Lattice QCD with Machine learning

Akio Tomiya (Assistant prof. in IPUT Osaka)
akio_at_yukawa.kyoto-u.ac.jp

Based on arXiv:
arXiv: 2103.11965,
2205.08860 etc

Self-introduction

Lattice QCD & Machine learning @ IPUT Osaka

What am I?

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

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 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 at T>0 with Mobius Domain-wall fermions

KAKENHI

PI: Grant-in-Aid for Transformative Research Areas (A)

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

Grant-in-Aid for Early-Career Scientists

CI: Grant-in-Aid for Scientific Research (C), etc

Biography

2015 : PhD in Osaka university
2015 - 2018 : Postdoc in CCNU (Wuhan, China)
2018 - 2021 : SPDR in Riken/BNL (New York, US)
2021 - : Assistant prof. in IPUT Osaka

Phys + ML

Seminar series: Deep learning and physics

- https://cometscome.github.io/DLAP2020/
- Online seminar series since 2020
- Machine learning + Physics
- Thursday morning, bi-weekly

Intensive lecture series

- https://akio-tomiya.github.io/lectures4mlphys/
  - #1 An introduction to optimal transport (in Japanese)
  - Wako Riken (Hybrid)
  - Date:12 Jan (Thu) 13:30-17:00 JST
  - Akinori Tanaka (RIKEN-AIP)

Spring school for computational physics

- https://hohno0223.github.io/comp_phys_spring_school2023/index.html
- @ Okinawa
- Hybrid (we will support traveling fee)
- March 13(Mon)-15(Wed)
- High energy physics + condensed matter + ML + quantum computing, etc

Outline

1. What and why QCD/lattice QCD?
2. Lattice QCD + Machine learning
    1. "Neural net = Smearing"

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

Theoretical Physics

Akio Tomiya

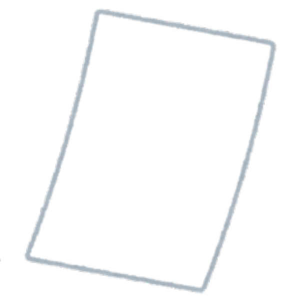
Appropriate description depends on the scale

1 m
Human

1cm = 1^[-2] m
Coins

1mm=10^[-3]m

10^[-4]m



10^[-5]m

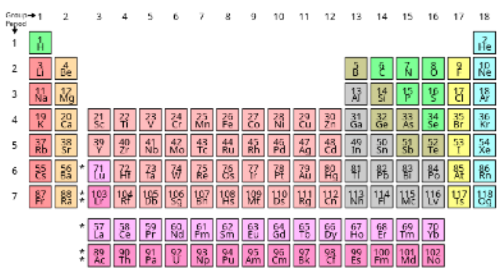
10^[-7]m

Biology

10^[-8]m

Chemistry

Condensed matter

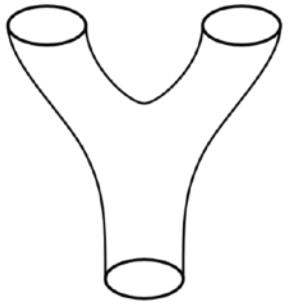


I am interested in
10^[-15]m

High energy phys

10^[-24]m

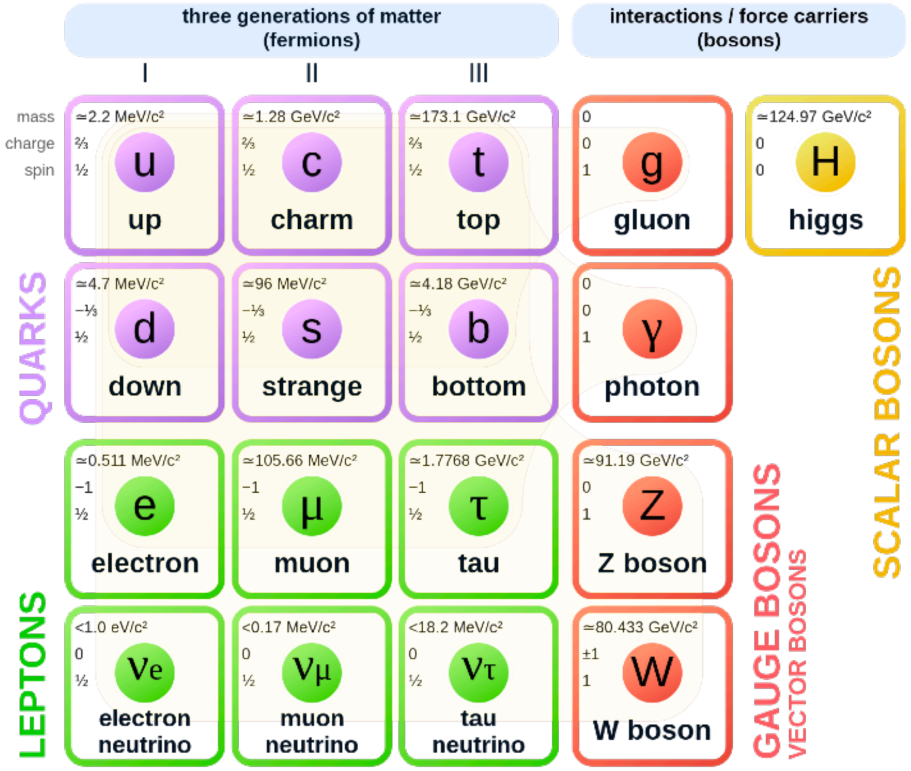
String



Introduction

Akio Tomiya

Particle physics = four fundamental forces + matters



- Gravity
  - Binding everything, mediated by graviton(?)
- The electromagnetic force
  - Binding nucleus and electrons, mediated by photon
- The weak force
  - Change particles, mediated by Z, W bosons
- The strong force (very strong)
  - Binding quarks, mediated by gluons

Introduction

QCD: a fundamental theory of particles inside of nuclei

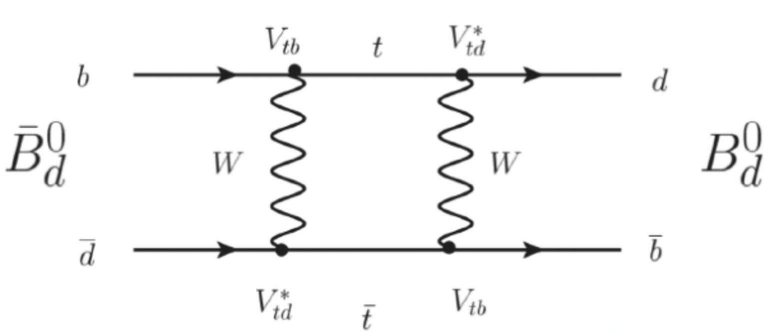
QCD (Quantum Chromo-dynamics) in 3 + 1 dimension

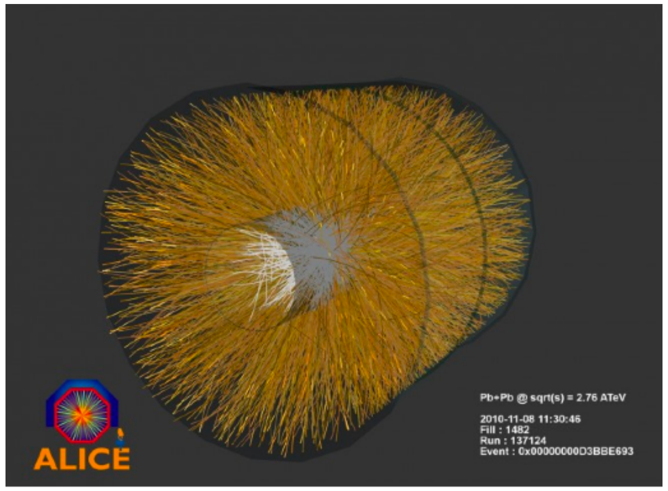
$$S = \int d^4x \left[-\frac{1}{2} \mathrm{tr}\, F_{\mu\nu}F^{\mu\nu} + \bar{\psi}\left(i\partial\!\!\!/ + gA\!\!\!/ - m\right)\psi \right]$$

$$F_{\mu\nu} = \partial_\mu A_\nu - \partial_\nu A_\mu - ig[A_\mu, A_\nu]$$

$$|\psi(t)\rangle = e^{-iHt}|\psi(0)\rangle$$

Expectation values with this are needed





- QCD is theory for the strong force and an extension of electro-magnetism
- QCD enables us to calculate (in principle):
  - Equation of state of neutron star, Tc
  - Scattering of quarks and gluons
  - Hadron spectrum (bound state energy of quarks)
- Strongly coupled quantum system
- Use lattice formulation + Monte-Carlo

Introduction

Lattice path integral > 1000 dim, Trapezoidal int is impossible

K. Wilson 1974

Imaginary time $t=-i\tau$

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

Lattice regularization $U_\mu=e^{aigA_\mu}\in SU(3)$

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

Both S give same expectation value for long range $\mathrm{Re}\,U_{\mu\nu}\sim\frac{-1}{2}g^2a^4F_{\mu\nu}^2+O(a^6)$

Quantum expectation values = multi-dimensional integral (Path integral)

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

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

$$S_{\mathrm{eff}}[U]=S_{\mathrm{gauge}}[U]-\log\det(D\!\!\!\!/[U]+m)$$

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

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

We cannot control numerical error

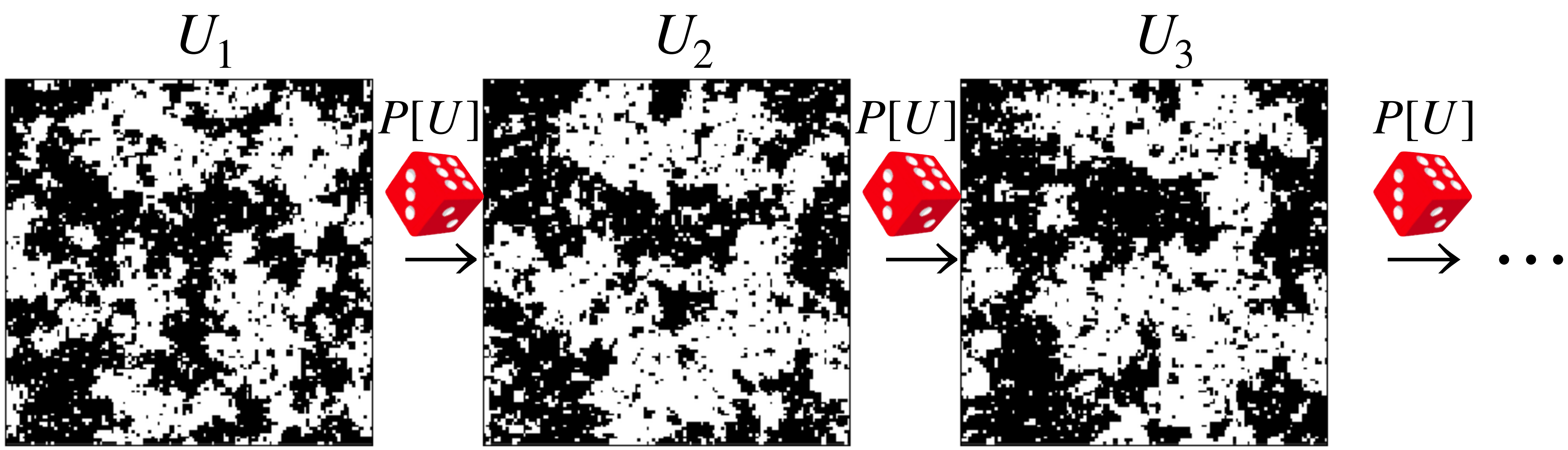
Introduction

Monte-Carlo integration is available

M. Creutz 1980

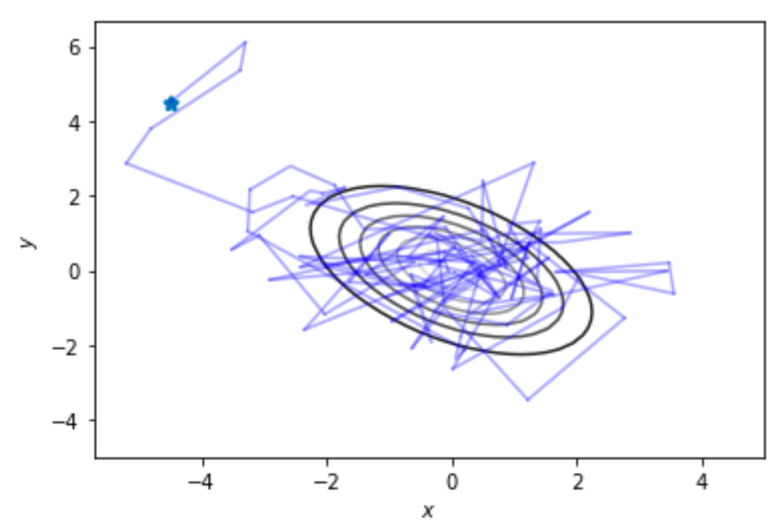
$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}U e^{-S_{\mathrm{eff}}[U]} \mathcal{O}(U) \qquad S_{\mathrm{eff}}[U] = S_{\mathrm{gauge}}[U] - \log\det(\mathbb{D}[U] + m)$$

Monte-Carlo: Generate field configurations with "$P[U] \propto e^{-S_{\mathrm{eff}}[U]}$". It gives expectation values



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

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



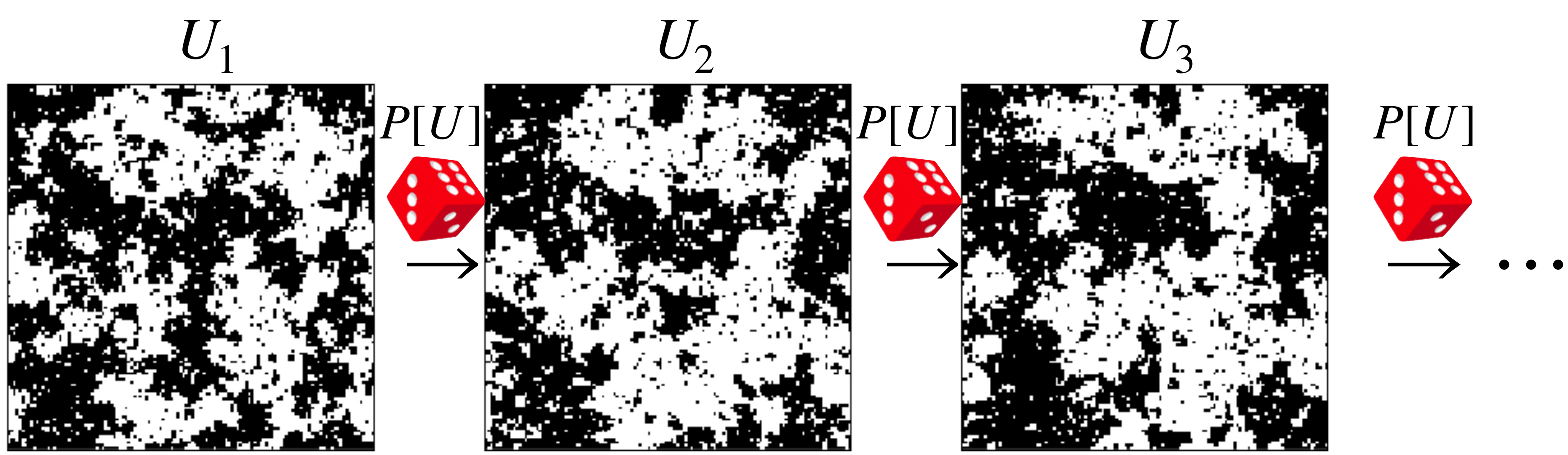
Introduction

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 “$P[U] \propto e^{-S_{\text{eff}}[U]}$”**. It gives expectation values**



Error of integration is determined by the number of sampling

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

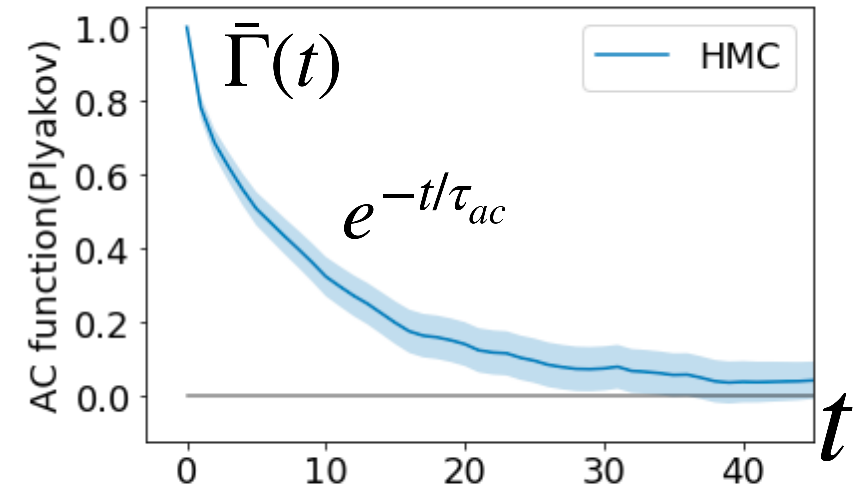
Introduction

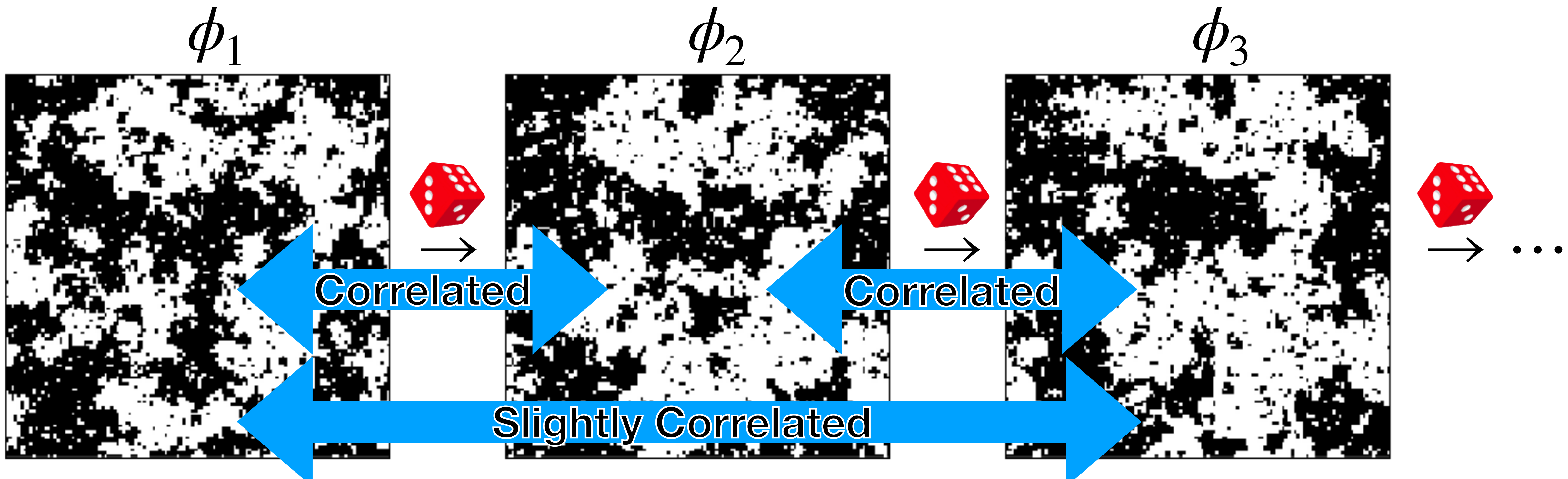
Correlation between samples = inefficiency of calculation

$$\langle O[\phi] \rangle = \frac{1}{N}\sum_{k}^{N} O[\phi_k] \pm O\left(\frac{1}{\sqrt{N_{\mathrm{indep}}}}\right)$$

$$N_{\mathrm{indep}} = \frac{N_{\mathrm{sample}}}{2\tau_{ac}}$$

$$\bar{\Gamma}(t) = \frac{1}{N-t}\sum_{k}(O[\phi_{k+t}] - \bar{O})(O[\phi_k] - \bar{O}) \sim e^{-t/\tau_{ac}}$$



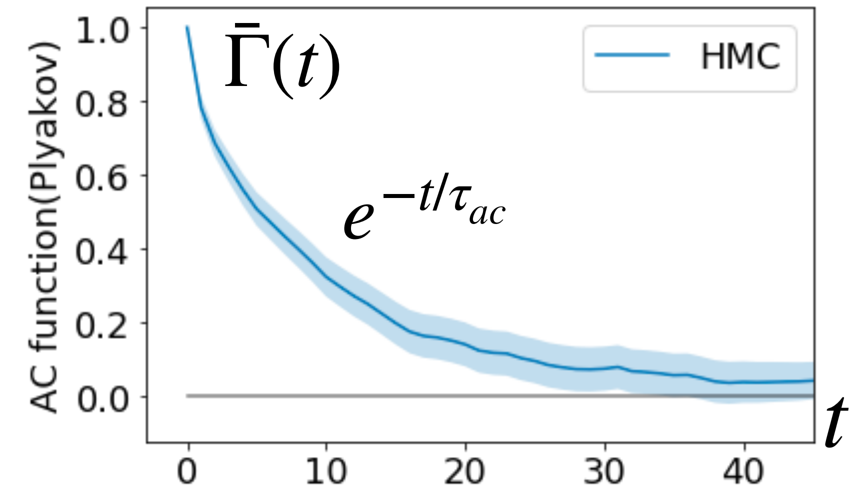


Large τ_{ac} means, such Monte-Carlo integration is inefficient

Introduction

Summary for now: long autocorrelation = inefficiency

$$\langle O[\phi] \rangle = \frac{1}{N}\sum_{k}^{N} O[\phi_k] \ \pm \ O\left(\frac{1}{\sqrt{N_{\text{indep}}}}\right)$$



$$N_{\text{indep}} = \frac{N_{\text{sample}}}{2\tau_{ac}}$$

$$\bar{\Gamma}(t) = \frac{1}{N-t}\sum_{k} (O[\phi_{k+t}] - \bar{O})(O[\phi_k] - \bar{O}) \sim e^{-t/\tau_{ac}}$$

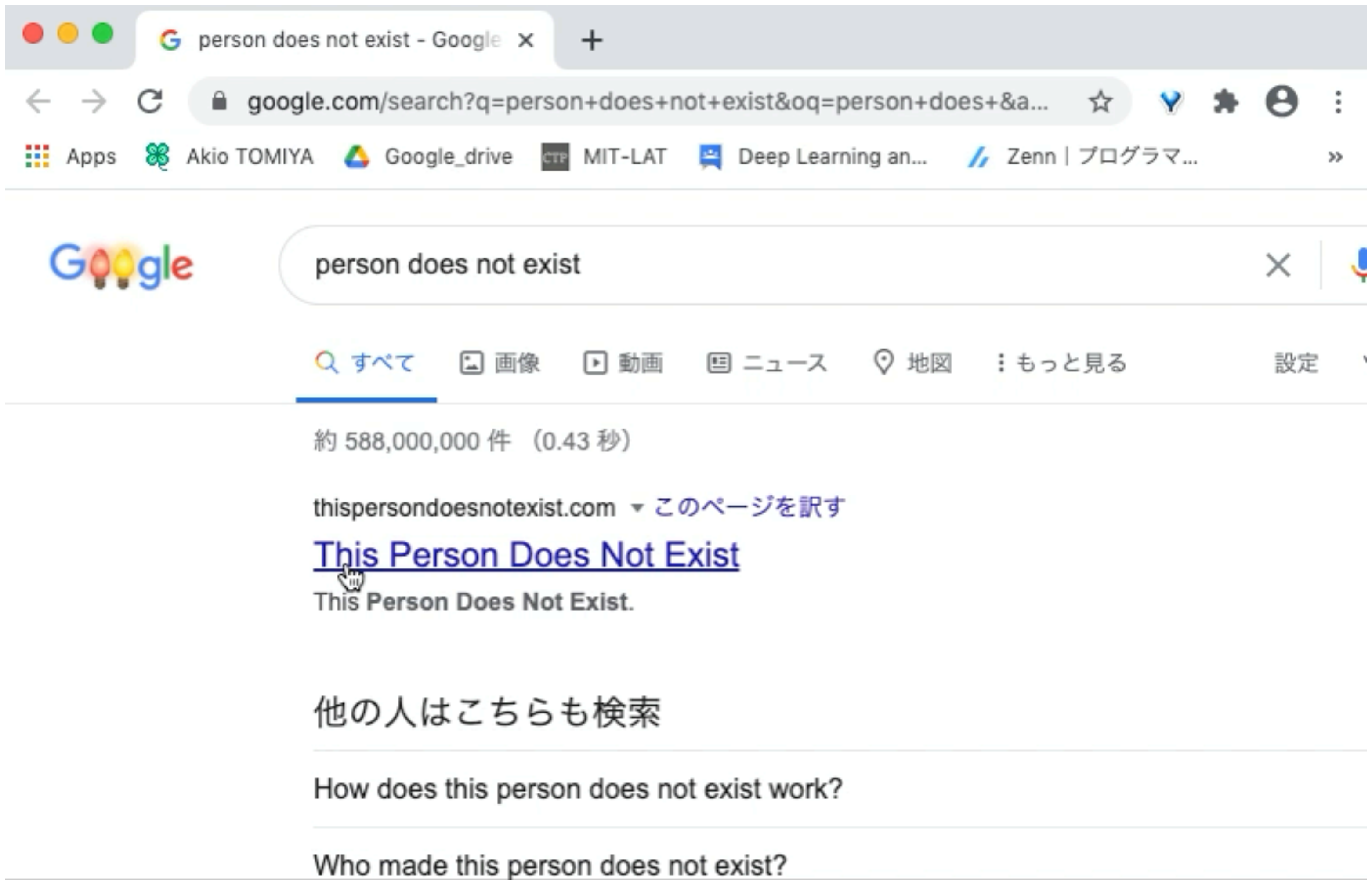
τ_{ac} is given by an update algorithm (N. Madras et. al 1988)

- Autocorrelation time τ_{ac} quantifies similarity between samples
- τ_{ac} is algorithm dependent quantity
- If τ_{ac} becomes half, we can get doubly precise results in the same time cost

Can we make this mild using machine learning?

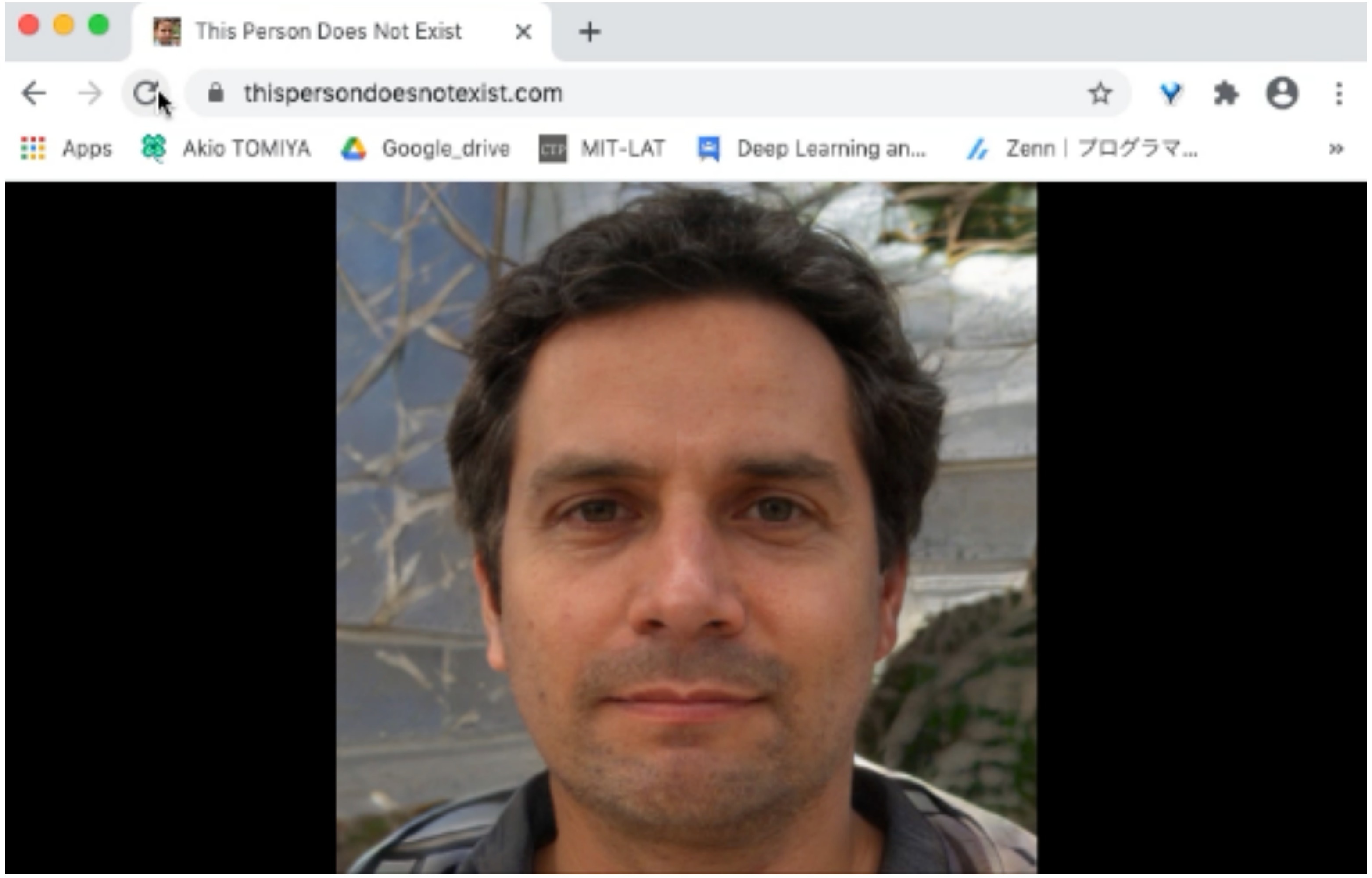
Introduction

Neural net can make human face images



Introduction
Neural net can make human face images

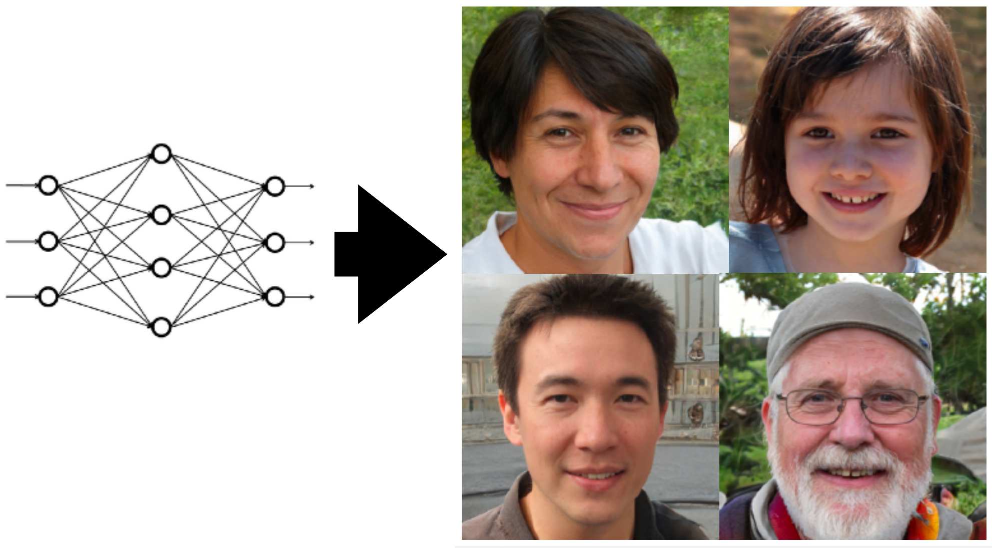
Neural nets can generate realistic human faces (Style GAN2)



Realistic Images can be generated by machine learning!
Configurations as well? (configuration ~ images?)

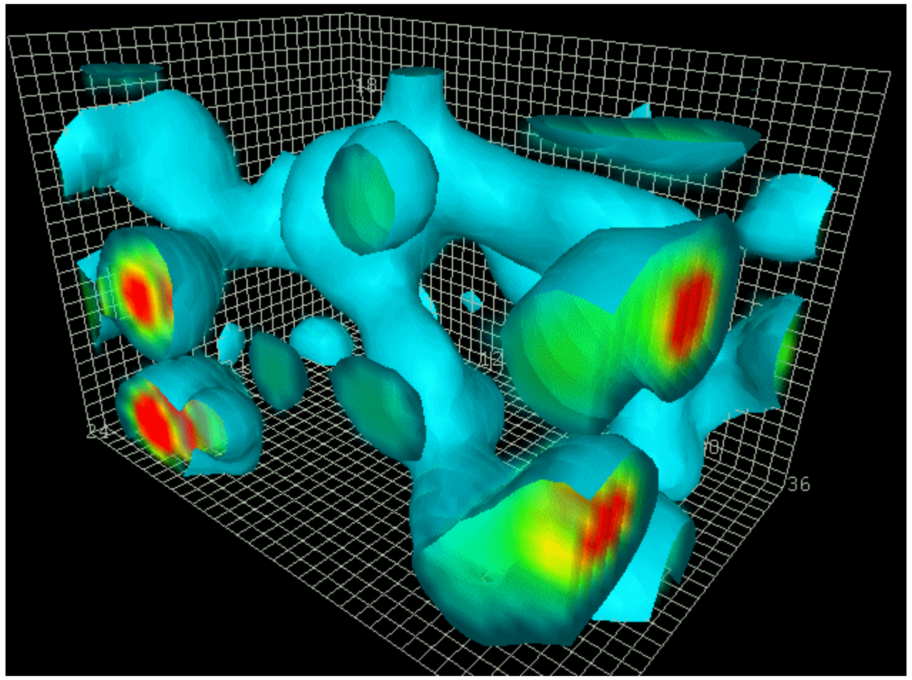
ML for LQCD is needed

- Machine learning/ Neural networks
  - Data processing techniques for 2d image in daily life (pictures = pixels = a set of real #)
  - Neural network can generate images! (arpproximately)



thispersondoesnotexist.com

- Lattice QCD is more complicated than pictures
  - 4 dimension
  - **Non-abelian gauge d.o.f. and symmetry**
  - Fermions
  - Exactness of algorithm is necessary
- Q. How can we deal with?



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		

+…

LQCD + Machine learning
How to deal gauge sym.

Introduction

Akio Tomiya

Neural network is a universal approximator of functions

Image classification, cats and dogs

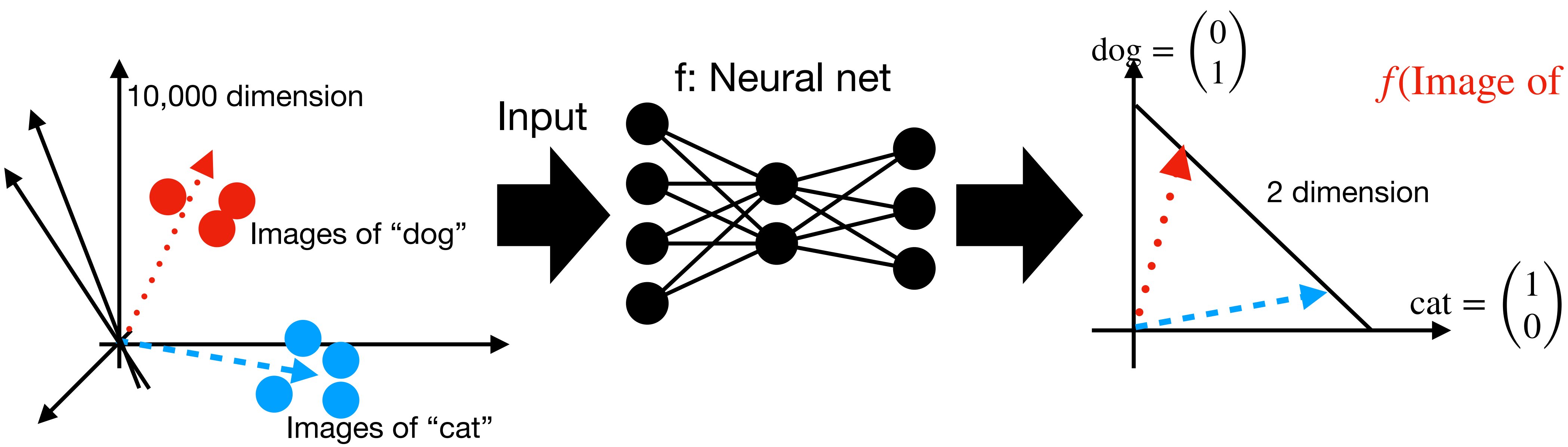
100x100

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

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

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

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



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

Akio Tomiya

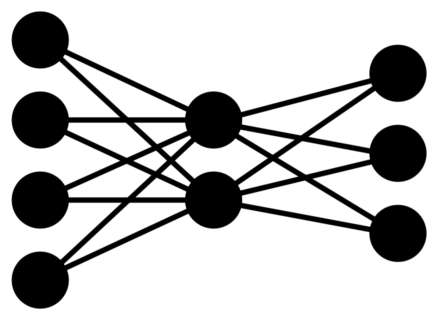
Introduction

Affine transformation + element-wise transformation

Fully connected neural networks

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

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



Component of neural net: $l = 2,3,\cdots$ and $\overrightarrow{u}^{(1)} = \overrightarrow{x}$

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

Matrix product vector addition (w, b determined in the training)

element-wise (local) Non-linear transf.

Typically σ ~ tanh shape

Neural network = (Variational) map between vector to vector

Introduction

Neural network is a universal approximator of functions

Image classification, cats and dogs

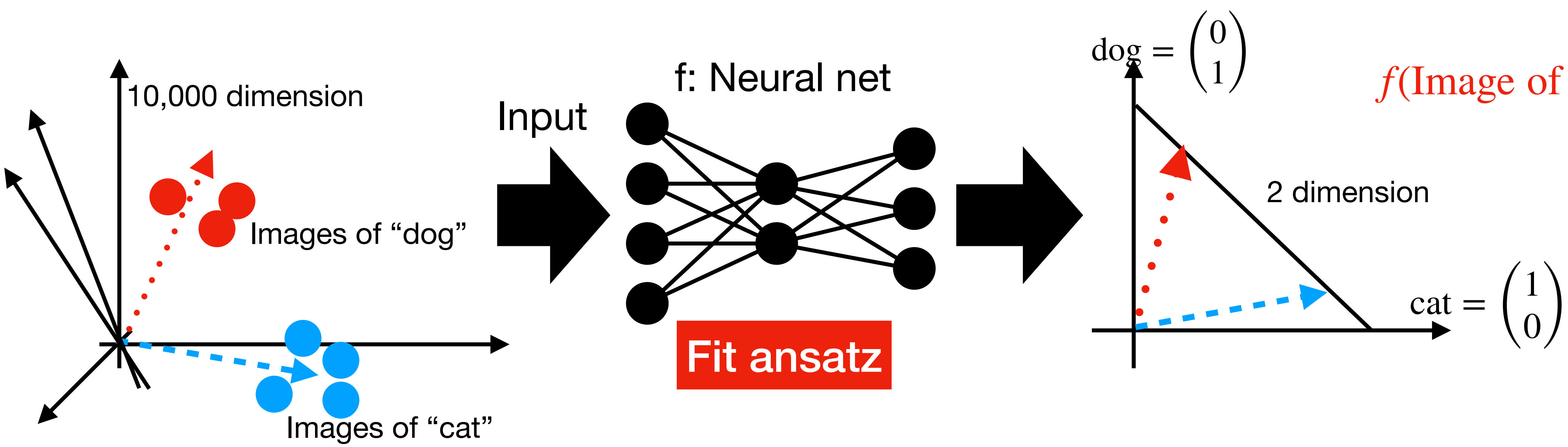
100x100

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

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

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

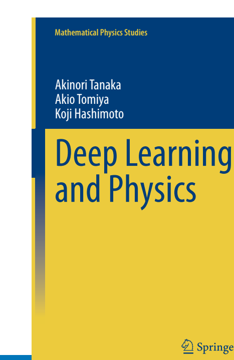
Label is 2 dim vector
(cat = $(1, 0)^t$)



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

<u>**Fact: neural network can mimic any function! (universal app. thm)**</u>

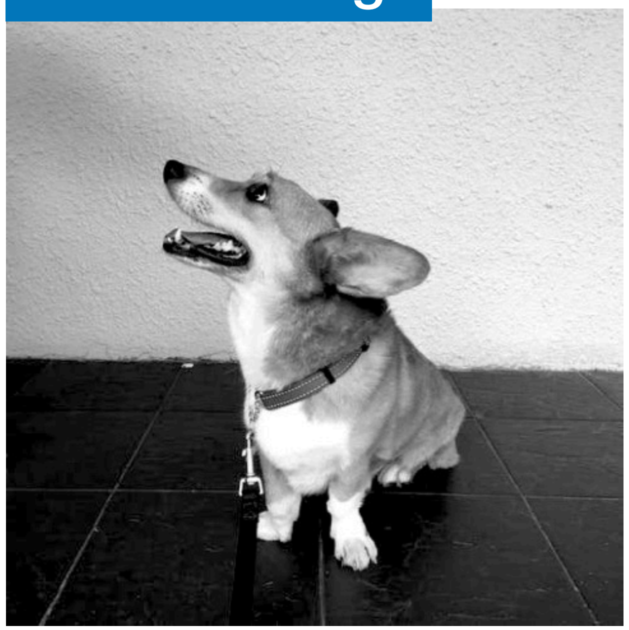
In this example, neural net mimics a map between
image (10,000-dim vector) and label (2-dim vector)



What is the neural networks?

Convolution layer = trainable filter

Filter on image



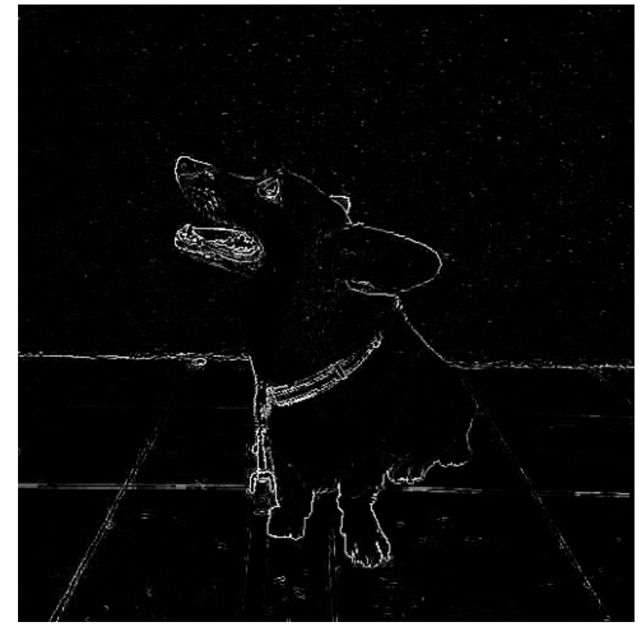
$*$

Laplacian filter

0	1	0
1	-2	1
0	1	0

(Discretization of ∂^2)

$=$



Edge detection

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

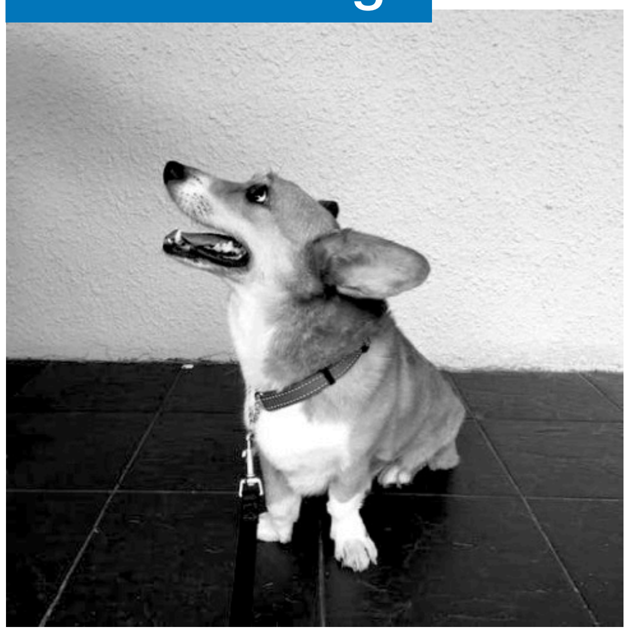
= translationally equivaliant (similar to covariance, operations just commute each other)

What is the neural networks?

Akio Tomiya

Convolution layer = trainable filter

Filter on image



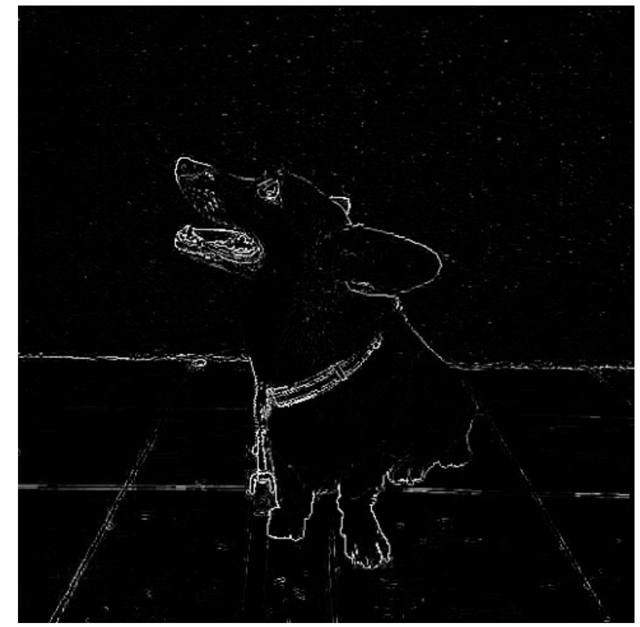
$*$

Laplacian filter

0	1	0
1	-2	1
0	1	0

(Discretization of ∂^2)

=



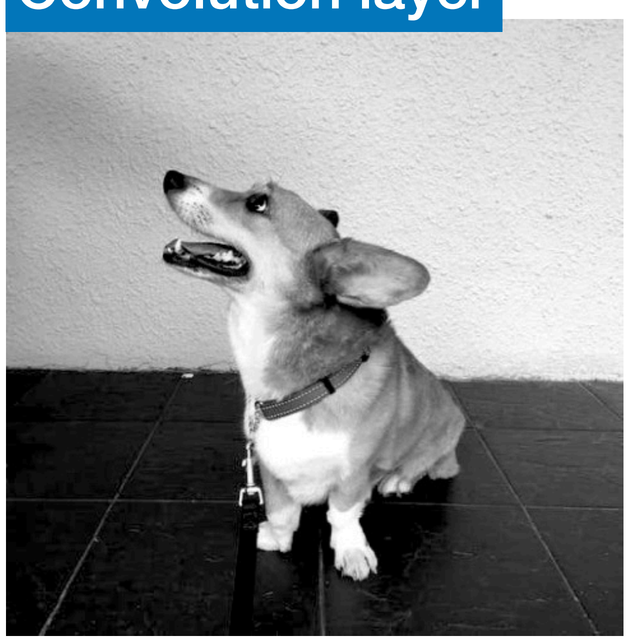
Edge detection

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

= translationally equivaliant (similar to covariance, operations just commute each other)

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

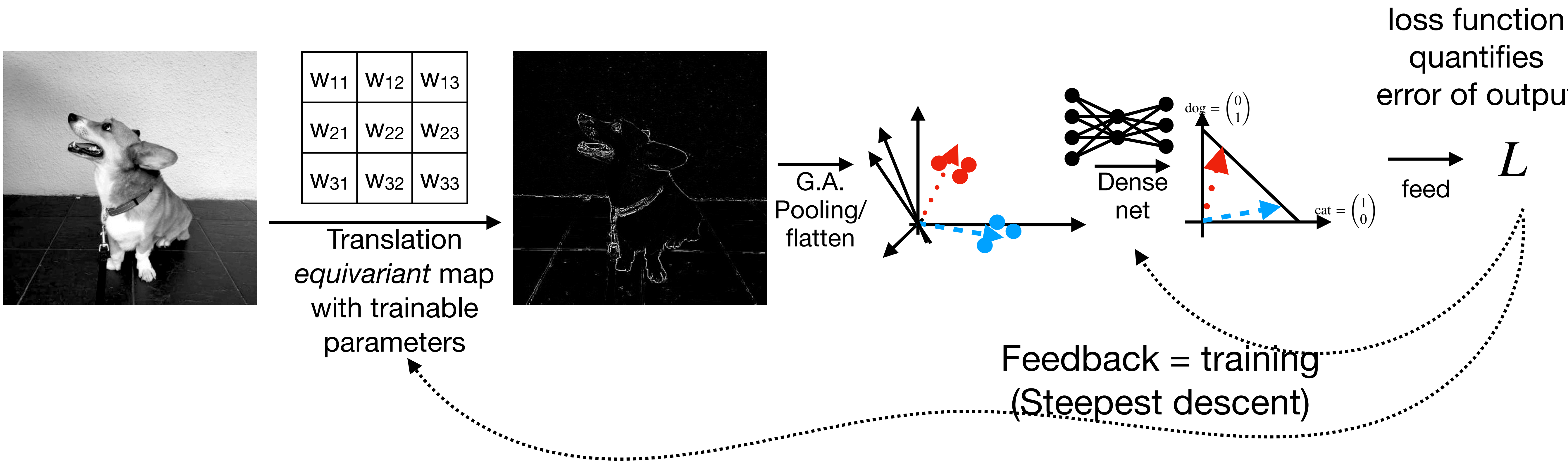
$\frac{1}{16}$

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

Training can be done with back propagation



Smearing

Akio Tomiya

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

Smearing

Akio Tomiya

Smoothing improves global properties

Eg.

Coarse image

Numerical derivative is unstable

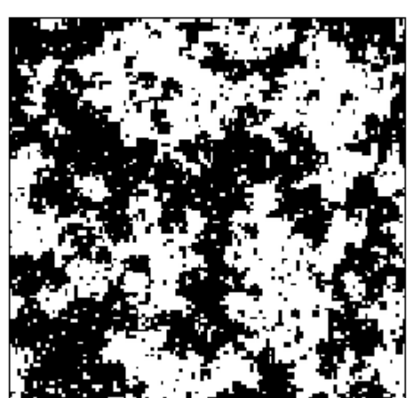
Gaussian filter

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

Smoothened image

Numerical derivative is stable

Gauge configurations have gauge, translation, 90 deg rot symmetries



Gauge transformation on the lattice

$$U_\mu(n) \to g(n) U_\mu(n) g^\dagger(n+\mu) \qquad g(n) \in SU(3)$$

$$S[U] \to S[U]$$

Smearing

Smoothing improves global properties

Eg.

Coarse image

Numerical derivative is unstable

Gaussian filter

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

Smoothened image

Numerical derivative is stable

We want to smoothen gauge configurations with keeping gauge symmetry

Two types:

APE-type smearing

Stout-type smearing

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

Smearing
Smoothing with gauge symmetry, APE type

Akio Tomiya

M. Albanese+ 1987
R. Hoffmann+ 2007

APE-type smearing

Covariant sum

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

Normalization

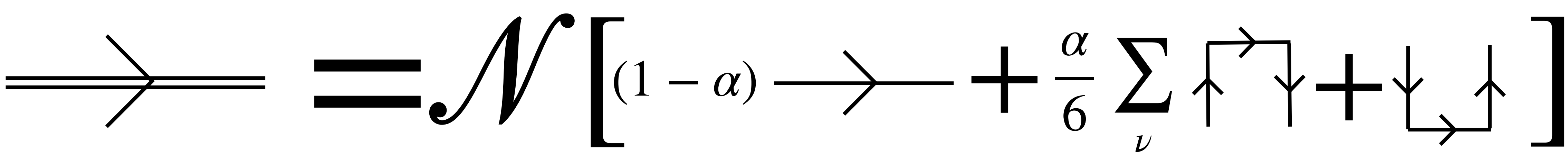
$$\mathcal{N}[M] = \frac{M}{\sqrt{M^\dagger M}}$$

Or projection

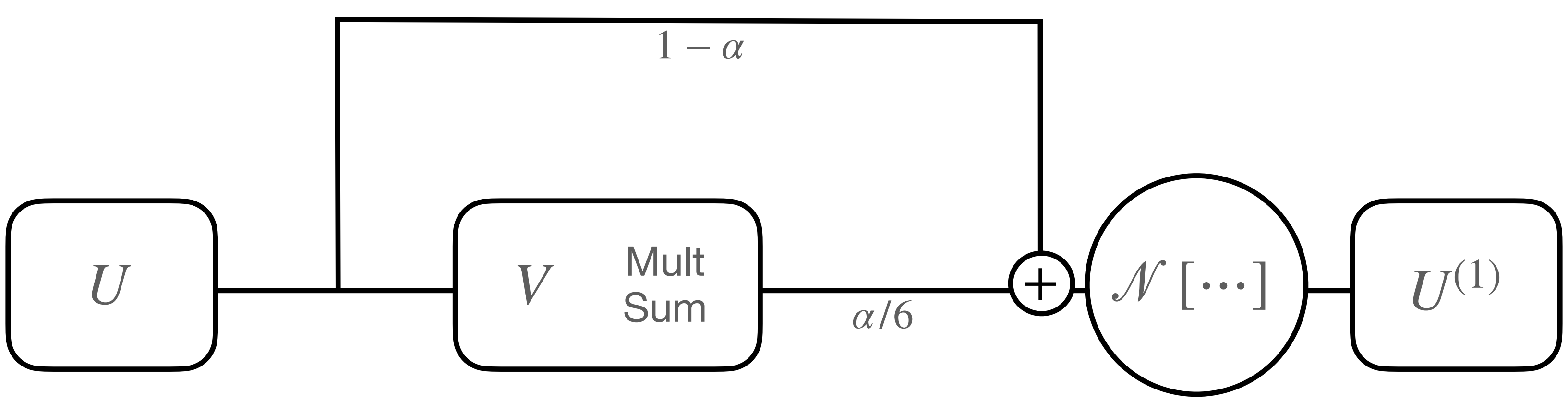
$$V_\mu^\dagger[U](n) = \sum_{\mu\neq\nu} U_\nu(n)U_\mu(n+\hat{\nu})U_\nu^\dagger(n+\hat{\mu}) + \cdots$$

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

Schematically,



In the calculation graph,



Smearing

Smoothing with gauge symmetry, stout type

C. Morningster+ 2003

Stout-type smearing

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

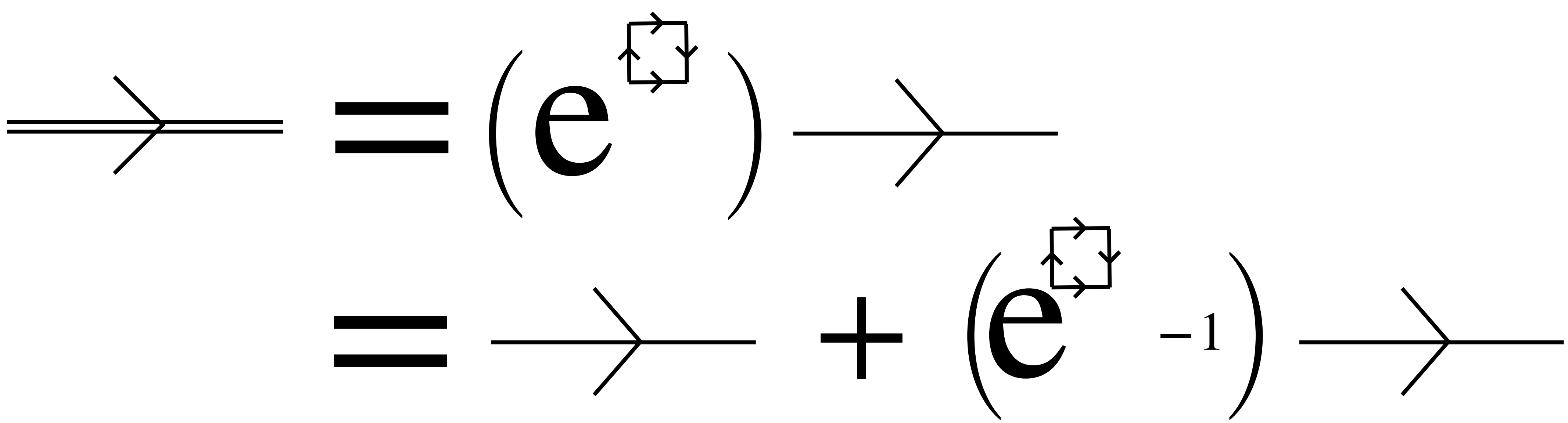
$$= U_\mu(n) + (\mathrm{e}^{Q} - 1) U_\mu(n)$$

Covariant sum

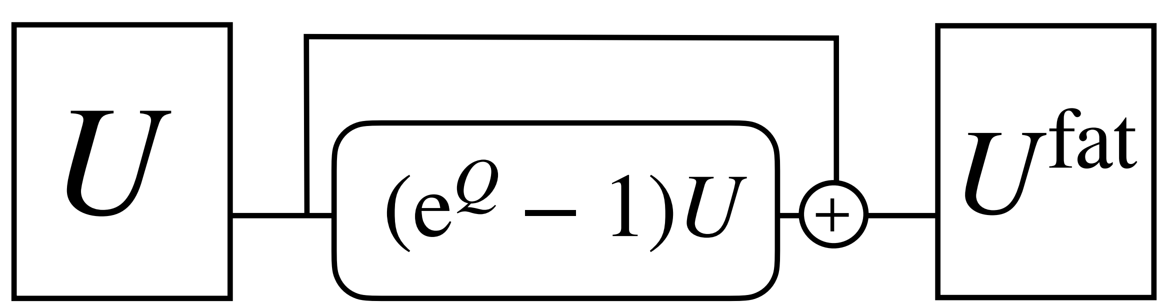
Q: anti-hermitian traceless plaquette

This is less obvious but this actually obeys same transformation

Schematically,



In the calculation graph,



Smearing

Smearing decomposes into two parts

General form of smearing (covariant transformation)

$$\begin{cases} z_\mu(n) = w_1 U_\mu(n) + w_2 \mathcal{G}[U] & \text{Gauge covariant sum} \\ U^{\text{fat}}_\mu(n) = \mathcal{N}(z_\mu(n)) & \text{A local function} \end{cases}$$

Smearing

Smearing ∼ neural network with fixed parameter!

AT Y. Nagai arXiv: 2103.11965

General form of smearing (covariant transformation)

$$\begin{cases} z_\mu(n) = w_1 U_\mu(n) + w_2 \mathcal{G}[U] & \text{Gauge covariant sum} \\ U_\mu^{\text{fat}}(n) = \mathcal{N}(z_\mu(n)) & \text{A local function} \end{cases}$$

It has similar structure with neural networks,

$$\begin{cases} z_i^{(l)} = \sum_j w_{ij}^{(l)} u_j^{(l-1)} + b_i^{(l)} & \text{Matrix product vector addition} \\ u_i^{(l)} = \sigma^{(l)}(z_i^{(l)}) & \text{element-wise (local) Non-linear transf.} \end{cases}$$

Typically σ ~ tanh shape

Actually, we can find a dictionary between them

Gauge covariant neural network

= trainable smearing

Akio Tomiya

AT Y. Nagai arXiv: 2103.11965

Dictionary

	(convolutional) Neural network	Smearing in LQCD	
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)

Takeaway message

Gauge Covariant Neural networks
= trainable smearing, training for SU(N) fields

Gauge covariant neural network
= trainable smearing

Akio Tomiya

AT Y. Nagai arXiv: 2103.11965

Gauge covariant neural network = general smearing with trainable parameters w

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

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

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

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

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

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

1. Gauge covariant composite function:
   Input = gauge field, Output = gauge field
2. Parameters in the network can be trainable using ML techniques.

Gauge covariant neural network

Akio Tomiya

Training can be done with (extended) back propagation

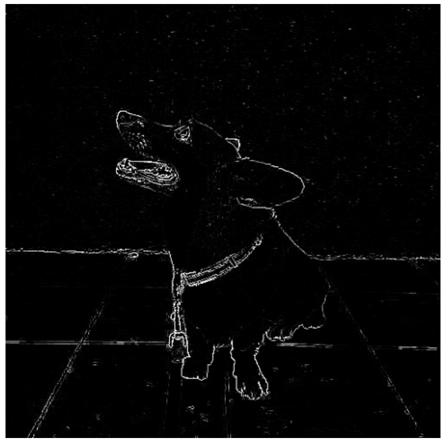
AT Y. Nagai arXiv: 2103.11965

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

Usual neural network

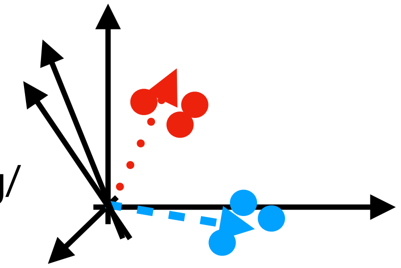
w_{11}	w_{12}	w_{13}
w_{21}	w_{22}	w_{23}
w_{31}	w_{32}	w_{33}

→ Translation *equivariant* map with trainable parameters

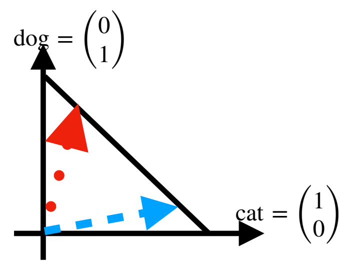


Translation *equivariant* = the image is shifted, output image is also shifted

→ G.A. Pooling/ flatten

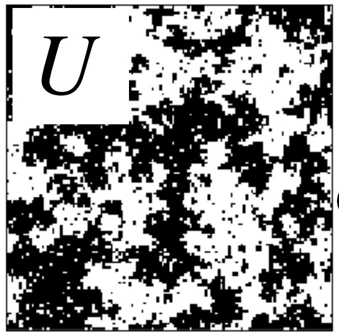


→ Dense net



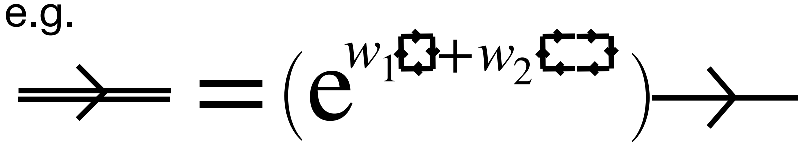
→ feed → Invariant loss function L

Covariant neural networks



$U_\mu^{\rm NN}(n)[U]$ → Gauge covariant map with trainable parameters

e.g.



Feed to Dirac op → Covariant $D\left[U_\mu^{\rm NN}(n)[U]\right]$ (Parametrized Dirac operator) → Parametrized action $S_{\rm ferm}\left[U^{\rm NN}\right]$ → feed → Invariant loss function L

Construct loops → Covariant $W\left[U_\mu^{\rm NN}(n)[U]\right]$ (Parametrized loop operators (e.g. plaquette, Polyakov loop))

→ $S_{\rm plaq}\left[U^{\rm NN}\right]$ → feed → L

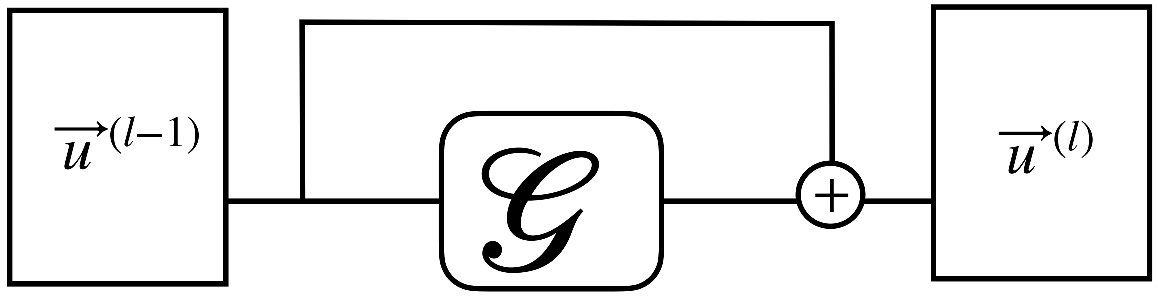
→ $Q\left[U^{\rm NN}\right]$ (Topological charge) → L

cf. Gauge equivariant neural net (M Favoni+)

Gauge covariant neural network

Neural ODE of Cov-Net = “gradient flow”

ResNet



arXiv: 1512.03385

Continuum Layer Limit

Neural ODE

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

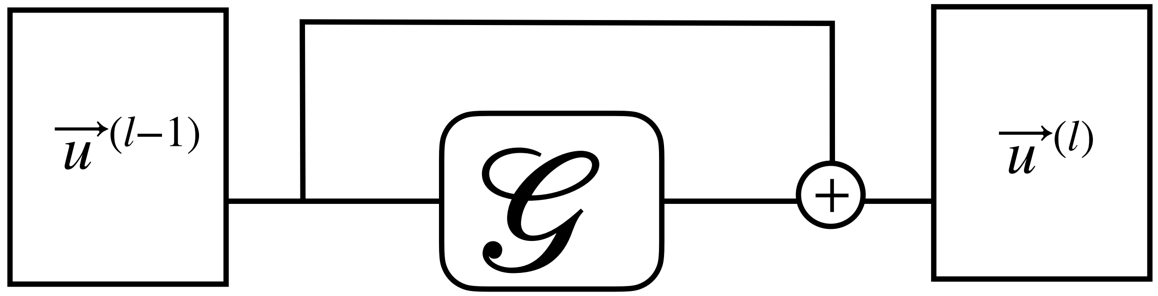
arXiv: 1806.07366
(Neural IPS 2018 best paper)

Gauge covariant neural network

Akio Tomiya

Neural ODE of Cov-Net = “gradient flow”

ResNet



arXiv: 1512.03385

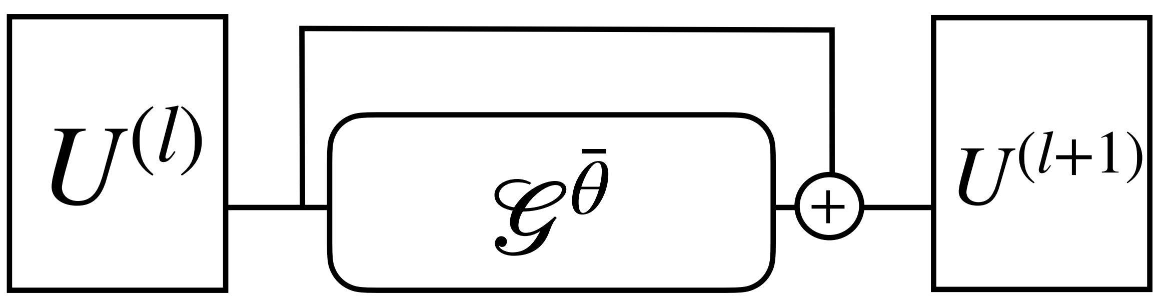
Continuum Layer Limit

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

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 (e.g Gaussian/ Laplacian)	ResNet: Neural ODE	Delta rule and backprop Gradient opt.
Gauge cov. net AT Y. Nagai arXiv: 2103.11965	Gauge covariance Translation equiv, 90° rotation equiv	Smearing	“Gradient flow”	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

An application
Self-learning HMC

Application for the staggered in 4d

Problems to solve

arXiv: 2103.11965

Our neural network enables us to **parametrize** gauge symmetric action covariant way. **It can be used in variational ansatz in gauge theory.**

e.g.

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

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

Test of our neural network?

Can we mimic a different Dirac operator using neural net?

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

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

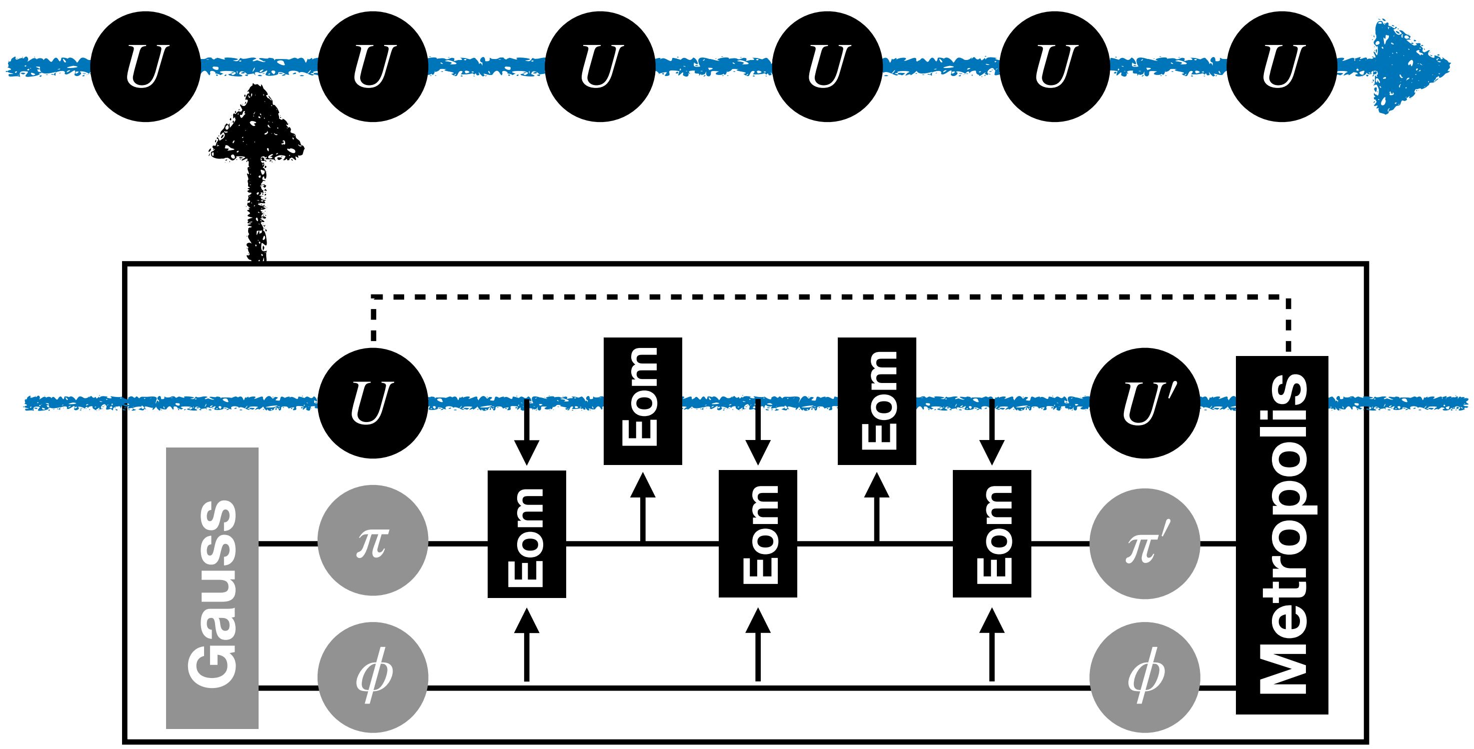
SLHMC = Exact algorithm with ML

Akio Tomiya

SLHMC for gauge system with dynamical fermions

arXiv: 2103.11965 and reference therein

HMC



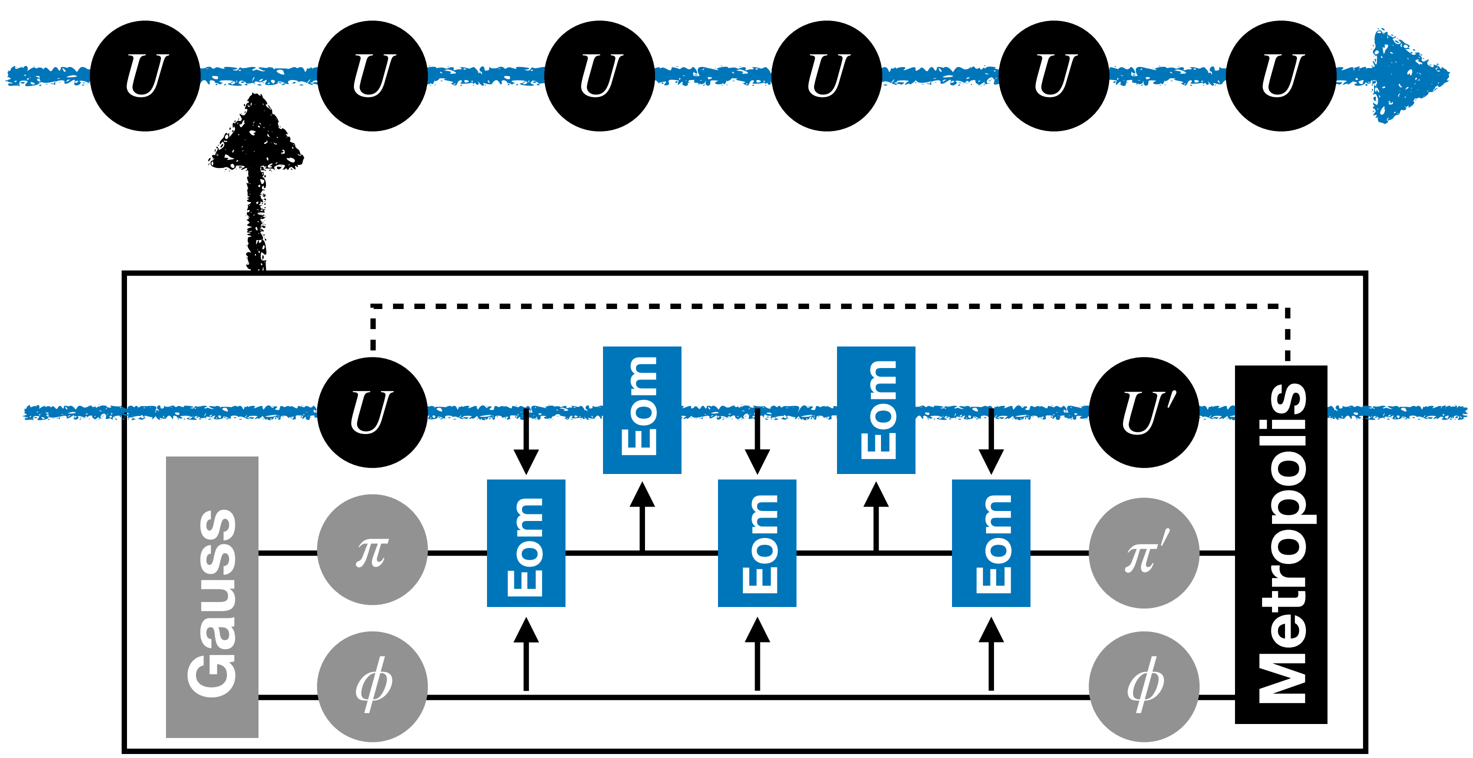
Eom **Metropolis**

Both use

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

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

Self Lerning 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

SLHMC works as an adaptive reweighting!

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), Productive as Python
3. Machine learning friendly (Julia ML packages + Python libraries w/ PyCall)
4. Supercomputers support Julia

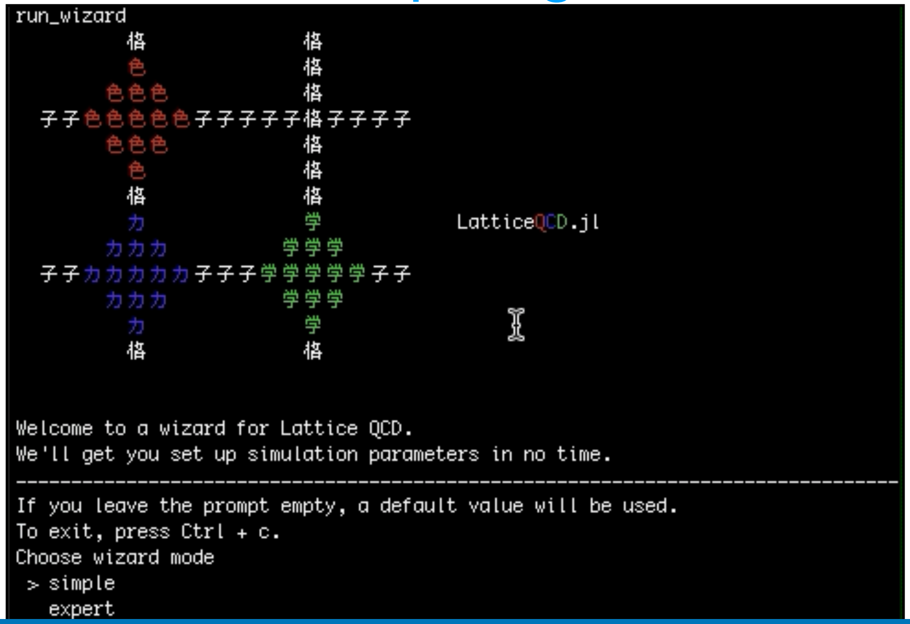
LatticeQCD.jl (Official package) : Laptop/desktop/PC-cluster/Jupyter (Google colab)

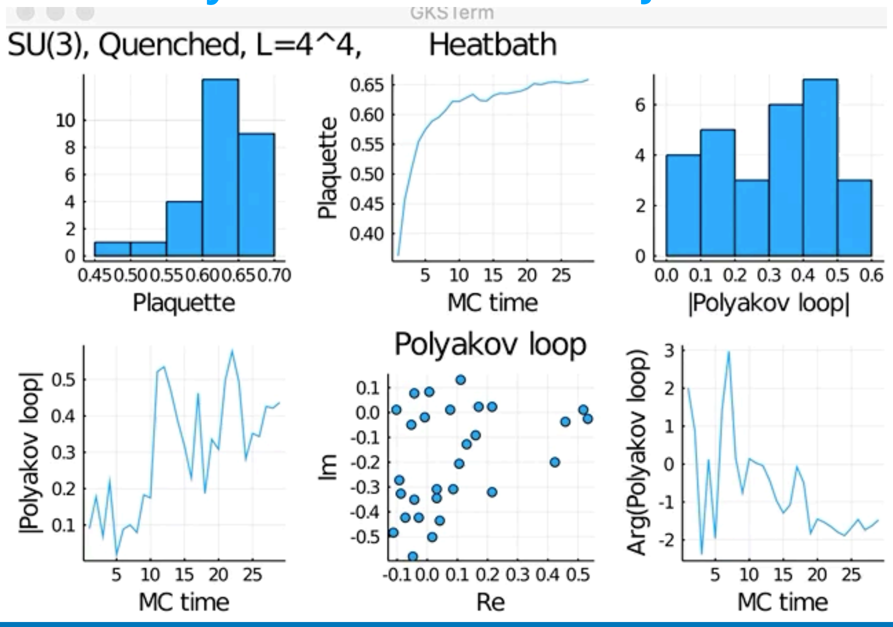
SU(Nc)-heatbath/SLHMC/SU(Nc) Stout/(R)HMC/staggered/Wilson-Clover
Domain-wall/Measurements (Now updating to v1.0, MPI ver is ready)

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





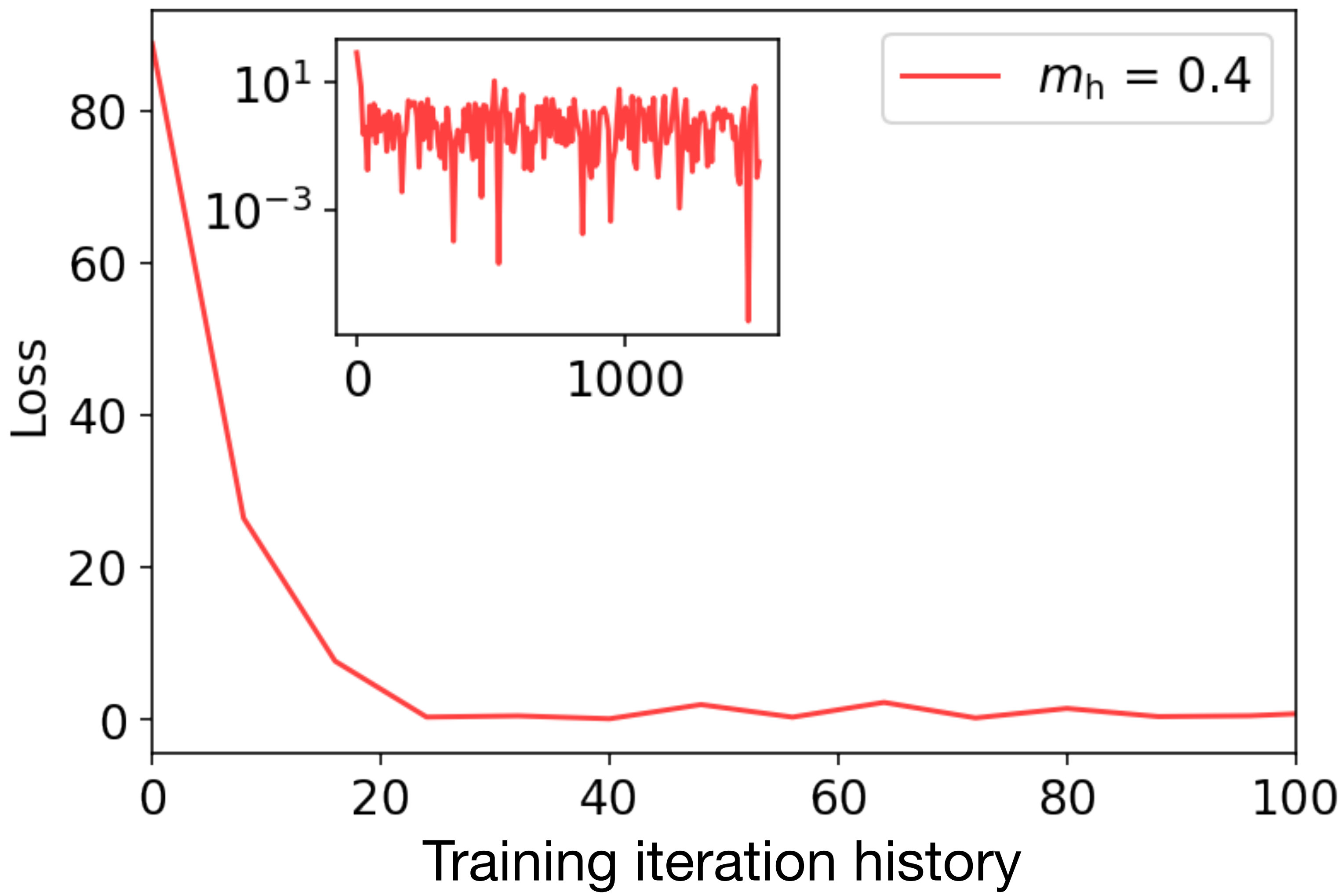
Details (skip)

Akio Tomiya

Results: Loss decreases along with the training

arXiv: 2103.11965

Loss function: $L_\theta[U] = \frac{1}{2}\left| S_\theta[U,\phi] - S[U,\phi] \right|^2,$ ~ -log(reweighting factor)



Without training, e^(-L)<< 1, this means that candidate with approximated action never accept.

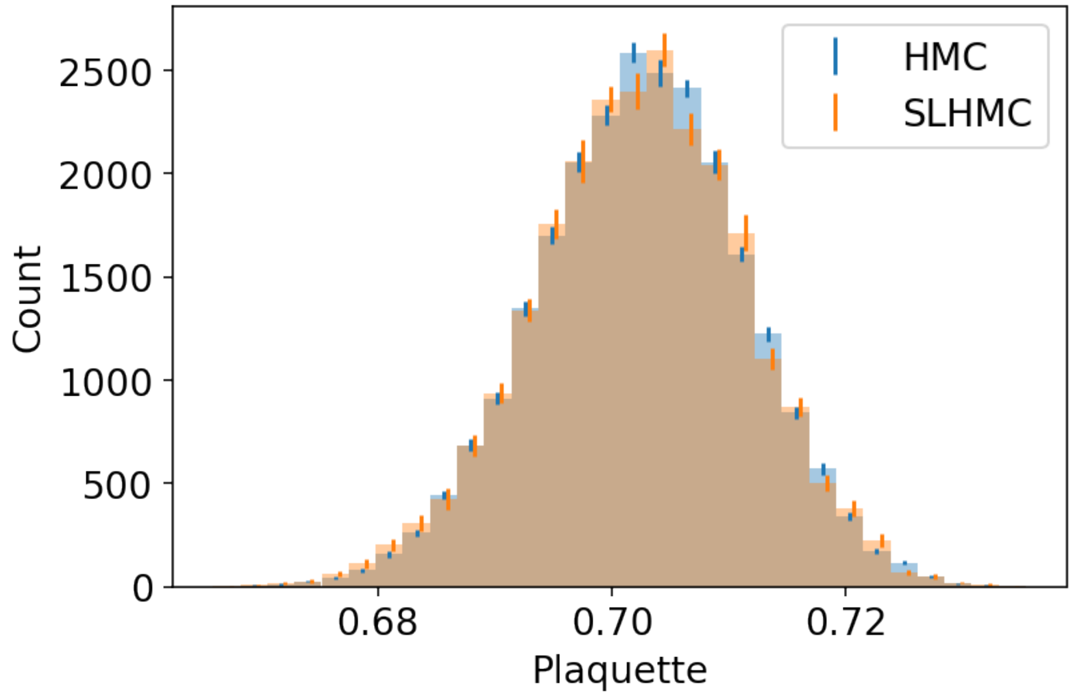
After training, e^(-L) ~1, and we get practical acceptance rate!

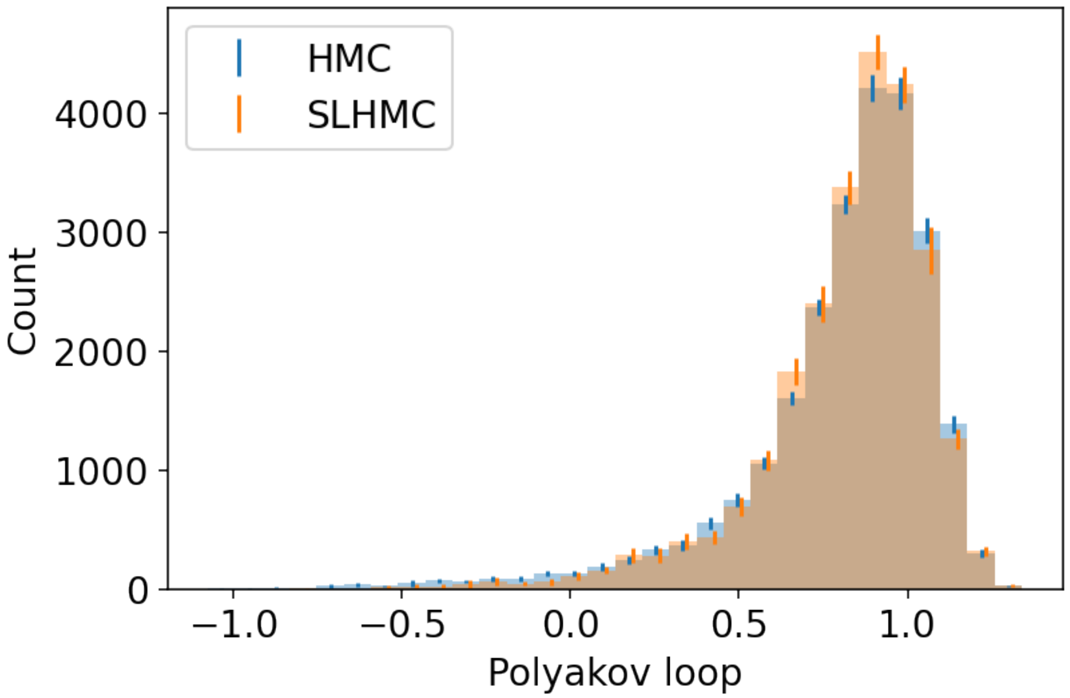
Application for the staggered in 4d

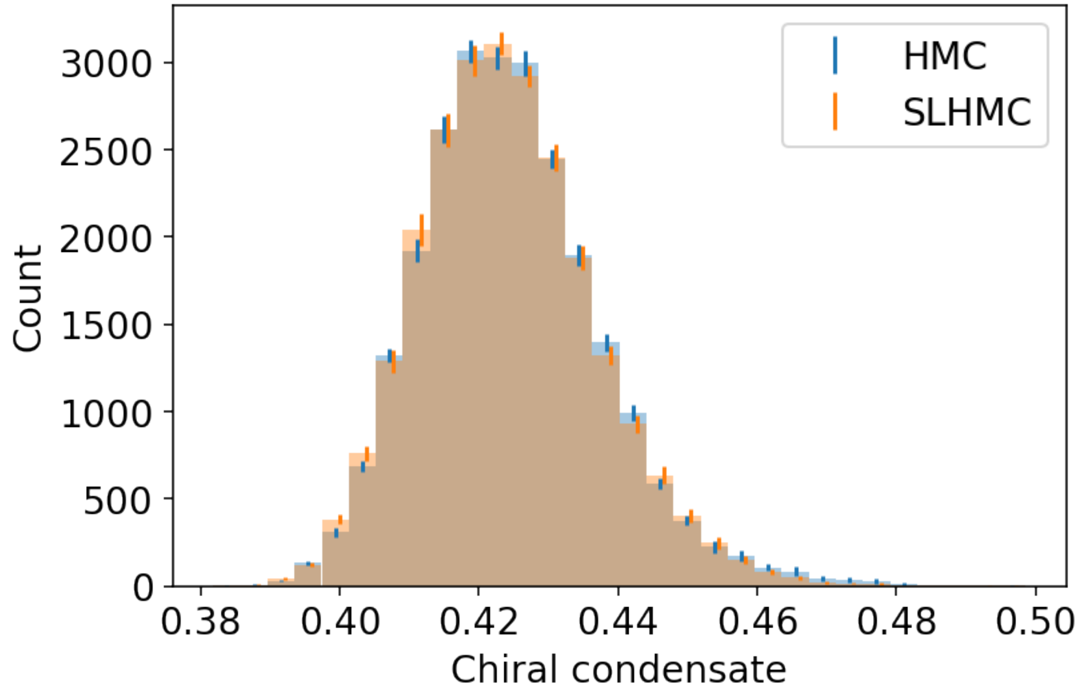
Akio Tomiya

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)

Acceptance = 40%

Other architecture:
Flow based sample algorithm

Related works

Gradient flow as a trivializing map

Trivializing map for lattice QCD has been demanded…

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \cdots \int \prod_{x\in 100} \prod_{y\in 100} \prod_{z\in 100} \prod_{t\in 100} d\phi_{x,y,z,t} \mathrm{e}^{-S(\phi)} \mathcal{O}[\phi_{x,y,z,t}]$$

$$\tilde{\phi} = \mathcal{F}_\tau(\phi)$$ Flow equation (change variable)

If the solution satisfies $S(\mathcal{F}_\tau(\phi)) + \ln \det(\mathrm{Jacobian}) = \sum_n \tilde{\phi}_n^2$,

M. Luscher arXiv:0907.5491

arxiv 1904.12072, 2003.06413, 2008.05456

Flow based sampling algorithm

Normalizing flow ~ Change of variables

Simplest example: Box Muller

Change of variables

$$\begin{cases} z = e^{-\frac{1}{2}(x^2+y^2)} \\ \tan\theta = y/x \end{cases}$$

$$\underbrace{\int_{-\infty}^{\infty} dx \int_{-\infty}^{\infty} dy\, e^{-\frac{1}{2}x^2-\frac{1}{2}y^2}}_{\text{Original integral: hard}} = \underbrace{\frac{1}{2}\int_0^{2\pi} d\theta \int_0^1 dz}_{\text{Easy}}$$

Point: Make problem easier with change of variables (make the measure flat)

RHS is flat measure
→We can sample like right eq.

$$\begin{cases} \xi_1 \sim (0,2\pi) \\ \xi_2 \sim (0,1) \end{cases}$$

We can reconstruct a “field config” x, y for original theory like right eq.

$$\begin{cases} x = r\cos\theta & \theta = \xi_1 \\ y = r\sin\theta & r = \sqrt{-2\log\xi_2} \end{cases}$$

A change of variable which $D\phi e^{-S[\phi]}$ makes flat = **Trivializing map**

Related works

Gradient flow as a trivializing map

Trivializing map for lattice QCD has been demanded…

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \cdots \int \prod_{x \in 100} \prod_{y \in 100} \prod_{z \in 100} \prod_{t \in 100} d\phi_{x,y,z,t} \mathrm{e}^{-S(\phi)} \mathcal{O}[\phi_{x,y,z,t}]$$

$$\tilde{\phi} = \mathcal{F}_\tau(\phi)$$ Flow equation (change variable)

If the solution satisfies $S(\mathcal{F}_\tau(\phi)) + \ln \det(\text{Jacobian}) = \sum_n \tilde{\phi}_n^2$,

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \cdots \int \prod_{x \in 100} \prod_{y \in 100} \prod_{z \in 100} \prod_{t \in 100} d\tilde{\phi} \mathcal{O}[\mathcal{F}_\tau(\phi)] \mathrm{e}^{-\sum \tilde{\phi}_n^2}$$

It becomes Gaussian integral! Easy to evaluate!!

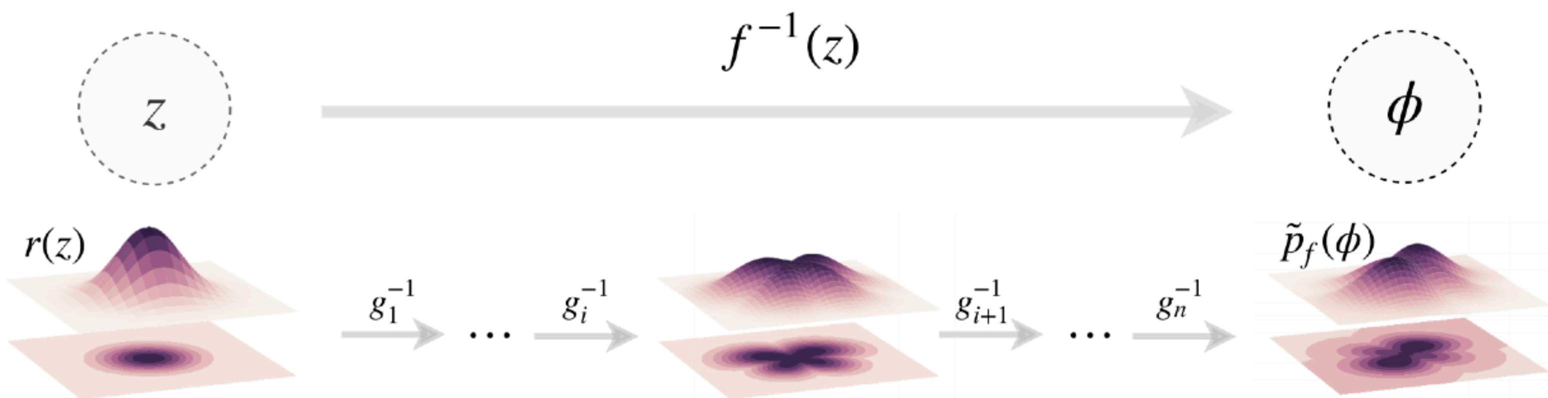
However, the Jacobian cannot evaluate easily, so it is not practical.
Life is hard.

M. Luscher arXiv:0907.5491
arxiv 1904.12072, 2003.06413, 2008.05456

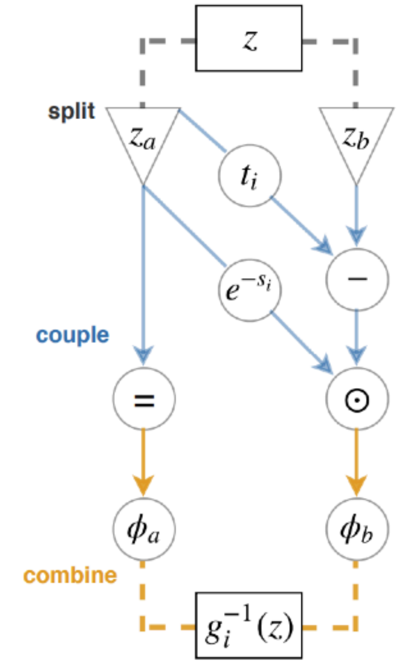
Akio Tomiya

Related works

Flow based algorithm = neural net represented flow algorithm



(a) Normalizing flow between prior and output distributions



(b) Inverse coupling layer

FIG. 1: In (a), a normalizing flow is shown transforming samples z from a prior distribution $r(z)$ to samples ϕ distributed according to $\tilde{p}_f(\phi)$. The mapping $f^{-1}(z)$ is constructed by composing inverse coupling layers g_i^{-1} as defined in Eq. (10) in terms of neural networks s_i and t_i and shown diagrammatically in (b). By optimizing the neural networks within each coupling layer, $\tilde{p}_f(\phi)$ can be made to approximate a distribution of interest, $p(\phi)$.

MIT + Google brain 2019~

Train a neural net as a "flow" $\tilde{\phi} = \mathscr{F}(\phi)$
If it is well approximated, we can sample from a Gaussian
It can be done "Normalizing flow" (Real Non-volume preserving map)
Moreover, Jacobian is tractable!

arxiv 1904.12072, 2003.06413, 2008.05456

Related works

Flow based algorithm = neural net represented flow algorithm

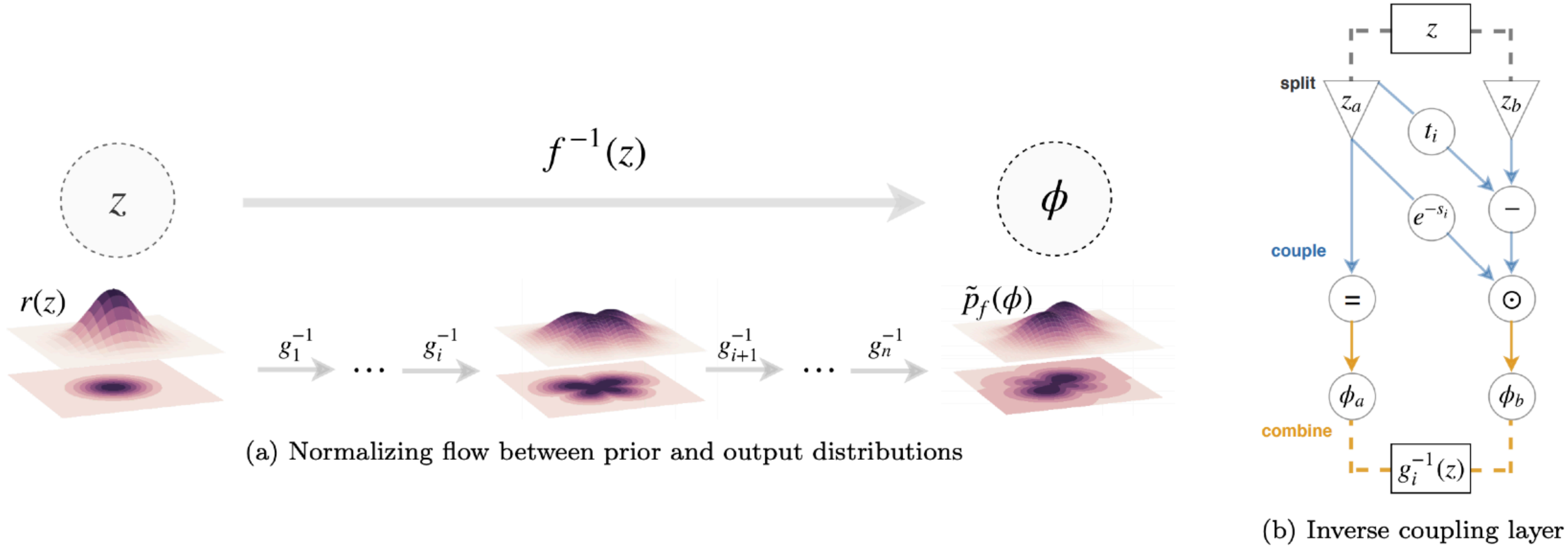


FIG. 1: In (a), a normalizing flow is shown transforming samples z from a prior distribution $r(z)$ to samples ϕ distributed according to $\tilde{p}_f(\phi)$. The mapping $f^{-1}(z)$ is constructed by composing inverse coupling layers g_i^{-1} as defined in Eq. (10) in terms of neural networks s_i and t_i and shown diagrammatically in (b). By optimizing the neural networks within each coupling layer, $\tilde{p}_f(\phi)$ can be made to approximate a distribution of interest, $p(\phi)$.

MIT + Google brain 2019~

Their sampling strategy

→ **Sampling from Gaussian**
→ **Apply an inverse trivializing map (neural network)**
→ **QFT configurations + Tractable Jacobian (by even-odd strategy)**
→ **Metropolis-Hastings test (Detailed balance), exact!**

Flow based sampling algorithm

Akio Tomiya

Flow based ML for QFT

MIT + Deepmind + …

$$\int D\phi e^{-S[\phi]} O[\phi] \propto \underline{\prod_i \int d\varphi_i e^{-V(\varphi_i)} J^{-1}[\varphi] O[F[\varphi]]}$$

Original integral: hard

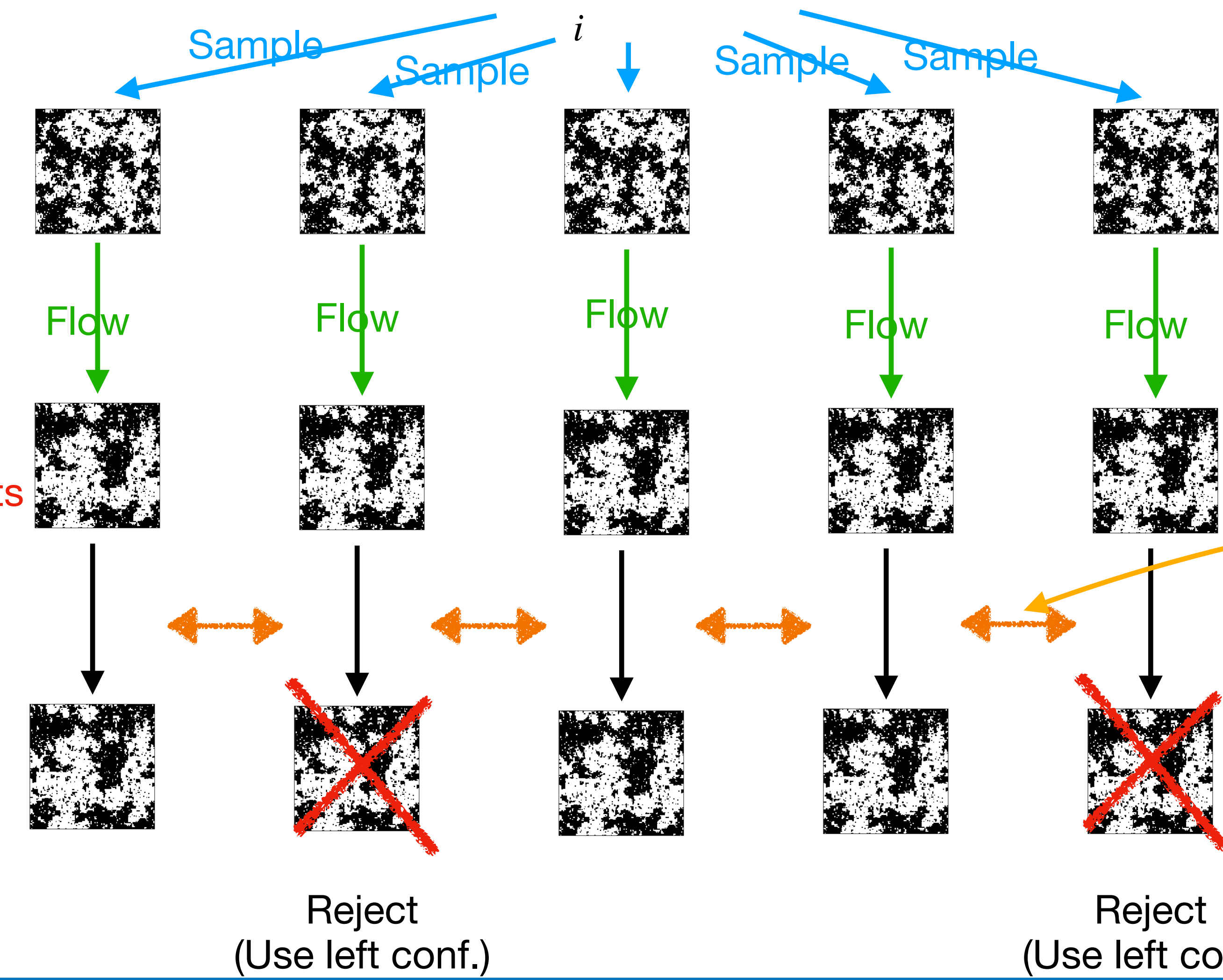
Easy

Flow-based sampling algorithm

$$\prod_i^{Vol} e^{-V(\varphi_i)}$$

Trivial theory
(no kinetic term, no topology)

No auto-correlation
No correlation for points



“Cooling = change of variable”
via trained neural net
“un-trivializing map”

No auto-correlation
Approx.correlation for points

Metropolis-*Hastings* with
$e^{-S}/e^{-V(\varphi_i)}J^{-1}[\varphi]$

Small auto-correlation
Correct correlations

Normalizing flow in Julia

Public code and improvement

GomalizingFlow.jl: A Julia package for Flow-based sampling algorithm for lattice field theory

Akio Tomiya

Faculty of Technology and Science, International Professional University of Technology, 3-3-1, Umeda, Kita-ku, Osaka, 530-0001, Osaka, Japan

Satoshi Terasaki

AtelierArith, 980-0004, Miyagi, Japan

A public code for Flow-based sampling algorithm not only 2d but also 3d, 4d

Combinational-convolution for flow-based sampling algorithm

Akio Tomiya
International Professional University of Technology in Osaka
Faculty of Technology and Science,
International Professional University of Technology,
3-3-1, Umeda, Kita-ku, Osaka, 530-0001, Osaka, Japan
akio@yukawa.kyoto-u.ac.jp

Satoshi Terasaki
AtelierArith
980-0004 Miyagi Japan
s.terasaki@atelier-arith.jp

Improvement of convolution for the flow has been reported in NurIPS2022 workshop

https://github.com/AtelierArith/GomalizingFlow.jl https://arxiv.org/abs/2208.08903 https://ml4physicalsciences.github.io/2022/

Summary

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

1. What and why QCD/lattice QCD?
    1. Problem: Long auto-correlation
2. Lattice QCD + Machine learning
    1. Trainable smearing + SLHMC = adaptive reweighting
    2. Flow-based sampling algorithm

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