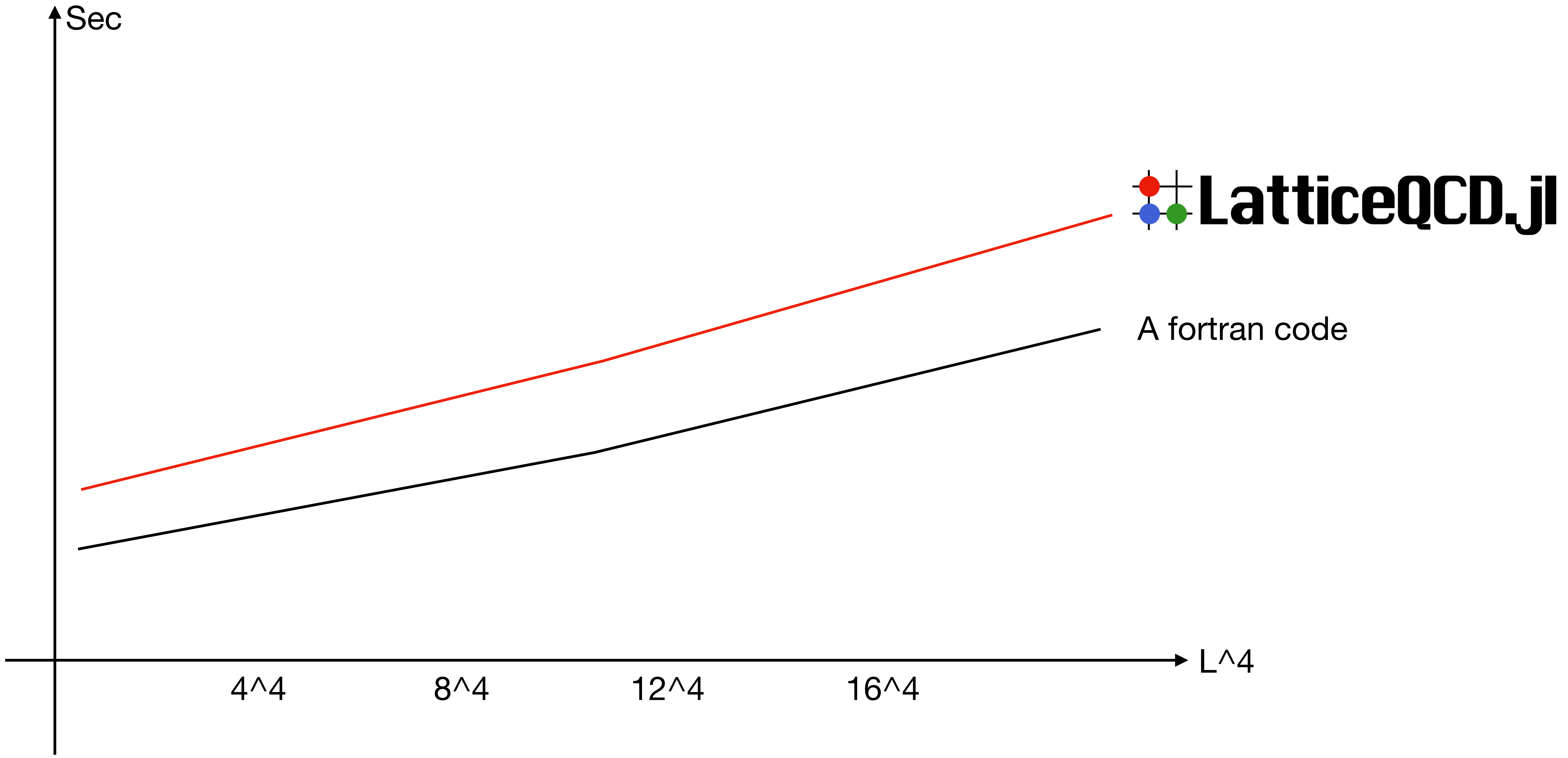
“Gradient” flow

(not has to be gradient of S)

“Continuous stout smearing is the Wilson flow” 2010 M. Luscher

Comparison

Single core is done effectively! Good so far!



Parameter:
x
xx
x
x
x
x
x

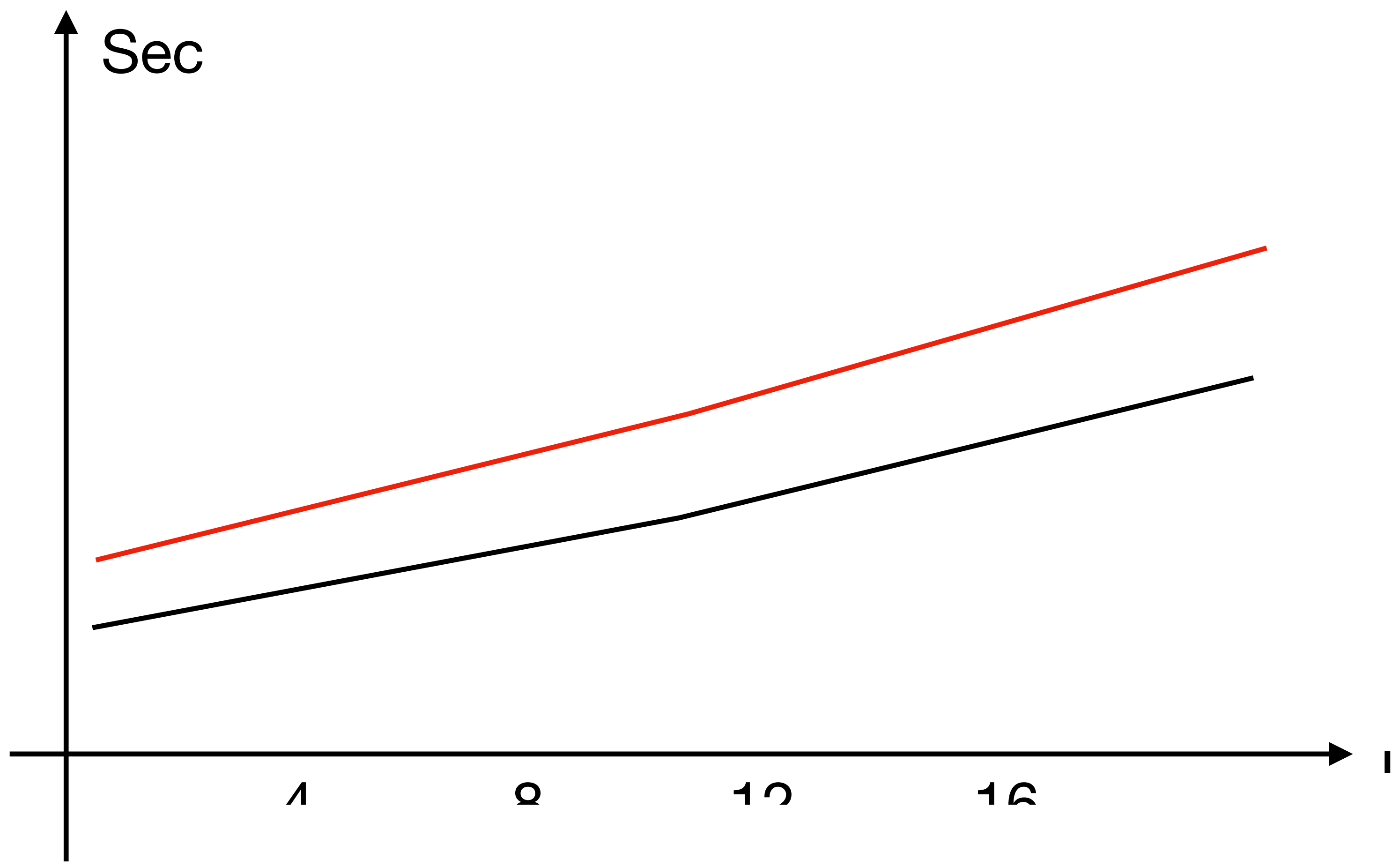
Bench mark
LTK(Wilson), LatticeQCD.jl(Wilson)
L -> 大きく

Julia is almost perfect but…

We need more speed!

- Lattice QCD is a theory in four dimensional spacetime
- <u>So, a code for Lattice QCD has a lot of for loop because it is 4 dimension</u>
- To treat them, we need massive parallelization!

- Parallelization in current Julia is not enough
  - Thread: Shared array
  - Process: Distributed array
  - Hybrid: ?
  - GPU control
- We need documents! Examples…
- We need contributer for parallelization with Julia
- Good “testbed” for parallelization



Summary

Julia + Quantum field theory = Good??

LatticeQCD.jl

- Lattice QCD has been run on supercomputers
- Lattice QCD is a good benchmark for language/hardware
  - ∵ It requires huge numerical resource
- Our code on desktop/laptop/PC-cluster (System size $< 16^4$)
  - Quicker than a Fortran code
- We need parallelization on Julia more!
- Note: Supercomputers support Julia (e.g. Summit, Fugaku)
- Can our code be used on supercomputers for real with practical speed?
- <u>We need contributers, who know parallelization on Julia</u>

Stay tuned!

Title
Subtitle

Something to write here