OsakaU, Sep 23, 2022
10:50-11:30 (40 min)

http://www-het.phys.sci.osaka-u.ac.jp/Onogi-workshop/

Neural net for lattice QCD, and quantum methods for finite-temperature and density

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

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

Based on arXiv:
arXiv: 2103.11965,
2205.08860 etc

Happy “Kann-reki”(full circle) birthday

Akio Tomiya

Onogi-san and me

還暦おめでとうございます!

When I entered to master course in Osaka university in 2010 (12 years ago!),
Tetsuya was the second year as a professor in Osaka university
(My supervisor Fukaya-san also joined HEP group as an assistant prof. in 2010)

We had a lot of study groups: String theory, lattice field theory, RG

http://kabuto.phys.sci.osaka-u.ac.jp/HETphoto/HETCamp2010.html

Happy "Kann-reki"(full circle) birthday

Akio Tomiya

Onogi-san and me

2015

Onogi-san = Supervisor's Supervisor's of me = Great master (大師匠 おおししょう)!

And a mentor of me (he always encouraged me).

He taught me quantum field theory (perturbation theory/SM/RG/GWW trs), lattice field theory, algorithms, how to read a code, general relativity, etc
In particular, my master thesis about many flavor QCD.

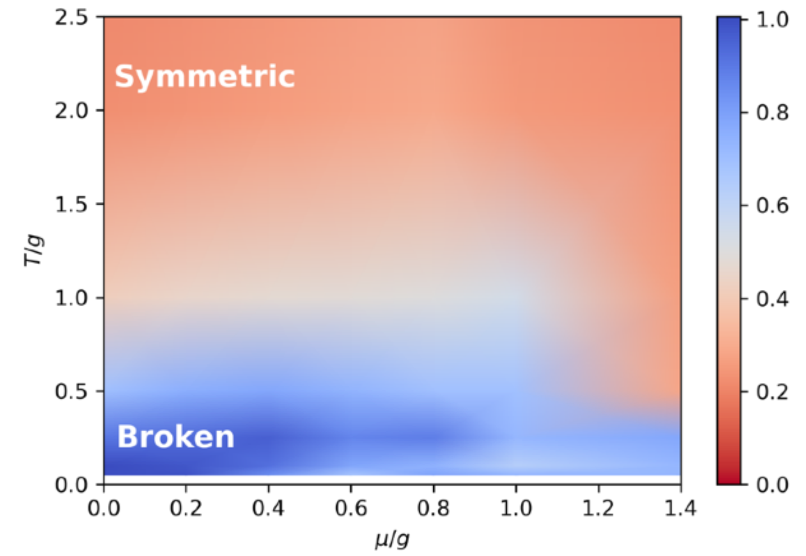
Thank you and congratulations Onogi-san!

Outline

Two exotic topics

1. What and why QCD/lattice QCD?
2. Lattice QCD + Machine learning
    1. “Neural net = Smearing”
3. Lattice QCD + Quantum algorithm
    1. Finite temp/dens for QFT

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



Introduction

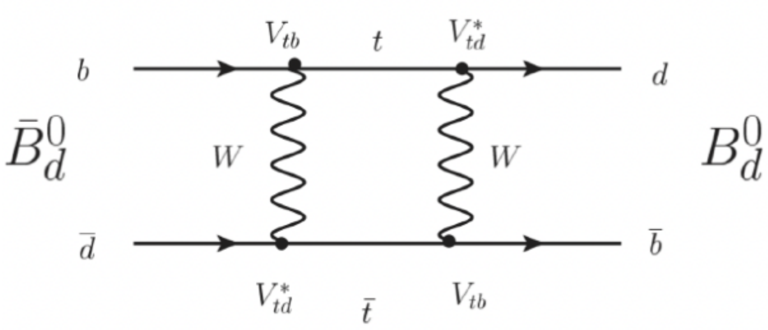
QCD: a fundamental theory of particles inside of nuclei

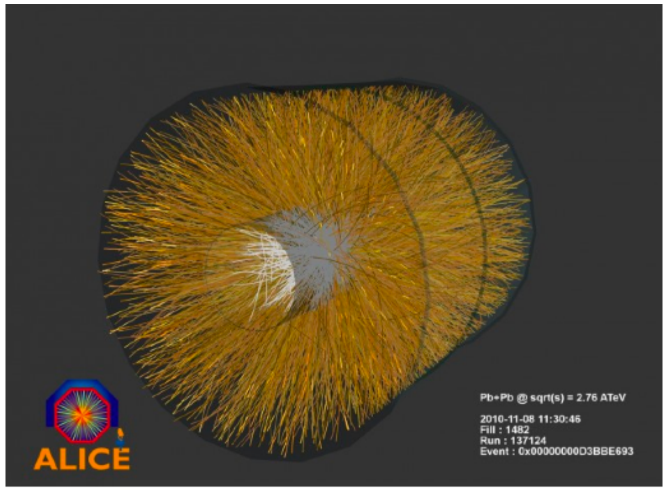
QCD (Quantum Chromo-dynamics) in 3 + 1 dimension

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

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

Quantization: $|\psi(t)\rangle = \mathrm{e}^{-\mathrm{i}Ht}|\psi(0)\rangle$ $\qquad [A_\mu, E_\nu] = \mathrm{i}\hbar\delta_{\mu\nu}$





- QCD enables us to calculate (in principle):
  - Equation of state of neutron star, Tc
  - Scattering of quarks and gluons, Parton distributions
  - Hadron spectrum!
- Strongly coupled quantum system
- Use lattice QCD + Monte-Carlo

Introduction

Lattice path integral > 1000 dim, Trapezoidal int is impossible

K. Wilson 1974

Imaginary time $t = -\mathrm{i}\tau$

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

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

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

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

Path integral formalism

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

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

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

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

We cannot control numerical error

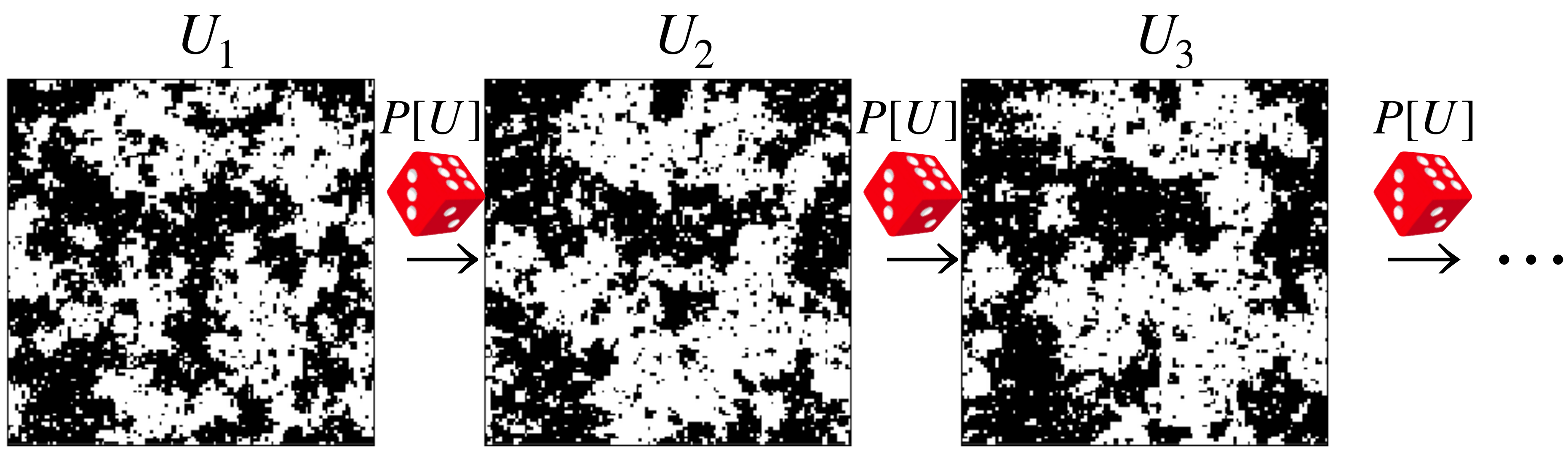
Introduction

Monte-Carlo integration is available

M. Creutz 1980

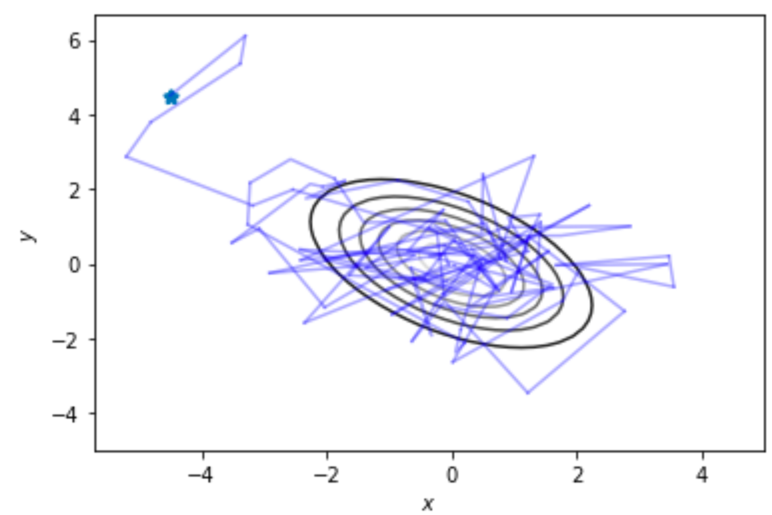
$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}U e^{-S_{\mathrm{eff}}[U]} \mathcal{O}(U) \qquad S_{\mathrm{eff}}[U] = S_{\mathrm{gauge}}[U] - \log \det(\mathbb{D}[U] + m)$$

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



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

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



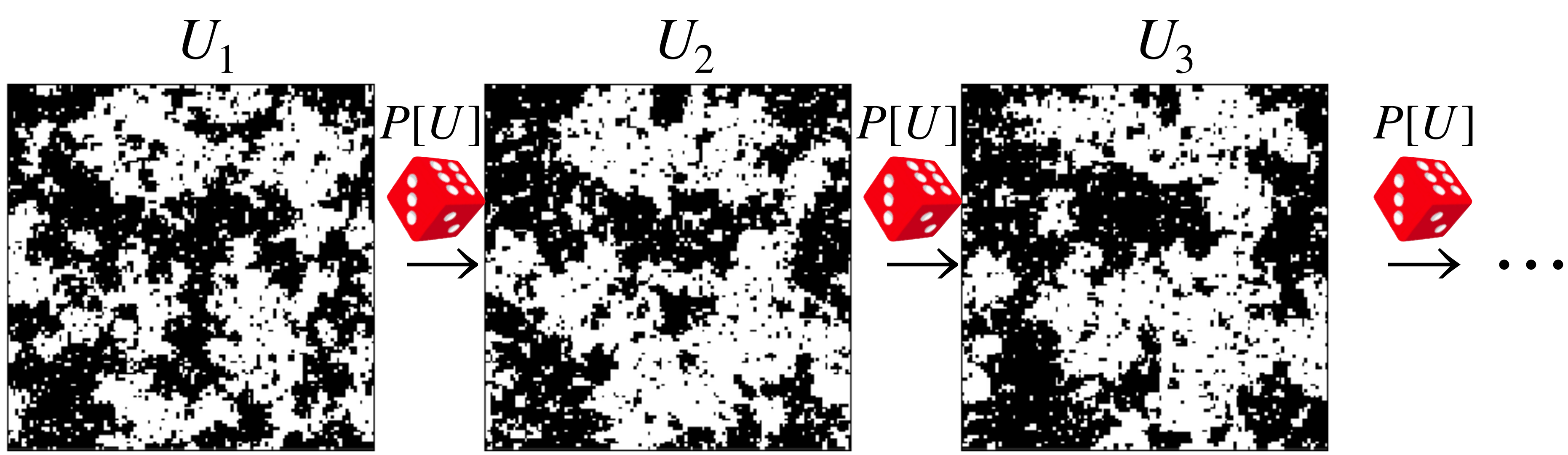
Introduction

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 "$P[U] = \frac{1}{Z} e^{-S_{\text{eff}}[U]}$". **It gives expectation value**



Error of integration is determined by the number of sampling

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

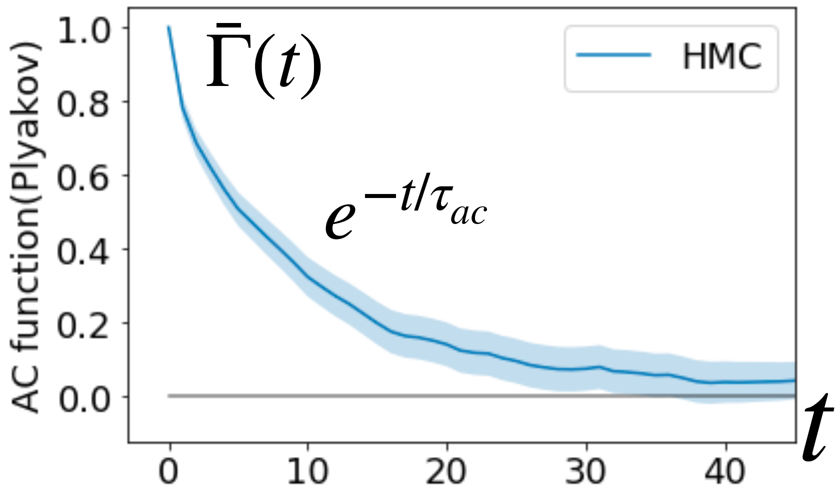
Introduction

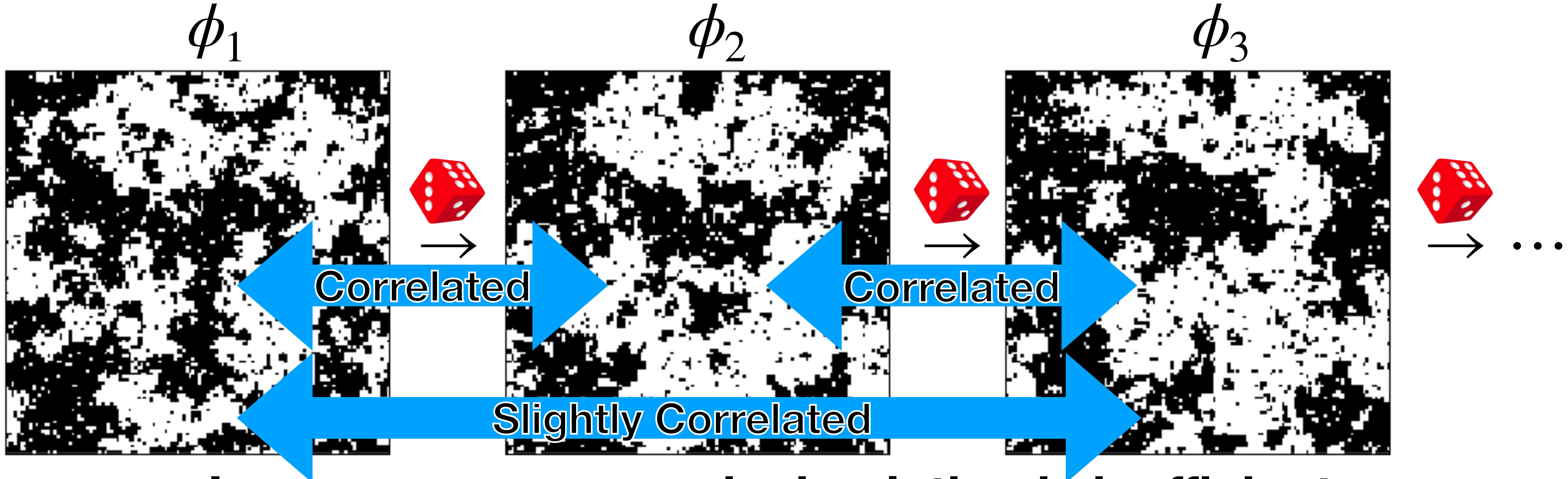
Correlation between samples = inefficiency of calculation

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

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

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





Large τ_{ac} means, such simulation is inefficient

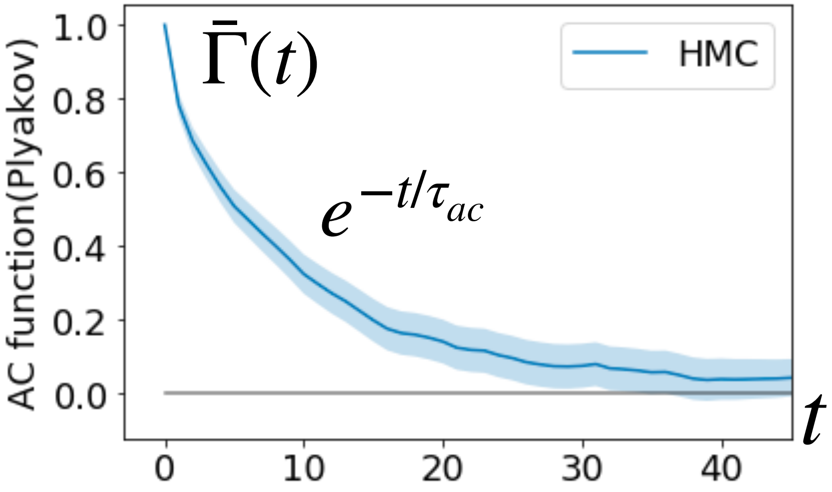
Introduction

Summary for now: long autocorrelation = inefficiency

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

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

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



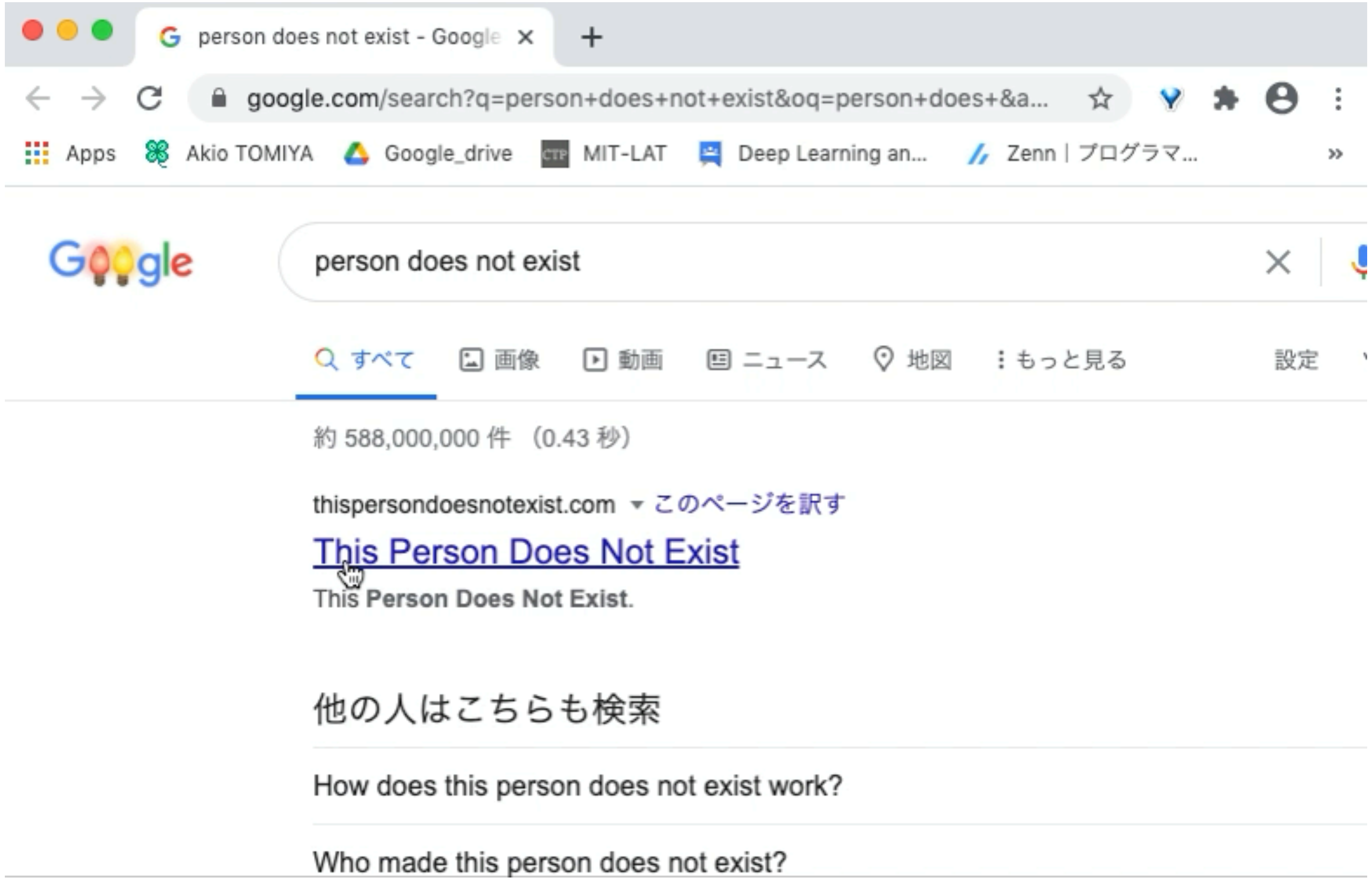
τ_{ac} is given by an update algorithm (N. Madras et. al 1988)

- 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

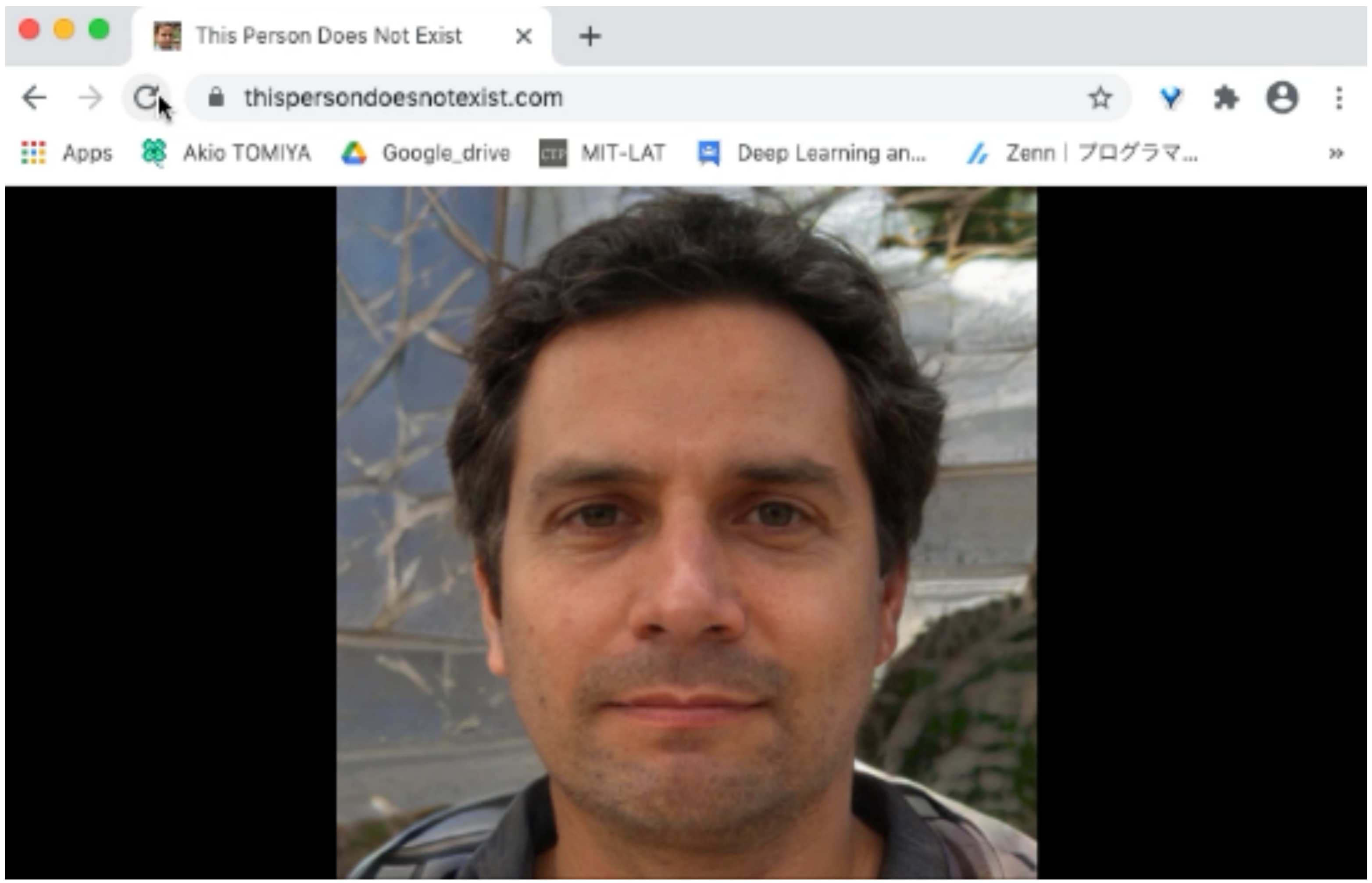
Neural net can make human face images



Introduction

Neural net can make human face images

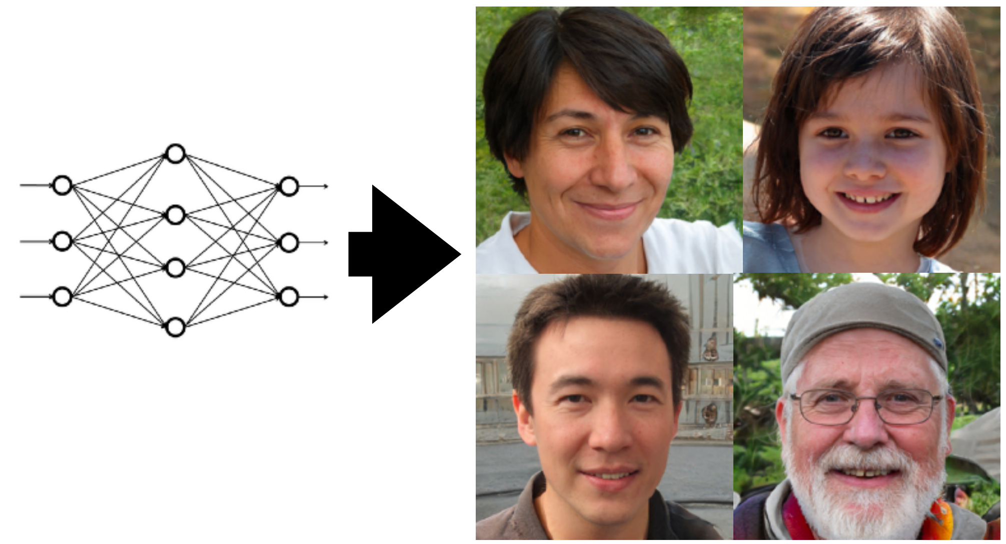
Neural nets can generate realistic human faces (Style GAN2)



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

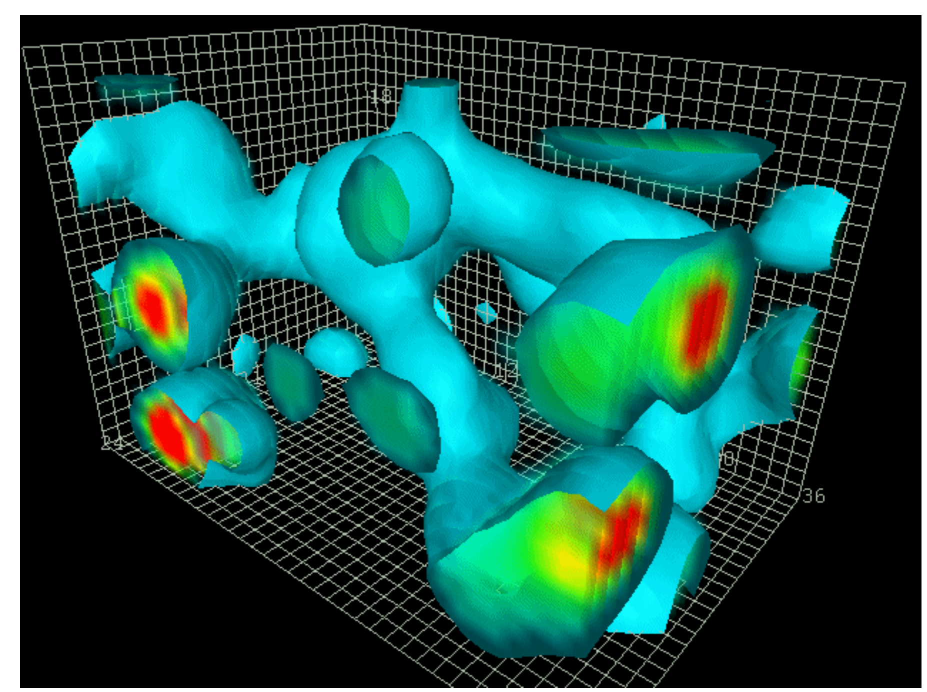
ML for LQCD is needed

- 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?	Lattice2021/ref
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

Neural network is a universal approximator of functions

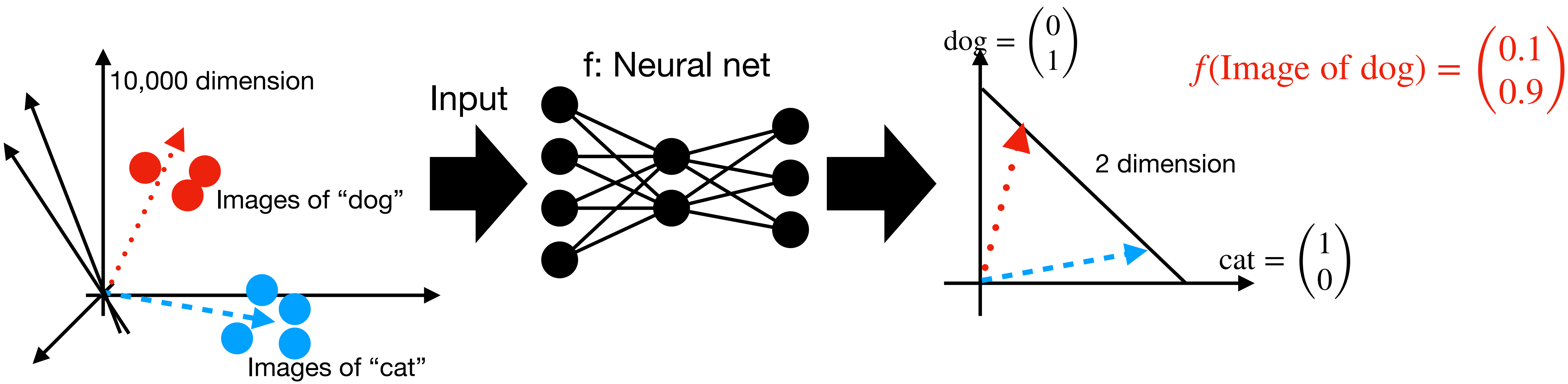
Image classification, cats and dogs

100x100

Flatten $\Longrightarrow$ $\begin{pmatrix} 0.000 \\ 0.000 \\ 0.8434 \\ 0.756 \\ 0.3456 \\ \vdots \end{pmatrix}$ Image is a vector (this is 10,000 dimension)

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

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



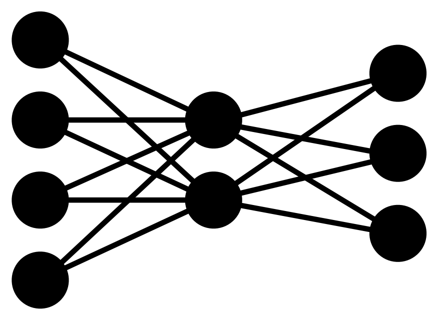
Introduction

Affine transformation + element-wise transformation

<u>Fully connected neural networks</u>

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

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



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

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

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

Akio Tomiya

Neural network is a universal approximator of functions

Image classification, cats and dogs

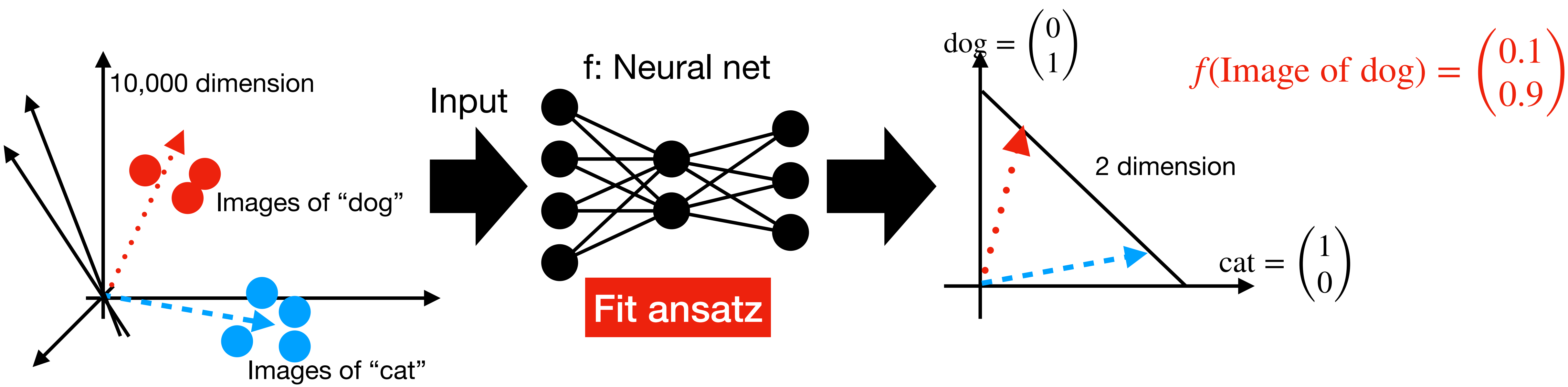
100x100

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

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

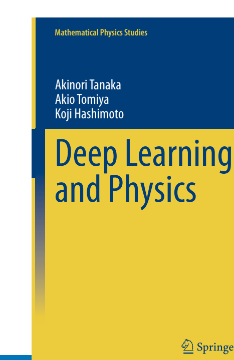
$$\text{dog} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

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



<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)

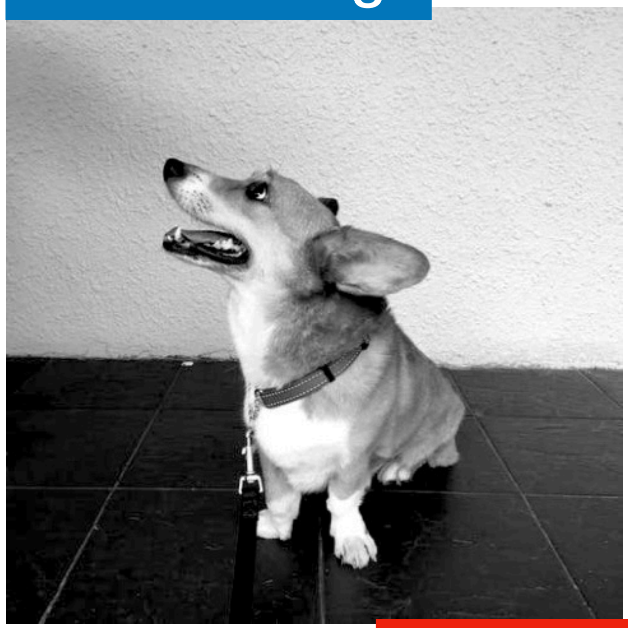


What is the neural networks?

Akio Tomiya

Convolution layer = trainable filter

Filter on image



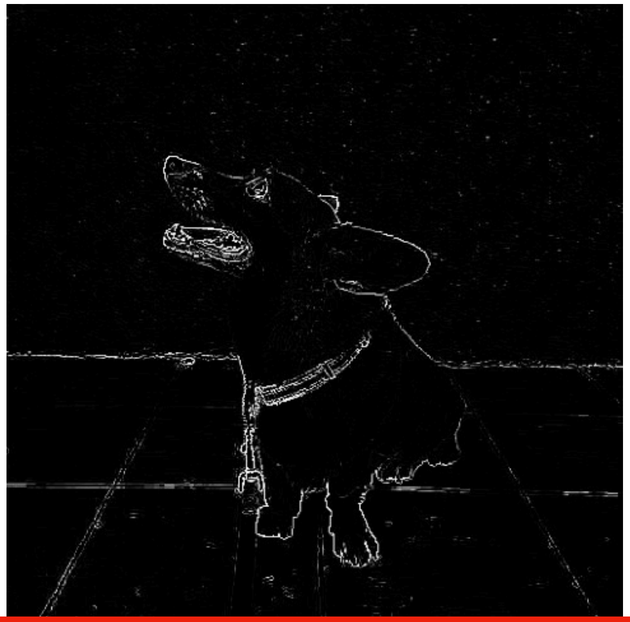
$*$

Laplacian filter

0	1	0
1	-2	1
0	1	0

(Discretization of ∂^2)

$=$



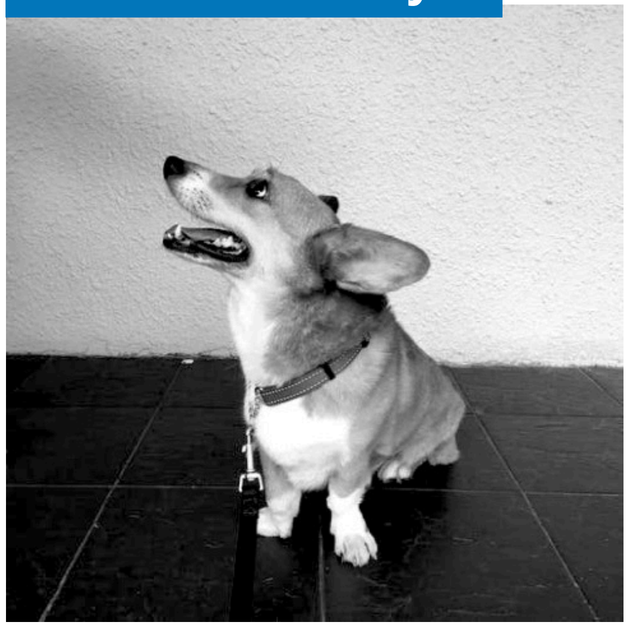
Edge detection

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

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

Convolution layer

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



$*$

Trainable filter

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

$\rightarrow$

Edge detection

Smoothing (Gaussian filter)

…

Gaussian filter

$\frac{1}{16}$

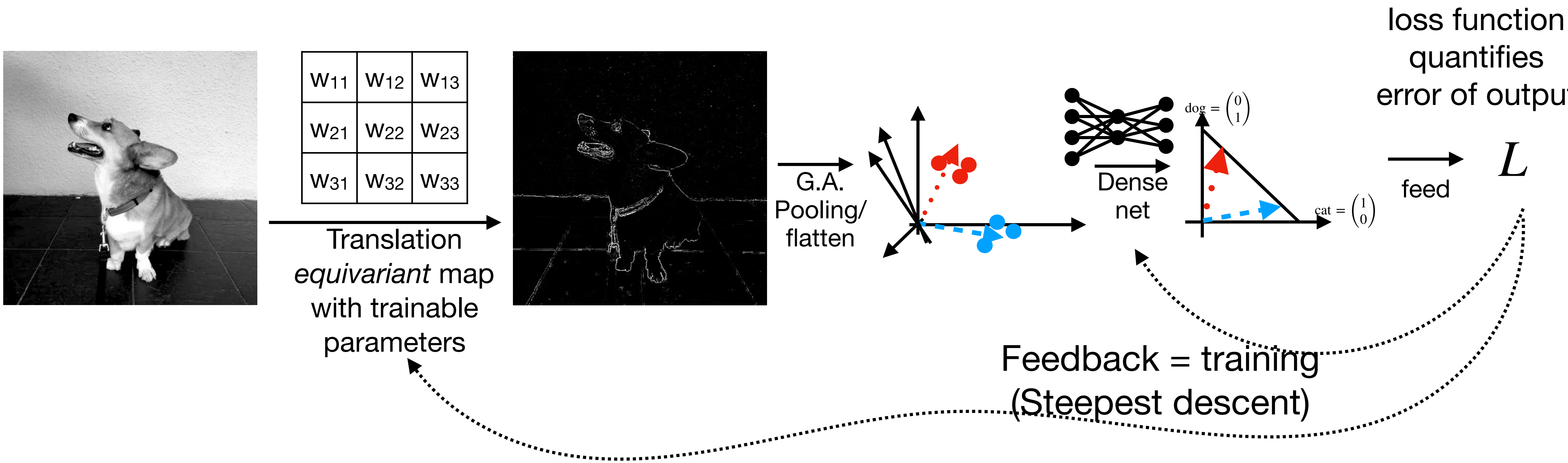
1	2	1
2	4	2
1	2	1

This can be any filter which helps feature extraction but still transitionally equivariant!

Convolution neural network

Akio Tomiya

Training can be done with back propagation



Smearing

Smoothing improves global properties

Eg.

Coarse image

Numerical derivative is unstable

Gaussian filter

$\frac{1}{16}$

1	2	1
2	4	1
1	2	1

Smoothened image

Numerical derivative is stable

We want to smoothen gauge configurations with keeping gauge symmetry

Two types:

APE-type smearing

Stout-type smearing

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

Smearing

Smoothing with gauge symmetry, APE type

Akio Tomiya

M. Albanese+ 1987
R. Hoffmann+ 2007

APE-type smearing

Covariant sum

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

Normalization

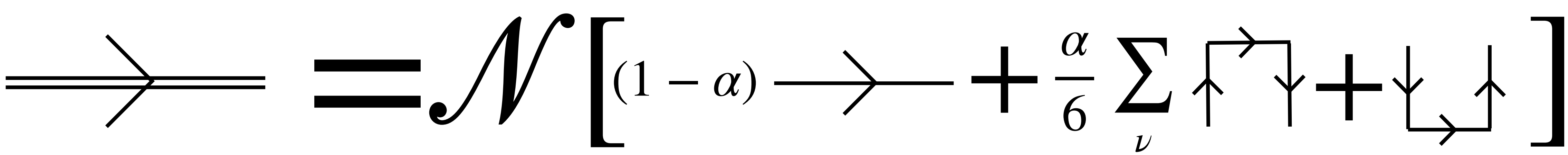
$$\mathcal{N}[M] = \frac{M}{\sqrt{M^\dagger M}}$$

Or projection

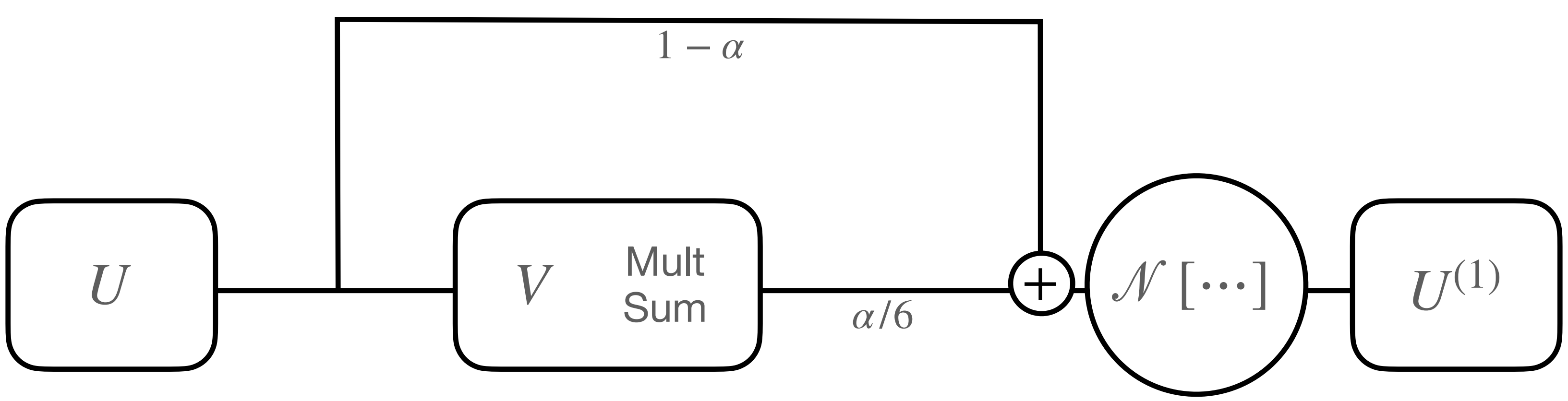
$$V_\mu^\dagger[U](n) = \sum_{\mu\neq\nu} U_\nu(n)U_\mu(n+\hat{\nu})U_\nu^\dagger(n+\hat{\mu}) + \cdots$$

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

Schematically,



In the calculation graph,



Smearing

Smoothing with gauge symmetry, stout type

C. Morningster+ 2003

Stout-type smearing

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

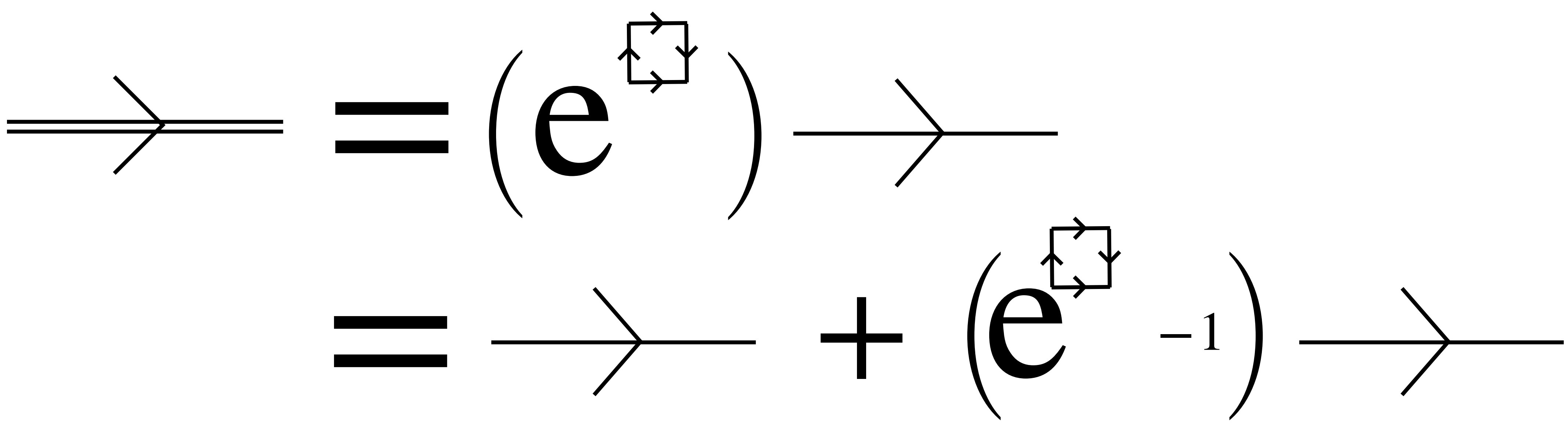
$$= U_\mu(n) + (\mathrm{e}^{Q} - 1) U_\mu(n)$$

Covariant sum

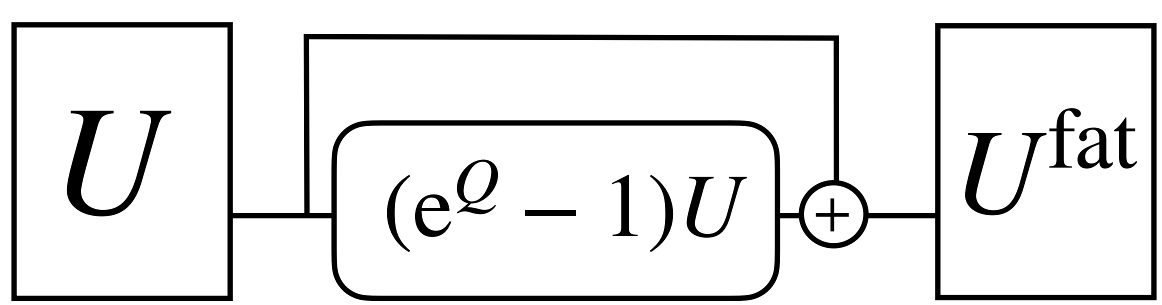
Q: anti-hermitian traceless plaquette

This is less obvious but this actually obeys same transformation

Schematically,



In the calculation graph,



Smearing

Akio Tomiya

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)

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

Gauge covariant sum

A local function

It has similar structure with neural networks,

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

Matrix product
vector addition

element-wise (local)
Non-linear transf.
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)
Training?	Backprop + Delta rule	AT Nagai 2103.11965

Well-known

(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

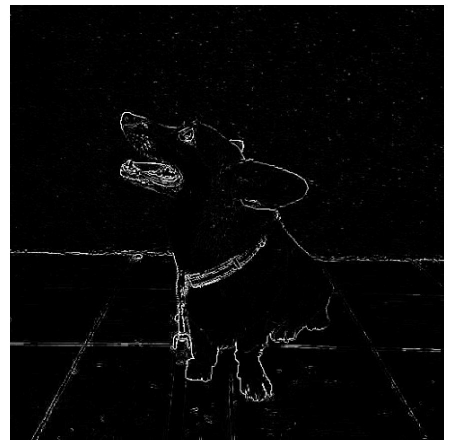
AT Y. Nagai arXiv: 2103.11965

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

Usual neural network

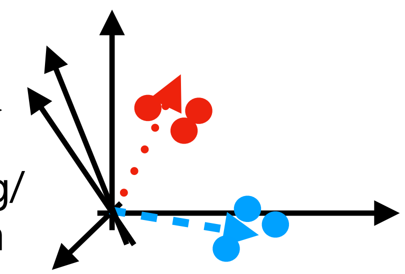
w_{11}	w_{12}	w_{13}
w_{21}	w_{22}	w_{23}
w_{31}	w_{32}	w_{33}

Translation *equivariant* map with trainable parameters

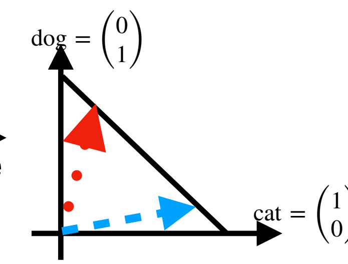


Translation *equivariant* = the image is shifted, output image is also shifted

G.A. Pooling/ flatten



Dense net

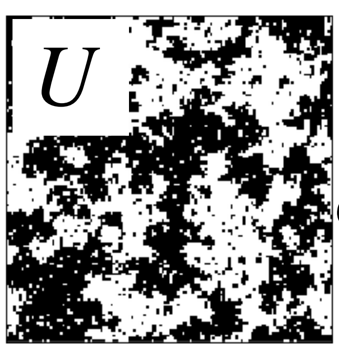


feed

Invariant loss function

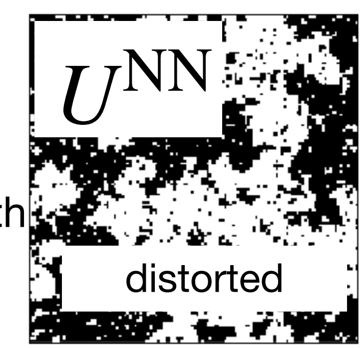
L

Covariant neural networks

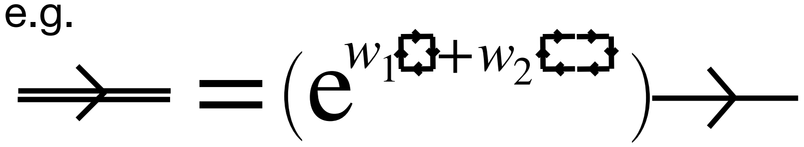


$U_\mu^{\rm NN}(n)[U]$

Gauge covariant map with trainable parameters



e.g.



Feed to Dirac op

Covariant

$D\left[U_\mu^{\rm NN}(n)[U]\right] \longrightarrow S_{\rm ferm}\left[U^{\rm NN}\right] \xrightarrow{\text{feed}} L$

Parametrized Dirac operator

Parametrized action

Invariant loss function

Construct loops

Covariant

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

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

$\searrow Q\left[U^{\rm NN}\right] \longrightarrow L$

Parametrized loop operators (e.g. plaquette, Polyakov loop)

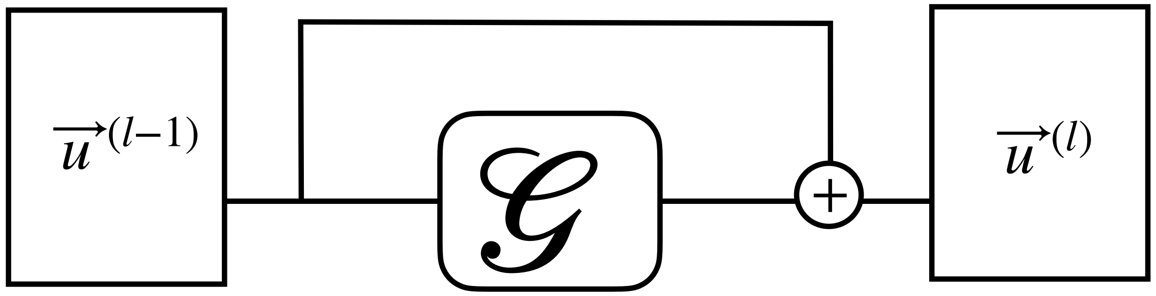
Topological charge

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

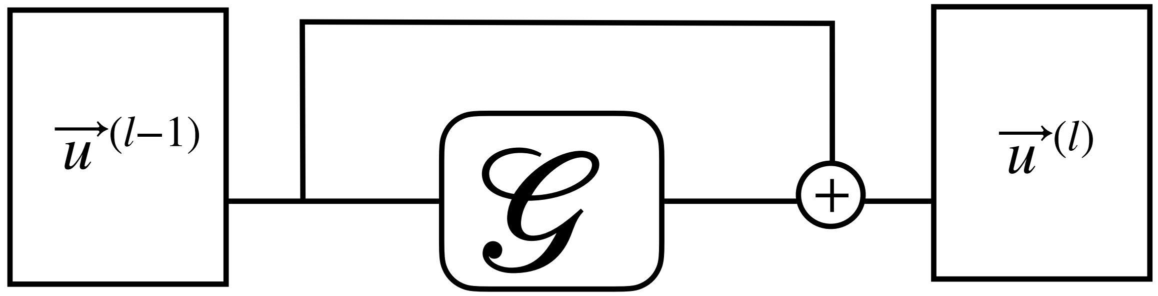
$$\frac{d\vec{u}^{(t)}}{dt} = \mathcal{G}(\vec{u}^{(t)})$$

arXiv: 1806.07366
(Neural IPS 2018 best paper)

Gauge covariant neural network

Neural ODE of Cov-Net = “gradient flow”

ResNet



arXiv: 1512.03385

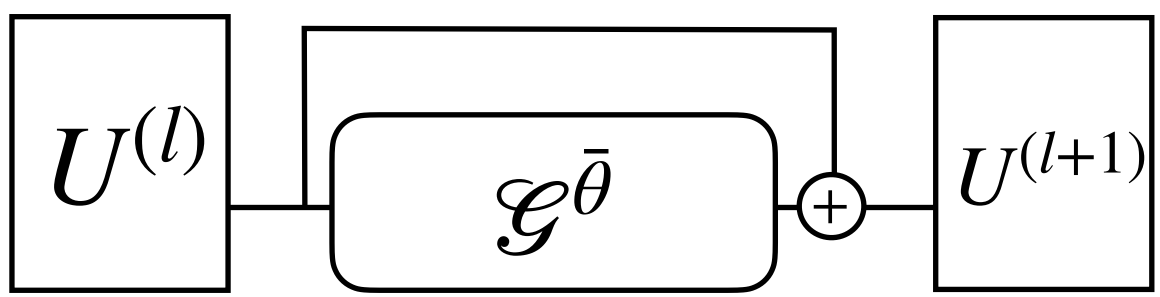
↓ Continuum Layer Limit

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

Akio Tomiya

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.

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

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

Test of our neural network?

Can we mimic a different Dirac operator using neural net?

Artificial example for HMC:

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

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

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

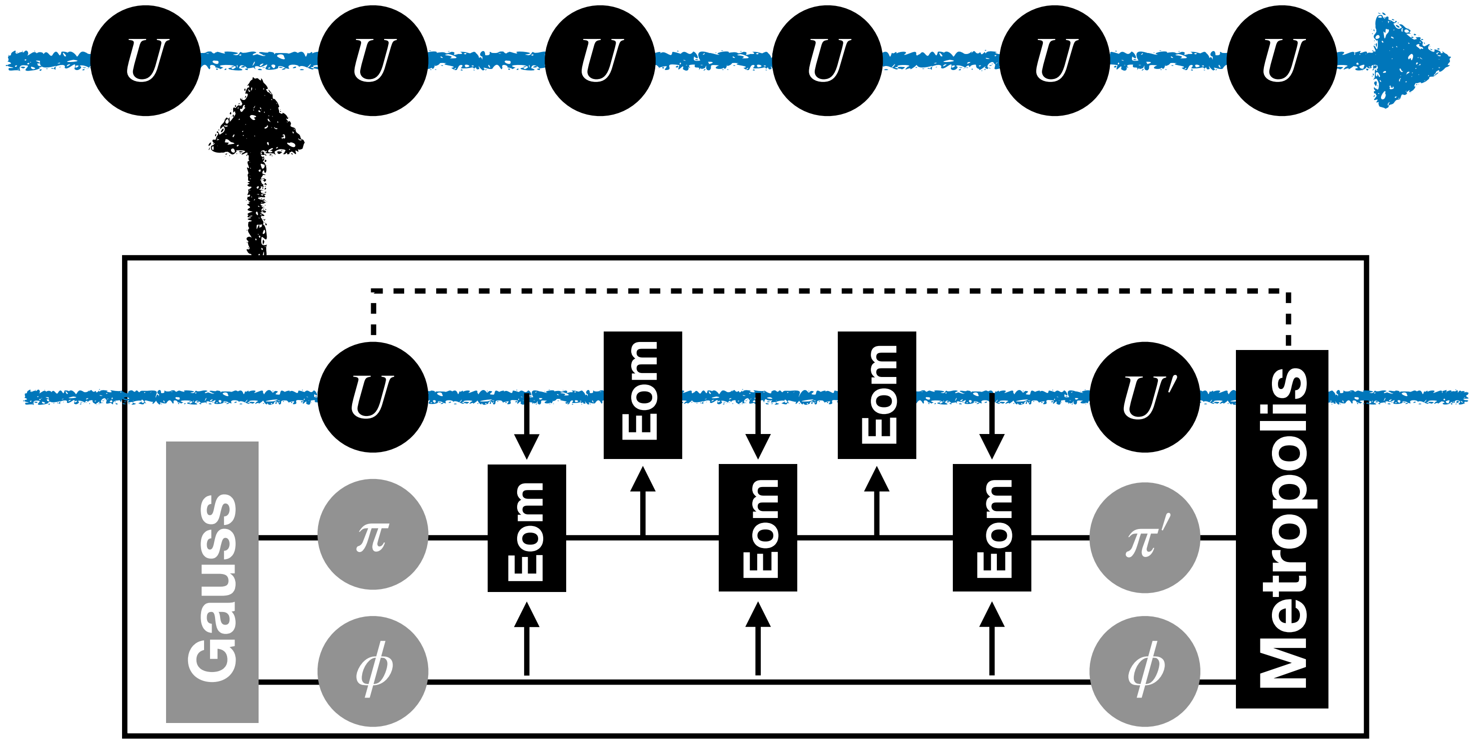
SLHMC = Exact algorithm with ML

Akio Tomiya

SLHMC for gauge system with dynamical fermions

arXiv: 2103.11965 and reference therein

HMC



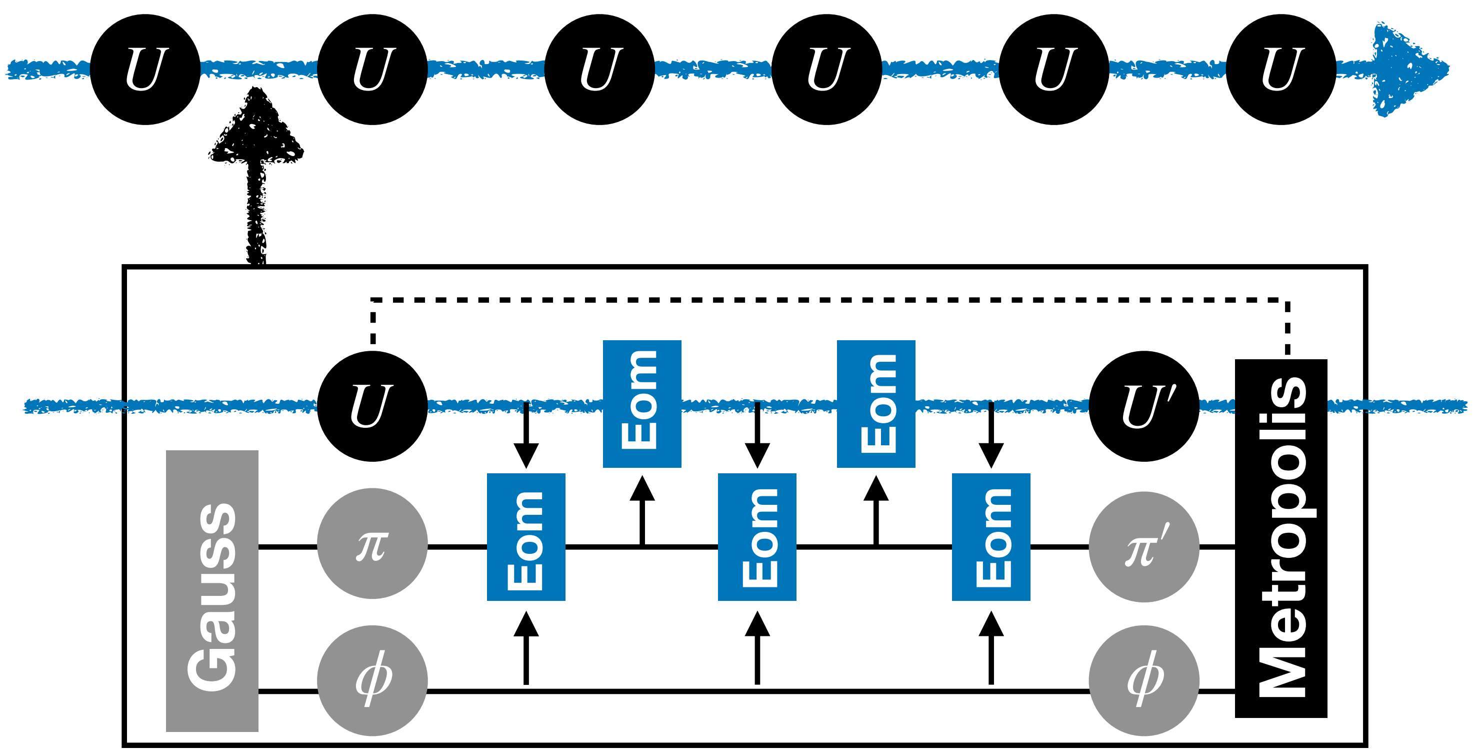
Eom **Metropolis**

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

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

(But we added some functions on the public version)

AT+ (in prep)

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

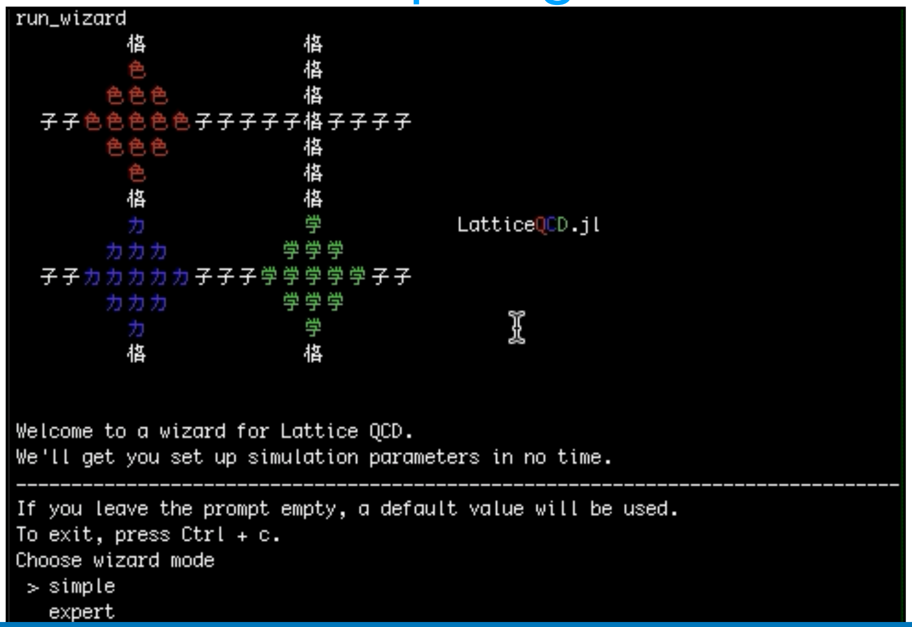
LatticeQCD.jl (Official package) : Laptop/desktop/PC-cluster/Jupyter (Google colab)

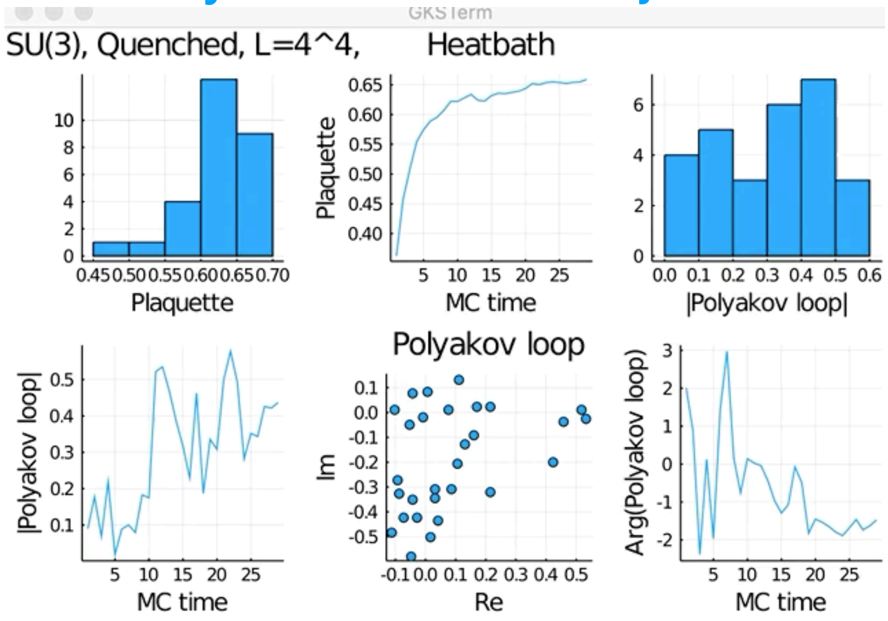
SU(Nc)-heatbath/SLHMC/SU(Nc) Stout/(R)HMC/staggered/Wilson-Clover
Domain-wall/Measurements (Now updating to v1.0, MPI ver is ready)

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



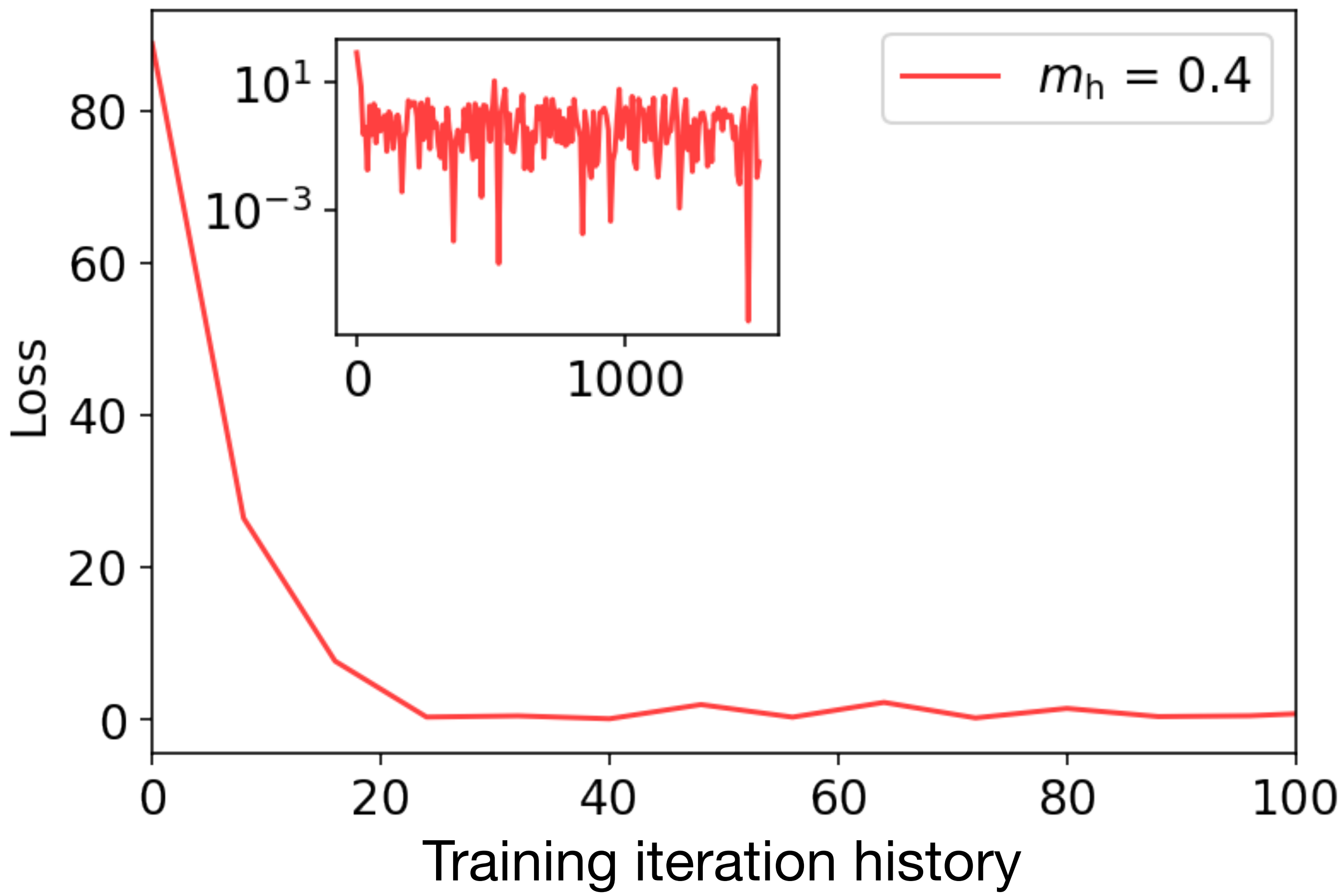


Details (skip)

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.

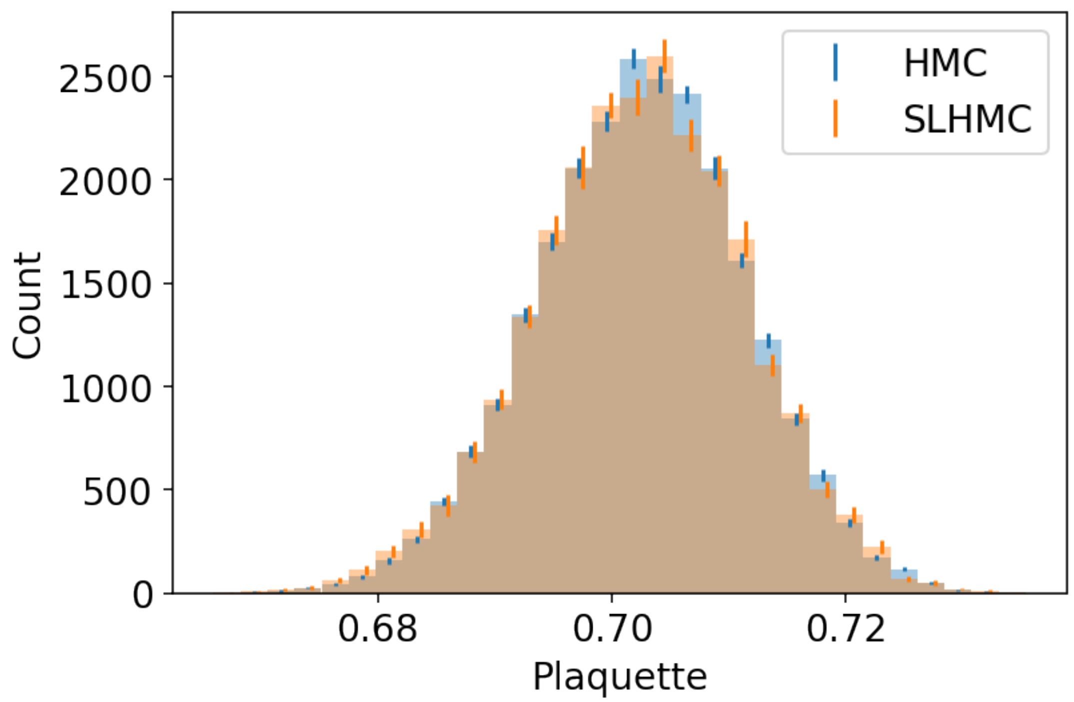
After training,   e^(-L) ~1, and we get practical acceptance rate!

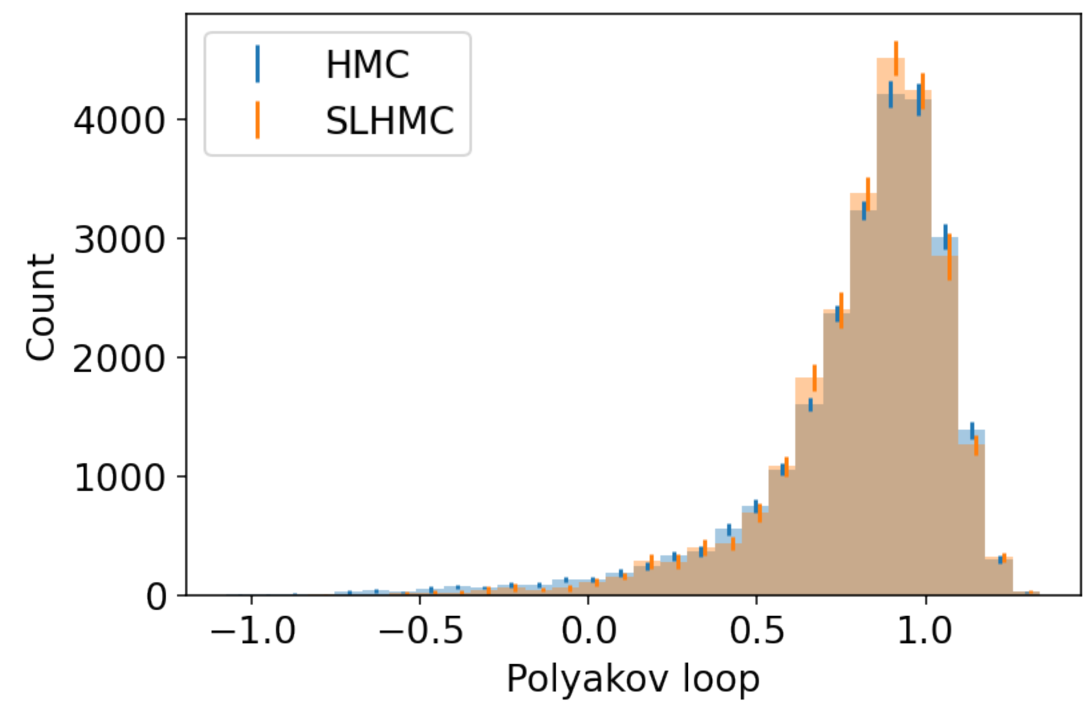
Akio Tomiya

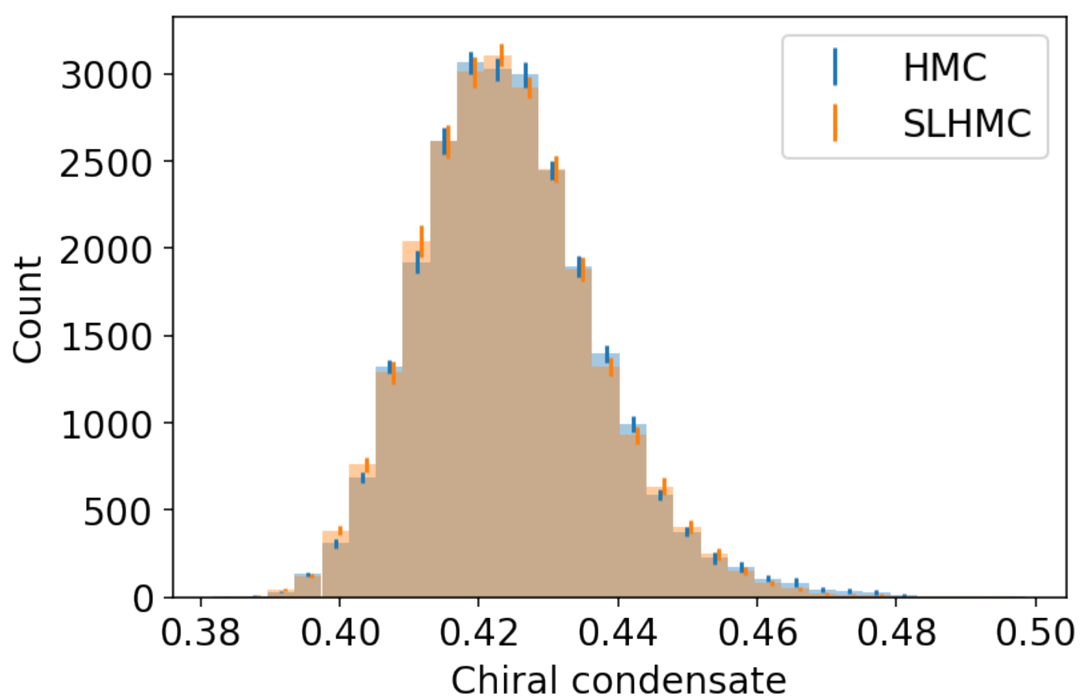
Application for the staggered in 4d

Results are consistent with each other

arXiv: 2103.11965







Expectation value

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

Acceptance = 40%

Future work: Domain-wall/Overlap SLHMC (?)

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

Akio Tomiya

Normalizing flow ~ Change of variables

Simplest example: Box Muller

Change of variables

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

$$\int_{-\infty}^{\infty} dx \int_{-\infty}^{\infty} dy \; e^{-\frac{1}{2}x^2 - \frac{1}{2}y^2} = \frac{1}{2}\int_0^{2\pi} d\theta \int_0^1 dz$$

Original integral: hard

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

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

It becomes Gaussian integral! Easy to evaluate!!

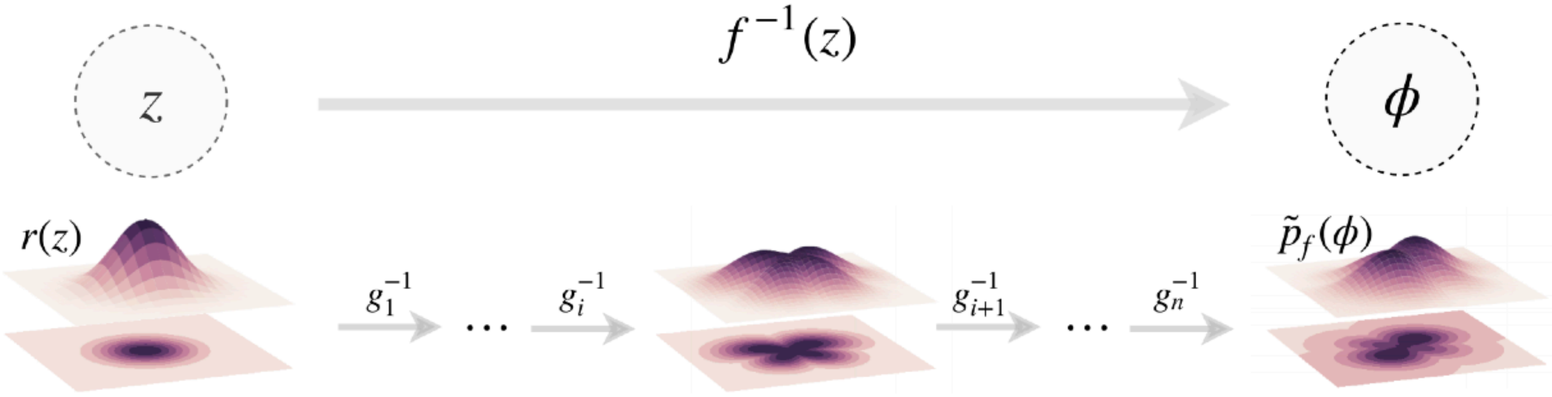
However, the Jacobian cannot evaluate easily, so it is not practical.
Life is hard.

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

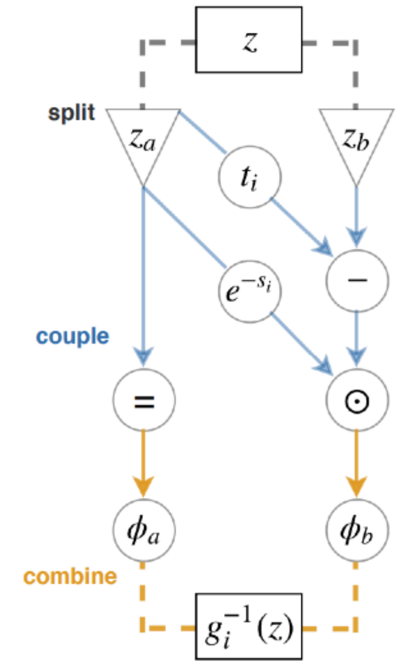
Related works

Flow based algorithm = neural net represented flow algorithm

MIT + Google brain 2019~



(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, $p(\phi)$.

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!

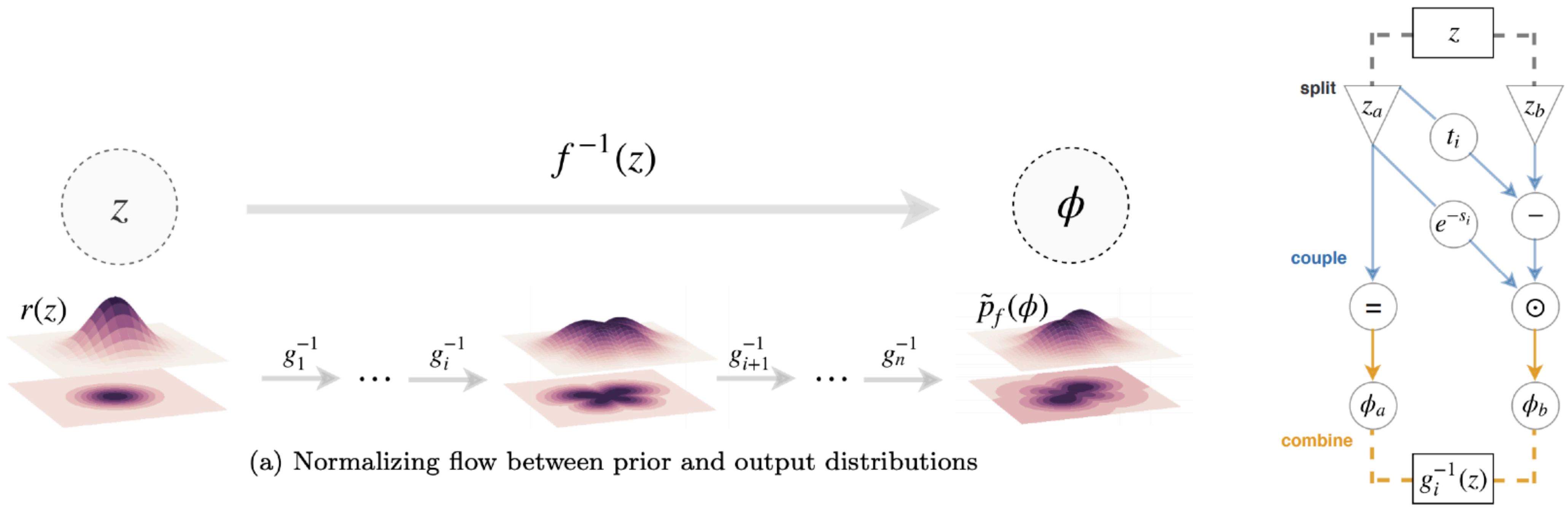
arxiv 1904.12072, 2003.06413, 2008.05456

Related works

Akio Tomiya

Flow based algorithm = neural net represented flow algorithm

MIT + Google brain 2019~



(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, $p(\phi)$.

Their sampling strategy

sample gaussian → inverse trivializing map → QFT configurations
Tractable Jacobian (by even-odd strategy)
After sampling, Metropolis-Hastings test (Detailed balance)→ exact!

arxiv 1904.12072, 2003.06413, 2008.05456

Akio Tomiya

Flow based sampling algorithm

Flow based ML for QFT

MIT + Deepmind + …

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

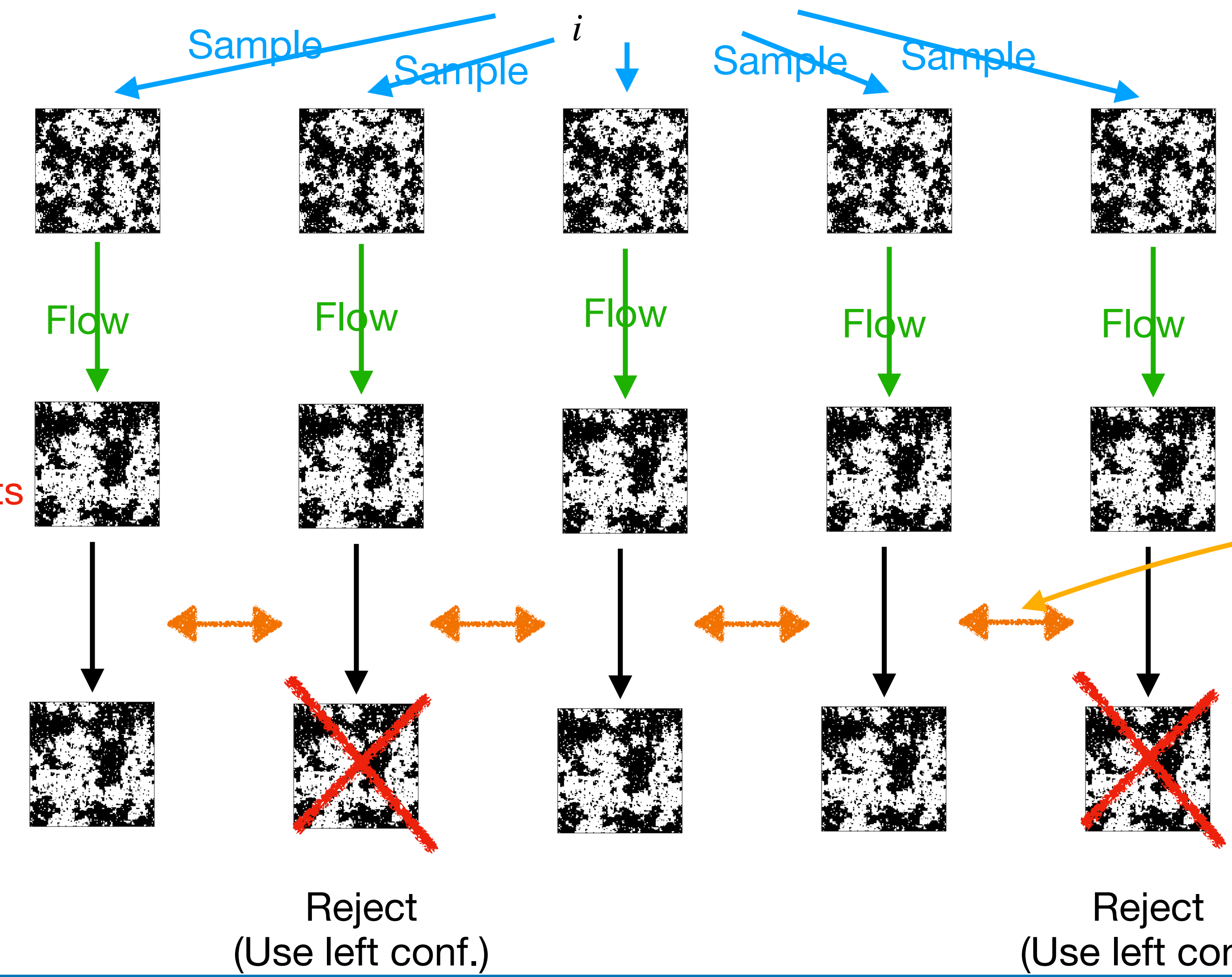
Original integral: hard

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

We made a public code in Julia Language

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

https://arxiv.org/abs/2208.08903

Abstract

GomalizingFlow.jl: is a package to generate configurations for quantum field theory on the lattice using the flow based sampling algorithm in Julia programming language. This software serves two main purposes: to accelerate research of lattice QCD with machine learning with easy prototyping, and to provide an independent implementation to an existing public Jupyter notebook in Python/PyTorch. GomalizingFlow.jl implements, the flow based sampling algorithm, namely, RealNVP and Metropolis-Hastings test for two dimension and three dimensional scalar field, which can be switched by a parameter file. HMC for that theory also implemented for comparison. This package has Docker image, which reduces effort for environment construction. This code works both on CPU and NVIDIA GPU.

Keywords: Lattice QCD, Particle physics, Machine learning, Normalizing flow, Julia

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

LQCD + Quantum galtorithm

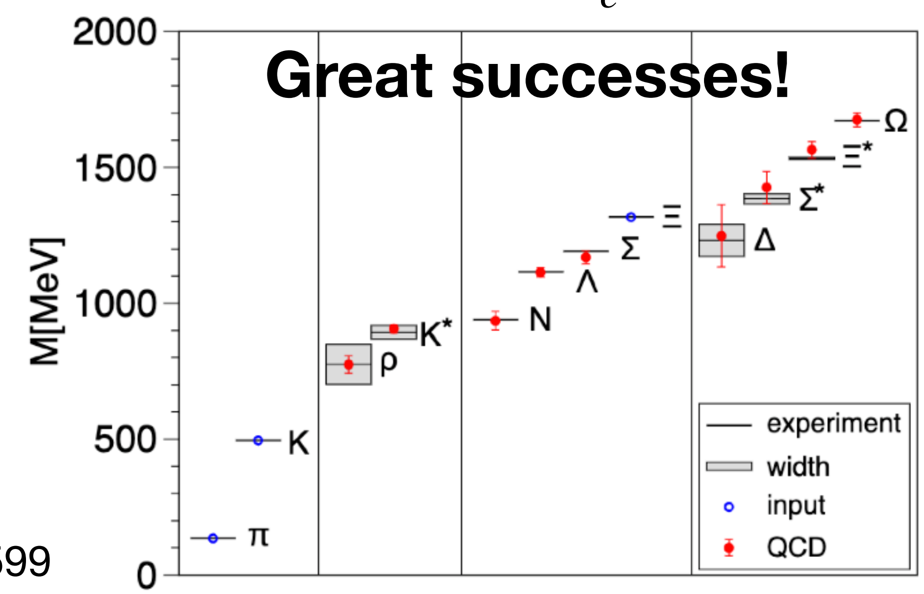
Motivation

Akio Tomiya

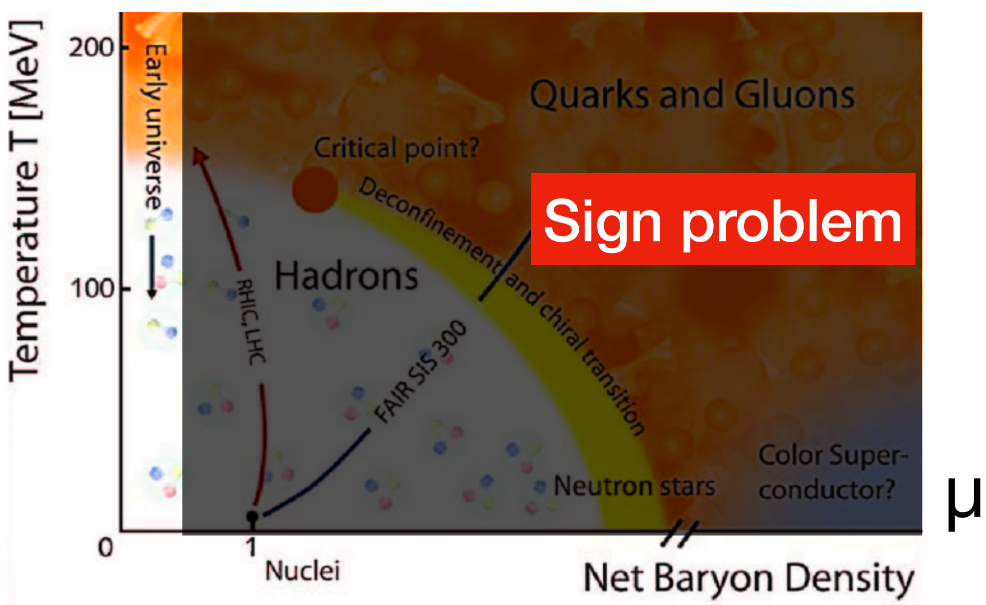
Sign problem prevents using Monte-Carlo

- Monte-Carlo enables us to evaluate expectation values for "statistical system", like lattice QCD in imaginary time

$$\langle O[U] \rangle = \frac{1}{N_{\rm conf}} \sum_{c}^{N_{\rm conf}} O[U_c] + \mathcal{O}\left(\frac{1}{\sqrt{N_{\rm conf}}} \right) \qquad U_c \leftarrow P(U) = \frac{1}{Z} e^{-S[U]} \in \mathbb{R}_+$$



arXiv:0906.3599



- If we turn on the baryon chemical potential μ, Monte-Carlo methods do not work because $e^{-S[U]}$ becomes complex. This is no more probability. (sign problem)
- Operator formalism does not have such problem! But it requires huge memory to store quantum states, which cannot be realized even on supercomputer.
- Quantum states should be stored on quantum device (Feynman)

Motivation

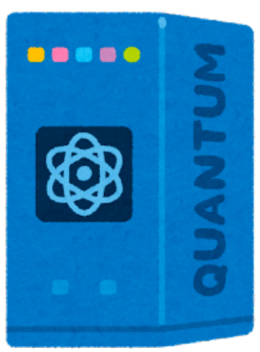
μ = 0 is good for Classical, T=0 is good for Quantum

Classical machine: Lattice field theory calculations rely on

$$P(U) = \frac{1}{Z} e^{-S[U]} \det(D[U] + m)^2 \in \mathbb{R}_+$$

Since 1980 (M. Creutz)~

- This P(U) cannot be regarded as probability if μ ≠ 0 (sign problem)



Quantum machines can realize (any) unitary evolutions (Solovay Kitaev theorem),

$$U(t) = \mathrm{e}^{-\mathrm{i}\hat{H}t}$$

Phys.Rev.D 105 (2022) 9, 094503
and references therein

- No problem for μ≠0 because we can only use unitary gates (operators)
- “Short time evolution” (shallow circuit) is preferred for near-term devices

	Classical Computers	Quantum Computers
Finite Density	Sign Problem	√
Finite Temperature	√	Challenging *

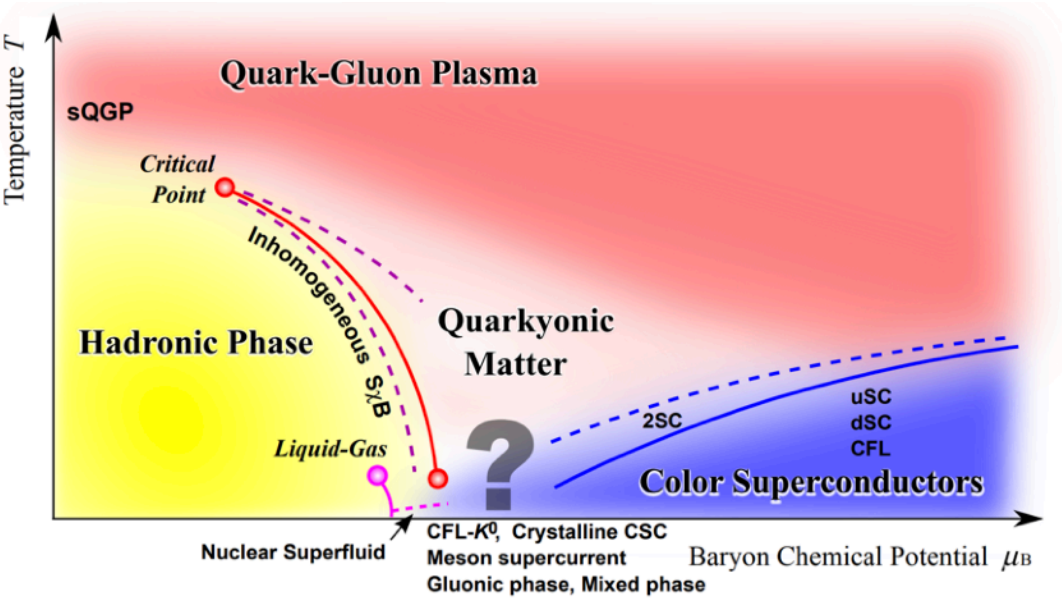
We need a method to calculate T>0 and μ≠0 for QCD
and for near-term quantum devices

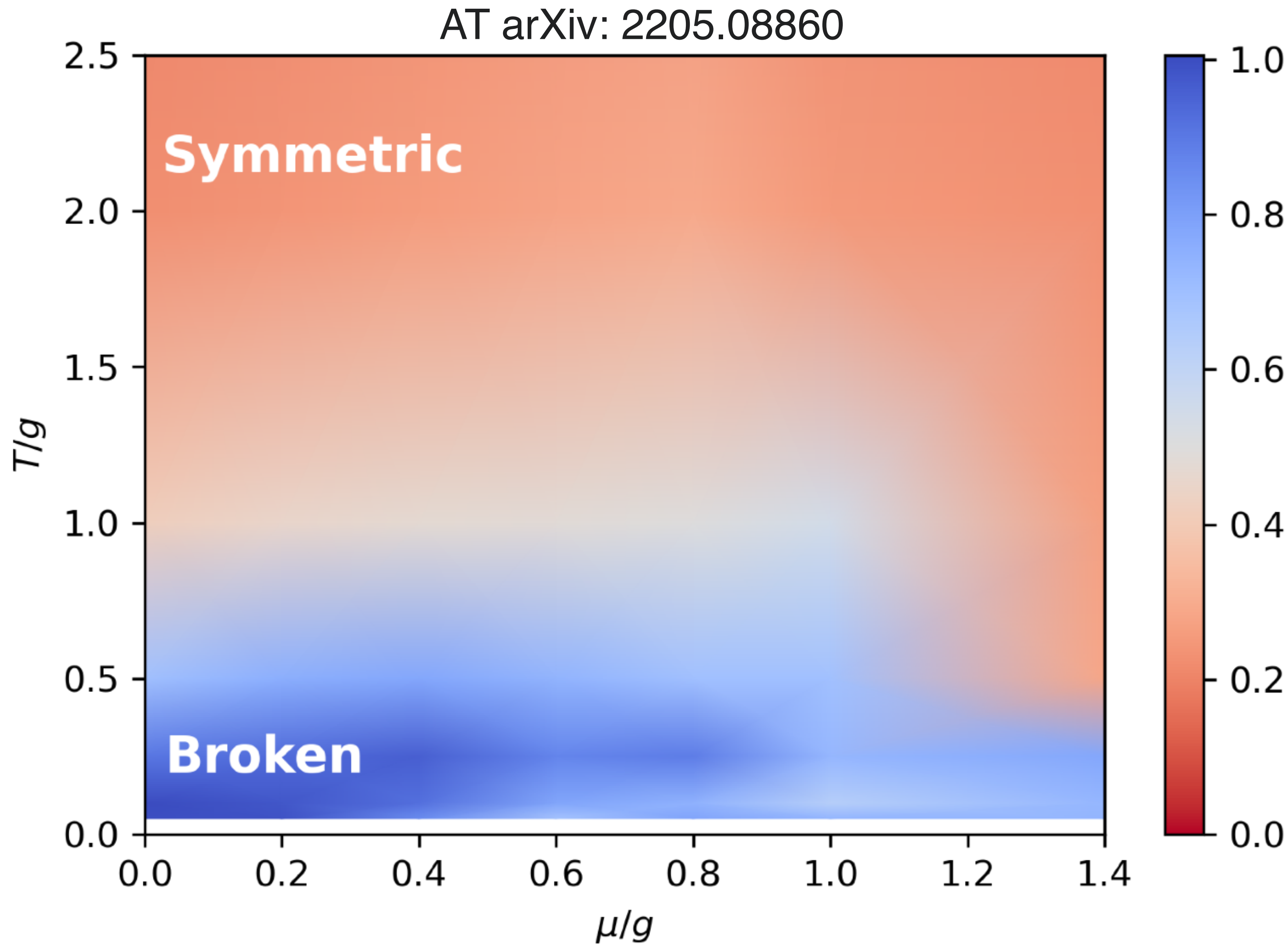
*https://indico.hiskp.uni-bonn.de/event/40/contributions/484/attachments/358/630/Powers%20Talk%20Final%20Draft%20Updated.pdf

Summary of this talk

Hybrid = Quantum algorithm + machine learning

Fukushima , Hatsuda
Rept.Prog.Phys.74:014001,2011





I investigated T-mu phase diagram using a <u>quantum algorithm</u> & <u>neural network</u> (β-VQE, No sign problem) for Schwinger model (toy model of QCD)

QFT with Hamiltonian

Akio Tomiya

Hamiltonian vs Lagrangian

Operator formalism (This work)

H : Hamiltonian in QFT

Real time

Finite temperature/imaginary time

Minkowski in M^{d+1}

Euclid in S^1 x M^d

$$U(t) = \mathrm{e}^{-\mathrm{i}tH}$$

Euclid(t → τ)

$t = -\mathrm{i}\tau$

Minkowski(τ → t)

$$U(\tau) = \mathrm{e}^{-\tau H}$$

$$\langle OO(\tau)\rangle = \mathrm{Tr}[O(0)O(\tau)\rho]$$

$$\rho = U(\tau)/Z$$

- Typical use case of quantum algorithm is for real-time. Unitary.
  - Time evolution: Correlators (e.g. 2pt on light-cone), etc
  - Main interest: $\langle\Omega|O|\Omega\rangle$, where $|\Omega\rangle$ is the exact ground state
- Difficulty: State preparation for exact ground state of H

State preparation, VQE and Beta-VQE

State preparation is hard

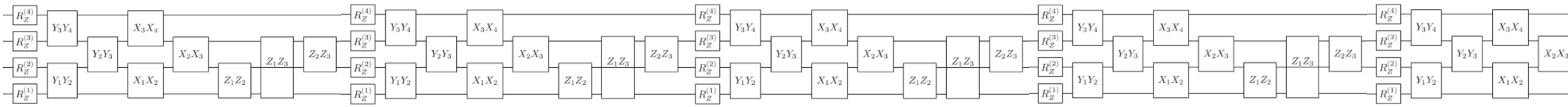
We are interested in expectation value with true ground state for Hamiltonian

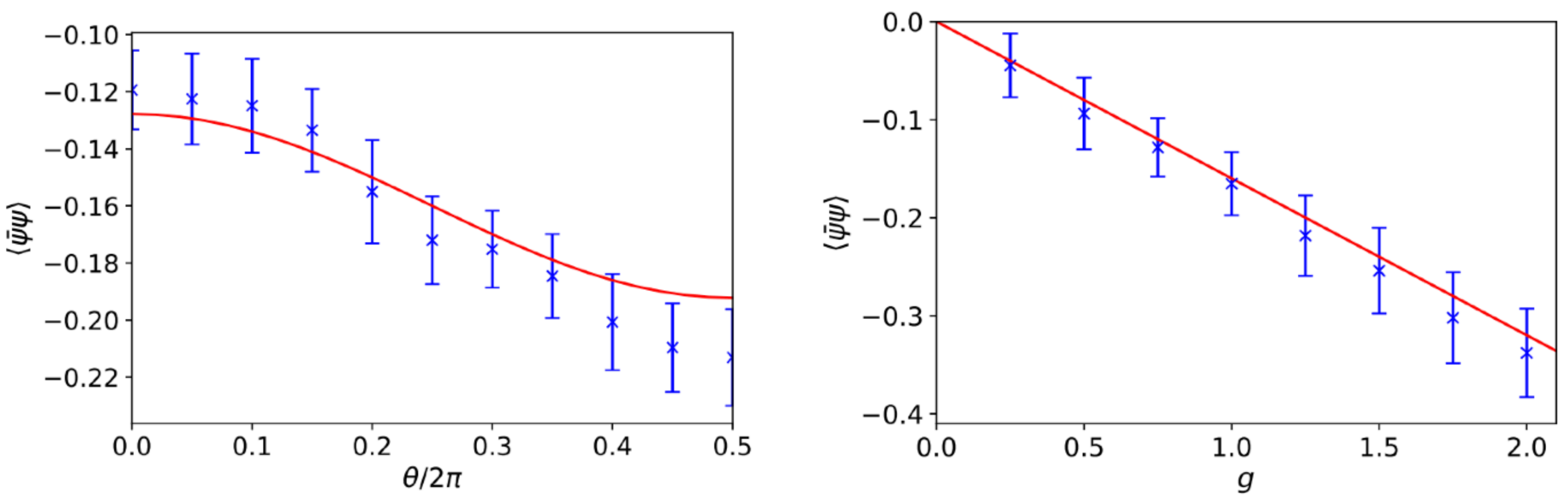
$$\langle O \rangle = \langle \Omega | O | \Omega \rangle$$

For the actual ground state $H|\Omega\rangle = E_0|\Omega\rangle$

The exact ground state can be prepared using adiabatic state preparation = long unitary evolution with gradually changing Hamiltonian

$$\mathrm{e}^{-\mathrm{i}Ht} \approx (\mathrm{e}^{-\mathrm{i}H_{\mathrm{kin}}t/N}\mathrm{e}^{-\mathrm{i}H_{\mathrm{mass}}t/N}\dots)^N$$





B Chakraborty, M Honda, T Izubuchi, Y Kikuchi, **AT**

Phys.Rev.D 105 (2022) 9, 094503

BUT, Near term quantum devices are only capable to deal with simple (short) circuit!

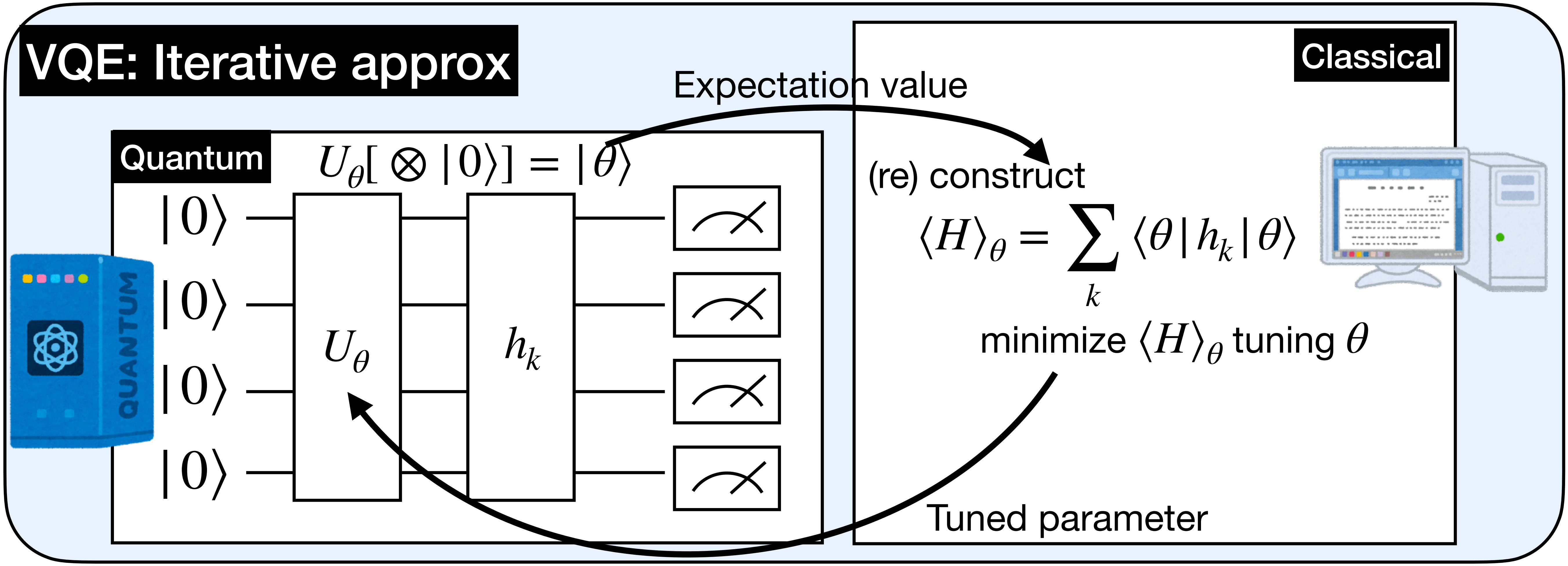
Variational approaches help to evaluate the ground state to evaluate the expectation value = Variational Quantum Eigen-solver (VQE), a quantum-classical hybrid algorithm

VQE and Beta VQE 1/2

Akio Tomiya

Background: VQE is a variational method

- Quantum machine: Exact ground state $|\Omega\rangle$ preparation is hard. In particular, it is difficult on near term devices
- Variational method for a *pure state* with a short circuit (VQE, variation quantum eigen-solver).
  - Quantum/Classical hybrid algorithm, iterative. U_θ is a short circuit.
  - Parametrized unitary circuit (~parametrized state $|\theta\rangle$, θ: a set of parameters)



- Systematic error since $|\theta\rangle = U_\theta[\otimes|0\rangle] \neq |\Omega\rangle$ but cheap

QFT with Hamiltonian

Hamiltonian vs Lagrangian

Akio Tomiya

Operator formalism (This work)

H : Hamiltonian in QFT

Real time

Finite temperature/imaginary time

Minkowski in M^{d+1}

Euclid in S^1 x M^d

$U(t) = \mathrm{e}^{-itH}$

Euclid(t → τ)

$t = -i\tau$

Minkowski(τ → t)

$U(\tau) = \mathrm{e}^{-\tau H}$

$\langle OO(\tau) \rangle = \mathrm{Tr}[O(0)O(\tau)\rho]$

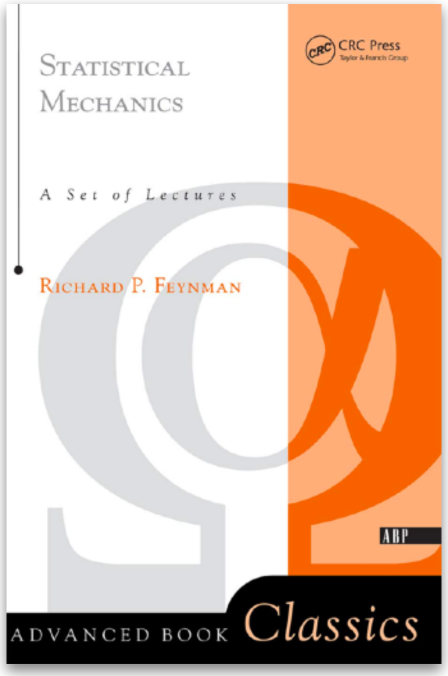
$\rho = U(\tau)/Z$

- Thermal state in quantum system?

  -> Density matrix formalism

Density matrix

unifies description of pure states and mixed states



Pure states: System is purely quantum

$$\rho_{\text{pure}} = |\Psi\rangle\langle\Psi| \qquad \langle O \rangle = \text{Tr}[O\rho_{\text{pure}}] = \langle \Psi | O | \Psi \rangle$$

Mixed states: States are classically mixed ($\neq$ superposition)

$$\rho_{\text{mixed}} = \sum_i w_i |\psi_i\rangle\langle\psi_i| \qquad \langle O \rangle = \text{Tr}[O\rho_{\text{mixed}}] = \sum_i w_i \langle \psi_i | O | \psi_i \rangle$$

$w_i \in \mathbb{R}_+$ represents probability to find a pure state $|\psi_i\rangle$

Thermal states (Grand-canonical):

$$\rho_{T,\mu} = \frac{1}{Z} \mathrm{e}^{-\frac{1}{T}(\hat{H}-\mu\hat{N})} \qquad \langle O \rangle_{T,\mu} = \text{Tr}[O\rho_{T,\mu}]$$

(Alternative approach TPQ: AT Yuki Nagai APLAT, 2020)

Thermal-quantum average in general

$$\langle O \rangle = \text{Tr}[O\rho]$$

Density matrix

Akio Tomiya

Quantum version of probability distribution

Thermal-quantum average in general

$$\langle O \rangle = \mathrm{Tr}[O\rho]$$

General Properties of density matrix ρ

- It unifies discretions of pure states and mixed states
- Normalized as $\mathrm{Tr}[\rho] = 1$
- ρ can be regarded as quantum version of probability distribution p(x)
  - e.g.) $S = -\int dx\, p(x) \log p(x)$ (Shannon entropy)

    <—> $S = -\mathrm{Tr}[\rho \log \rho]$ (Von-Neumann entropy)
- Distance between two density matrices = quantum relative entropy (next)

VQE and Beta VQE 2/2

Akio Tomiya

Beta VQE is a variational method for mixed states

- KL divergence for ρ = Kullback–Leibler *Umegaki* divergence (Pseudo-distance for ρ)
- Classical ver: $D(p\,|\,q) = \int dx\; p(x) \log p(x)/q(x)$ (KL divergence)
  - Relative entropy. Difference of two distributions (~distance)
  - Positive definite, Used in machine learning
  - D=0 if and only if p, q are equal
- **Quantum** $D(\rho_1\,|\,\rho_2) = \mathrm{Tr}[\rho_1 \log \rho_1 \big/ \rho_2]$ (KL-Umegaki divergence ~ distance)
  - Positive definite
  - D=0 if and only if ρ_1, ρ_2 are equal
- Kullback–Leibler *Umegaki* divergence can be used for variational approaches

Ansatz for ρ?

VQE and Beta VQE

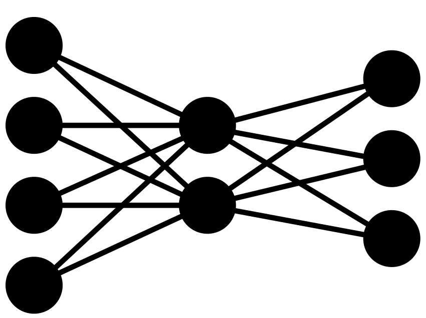
Akio Tomiya

Beta VQE is a variational method for mixed states

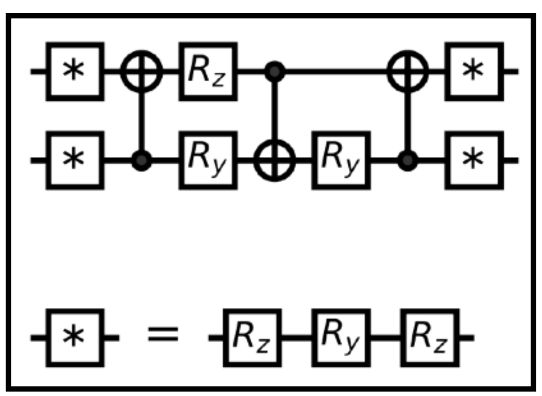
J. -Guo Liu+ 1902.02663

*M. Germain+ 1502.03509

- Variational ansatz for thermal quantum system:
  - $$\rho_\Theta = \sum_{\{\vec{x}\}} p_\phi[\vec{x}]\, U_\theta |\vec{x}\rangle\langle\vec{x}| U_\theta^\dagger, \quad \Theta = \theta \cup \phi \text{ (parameters)}$$



Neural network (Thermal)



Variational quantum circuit (Entanglement)

- $\vec{x} = (x_1, x_2, x_3, \cdots, x_k, \cdots)^\top$, and $x_k \in \{0,1\}$: (roughly) fermion occupation
- $|\vec{x}\rangle = |x_1\rangle \otimes |x_2\rangle \otimes |x_3\rangle \otimes \cdots$:

Product state (Easy to prepare)

Beta VQE

Extended VQE for mixed states

Jin-Guo Liu+ 1902.02663

- We minimize $\mathscr{L}(\Theta) = D(\rho_\Theta | \rho_{T,\mu}) - \ln Z_{T,\mu} = \mathrm{Tr}[\rho_\Theta \ln \rho_\Theta] + \frac{1}{T}\mathrm{Tr}[\rho_\Theta(\hat{H} - \mu\hat{N})]$
  - Variational bound: $\mathscr{L}(\Theta) \geq -\log Z_{T,\mu}$
- *Advantage* of beta VQE
  - No sign problem, even with the chemical potential
  - Bounded variational approximation
- *Disadvantage*
  - Systematic error
  - Need numerical resource if we use a classical machine

Simulation results

Akio Tomiya

Simulation setup (mostly skip)

- We apply beta-VQE for Schwinger model (= QED in 1+1d). Toy model of QCD, confinement, chiral symmetry breaking

$$S = \int d^2x \left[-\frac{1}{4} F_{\mu\nu}F^{\mu\nu} + \bar{\psi}\left(i\partial\!\!\!/ - gA\!\!\!/ - m\right)\psi \right] \Longleftrightarrow H = \int dx \left[-i\bar{\psi}\gamma^1(\partial_1 + igA_1)\psi + m\bar{\psi}\psi + \frac{1}{2}\Pi^2 \right]$$

$$\partial_x E = g\bar{\psi}\gamma^0\psi$$

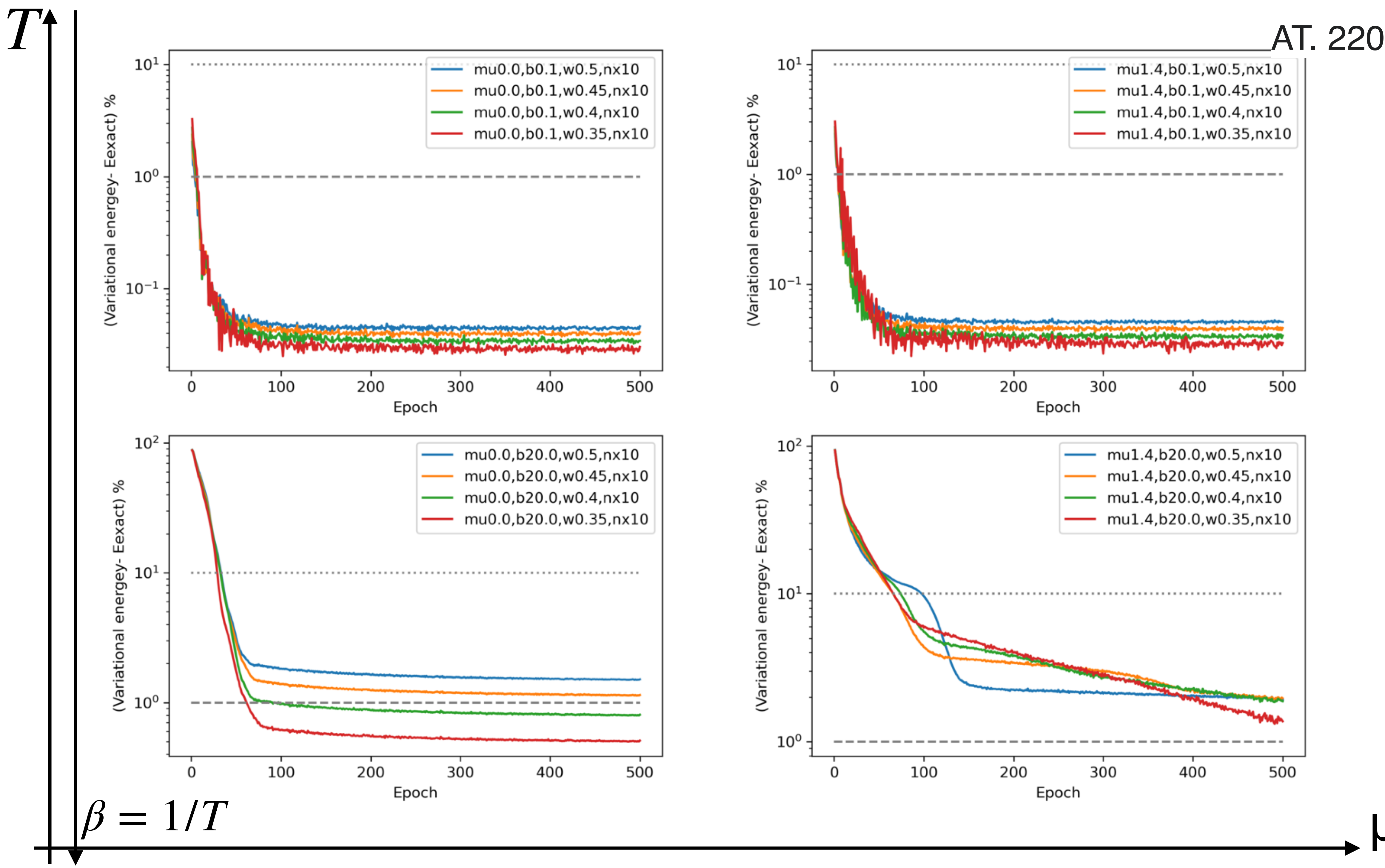
- Staggered fermion
- Jordan-Wigner transformation. Open Boundary condition.
- g = 1, Nx = (4, 6), 8, 10, 1/T = [0.5-20.0], mu= [0-1.4], 4 lattice spacings 1/2a = [0.5-0.35]
- We do not take large volume limit but take continuum limit
  - (Practically, Nx>10 cannot be calculated on our numerical resources)
  - (My previous work shows data from Nx>12 are essential to take stable large volume limit though)
- Setup for beta VQE:
  - Unitary part = SU(4) ansatz
  - Classical weight = Masked Auto-Encoder for Distribution Estimation (MADE)
- Training epoch is 500. Sampling = 5000 for classical distribution
- Observables
  - Variational free energy (exact and variational one)
  - (Translationally invariant) Chiral condensate
- **Check point: Dependence of variational error on <u>temperature and mu</u>**

Simulation results

Akio Tomiya

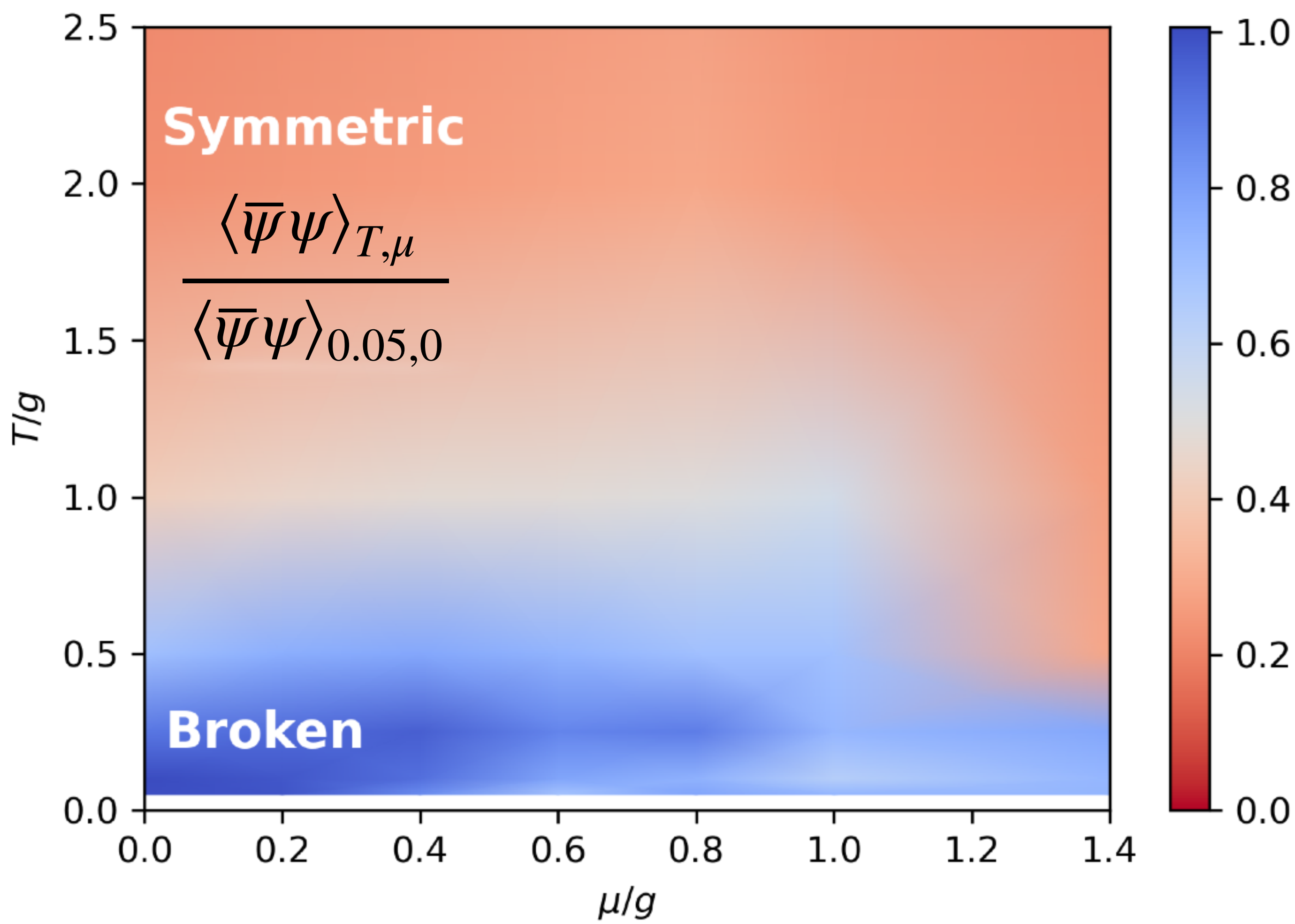
Variational free energy is O(1), Nx=10

AT. 2205.08860



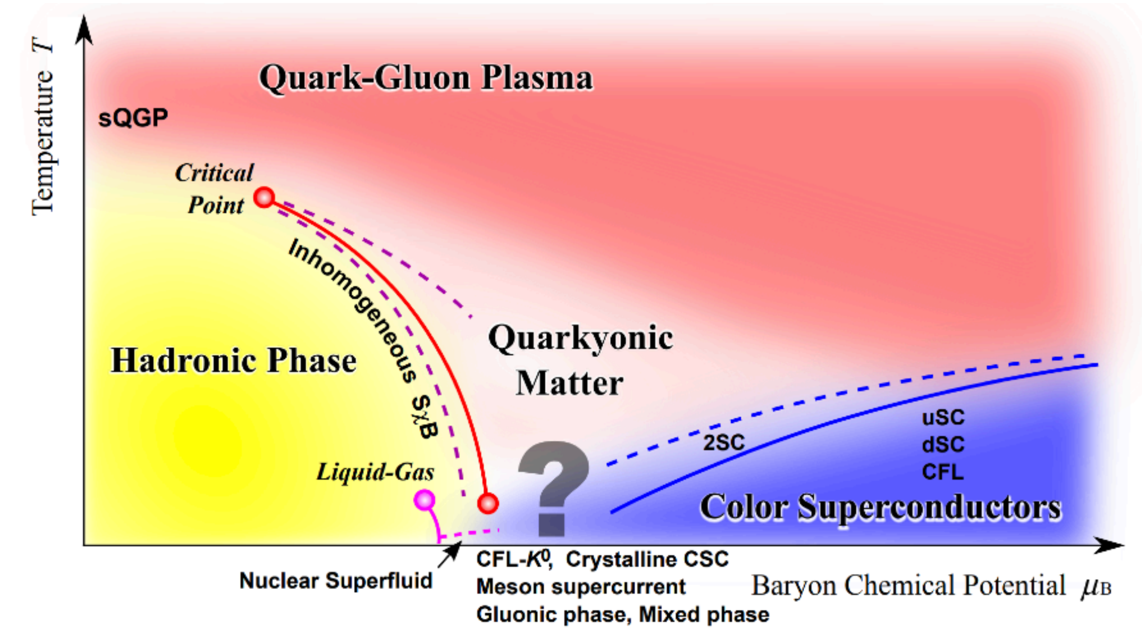
1. Mild dependence on μ (not fatal)
2. Hard for T -> 0 (large deviation) as expected

Summary for T-mu



AT. 2205.08860

Fukushima , Hatsuda
Rept.Prog.Phys.74:014001,2011



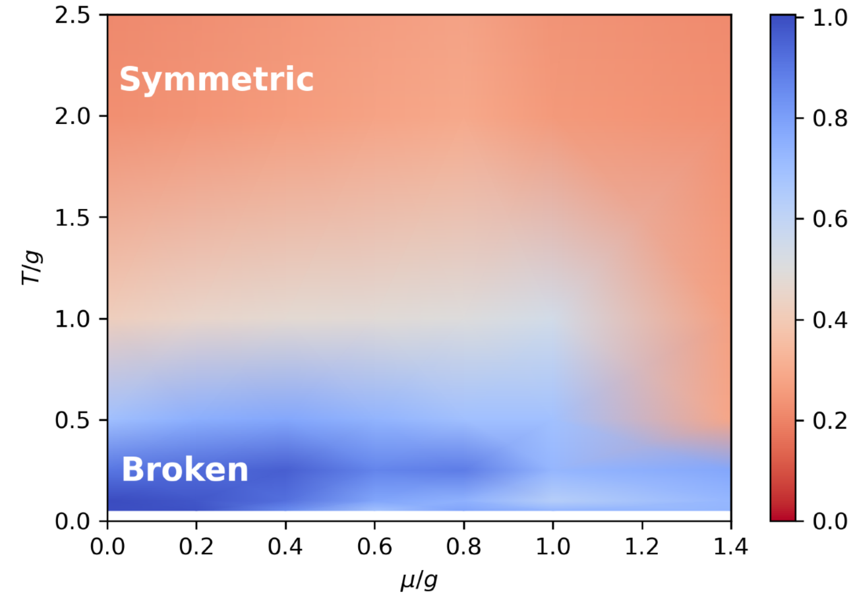
- We investigate T-μ phase diagram for Schwinger model
- Continuum extrapolation has been evaluated
  (except for additive mass renormalization by 2206.05308)
- The variational approach does not show difficulty for our parameter regime
- Towards to go large volume, optimization of code, GPU version, tensor network. (noise-free) real device!

Summary

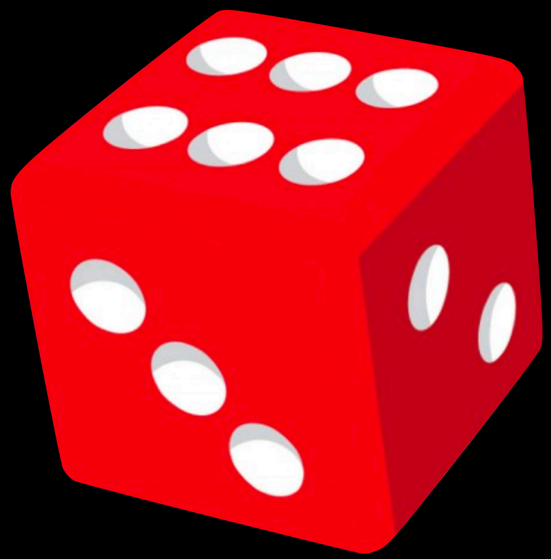
MLPhYs Foundation of "Machine Learning Physics"
Grant-in-Aid for Transformative Research Areas (A)

1. What and why QCD/lattice QCD?
    1. Problem: Long auto-correlation, Sign problem
2. Lattice QCD + Machine learning
    1. Trainable smearing + SLHMC = adaptive reweighting
3. Lattice QCD + Quantum algorithm
    1. Sign problem + non-unitary -> classical/quantum hybrid!

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



Congratulations again, Onogi-san!



Self-introduction
Lattice QCD & Machine learning

What/who am I?

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

Schwinger model at finite temperature and density with beta VQE
A Tomiya
arXiv preprint arXiv:2205.08860 — Phase diagram via Quantum/Classical algorithm

Biography

2010 - 2015 : Osaka university (Master& PhD)
2015 - 2018 : Postdoc in CCNU (Wuhan, China)
2018 - 2021 : SPDR in RIKEN/BNL (Brookhaven, US)
2021 - : Faculty in IPUT Osaka

KAKENHI (Grants-in-Aid for Scientific Research)

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

Details (skip)

Akio Tomiya

Network: trainable stout (plaq+poly)

arXiv: 2103.11965

Structure of NN

(Polyakov loop+plaq in the stout-type)

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

All ρ is weight

O meas an loop operator

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

TA: Traceless, anti-hermitian operation

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

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

2- layered stout

with 6 trainable parameters

Neural network Parametrized action:

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

Action for MD is built by gauge covariant NN

Loss function:

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

Invariant under, rot, transl, gauge trf.

Training strategy:

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

Details (skip)

Results: Loss decreases along with the training

arXiv: 2103.11965

Loss function:

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

Intuitively, e^(-L) is understood as Boltzmann weight or reweighting factor.

Prior HMC run (training)

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

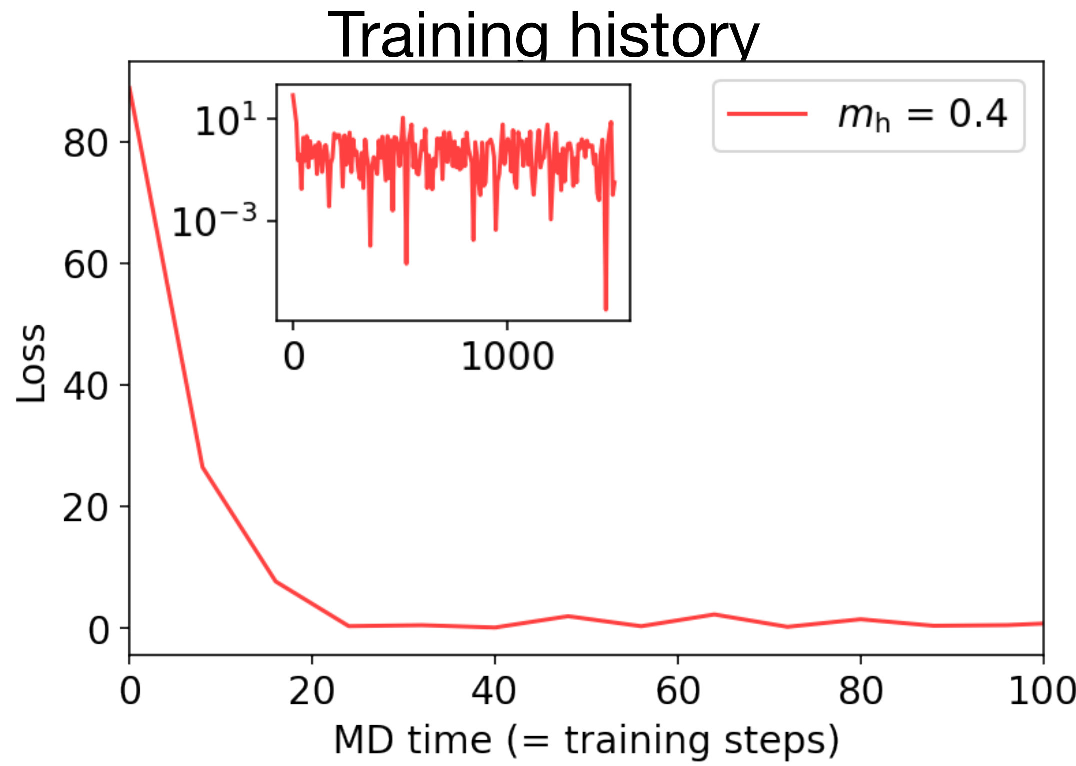
$$\theta \leftarrow \theta - \eta \frac{\partial L_\theta(\mathcal{D})}{\partial \theta},$$

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

Ω: sum of un-traced loops

C: one U removed Ω

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



Without training, e^(-L)<< 1,
this means that candidate with approximated action never accept.
After training, e^(-L) ~1, and we get practical acceptance rate!

We perform SLHMC with these values!

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

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

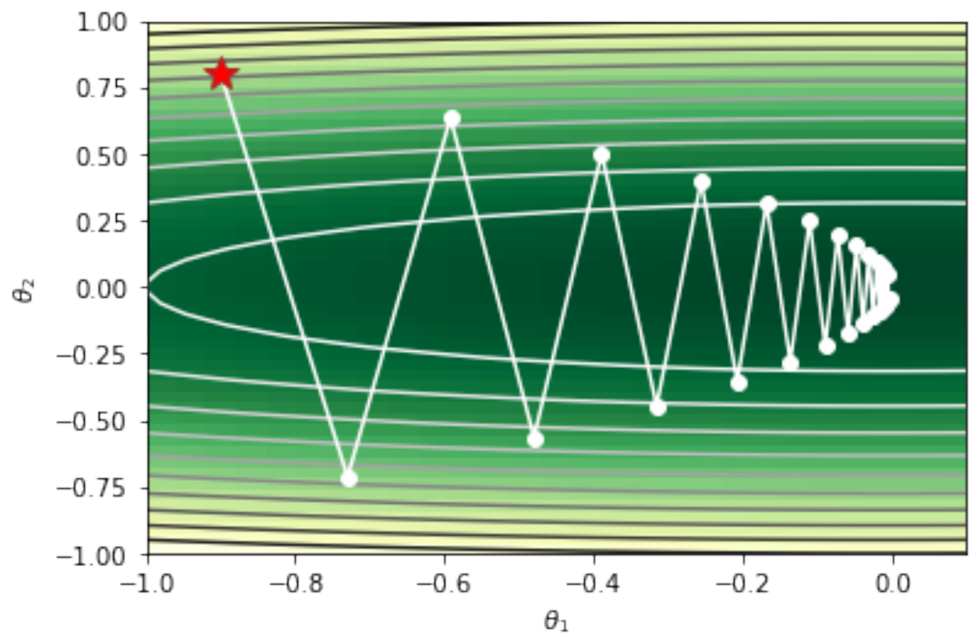
Training: We can use "gradient descent" (also "Adam" (adaptive-momentum) is applicable)

Repeat update (until converge)

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

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

Example of Gradient descent



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

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

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

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

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

Repeat update (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 developed extended delta rule:

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

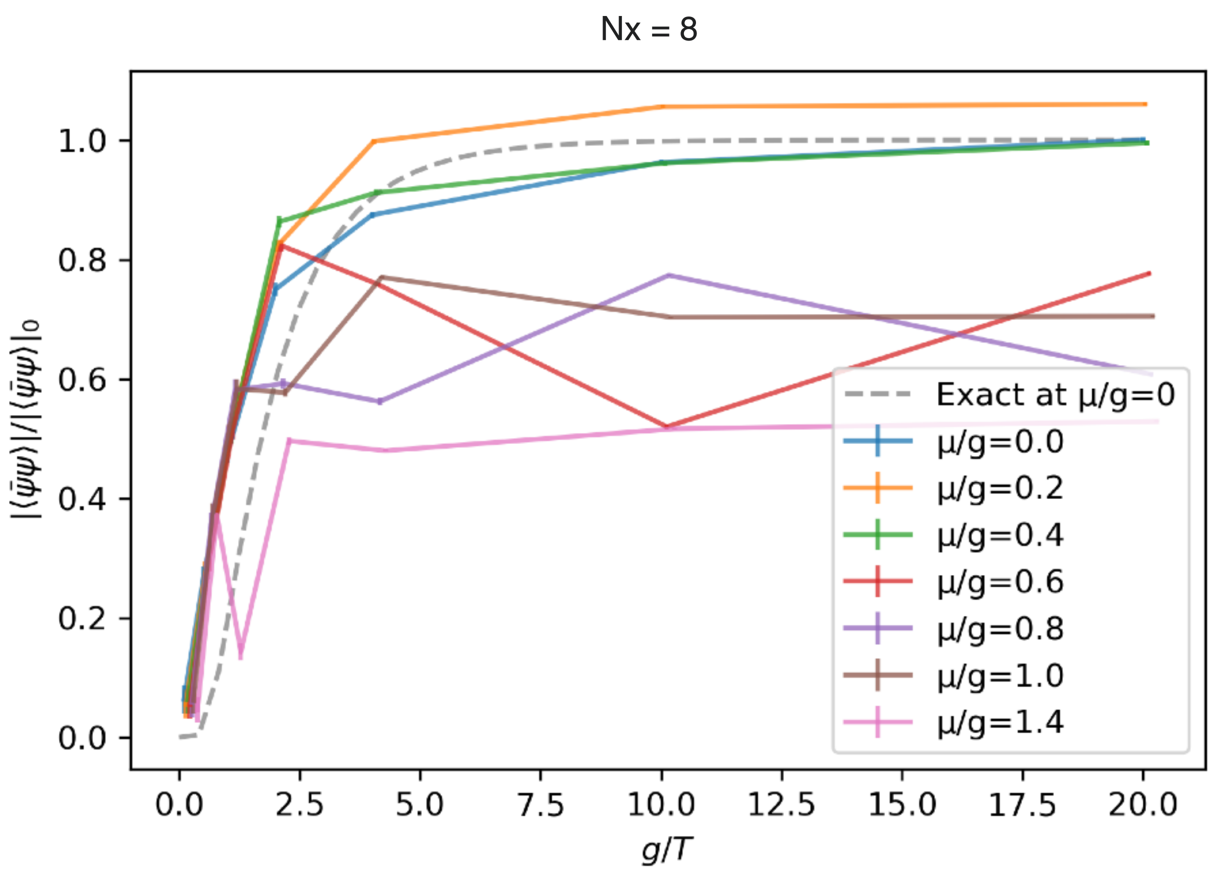
This matrix derivative is common to **the stout force** (namely well known)

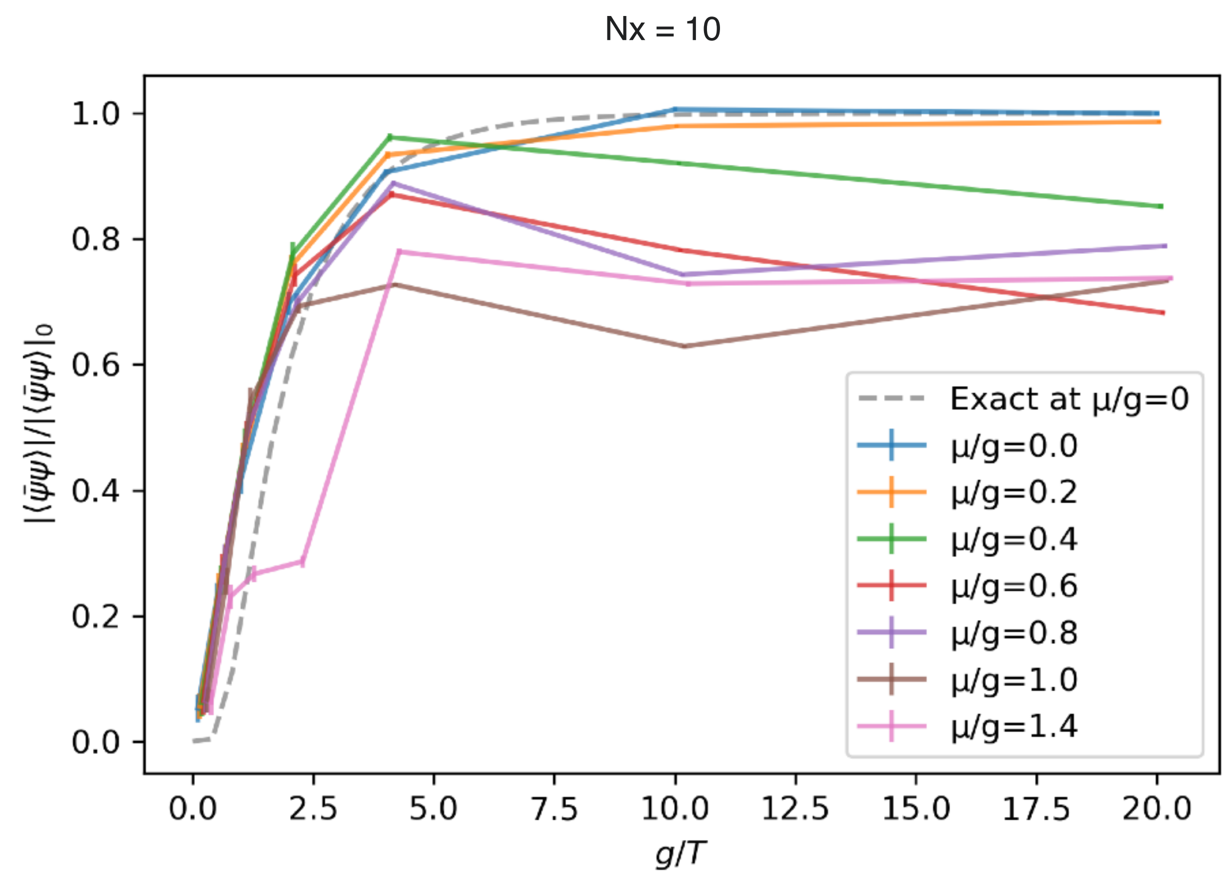
Simulation results

Continuum extrapolation for Nx = 8, 10

AT. 2205.08860

Continuum limit with a polynomial ansatz
it looks good So far*





*(I did not include additive mass shift (Ross Dempsey+ arXiv: 2206.05308).
I thank to Takis Angelides (DESY) and Etsuko Itou (RIKEN) for letting me know this important reference!)

We use Nx = 10 results for the phase diagram

VQE and Beta VQE 2/2

Akio Tomiya

Beta VQE is a variational method for mixed states

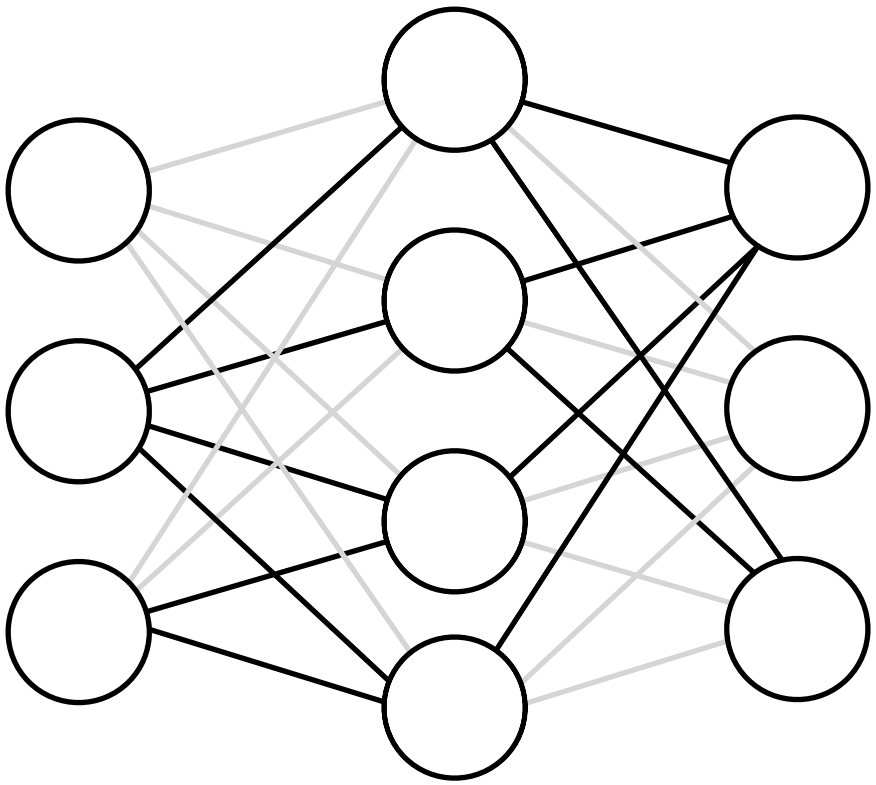
J. -Guo Liu+ 1902.02663
*M. Germain+ 1502.03509

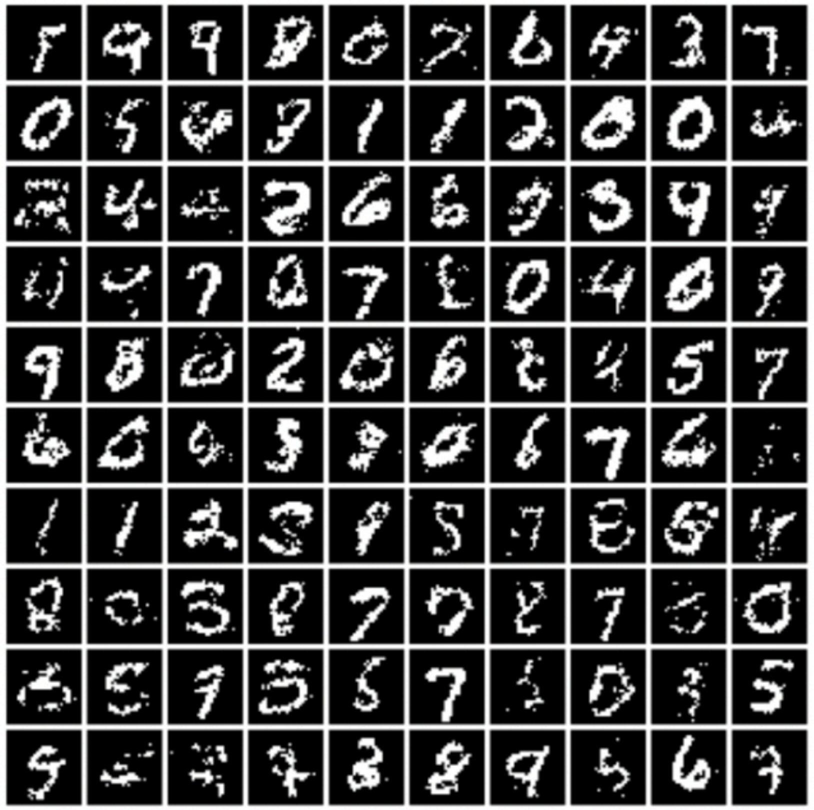
- Variational method for mixed states: Variational method on ρ
  - $\rho_\Theta = \sum_{\{\vec{x}\}} p_\phi[\vec{x}]\, U_\theta |\vec{x}\rangle\langle\vec{x}| U_\theta^\dagger$, $\Theta = \theta \cup \phi$ (parameters)
  - $\vec{x} = (x_1, x_2, x_3, \cdots, x_k, \cdots)^{\mathsf{T}}$, and $x_k \in \{0,1\}$: (roughly) fermion occupation
  - $|\vec{x}\rangle = |x_1\rangle \otimes |x_2\rangle \otimes |x_3\rangle \otimes \cdots$: easy to prepare
  - $U_\theta|\vec{x}\rangle$: parametrized pure states, similar to the conventional VQE
- $p_\phi[\vec{x}]$: Classically approximated distribution for a configuration of $\vec{x}$,
  Neural network (MADE*) is used. ϕ = parameters
  This can generate configurations of $\vec{x}$

MADE?

(masked) Auto-encoder for binary variable distribution

- MADE (neural network) mimics joint probability distribution e.g. $p(x_1, x_2, x_3)$, whose input is binary array $(x_1, x_2, x_3), x_i = 0,\ 1$





Reconstructed MNIST (Binarized)

Auto-encoder with a mask -> Generative model for binary array
(Please ask me later in detail)

*M. Germain+ 1502.03509

Beta VQE

Extended VQE for mixed states

Jin-Guo Liu+ 1902.02663

NN VQE

- We approximate $\rho = \frac{1}{Z}\mathrm{e}^{-\frac{1}{T}(\hat{H}-\mu\hat{N})}$ by $\rho_\Theta = \sum_{\{\vec{x}\}} p_\phi[\vec{x}]\, U_\theta|\vec{x}\rangle\langle\vec{x}|U_\theta^\dagger$
  - $\langle O\rangle_{T,\mu} \approx \mathrm{Tr}[\rho_\Theta O] = \sum_{\{\vec{x}\}} p_\phi[\vec{x}]\, \langle\vec{x}|U_\theta^\dagger O U_\theta|\vec{x}\rangle$
  - Quantum machine can store a state $U_\theta|\vec{x}\rangle$ (test wave function)
  - Classical machine can sample thermal distribution from $p_\phi[\vec{x}]$ (neural net)
- All parameters are tuned such that minimizing $D(\rho_\Theta|\rho)$
  - Optimization of parameters is done with a optimizer (as in machine learning)

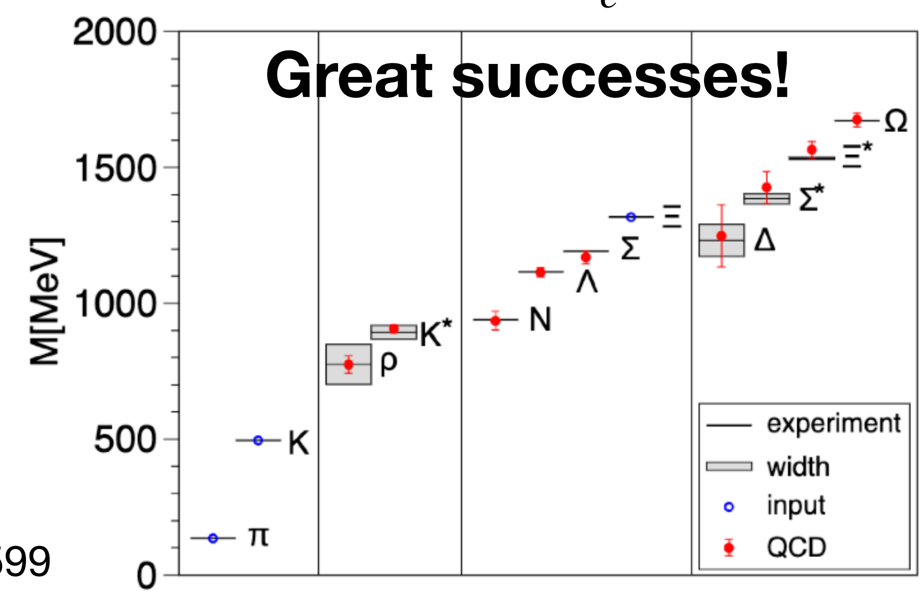
Akio Tomiya

Motivation

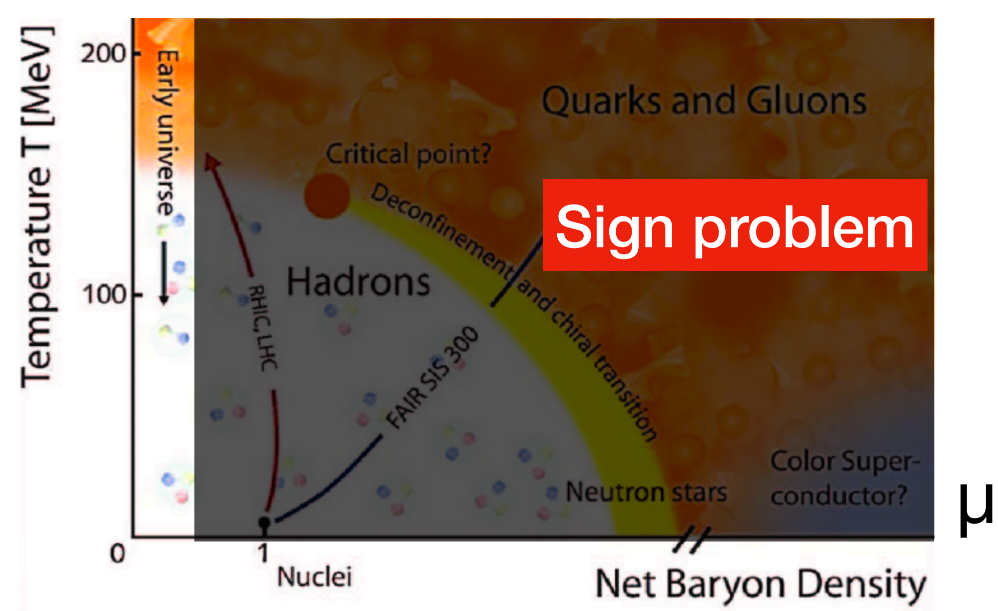
Sign problem prevents using Monte-Carlo

- Monte-Carlo enables us to evaluate expectation values for "statistical system", like lattice QCD in imaginary time

$$\langle O[U] \rangle = \frac{1}{N_{\mathrm{conf}}} \sum_{c}^{N_{\mathrm{conf}}} O[U_c] + \mathcal{O}\left(\frac{1}{\sqrt{N_{\mathrm{conf}}}} \right) \qquad U_c \leftarrow P(U) = \frac{1}{Z} e^{-S[U]} \in \mathbb{R}_+$$



arXiv:0906.3599



- If we turn on the baryon chemical potential μ, Monte-Carlo methods do not work because $e^{-S[U]}$ becomes complex. This is no more probability. (sign problem)
- Operator formalism does not have such problem! But it requires huge memory to store quantum states, which cannot be realized even on supercomputer.
- Quantum states should be stored on quantum device (Feynman)

Motivation

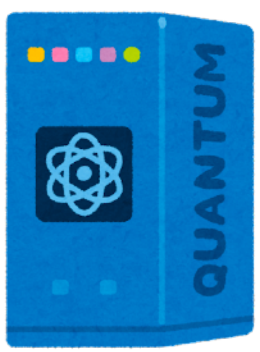
μ = 0 is good for Classical, T=0 is good for Quantum

Classical machine: Lattice field theory calculations rely on

$$P(U) = \frac{1}{Z} e^{-S[U]} \det(D[U] + m)^2 \in \mathbb{R}_+$$

Since 1980 (M. Creutz)~

- This P(U) cannot be regarded as probability if $\mu \neq 0$ (sign problem)



Quantum machines can realize (any) unitary evolutions (Solovay Kitaev theorem),

$$U(t) = \mathrm{e}^{-\mathrm{i}\hat{H}t}$$

Phys.Rev.D 105 (2022) 9, 094503
and references therein

- No problem for μ≠0 because we can only use unitary gates (operators)
- "Short time evolution" (shallow circuit) is preferred for near-term devices

	Classical Computers	Quantum Computers
Finite Density	Sign Problem	√
Finite Temperature	√	Challenging *

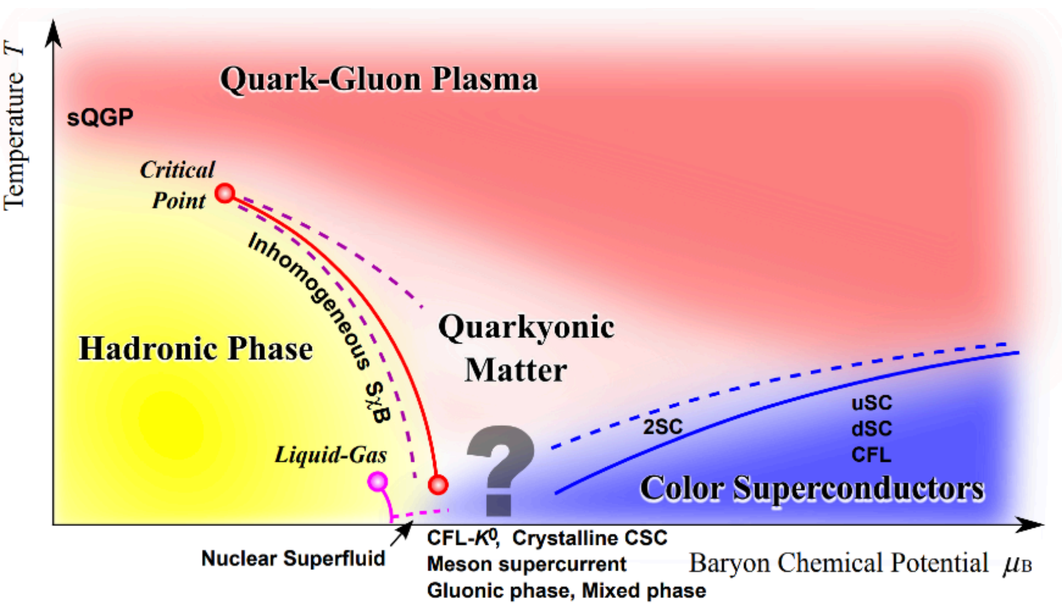
We need a method to calculate T>0 and μ≠0 for QCD and for near-term quantum devices

*https://indico.hiskp.uni-bonn.de/event/40/contributions/484/attachments/358/630/Powers%20Talk%20Final%20Draft%20Updated.pdf

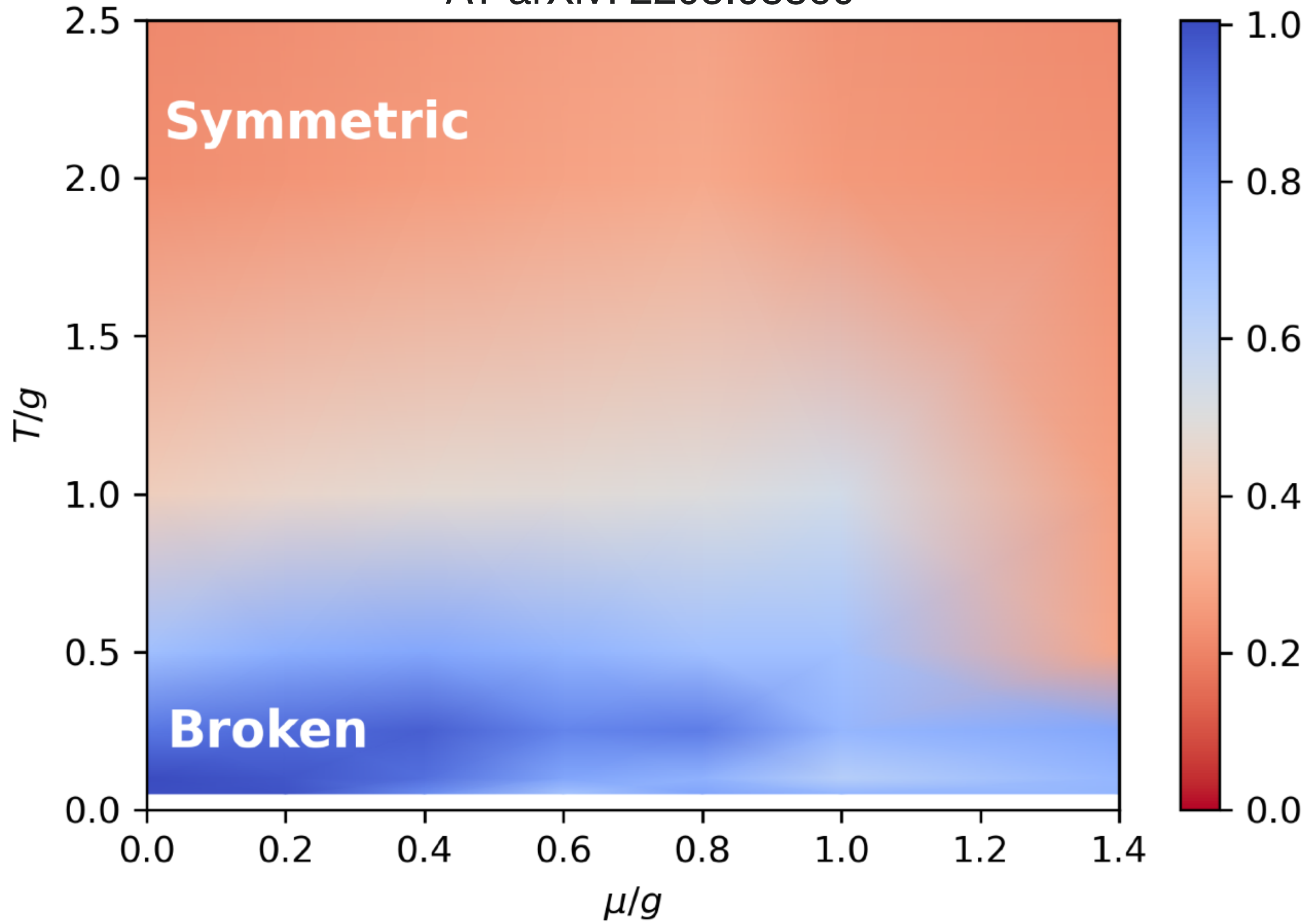
Summary of this talk

Hybrid = Quantum algorithm + machine learning

Fukushima , Hatsuda
Rept.Prog.Phys.74:014001,2011



AT arXiv: 2205.08860



I investigated T-mu phase diagram using a <u>quantum algorithm</u> & <u>neural network</u> (β-VQE, No sign problem) for Schwinger model (toy model of QCD)

QFT with Hamiltonian

Hamiltonian vs Lagrangian

Operator formalism (This work)

H : Hamiltonian in QFT

Real time

Finite temperature/imaginary time

Minkowski in M^{d+1}

Euclid in S^1 x M^d

$$U(t) = \mathrm{e}^{-\mathrm{i}tH}$$

Euclid(t → τ)

$t = -\mathrm{i}\tau$

Minkowski(τ → t)

$$U(\tau) = \mathrm{e}^{-\tau H}$$

$$\langle OO(\tau) \rangle = \mathrm{Tr}[O(0)O(\tau)\rho]$$

$$\rho = U(\tau)/Z$$

- Typical use case of quantum algorithm is for real-time. Unitary.
  - Time evolution: Correlators (e.g. 2pt on light-cone), etc
  - Main interest: $\langle \Omega | O | \Omega \rangle$, where $| \Omega \rangle$ is the exact ground state
- Difficulty: State preparation for exact ground state of H

State preparation, VQE and Beta-VQE

State preparation is hard

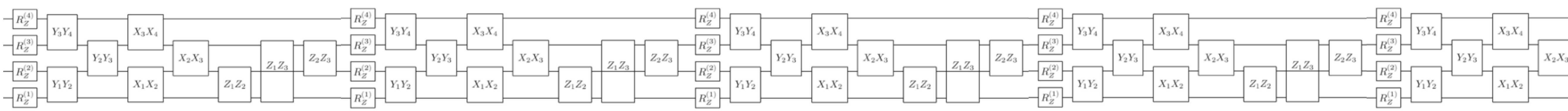
We are interested in expectation value with true ground state for Hamiltonian

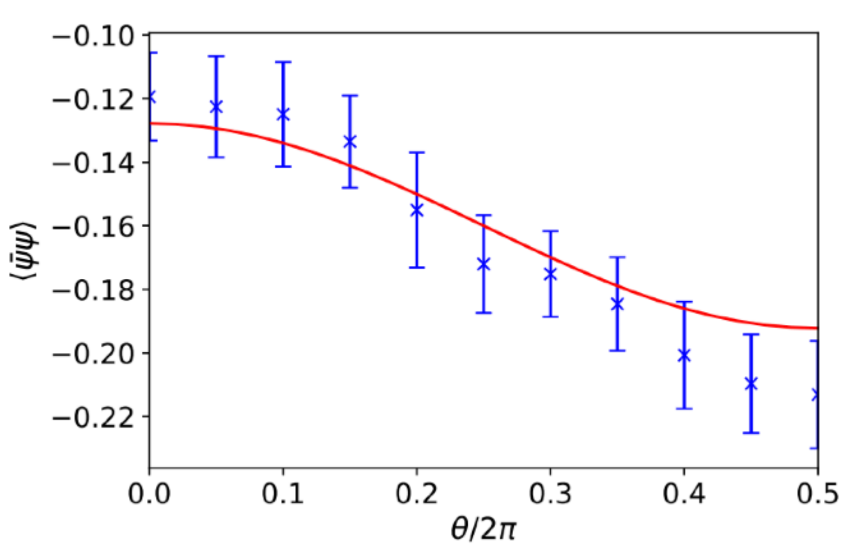
$$\langle O \rangle = \langle \Omega | O | \Omega \rangle$$

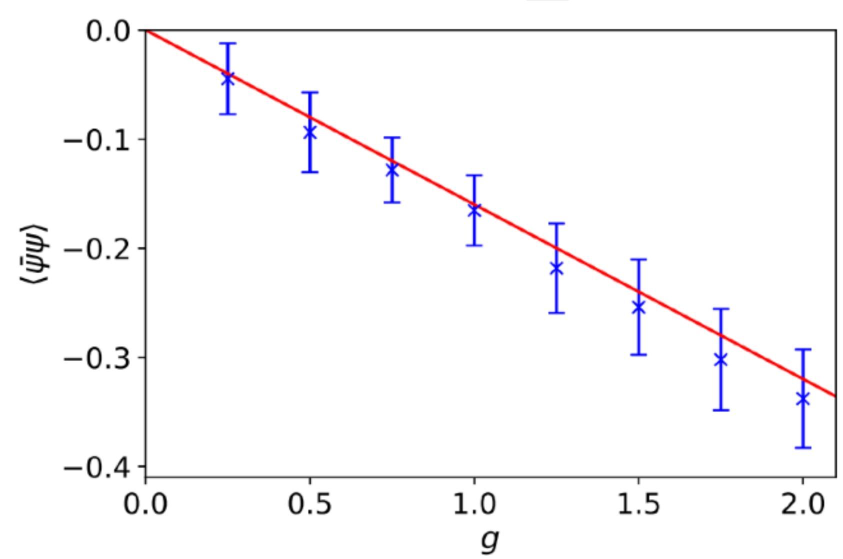
For the actual ground state $H|\Omega\rangle = E_0|\Omega\rangle$

The exact ground state can be prepared using adiabatic state preparation = long unitary evolution with gradually changing Hamiltonian

$$e^{-iHt} \approx (e^{-iH_{\rm kin}t/N} e^{-iH_{\rm mass}t/N} \dots)^N$$







B Chakraborty, M Honda, T Izubuchi, Y Kikuchi, **AT**

Phys.Rev.D 105 (2022) 9, 094503

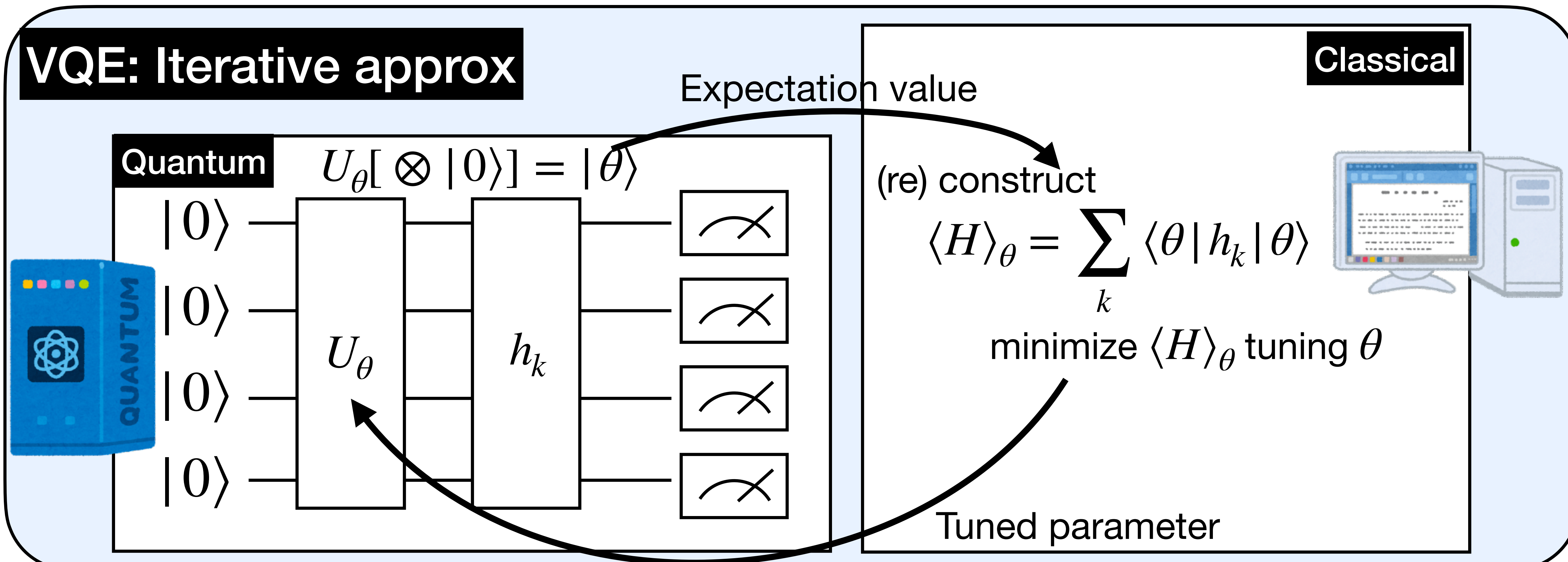
BUT, Near term quantum devices are only capable to deal with simple (short) circuit!

Variational approaches help to evaluate the ground state to evaluate the expectation value = Variational Quantum Eigen-solver (VQE), a quantum-classical hybrid algorithm

VQE and Beta VQE 1/2

Background: VQE is a variational method

- Quantum machine: Exact ground state $|\Omega\rangle$ preparation is hard. In particular, it is difficult on near term devices
- Variational method for a *pure state* with a short circuit (VQE, variation quantum eigen-solver).
  - Quantum/Classical hybrid algorithm, iterative. U_θ is a short circuit.
  - Parametrized unitary circuit (~parametrized state $|\theta\rangle$, θ: a set of parameters)



- Systematic error since $|\theta\rangle = U_\theta[\otimes|0\rangle] \neq |\Omega\rangle$ but cheap

QFT with Hamiltonian

Hamiltonian vs Lagrangian

Operator formalism (This work)

H : Hamiltonian in QFT

Real time

Finite temperature/imaginary time

Minkowski in M^{d+1}

Euclid in S^1 x M^d

$U(t) = e^{-itH}$

Euclid(t → τ)

$t = -i\tau$

Minkowski(τ → t)

$U(\tau) = e^{-\tau H}$

$\langle OO(\tau)\rangle = \mathrm{Tr}[O(0)O(\tau)\rho]$

$\rho = U(\tau)/Z$

- Thermal state in quantum system?

  -> Density matrix formalism

Density matrix

unifies description of pure states and mixed states

Pure states: System is purely quantum

$$\rho_{\rm pure} = |\Psi\rangle\langle\Psi| \qquad \langle O\rangle = {\rm Tr}[O\rho_{\rm pure}] = \langle\Psi|O|\Psi\rangle$$

Mixed states: States are classically mixed ($\neq$ superposition)

$$\rho_{\rm mixed} = \sum_i w_i|\psi_i\rangle\langle\psi_i| \qquad \langle O\rangle = {\rm Tr}[O\rho_{\rm mixed}] = \sum_i w_i\langle\psi_i|O|\psi_i\rangle$$

w_i represents probability to find a pure state $|\psi_i\rangle$

Thermal states (Grand-canonical):

$$\rho_{T,\mu} = \frac{1}{Z}{\rm e}^{-\frac{1}{T}(\hat{H}-\mu\hat{N})} \qquad \langle O\rangle_{T,\mu} = {\rm Tr}[O\rho_{T,\mu}]$$

(Alternative approach TPQ: AT Yuki Nagai APLAT, 2020)

Thermal-quantum average in general

$$\langle O\rangle = {\rm Tr}[O\rho]$$

Density matrix

Quantum version of probability distribution

Thermal-quantum average in general

$$\langle O \rangle = \mathrm{Tr}[O\rho]$$

General Properties of density matrix ρ

- It unifies discretions of pure states and mixed states
- Normalized as $\mathrm{Tr}[\rho] = 1$
- ρ can be regarded as quantum version of probability distribution p(x)
  - e.g.) $S = -\int dx\, p(x) \log p(x)$ (Shannon entropy)

    <—> $S = -\mathrm{Tr}[\rho \log \rho]$ (Von-Neumann entropy)
- Distance between two density matrices = quantum relative entropy (next)

VQE and Beta VQE 2/2

Akio Tomiya

Beta VQE is a variational method for mixed states

- KL divergence for ρ = Kullback–Leibler *Umegaki* divergence (Pseudo-distance for ρ)
- Classical ver: $D(p\,|\,q) = \int dx\; p(x) \log p(x)/q(x)$ (KL divergence)
  - Relative entropy. Difference of two distributions (~distance)
  - Positive definite, Used in machine learning
  - D=0 if and only if p, q are equal
- **Quantum** $D(\rho_1\,|\,\rho_2) = \mathrm{Tr}[\rho_1 \log \rho_1 \big/ \rho_2]$ (KL-Umegaki divergence ~ distance)
  - Positive definite
  - D=0 if and only if ρ_1, ρ_2 are equal
- Kullback–Leibler *Umegaki* divergence can be used for variational approaches

Ansatz for ρ?

VQE and Beta VQE 2/2

Akio Tomiya

Beta VQE is a variational method for mixed states

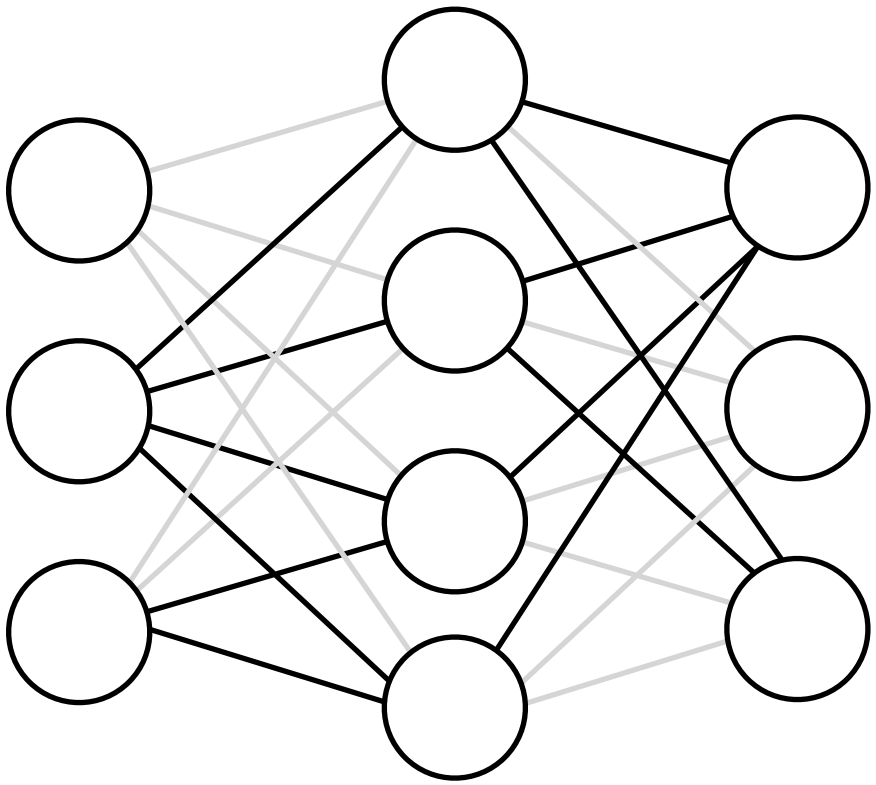
J. -Guo Liu+ 1902.02663
*M. Germain+ 1502.03509

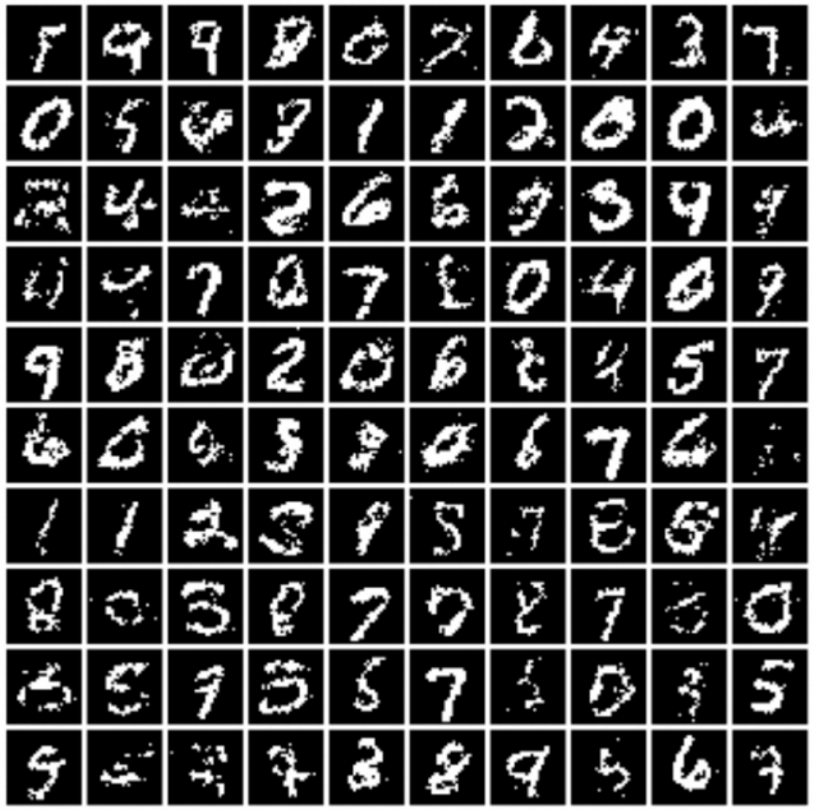
- Variational method for mixed states: Variational method on ρ
  - $\rho_\Theta = \sum_{\{\vec{x}\}} p_\phi[\vec{x}]\; U_\theta |\vec{x}\rangle\langle\vec{x}| U_\theta^\dagger, \quad \Theta = \theta \cup \phi$ (parameters)
  - $\vec{x} = (x_1, x_2, x_3, \cdots, x_k, \cdots)^\top$, and $x_k \in \{0,1\}$: (roughly) fermion occupation
  - $|\vec{x}\rangle = |x_1\rangle \otimes |x_2\rangle \otimes |x_3\rangle \otimes \cdots$: easy to prepare
  - $U_\theta |\vec{x}\rangle$: parametrized pure states, similar to the conventional VQE
- $p_\phi[\vec{x}]$: Classically approximated distribution for a configuration of $\vec{x}$,
  Neural network (MADE*) is used. ϕ = parameters
  This can generate configurations of $\vec{x}$

MADE?

(masked) Auto-encoder for binary variable distribution

- MADE (neural network) mimics joint probability distribution e.g. $p(x_1, x_2, x_3)$, whose input is binary array $(x_1, x_2, x_3), x_i = 0,\ 1$





Reconstructed MNIST (Binarized)

Auto-encoder with a mask -> Generative model for binary array
(Please ask me later in detail)

*M. Germain+ 1502.03509

Beta VQE

Akio Tomiya

Extended VQE for mixed states

Jin-Guo Liu+ 1902.02663

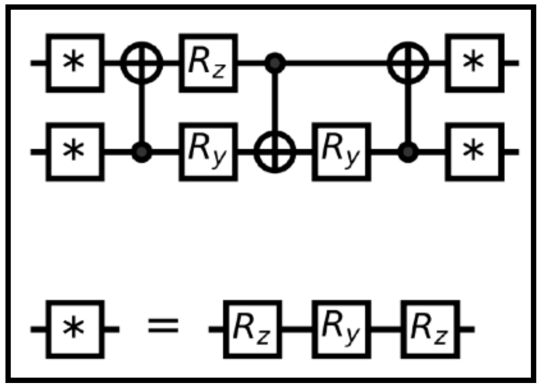
- We approximate $\rho = \frac{1}{Z} \mathrm{e}^{-\frac{1}{T}(\hat{H}-\mu\hat{N})}$ by $\rho_\Theta = \sum_{\{\vec{x}\}} p_\phi[\vec{x}]\, U_\theta | \vec{x} \rangle \langle \vec{x} | U_\theta^\dagger$
  - $\langle O \rangle_{T,\mu} \approx \mathrm{Tr}[\rho_\Theta O] = \sum_{\{\vec{x}\}} p_\phi[\vec{x}]\, \langle \vec{x} | U_\theta^\dagger O U_\theta | \vec{x} \rangle$
  - Quantum machine can store a state $U_\theta | \vec{x} \rangle$ (test wave function)
  - Classical machine can sample thermal distribution from $p_\phi[\vec{x}]$ (neural net)
- All parameters are tuned such that minimizing $D(\rho_\Theta | \rho)$
  - Optimization of parameters is done with a optimizer (as in machine learning)

Beta VQE 4/4

Extended VQE for mixed states

Jin-Guo Liu+ 1902.02663

- We minimize the loss function $\mathscr{L}(\Theta) = D - \ln Z = \mathrm{Tr}[\rho_\Theta \ln \rho_\Theta] + \frac{1}{T}\mathrm{Tr}[\rho_\Theta(\hat{H} - \mu\hat{N})]$
    - Variational bound: $\mathscr{L}(\Theta) - \log Z_{T,\mu} \geq 0$
    - We use SU(4) ansatz for each 2 qubits for U_θ



- *Advantage* of beta VQE
    - No sign problem, even with the chemical potential
    - Bounded variational approximation
- *Disadvantage*
    - Systematic error
    - Need numerical resource if we use a classical machine

Simulation results

Simulation setup (mostly skip)

- We apply beta-VQE for Schwinger model (= QED in 1+1d). Toy model of QCD, confinement, chiral symmetry breaking

$$S = \int d^2x \Big[-\frac{1}{4} F_{\mu\nu} F^{\mu\nu} + \bar{\psi}\big(i\partial\!\!\!/ - gA\!\!\!/ - m \big)\psi \Big] \quad \Longleftrightarrow \quad H = \int dx \left[-i\bar{\psi}\gamma^1(\partial_1 + igA_1)\psi + m\bar{\psi}\psi + \frac{1}{2}\Pi^2 \right]$$

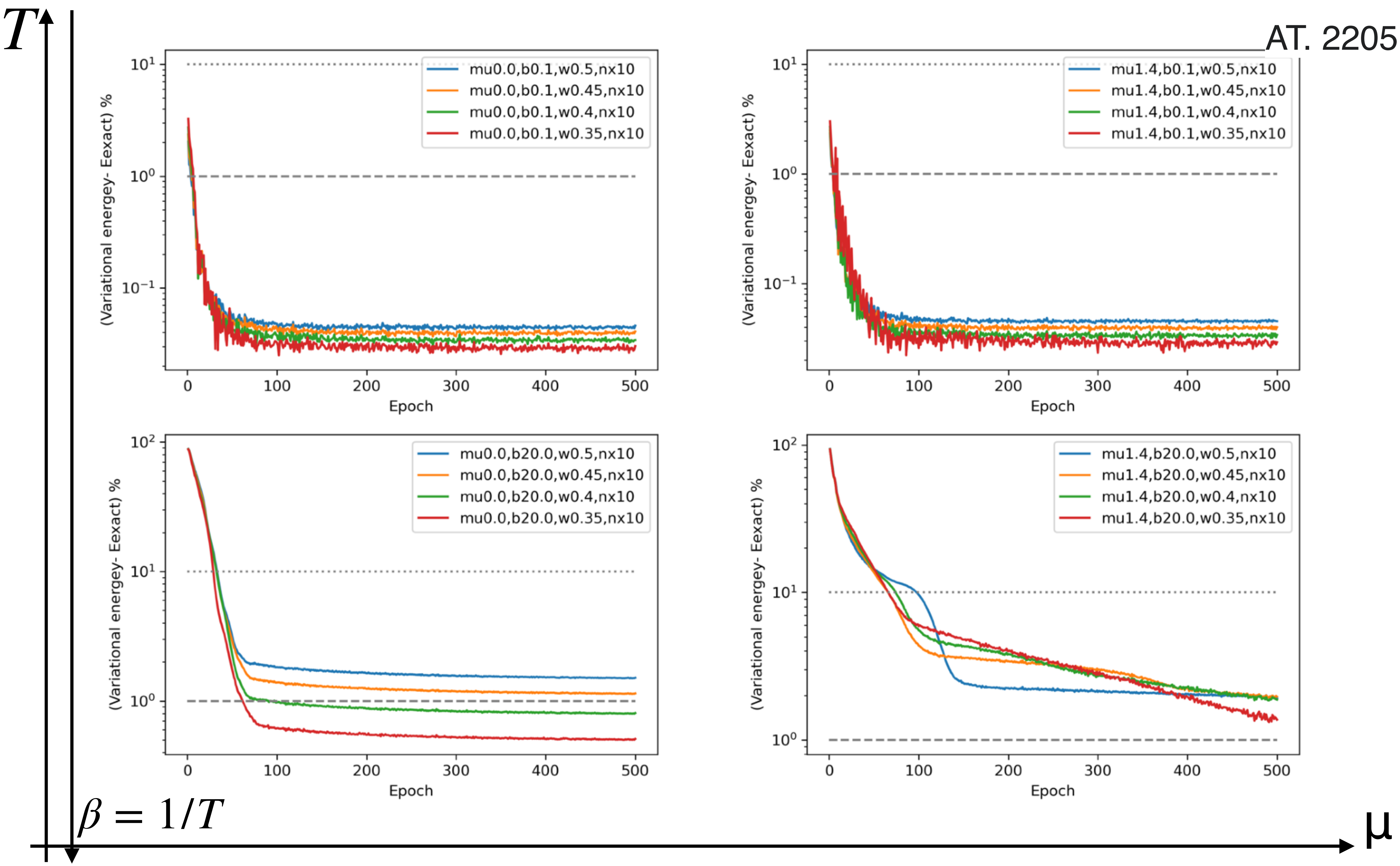
$$\partial_x E = g\bar{\psi}\gamma^0\psi$$

- Staggered fermion
- Jordan-Wigner transformation. Open Boundary condition.
- g = 1, Nx = (4, 6), 8, 10, 1/T = [0.5-20.0], mu= [0-1.4], 4 lattice spacings 1/2a = [0.5-0.35]
- We do not take large volume limit but take continuum limit
  - (Practically, Nx>10 cannot be calculated on our numerical resources)
  - (My previous work shows data from Nx>12 are essential to take stable large volume limit though)
- Setup for beta VQE:
  - Unitary part = SU(4) ansatz
  - Classical weight = Masked Auto-Encoder for Distribution Estimation (MADE)
- Training epoch is 500. Sampling = 5000 for classical distribution
- Observables
  - Variational free energy (exact and variational one)
  - (Translationally invariant) Chiral condensate
- **Check point: Dependence of variational error on temperature and mu**

Simulation results

Variational free energy is O(1), Nx=10

AT. 2205.08860



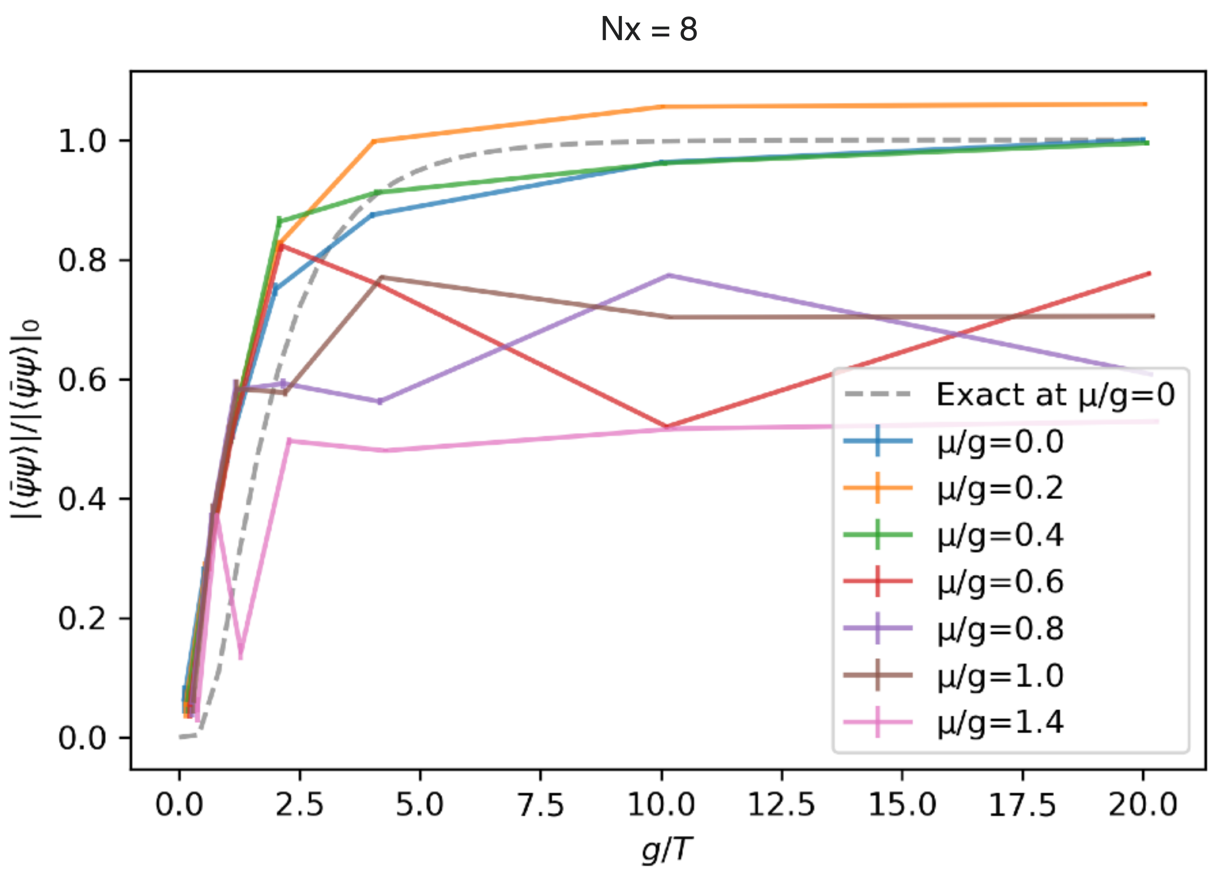
1. Mild dependence on μ (not breaking)
2. Hard for T -> 0 (large deviation) as expected

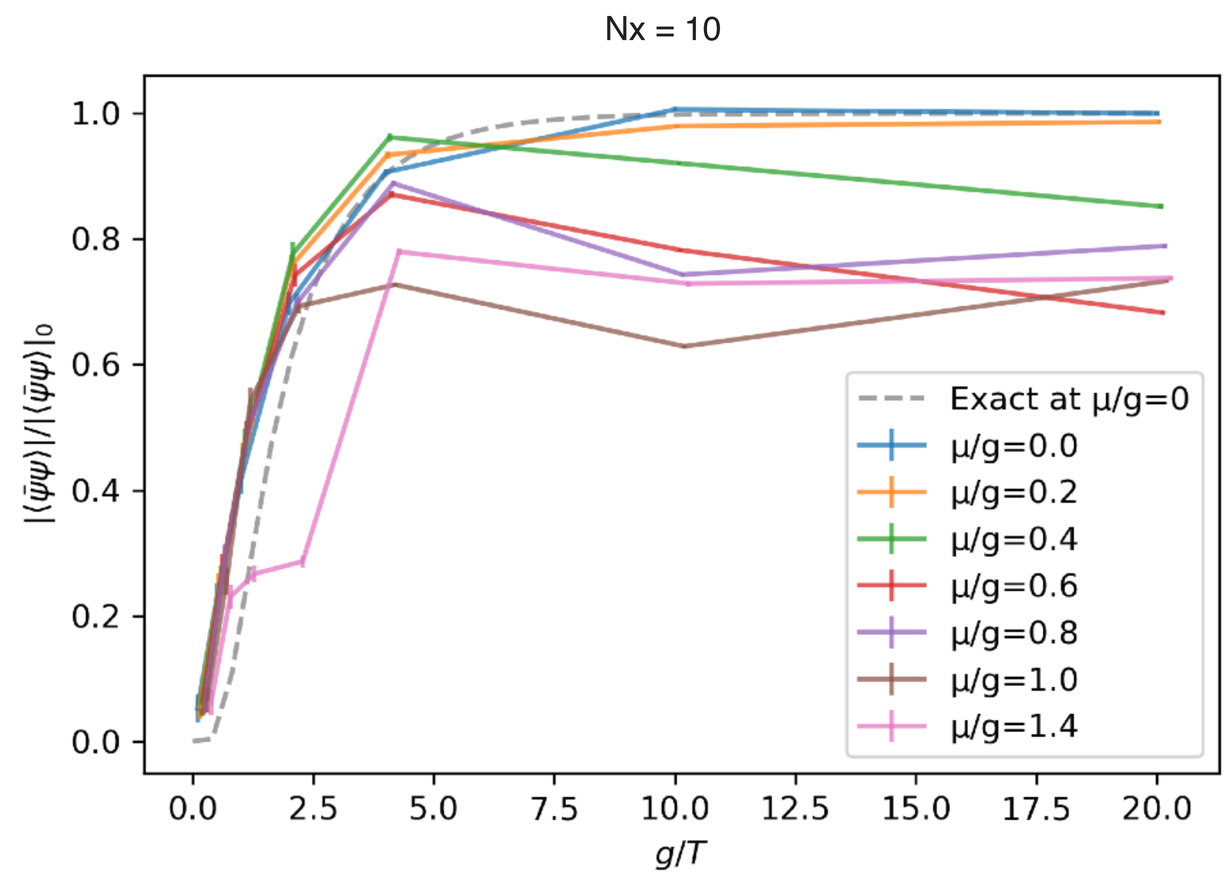
Simulation results

Continuum extrapolation for Nx = 8, 10

AT. 2205.08860

Continuum limit with a polynomial ansatz
it looks good So far*



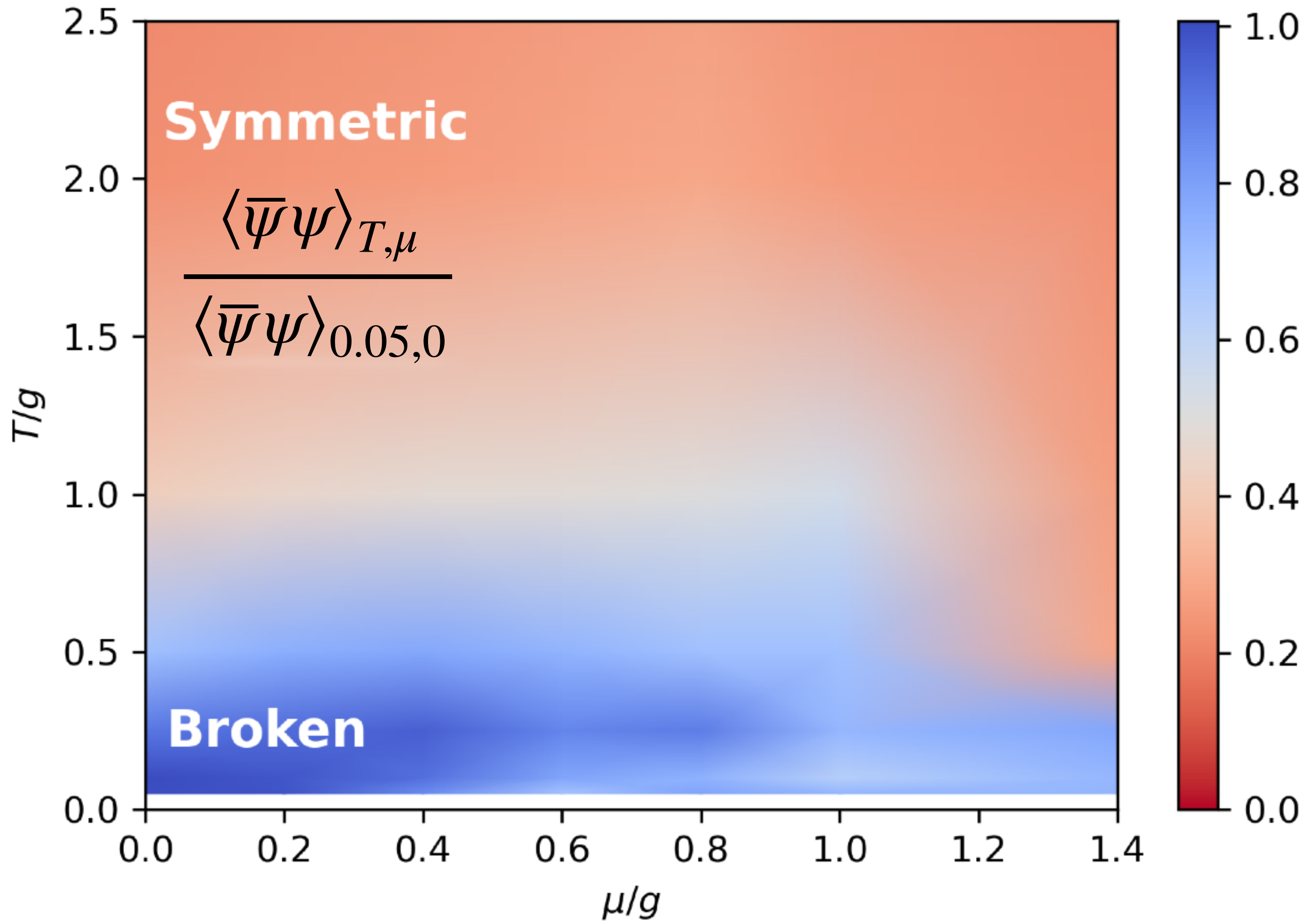


*(I did not include additive mass shift (Ross Dempsey+ arXiv: 2206.05308).
I thank to Takis Angelides (DESY) and Etsuko Itou (RIKEN) for letting me know this important reference!)

We use Nx = 10 results for the phase diagram

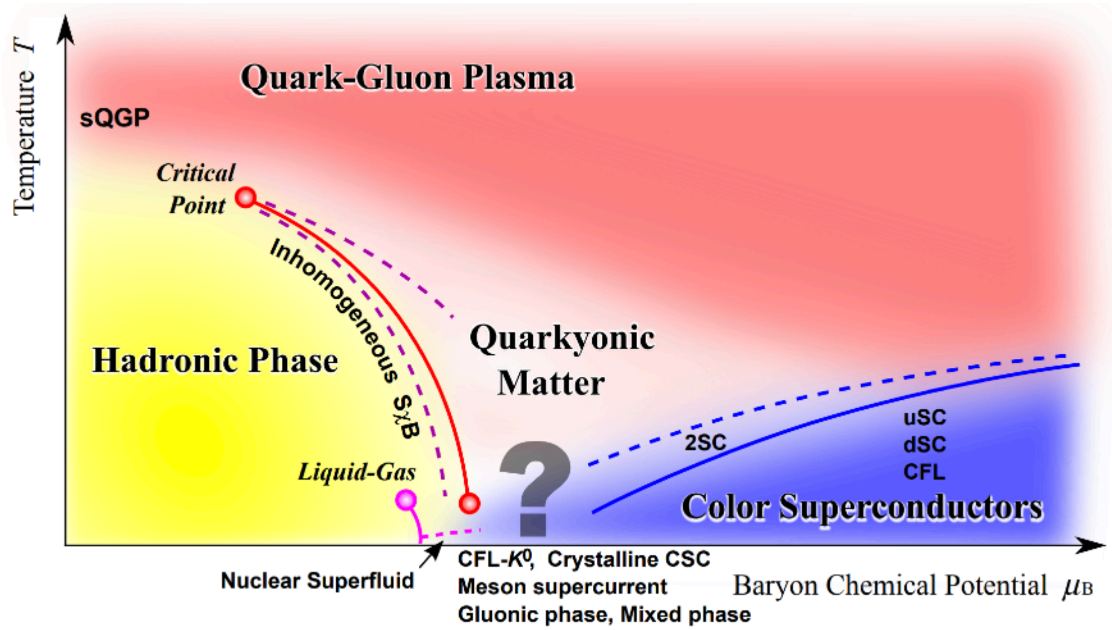
Summary for T-mu

Akio Tomiya

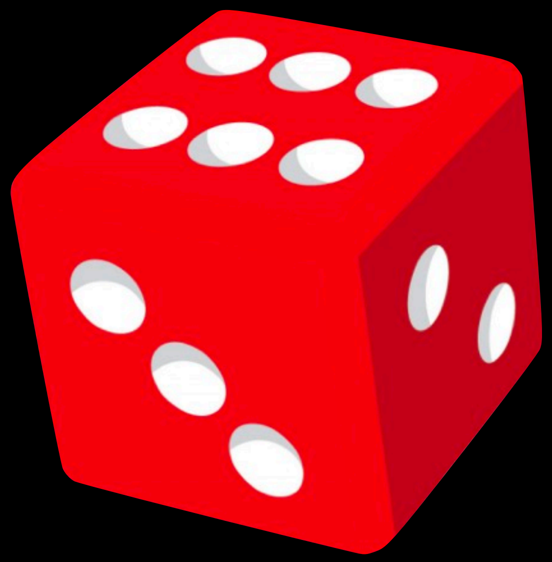


AT. 2205.08860

Fukushima , Hatsuda
Rept.Prog.Phys.74:014001,2011



- We investigate T-μ phase diagram for Schwinger model
- Continuum extrapolation has been evaluated (except for additive mass renormalization by 2206.05308)
- The variational approach does not show difficulty for our parameter regime
- Towards to go large volume, optimization of code, GPU version, tensor network. (noise-free) real device!



QFT with Hamiltonian

Akio Tomiya

Same hamiltonian

H : Hamiltonian for QFT)

Real time

Finte temperature

Operator formalism $U(t) = \mathrm{e}^{-\mathrm{i}tH}$

To Euclid (t → τ)

$t = -\mathrm{i}\tau$

To Minkowski (τ → t)

$U(\tau) = \mathrm{e}^{-\tau H}$

perturbation $\langle OO(t) \rangle = \langle \Omega | \hat{\mathrm{T}} O(0) O(t) | \Omega \rangle$

$|\Omega\rangle \sim \lim U(t) | 0 \rangle$

path int

$$Z = \int \mathscr{D}\bar{\psi}\mathscr{D}\psi \mathrm{e}^{\mathrm{i}S^M}$$

perturbation

$$S^M = \int_{-\infty}^{\infty} dt \int d^d x \, \mathscr{L}^M(x, t)$$

$\langle OO(\tau) \rangle = \mathrm{Tr}[O(0)O(\tau)\rho]$

This work

$\rho = U(\tau)/Z$

Path int

$$Z = \int \mathscr{D}\bar{\psi}\mathscr{D}\psi \mathrm{e}^{-S^E}$$

Conventional QFT

$$S^E = \int_0^{1/T} d\tau \int d^d x \, \mathscr{L}^E(x, \tau)$$

Fermion has Anti-PBC for imaginary time direction.
This is necessary to connect get trace formula
in the operator formalism

$$\psi(\tau + 1/T, \vec{x}) = -\psi(\tau, \vec{x})$$

Simulation results

Akio Tomiya

Variational free energy is O(1), Nx=10

AT. 2205.08860

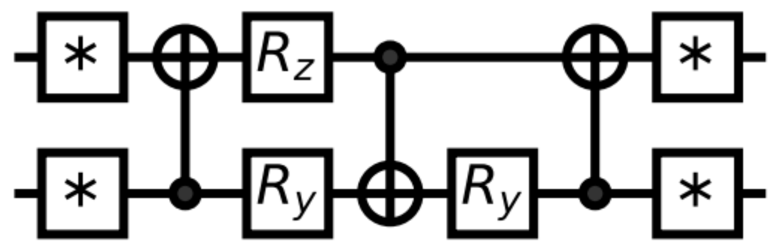
μ/g	g/T	N_x	~1/a w/g	Approx $\mathcal{L} - \ln Z$	Exact $-\ln Z$	Diff (%)
0.0	0.1	4	0.5	-27.779	-27.781	0.00804
0.0	0.1	4	0.35	-27.807	-27.808	0.005
0.0	0.1	10	0.5	-70.686	-70.718	0.0459
0.0	0.1	10	0.35	-71.744	-71.765	0.0302
0.0	0.5	4	0.5	-5.792	-5.802	0.185
0.0	0.5	4	0.35	-5.885	-5.891	0.105
0.0	0.5	10	0.5	-17.133	-17.25	0.68
0.0	0.5	10	0.35	-18.849	-18.934	0.448
0.0	10.0	4	0.5	-1.748	-1.75	0.161
0.0	10.0	4	0.35	-1.829	-1.829	0.0184
0.0	10.0	10	0.5	-8.218	-8.341	1.48
0.0	10.0	10	0.35	-9.98	-10.03	0.496
0.0	20.0	4	0.5	-1.492	-1.739	14.2
0.0	20.0	4	0.35	-1.653	-1.806	8.46
0.0	20.0	10	0.5	-8.202	-8.328	1.51
0.0	20.0	10	0.35	-9.955	-10.006	0.509

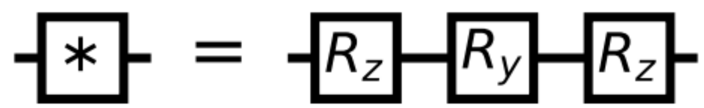
			~1/a	Approx	Exact	
1.4	0.1	4	0.5	-28.021	-28.023	0.00697
1.4	0.1	4	0.35	-27.989	-27.991	0.00755
1.4	0.1	10	0.5	-70.842	-70.874	0.0453
1.4	0.1	10	0.35	-71.742	-71.763	0.0291
1.4	0.5	4	0.5	-6.784	-6.789	0.0609
1.4	0.5	4	0.35	-6.644	-6.647	0.0327
1.4	0.5	10	0.5	-17.989	-18.104	0.636
1.4	0.5	10	0.35	-19.445	-19.534	0.456
1.4	10.0	4	0.5	-3.708	-3.71	0.0728
1.4	10.0	4	0.35	-3.63	-3.669	1.07
1.4	10.0	10	0.5	-10.067	-10.243	1.71
1.4	10.0	10	0.35	-11.763	-11.862	0.837
1.4	20.0	4	0.5	-3.673	-3.681	0.218
1.4	20.0	4	0.35	-3.621	-3.669	1.31
1.4	20.0	10	0.5	-10.028	-10.224	1.92
1.4	20.0	10	0.35	-11.699	-11.862	1.37

1. Mild dependence on μ
2. Hard for T -> 0 (large deviation) as expected

SU(4)
Variational ansatz

J. -Guo Liu+ 1902.02663





The general gate consists of 15 single qubit gates and 3 CNOT gates.
Each two qubit unitary is parametrized by 15 parameters in the rotational gates, which parametrizes the SU(4) group.

VQE and Beta VQE 1/2

Akio Tomiya

Background: VQE is a variational method

- Quantum machine: Exact ground state preparation is hard. In particular, it is difficult on near term devices
- Variational method for a *pure state* with a short circuit (VQE, variation quantum eigen-solver).
  - Quantum/Classical hybrid algorithm, iterative
  - Parametrized unitary circuit (~parametrized state $|\theta\rangle$, θ: a set of parameters)
    - $|\theta\rangle = \hat{U}(\theta)\big(|0\rangle_1 |0\rangle_2 |0\rangle_3 \cdots\big)$, and $\hat{U}(\theta)$ is a short circuit (entanglement + rotations)
  - If $\langle\theta|H|\theta\rangle = 0$, $|\theta\rangle \approx |\Omega\rangle$, where $|\Omega\rangle$ is the exact ground state
    = Variational approach for quantum system
-

VQE and Beta VQE (skip)

Beta VQE is a variational method for mixed states

J. -Guo Liu+ 1902.02663
*M. Germain+ 1502.03509

- Variational method for mixed states: Variational method on ρ
  - $$\rho_\Theta = \sum_{\{\vec{x}\}} p_\phi[\vec{x}]\, U_\theta |\vec{x}\rangle\langle\vec{x}| U_\theta^\dagger, \quad \Theta = \theta \cup \phi \text{ (parameters)}$$
  - $\vec{x} = (x_1, x_2, x_3, \cdots, x_k, \cdots)^\mathsf{T}$, and $x_k \in \{0,1\}$: (roughly) fermion excitation
  - $U_\theta |\vec{x}\rangle$: parametrized pure states, similar to the conventional VQE
  - $p_\phi[\vec{x}]$: Classically approximated distribution for a configuration of $\vec{x}$, Neural network (MADE*) is used. ϕ = parameters
- Minimizing $D(\rho_\Theta | \rho_{T,\mu}^{\text{exact}})$, we get approximated a set of states (= thermal)
- Shifted one (by a constant) is used in practice:
  - $$\mathscr{L}(\Theta) \equiv D(\rho_\Theta | \rho_{T,\mu}^{\text{exact}}) - \underbrace{\ln Z}_{\text{const}} = \mathrm{Tr}[\rho_\Theta \ln \rho_\Theta] + \frac{1}{T}\mathrm{Tr}[\rho_\Theta(\hat{H} - \mu\hat{N})]$$

The two language problem and solution?

The two language problem and solution?

- Programs for machine learning are usually implemented in Python
- LatticeQCD is in C++ (+CUDA)
- Two different languages used = “2 (programming) language problem”
- Use of one language is better for productivity
    - Python + LQCD: GPT for Grid, PyBridge++ for Bridge++, PyQCD
- Julia language* could be a solution of the problem
    - High performance as C++, Write like Python
    - NASA uses Julia 😲. Works on supercomputers
    - Machine learning, GPU and MPI friendly (Flux.jl, CUDA.jl, MPI.jl etc)
- LatticeQCD.jl, AT & Y. Nagai (updating to 1.0): LatticeQCD.jl
MPI-Parallel, stout smearing, domain-wall, staggered, (R)HMC, improved gauge actions, SU(Nc), gauge-covariant-neural net, ILDG support, etc…

* introduction: https://www.youtube.com/c/TheJuliaLanguage/playlists, https://akio-tomiya.github.io/julia_in_physics/ (japanese)

The two language problem and solution?

- Programs for machine learning are usually implemented in Python
- LatticeQCD is in C++ (+CUDA)
- Two different languages used = “2 (programming) language problem”
- Use of one language is better for productivity
  - Python + LQCD: GPT for Grid, PyBridge++ for Bridge++, PyQCD
- **Julia language* could be a solution of the problem**
  - High performance as C++, Write like Python
  - NASA uses Julia 😲. Works on supercomputers
  - Machine learning, GPU and MPI friendly (Flux.jl, CUDA.jl, MPI.jl etc)
- LatticeQCD.jl, AT & Y. Nagai (updating to 1.0): LatticeQCD.jl
  MPI-Parallel, stout smearing, domain-wall, staggered, (R)HMC, improved gauge actions, SU(Nc), gauge-covariant-neural net, ILDG support, etc…

* introduction: https://www.youtube.com/c/TheJuliaLanguage/playlists, https://akio-tomiya.github.io/julia_in_physics/ (japanese)

Beta VQE 2/4

Extended VQE for mixed states

Jin-Guo Liu+ 1902.02663

- How can we realize $\rho_\Theta \approx \rho$ for $\rho = \frac{1}{Z} \mathrm{e}^{-\frac{1}{T}(\hat{H}-\mu\hat{N})}$
- Minimize Kullback–Leibler–*Umegaki* divergence (pseudo-distance)
  - $D(\rho_\Theta \,|\, \rho) = \mathrm{Tr}[\rho_\Theta \ln \frac{\rho_\Theta}{\rho}] = \mathrm{Tr}[\rho_\Theta \ln \rho_\Theta] - \mathrm{Tr}[\rho_\Theta \ln \rho]$
  - Relative entropy for density matrices (Classical ver. is called KL div.)
  - This is bounded $D(\rho_\Theta \,|\, \rho) \geq 0$ and saturated iff $\rho_\Theta = \rho$
  - In practice, we minimize shifted one,

$$\mathscr{L}(\Theta) = D(\rho_\Theta \,|\, \rho) - \underbrace{\ln Z}_{\text{const}} = \mathrm{Tr}[\rho_\Theta \ln \rho_\Theta] + \frac{1}{T}\mathrm{Tr}[\rho_\Theta(\hat{H} - \mu\hat{N})]$$

Note (skip)

We can define, a loss function, $\tilde{\mathscr{L}}(\Theta) = D(\rho_\Theta || \rho)$

$$\rho_{T,\mu} = \frac{1}{Z_{T,\mu}} \mathrm{e}^{-\frac{1}{T}(\hat{H}-\mu\hat{N})}$$

$$\begin{aligned}
D(\rho_\Theta || \underset{T,\mu}{\rho}) &= \mathrm{Tr}\,[\rho_\Theta \log \frac{\rho_\Theta}{\rho_{T,\mu}}], && (24)\\
&= \mathrm{Tr}\,[\rho_\Theta \log \rho_\Theta] - \mathrm{Tr}\,[\rho_\Theta \log \rho_{T,\mu}], && (25)\\
&= \mathrm{Tr}\,[\rho_\Theta \log \rho_\Theta] - \mathrm{Tr}\,[\rho_\Theta \log \frac{1}{Z_{T,\mu}} \mathrm{e}^{-\frac{1}{T}(\hat{H}-\mu\hat{N})}], && (26)\\
&= \mathrm{Tr}\,[\rho_\Theta \log \rho_\Theta] + \mathrm{Tr}\,[\rho_\Theta \log Z_{T,\mu}] + \frac{1}{T}\mathrm{Tr}\,[\rho_\Theta(\hat{H} - \mu\hat{N})], && (27)\\
&= \mathrm{Tr}\,[\rho_\Theta \log \rho_\Theta] + \mathrm{Tr}\,[\rho_\Theta] \log Z_{T,\mu} + \frac{1}{T}\mathrm{Tr}\,[\rho_\Theta(\hat{H} - \mu\hat{N})], && (28)\\
&= \mathrm{Tr}\,[\rho_\Theta \log \rho_\Theta] + \underset{\text{(const in }\Theta)}{\log Z_{T,\mu}} + \frac{1}{T}\mathrm{Tr}\,[\rho_\Theta(\hat{H} - \mu\hat{N})]. && (29)
\end{aligned}$$

The last line follows because ρ_Θ is normalized.

In practice, we use,

$$\mathscr{L}(\Theta) = \tilde{\mathscr{L}}(\Theta) - \log Z_{T,\mu} = \mathrm{Tr}\,[\rho_\Theta \log \rho_\Theta] + \frac{1}{T}\mathrm{Tr}\,[\rho_\Theta(\hat{H} - \mu\hat{N})]. \qquad (30)$$

Namely,

$$\mathscr{L}(\Theta) = \mathrm{Tr}\,[\rho_\Theta \log \rho_\Theta] + \frac{1}{T}\mathrm{Tr}\,[\rho_\Theta \mathscr{H}], \qquad (31)$$

Beta VQE 3/4

Extended VQE for mixed states

Jin-Guo Liu+ 1902.02663

- $\mathscr{L}(\Theta) = \mathrm{Tr}[\rho_\Theta \ln \rho_\Theta] + \frac{1}{T}\mathrm{Tr}[\rho_\Theta(\hat{H} - \mu\hat{N})]$
    - $\mathrm{Tr}[\rho_\Theta \log \rho_\Theta] = \sum_{\{\vec{x}\}} p_\phi(\vec{x}) \log p_\phi(\vec{x})$
- We need two derivatives
    - $\frac{\partial}{\partial\phi}\mathscr{L}(\Theta) = \frac{\partial}{\partial\phi}\sum_{\{\vec{x}\}} p_\phi(\vec{x})[\log p_\phi(\vec{x})]$: Classical

p: a neural network
-> gradient descent

-
    - $\frac{\partial}{\partial\theta}\mathscr{L}(\Theta) = \frac{1}{T}\frac{\partial}{\partial\theta}\langle \vec{x} \,|\, U_\theta^\dagger \mathscr{H} U_\theta \,|\, \vec{x} \rangle]$: Quantum

REINFORCE algorithm

MADE: Masked Auto-encoder for Distribution Estimation

1502.03509

I (mostly) skip this section in the seminar

arXiv: 1502.03509

Summary of MADE

(simple) Neural network for probability estimation

- MADE = Masked Auto-encoder for Distribution Estimation
- Auto-encoder is a neural network
- It can mimic a joint distribution of binary variables (0, 1)
    - (x_1, x_2, x_3, x_4) is distributed as $p(x_1, x_2, x_3, x_4) \equiv p[\overrightarrow{x}]$
- It is categorized as a generative model (as the normalizing flow)
- It is correctly normalized

arXiv: 1502.03509

Basics (skip)

Product rule in the probability theory

- A configuration of variables (x_1, x_2, x_3, x_4) is distributed as $p(x_1, x_2, x_3, x_4) \equiv p[\overrightarrow{x}]$
- Probability distribution is normalized.
- For binary variables,

$$1 = \sum_{x_1=0}^{1} \sum_{x_2=0}^{1} \sum_{x_3=0}^{1} \sum_{x_4=0}^{1} p(x_1, x_2, x_3, x_4) = \sum_{\{\overrightarrow{x}\}} p_\phi[\overrightarrow{x}]$$

arXiv: 1502.03509

Basics (skip)

Product rule in the probability theory

- definition of the conditional probability is $p(x_2 \mid x_1) \equiv \dfrac{p(x_1, x_2)}{p(x_1)}$
    - Equivalently, $p(x_1, x_2) = p(x_1)p(x_2 \mid x_1)$: Product rule
- We can generalize to more than 2 variables
    - $p(x_3 \mid x_1, x_2) = \dfrac{p(x_1, x_2, x_3)}{p(x_1, x_2)} \Leftrightarrow p(x_1, x_2, x_3) = p(x_3 \mid x_1, x_2)p(x_2 \mid x_1)p(x_1)$
    - $p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2)p(x_4 \mid x_1, x_2, x_3)$
- We abbreviate this as $p(x_1, x_2, x_3, x_4) = \prod_{k=1}^{4} p(x_k \mid x_{<k})$

arXiv: 1502.03509

Bernoulli process (skip)

un-fair coin

- A (un-)fair coin, which takes face for a probability p, Tail for 1-p
- This process is called “Bernoulli trial” in Math
- Let us denote it as $\mathrm{Bernoulli}(p)$

arXiv: 1502.03509

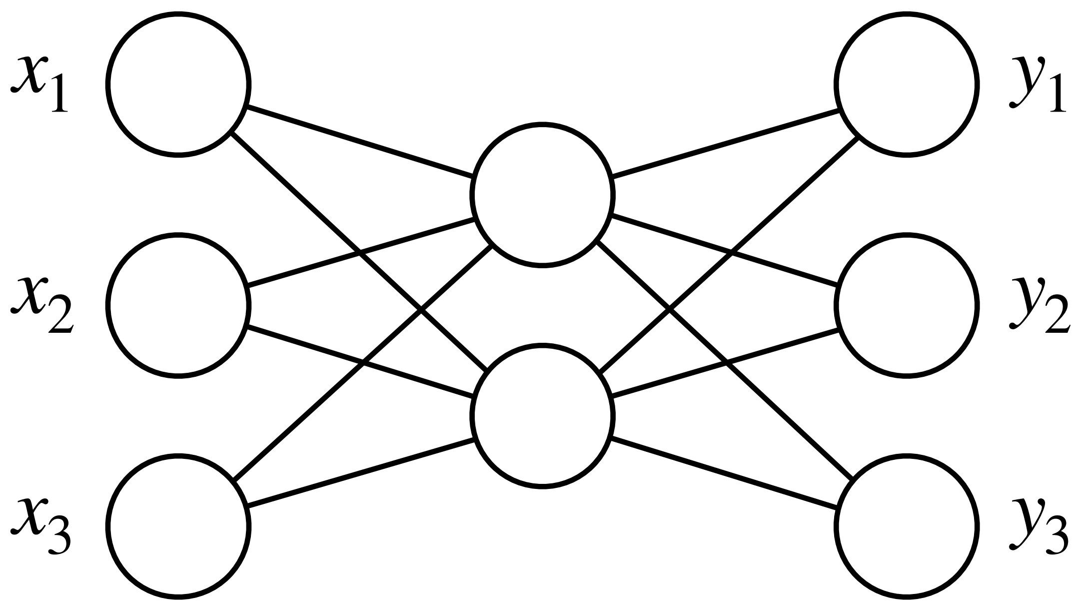
Basics (skip)

Product rule in the probability theory

- Neural network (NN) mimics $p(x_1, x_2, x_3) = p(x_1)p(x_2 | x_1)p(x_3 | x_1, x_2)$, whose input is binary array (x_1, x_2, x_3): 3 correlated coins
  - We can draw a sample using $\hat{x}_1 \sim y_1 \approx p(x_1)$
  - How can we construct $\hat{x}_2 \sim y_2 \approx p(x_2 | x_1)$
    - input only depends on x_1
  - How can we construct $\hat{x}_3 \sim y_3 \approx p(x_3 | x_1, x_2)$
    - input only depends on x_1, x_2

Auto-encoder (skip)

Auto-encoder ~ (un normalized) flow



$$-E[x] = \sum_i x_i \log y_i + (1 - x_i)\log(1 - y_i)$$

$$e^{-E[x]} = \prod_i y_i^{-x_i}(1 - y_i)^{-(1-x_i)}$$

$$\sum_{\{x\}} e^{-E[x]} \neq 1$$ Not-normalized

arXiv: 1502.03509

Auto-regressive property (skip)

Product rule

$$y_1 = p(x_1 = 1),\ \ y_2 = p(x_2 = 1\,|\,x_1),\ \ y_3 = p(x_3 = 1\,|\,x_1, x_2)$$

$$\longrightarrow\ \ p(x_1 = 0) = 1 - y_1,\ \ p(x_2 = 0\,|\,x_1) = 1 - y_2,\ \ p(x_3 = 0\,|\,x_1, x_2) = 1 - y_3$$

$$\longrightarrow\ \ y_d = p(x_d = 1\,|\,x_{<d}) \qquad p(x_d = 0\,|\,x_{<d}) = 1 - y_d$$

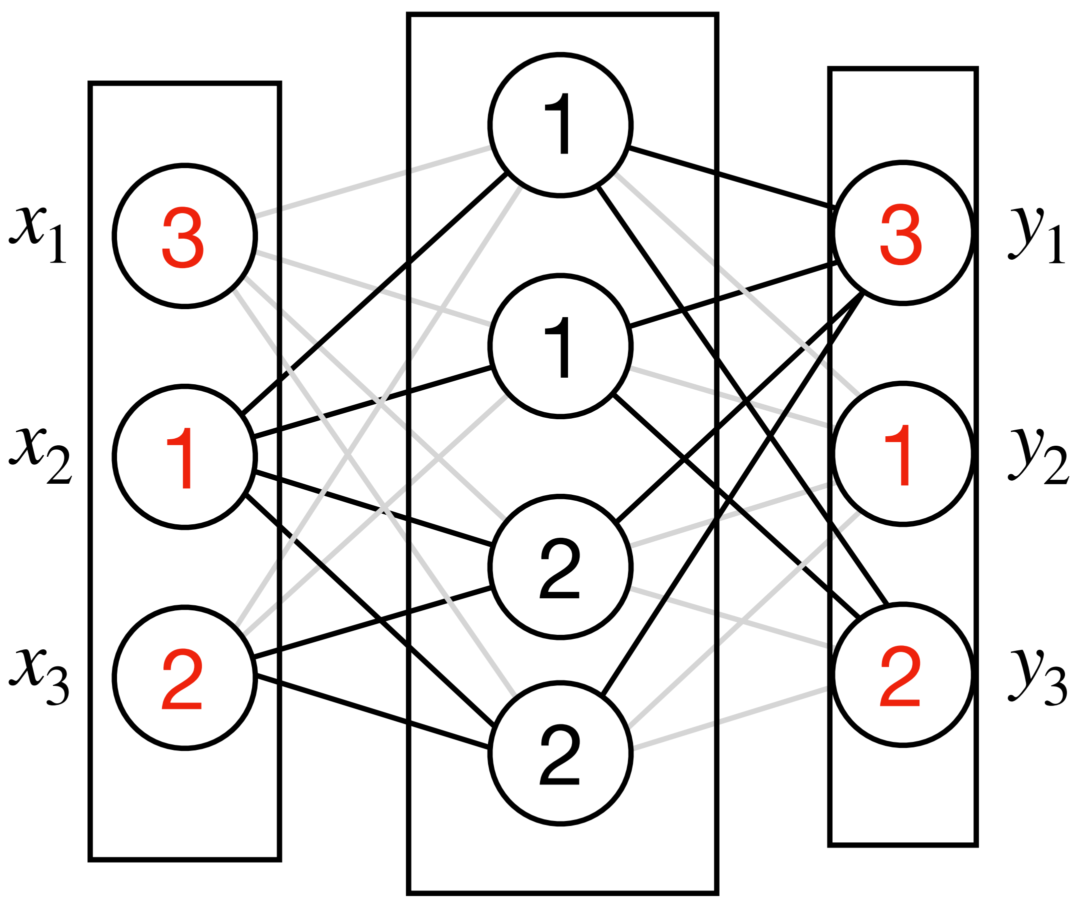
$$p(x_1, x_2, x_3, x_4) = \prod_{k=1}^{4} p(x_k\,|\,x_{<k})$$

$$\longrightarrow\ \ -\log p(x_1, x_2, x_3, x_4) = -\sum_{k=1}^{4} \log p(x_k\,|\,x_{<k})$$

arXiv: 1502.03509

MADE (skip)

Masked auto-encoder for density estimation



$$\hat{x}_1 \sim y_1 \approx p(x_1 \,|\, x_2, x_3)$$

$$\hat{x}_2 \sim y_2 \approx p(x_2)$$

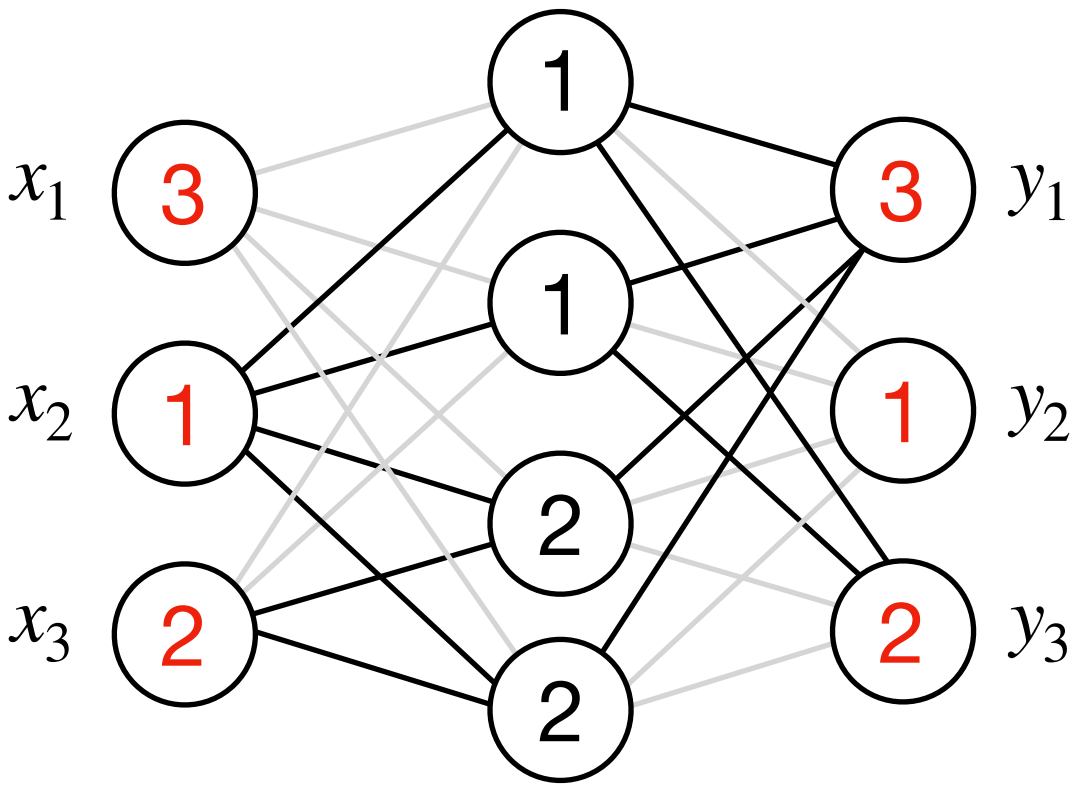
$$\hat{x}_3 \sim y_3 \approx p(x_3 \,|\, x_2)$$

Assign numbers on node:
Input& output node = assign

arXiv: 1502.03509

MADE (skip)

Masked auto-encoder for density estimation



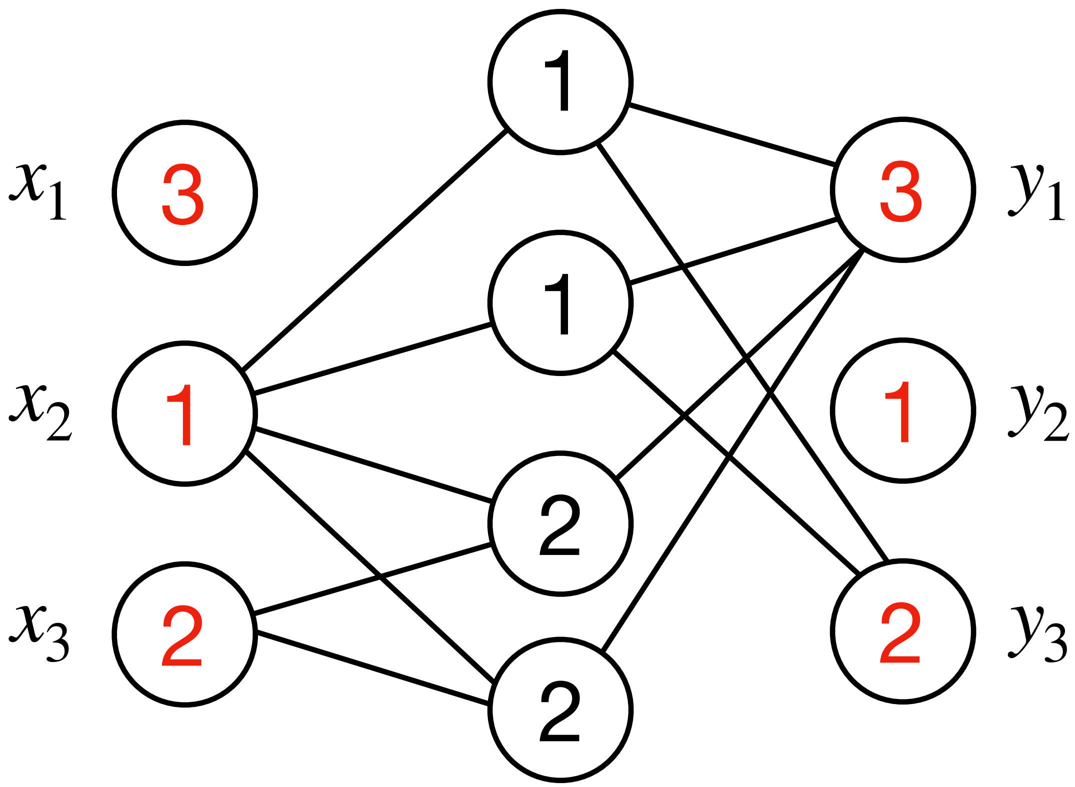
$\hat{x}_1 \sim y_1 \approx p(x_1 | x_2, x_3)$

$\hat{x}_2 \sim y_2 \approx p(x_2)$

$\hat{x}_3 \sim y_3 \approx p(x_3 | x_2)$

MADE (skip)

Masked auto-encoder for density estimation



$\hat{x}_1 \sim y_1 \approx p(x_1 \,|\, x_2, x_3)$

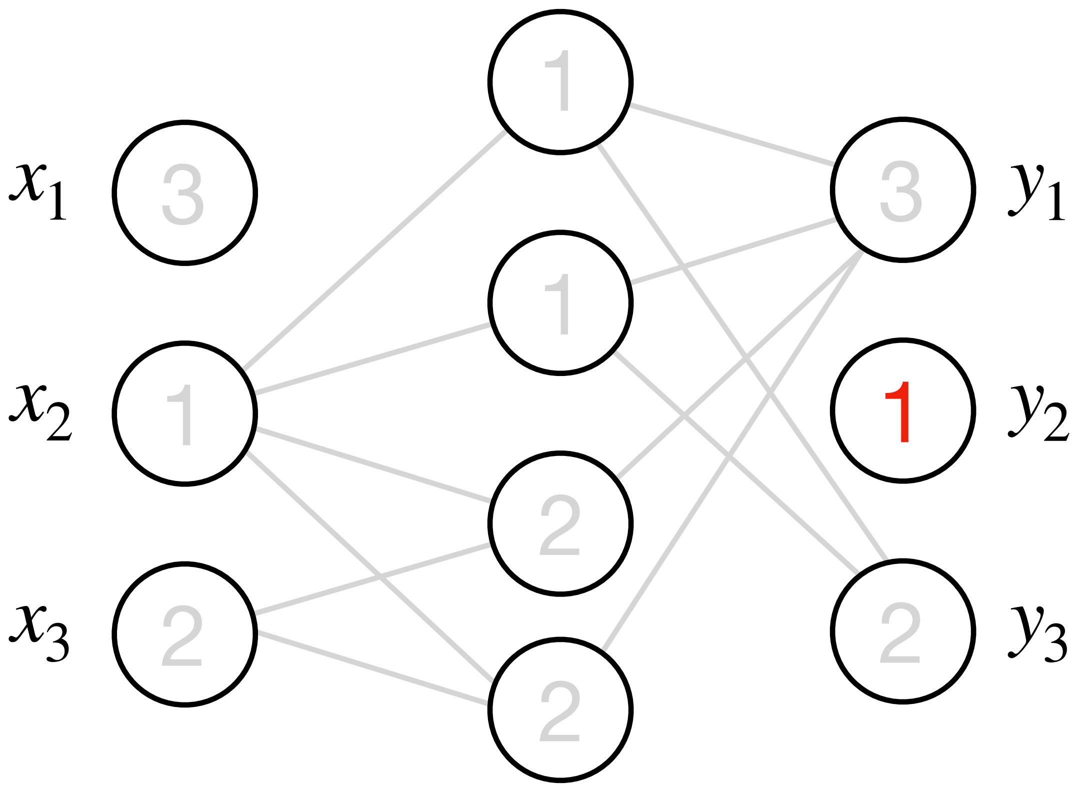
$\hat{x}_2 \sim y_2 \approx p(x_2)$

$\hat{x}_3 \sim y_3 \approx p(x_3 \,|\, x_2)$

arXiv: 1502.03509

MADE (skip)

Masked auto-encoder for density estimation



$\hat{x}_1 \sim y_1 \approx p(x_1 \,|\, x_2, x_3)$

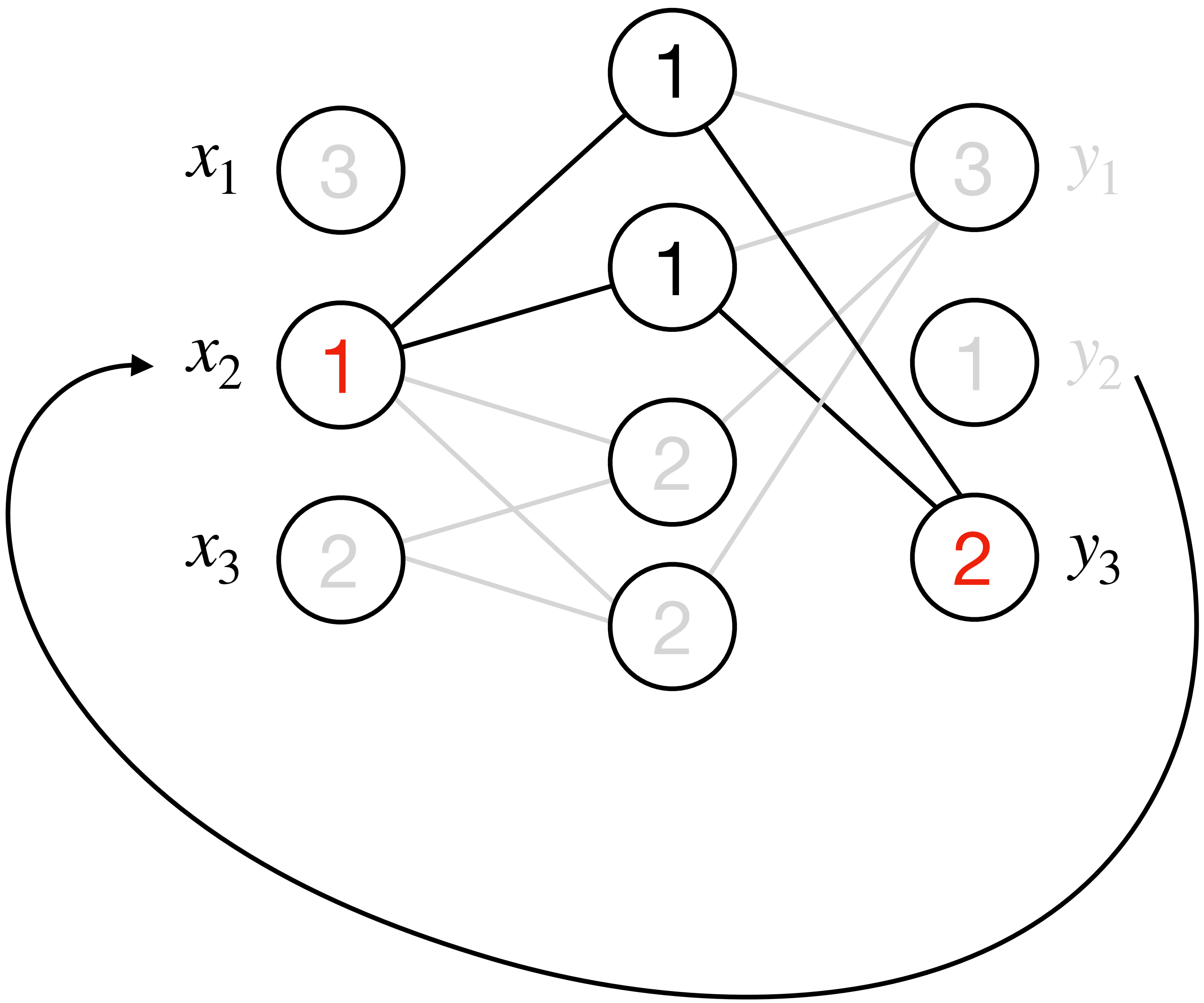
$\hat{x}_2 \sim y_2 \approx p(x_2)$

$\hat{x}_3 \sim y_3 \approx p(x_3 \,|\, x_2)$

arXiv: 1502.03509

MADE (skip)

Masked auto-encoder for density estimation



$\hat{x}_1 \sim y_1 \approx p(x_1 \,|\, x_2, x_3)$

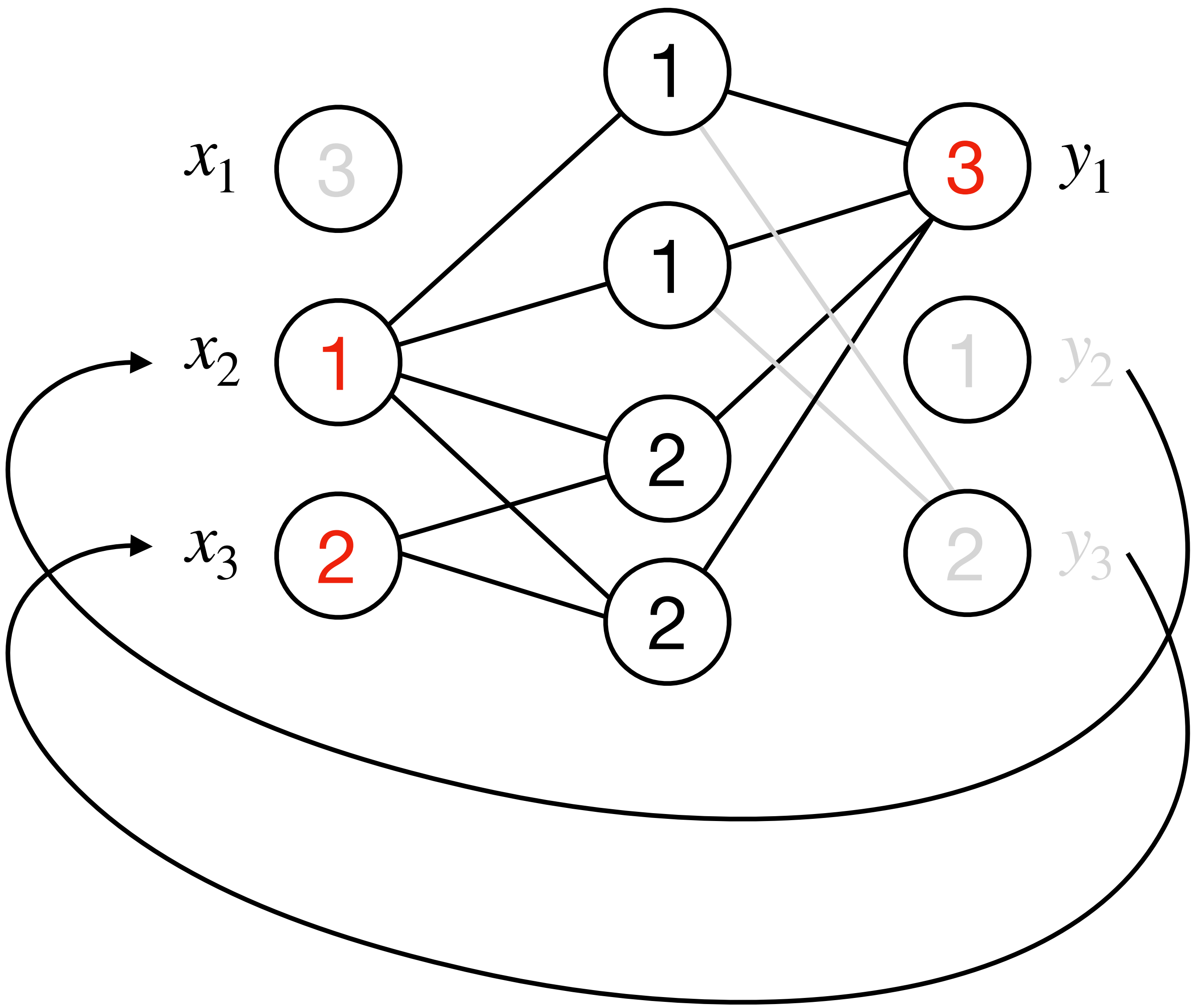
$\hat{x}_2 \sim y_2 \approx p(x_2)$

$\hat{x}_3 \sim y_3 \approx p(x_3 \,|\, x_2)$

arXiv: 1502.03509

MADE (skip)

Masked auto-encoder for density estimation



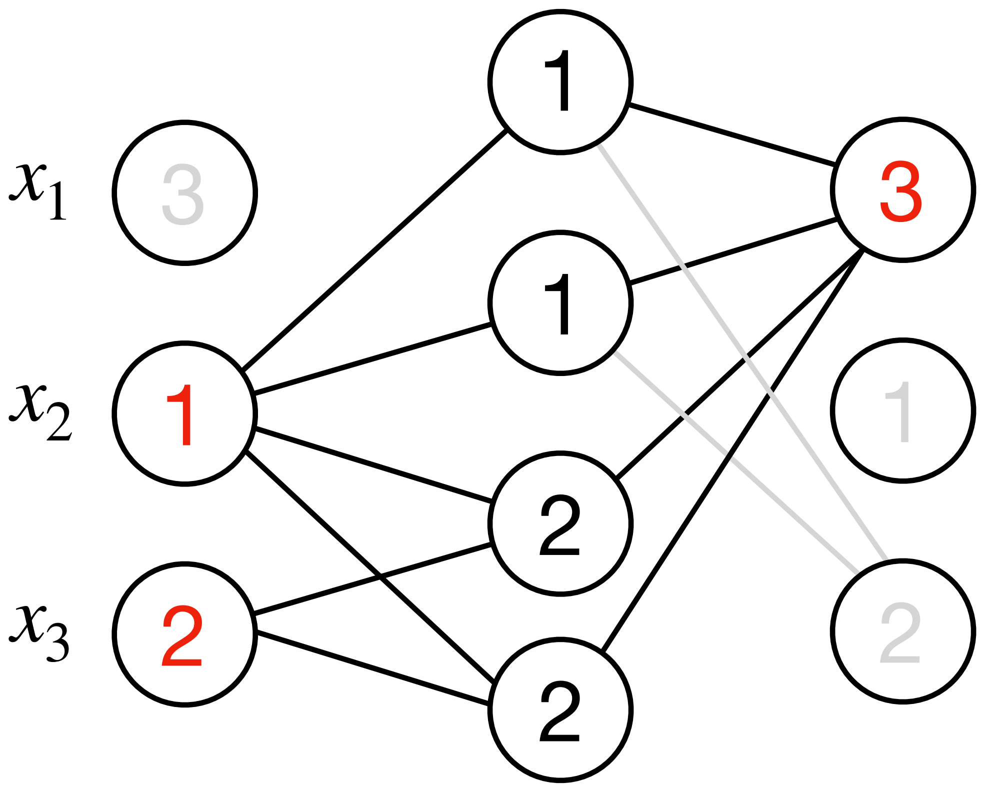
$\hat{x}_1 \sim y_1 \approx p(x_1 | x_1, x_3)$

$\hat{x}_2 \sim y_2 \approx p(x_2)$

$\hat{x}_3 \sim y_3 \approx p(x_3 | x_2)$

MADE (skip)

Masked auto-encoder for density estimation



$y_1 \approx p(x_1 \,|\, x_2, x_3)$ $\hat{x}_1 \sim y_1(x_2, x_3)$

$y_2 \approx p(x_2)$ $\hat{x}_2 \sim y_2$

$y_3 \approx p(x_3 \,|\, x_2)$ $\hat{x}_3 \sim y_3(x_2)$

We can draw a set of sample $(\hat{x}_1, \hat{x}_2, \hat{x}_3)$ from $p_\phi(x_1, x_2, x_3)$ where ϕ is network param.