Physics and Machine Learning

Hideaki Iida (FEFU)

Contents

- History of computer vs human on board game
- Basics of deep learning
- Application to Physics

History of computer vs human on board games

Shogi

- **Shogi=Chess-like game (Japanese Chess)**
  - **Rule:**
    Moving one piece in turn, following the rule of movement for each piece.
    You win when you get 王将 or 玉将.
  - **Correspondence btw Shogi & Chess:**
    (Note: similar but there are differences)

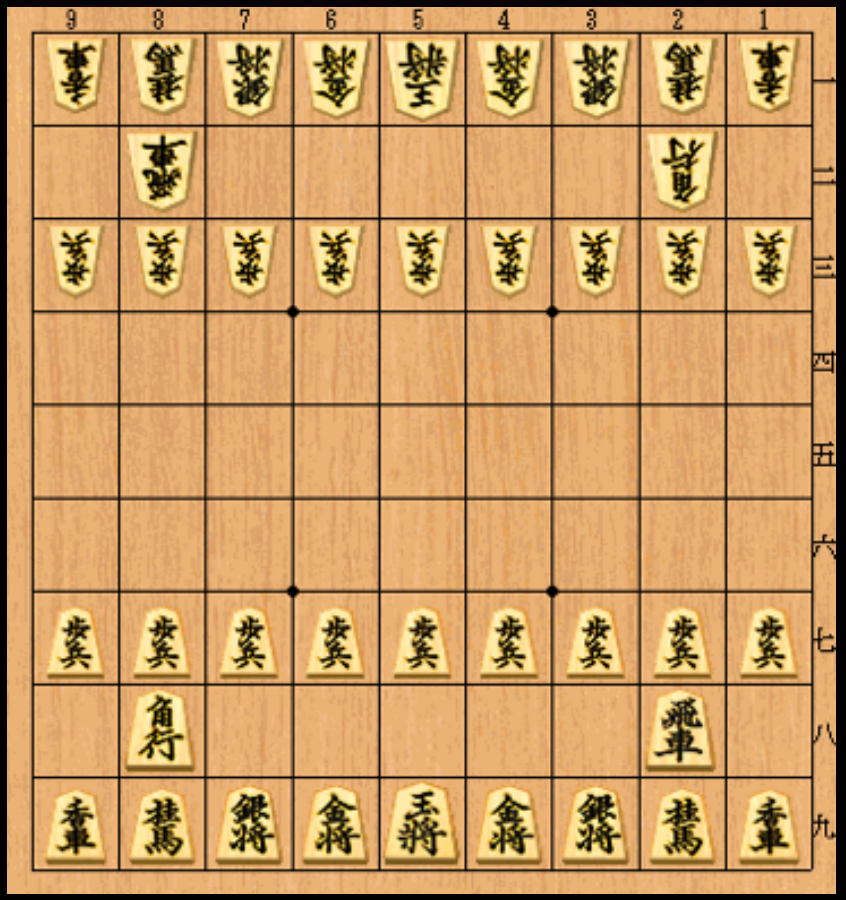
    王将,玉将 (Ohsho, Gyoku)~King

    桂馬(Keima)~Knight

    角行(Kaku)~Bishop

    飛車(Hisha)~Luke

    歩兵(Hu)~Pawn
  - **Uniqueness of Shogi:**
    You can use the pieces you get
    … this feature makes Shogi complex



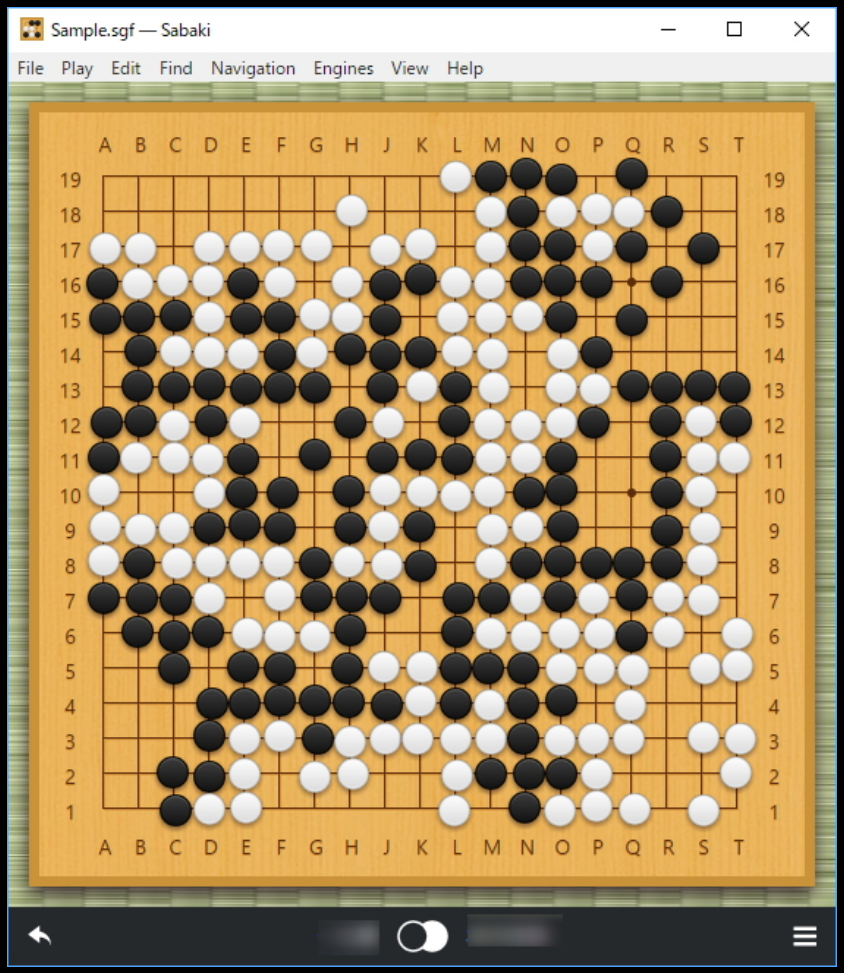
Initial state of Shogi

http://www.shogi-rule.com/将棋の駒の使い方/将棋の駒の初期配置.html

Go

- **Go (or Igo)**
  - **Rule:**
    The players put “stones” (black or white)
    in turn on the vacant intersections of a board.
    When stones surround the opposite stones,
    they (opposite stones) are removed.
    The winner is determined by counting
    each player's surrounded territory
    along with captured stones and komi
    (points added to the score of the player
    with the white stones as compensation
    for playing second).
    referred from Wikipedia of “Go”
  - **Features of Go:**
    The board is large (19x19 grid)
    -> “game tree complexity” is so high:

Chess	10^{123}
Shogi	10^{226}
Go	**10^{360}**



https://forest.watch.impress.co.jp/img/wf/docs/1114/356/html/image1.jpg.htm

History of computer vs human

- **Chess**
  1996 & 1997 Deep Blue vs Гарри Кимович Каспаров
  Result: 1(comp.)-3(human) & 2 draws (1996)
  2-1 & 3 draws (1997)

- **Shogi**
  2010-2017 Denno-Sen (電王戦)
  - 2016: ponanza* vs Takayuki Yamazaki (叡王, 八段)

    Result: 2-0
  - 2017: ponanza vs Amahiko Sato (叡王, 名人)

    Result: 2-0

***ponanza: name of program**
machine: Intel Core i7 6700 3.4GHz 4 cores

History of computer vs human

Go is distinctive: Chess & Shogi are “easier” for computers

- **Shogi**
  Actually, **computer was already enough strong in 2014**
  - The regulation of machine of Denno-Sen
    Before 2014: cluster computer (about 700 PCs)
    From 2014: laptop PC (!!!)
    because otherwise computer is too strong

- **Go**
  In contrast to Chess & Shogi, before Oct. 2015,
  no computer program wins professional Go player, even with some advantages!

Then AlphaGo was descended to our world

History of computer vs human

- **Go**
  - Oct.2015
    AlphaGo Fan vs 樊麾 (Fan Hui, Europe Champion)

Result 5-0

  - Mar.2016
    AlphaGo Lee vs 李世乭

(Lee Sedol, World Champ. for 18 times)

Result 4-1

柯潔 (Ke Jie, strongest Go player)

said “AI can win against 李, but not against me”

History of computer vs human

However…

- May2017
  AlphaGo Master vs **柯潔**

Result 3-0

*AlphaGo Master won 60 games (no loss) including
many world champions (not official & on-line play)

Recently, **柯潔** loses against “絶芸” (Tencent)

with handicap for 絶芸

But this is not the end of the story

History of computer vs human

- **AlphaGo Zero**
  trained by playing with itself, without referring any human games.
  - After 3 days, exceeded AlphaGo Lee (after 4,900,000 games)
  - After 21 days, exceeded AlphaGo Master
  - After 40 days, exceeded all of the previous versions of AlphaGo
- **AlphaZero**
  based on AlphaGo Zero (trained by self-playing)
  but can play not only Go but also Chess and Shogi.
  It becomes stronger than the strongest programs of
  - Chess (Stockfish) after 4 hours
  - Shogi (elmo) within 24 hours
  - Igo (AlphaGo Zero with 3 days training) within 24 hours.

Computer, which can play games extremely many times, does not need the training data of human

History of computer vs human

- **Key feature of AlphaGo**

  Deep neural network

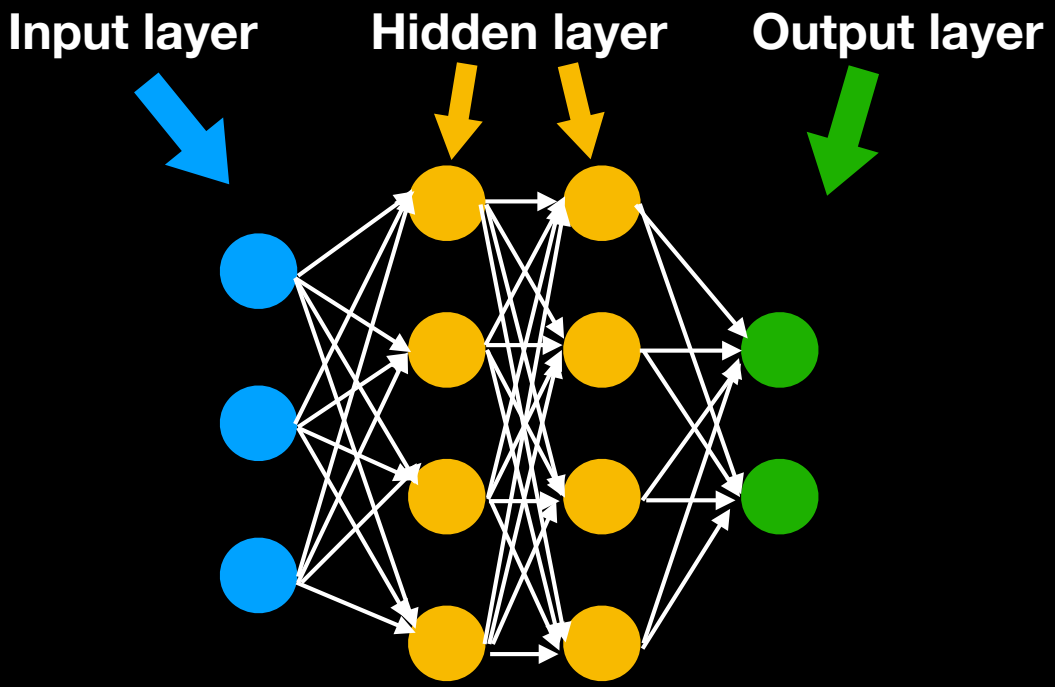
  - neural network: explain later
  - “Deep”: neural network with many layers

- **Deep neural network mimics the network of human brain: extremely “flexible” function, not biased by human**

  … fit to various regressions and optimizations with Machine Learning

Basics of Deep Learning

Neural Network



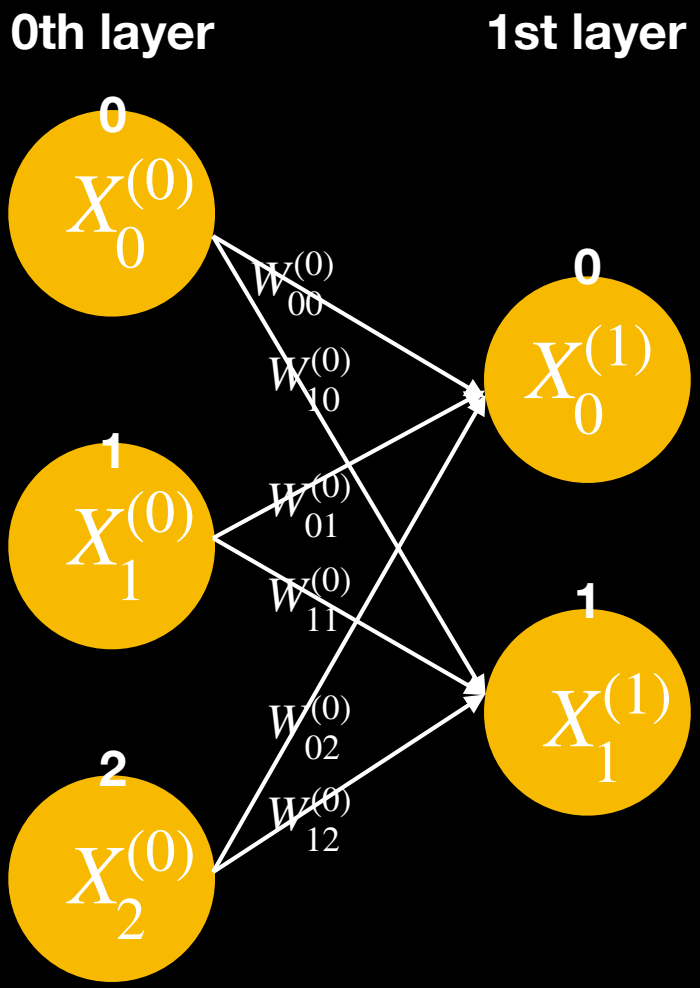
Neural network

network composed of many units
(units: denoted by circles in the left pic.)
… **mimics human brain**

Deep neural network

more than 3 hidden layers
(from Wikipedia)

Perceptron



A unit is made of two kinds of transformations

1. Linear transformation

$$u_0^{(1)} = W_{00}^{(0)} X_0^{(0)} + W_{01}^{(0)} X_1^{(0)} + W_{02}^{(0)} X_2^{(0)}$$

2. Nonlinear transformation

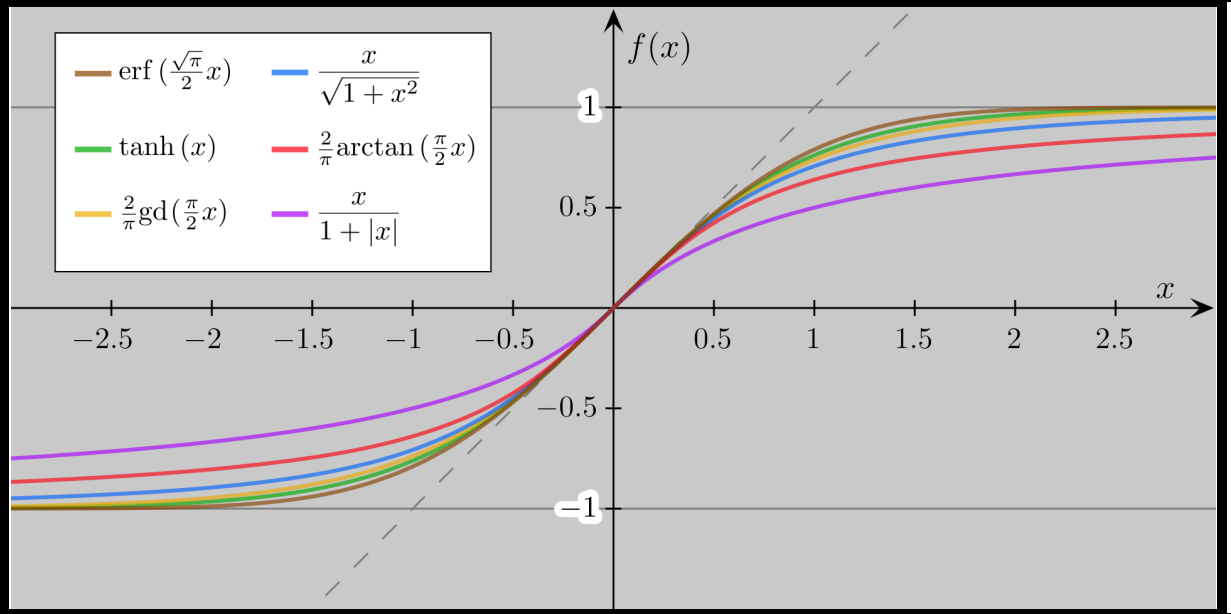
$$z_0^{(1)} = f(u_0^1)$$

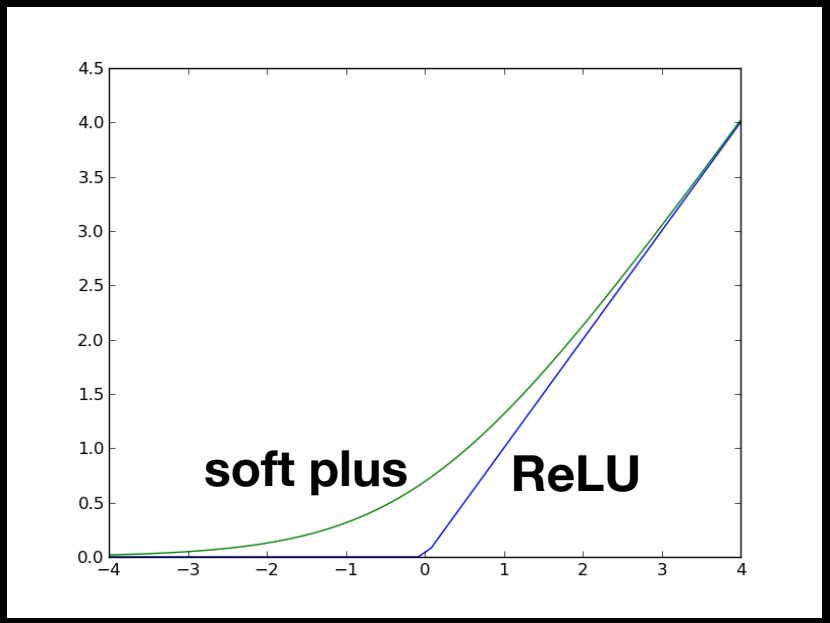
f : activation function (nonlinear)

Sigmoid
Softsign
softplus
ReLU
etc…

Activation functions

f(x) is increasing function and non-linear
Typially they have a threshold
… mimics firing of neuron





Graphs from Wikipedia “活性化関数”

Convolutional Neural Network (CNN)

- Another very important network
- I believe Nikolai and/or Sergei will explain it

Supervised training

Input $\vec{x}$

Neural Network
$\vec{f}_{\mathrm{NN}}(\vec{x}; W_{ij}^{(k)})$

$W_{ij}^{(k)}$: parameters

Output
$\vec{f}_{\mathrm{NN}}(\vec{x}; W_{ij}^{(k)})$

ex) $\vec{x}$: (R,G,B) for each pixel of a photo

$\vec{f}_{\mathrm{NN}}$: If the photo is dog: $\vec{f}_{\mathrm{NN}} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ 

cat: $\vec{f}_{\mathrm{NN}} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ 

Supervised learning:
Optimisation wrt W by using training data (like χ^2 fit of a function)
… in this case, photos of dogs & cats (typically, several thousands)

$\vec{x}^{(0)}$  $\vec{y}^{(0)} = (1,0)$

$\vec{x}^{(1)}$  $\vec{y}^{(1)} = (1,0)$

$\vec{x}^{(2)}$  $\vec{y}^{(2)} = (0,1)$

$\vec{x}^{(3)}$  $\vec{y}^{(3)} = (1,0)$

$\vec{x}^{(4)}$  $\vec{y}^{(4)} = (0,1)$

$\vec{x}^{(5)}$  $\vec{y}^{(5)} = (0,1)$

...

Loss function

Data set of the training: $(\vec{x}^{(l)}, \vec{y}^{(l)})$

l : numbering of pictures $\vec{x}^{(l)}$: data of l-th picture $\vec{y}^{(l)}$: answers of l-th picture

What is a criterion of optimisation?
… minimisation of loss function $L(\vec{f}_{\mathrm{NN}})$ ($\sim\chi^2$ of fit)

mean square (regression)

$$L(\vec{f}_{\mathrm{NN}}) \equiv \sum_{i,l} (f^{(l)}_{\mathrm{NN,i}} - y^{(l)}_i)^2$$

cross entropy (class identification)

$$L(\vec{f}_{\mathrm{NN}}) \equiv -\sum_{i,l} y^{(l)}_i \log f^{(l)}_{\mathrm{NN,i}}$$

(in this case f_i should be in [0,1])

Optimization of parameters

How to optimize $W_{ij}^{(k)}$?

Update the parameters by

$$W_{ij}^{(k;t+1)} = W_{ij}^{(k;t)} + \Delta W_{ij}^{(k;t)}, \qquad \Delta W_{ij}^{(k;t)} = -\eta \frac{\partial L}{\partial W_{ij}^{(k)}}\bigg|_{W_{ij}^{(k)}=W_{ij}^{(k;t)}}$$

$W_{ij}^{(k;t)}$: $W_{ij}^{(k)}$ at t-th updation

… update in the direction of -grad

There are some variants of the "gradient" method:
SGD, Momentum SGD, AdaGrad, RMSprop, AdaDelta, Adam…

(from the Textbook of Machine Learning by M.Taki)

How to calculate $\frac{\partial L}{\partial W_{ij}^{(k)}}$ **?**

$$\frac{\partial L}{\partial W_l} \simeq \frac{L(W_1, \cdots, W_l + \epsilon, \cdots, W_D) - L(W_1, \cdots, W_l, \cdots, W_D)}{\epsilon}$$

(l : 1dim. index of k, i, j, D : total number of the combination of i, j, k)

Too much calculation cost! (∼D-times forwarding needed)

→Back propagation: cost∼O(1) forwarding!!

1. devide into two parts:

$$\frac{\partial L}{\partial W^{(k)}} = \frac{\partial L}{\partial u^{(k)}} \frac{\partial u^{(k)}}{\partial W^{(k)}} = \frac{\partial L}{\partial u^{(k)}} z^{(k-1)} \equiv \delta^{(k)} z^{(k-1)}$$

$$(u^{(k)} \equiv W^{(k)} z^{(k-1)}, z^{(k)} = f(u^{(k)}))$$

$z^{(k)}$ can be calculated with 1 forwarding process. How about δ?

2. calculate $\delta^{(k)}$ **backward:**

$$\delta^{(k)} = \frac{\partial L}{\partial u^{(k)}} \frac{\partial u^{(k+1)}}{\partial u^{(k)}} = \delta^{(k+1)} W^{(k+1)} f'(u^{(k)})$$

(f': derivative of activation function)

Calc. cost of backward process is the same as that of 1 forwarding!

$$z^{(1)} \to u^{(2)}, z^{(2)} \to \cdots \to u^{(L-1)}, z^{(L-1)} \to u^{(L)}, z^{(L)}$$

$$\delta^{(1)} \leftarrow \delta^{(2)} \leftarrow \cdots \leftarrow \delta^{(L)}$$

Application to Physics

Application to physics

… especially related to phase transition & finite density system

- Akihiro Tanaka & A.Tomiya
  “Detection of Phase Transition via Convolutional Neural Networks”

- Kai Zhou,Gergely Endrődi and Long-Gang Pang
  “Regressive and generative neural networks for scalar field theory”

- Yuto Mori, Kouji Kashiwa and Akira Ohnishi
  “Application of a neural network to the sign problem via the path optimization method”

- Y.Fujimoto, K,Fukushima and K.Murase
  “Methodology study of machine learning for the neutron star equation of state”

A.Tanaka and A.Tomiya

“Detection of Phase Transition via Convolutional Neural Networks”

J. Phys. Soc. Jpn. 86, 063001 (2017)

Phase transition of 2d Ising model with ML

… 2nd ordr phase transition exists

A.Tanaka and A.Tomiya, Journal of the Physical Society of Japan 86 (6), 063001, 2017. 41, 2017.

Structure of neural network

$$
\left[\begin{array}{l}
\mathcal{I} = \left\{ \{\sigma_{xy}\} \,\middle|\, \text{Ising config on } L \times L \text{ lattice.} \right\} \\
\downarrow \left\{ \begin{array}{l} \text{Convolution}_{[N_f^2\text{-filter, } (s,s)\text{-stride, } C\text{-channels}]} \\ \text{ReLU or ELU activation} \\ \text{Flatten} \end{array} \right. \\
\mathbb{R}^{L^2/s^2 \times C} \\
\downarrow \left\{ \begin{array}{l} \text{Fully connected} \\ \text{Softmax} \end{array} \right. \\
\mathbb{R}^N = \mathcal{O}
\end{array}\right. \tag{1}
$$

F^a_{ij} in convolution, $\left[\begin{array}{l} a = 1, \ldots, C \\ i, j = 1, \ldots, N_f \end{array}\right]$ W^m_I in fully connected layer. $\left[\begin{array}{l} m = 1, \ldots, L^2/s^2 \times C \\ I = 1, \ldots N \end{array}\right]$

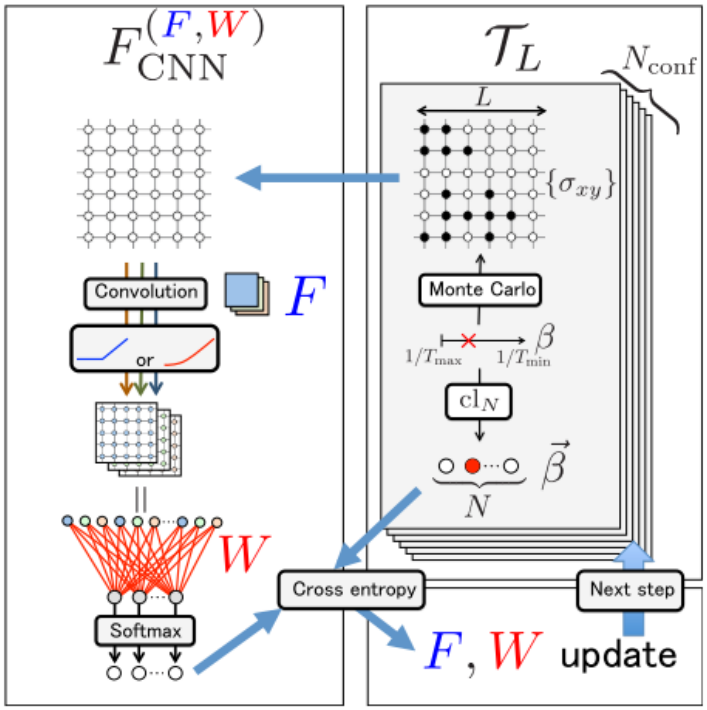


FIG. 2: Whole picture of our model. The data $(\{\sigma_{xy}\}, \vec{\beta}) \in \mathcal{T}_L$ for (8) is picked randomly in each step of SGD (A5).

Input and training data

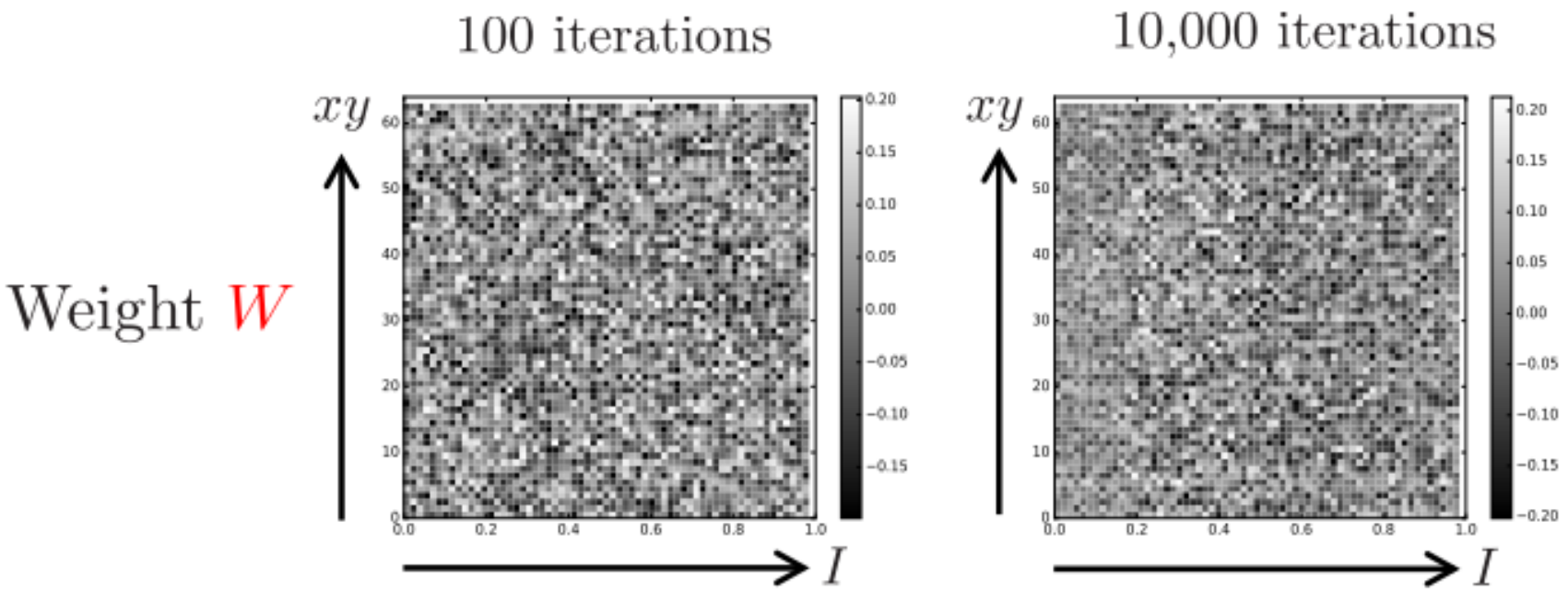
$$\left\{ \left(\{\sigma_{xy}^{(n)}\}, \beta_n \right) \Big| \frac{1}{\beta_n} = T_{\min} + n\delta \right\}_{n=0,\ldots,(N_{\text{conf}}-1)}, \quad \text{cl}_N : \beta \to \vec{\beta} = \begin{cases} (1,0,\ldots,0,0) & \text{for } \beta < 0 \\ (0,1,\ldots,0,0) & \text{for } \beta \in [0, \frac{1}{N-2}) \\ \ldots & \\ (0,0,\ldots,1,0) & \text{for } \beta \in [\frac{N-3}{N-2}, 1) \\ (0,0,\ldots,0,1) & \text{for } 1 \leq \beta \end{cases} . \quad (6)$$

Loss function

$$E(\vec{\beta}^{\text{CNN}}, \vec{\beta}) = -\sum_{I=1}^{N} \beta_I \log \beta_I^{\text{CNN}}, \qquad \text{minimized}$$

2d Plot of Weight W (flatten feature map):

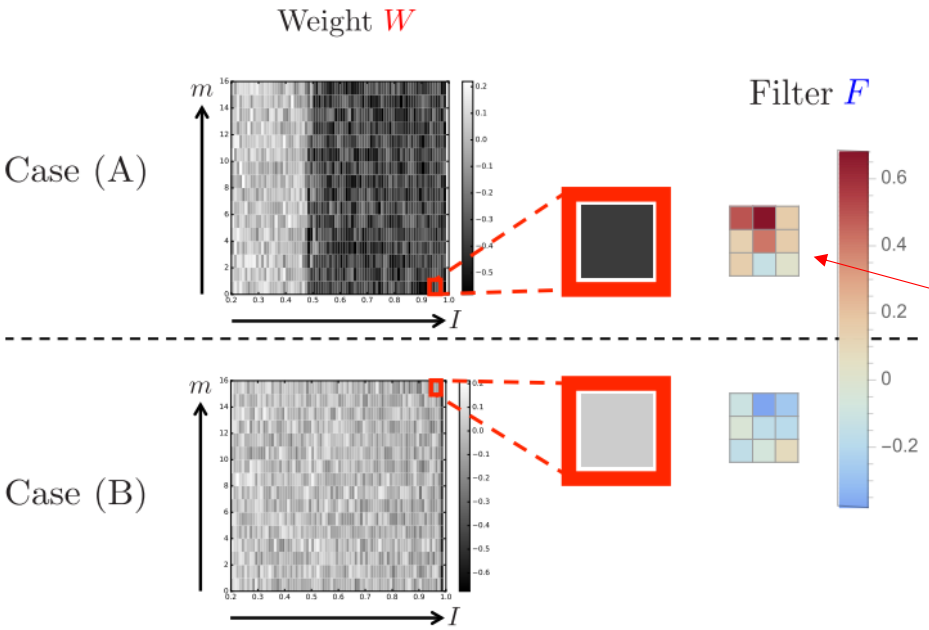
Without CNN: one hidden layer



I: Corresp. to Temperature

No feature of phase transition

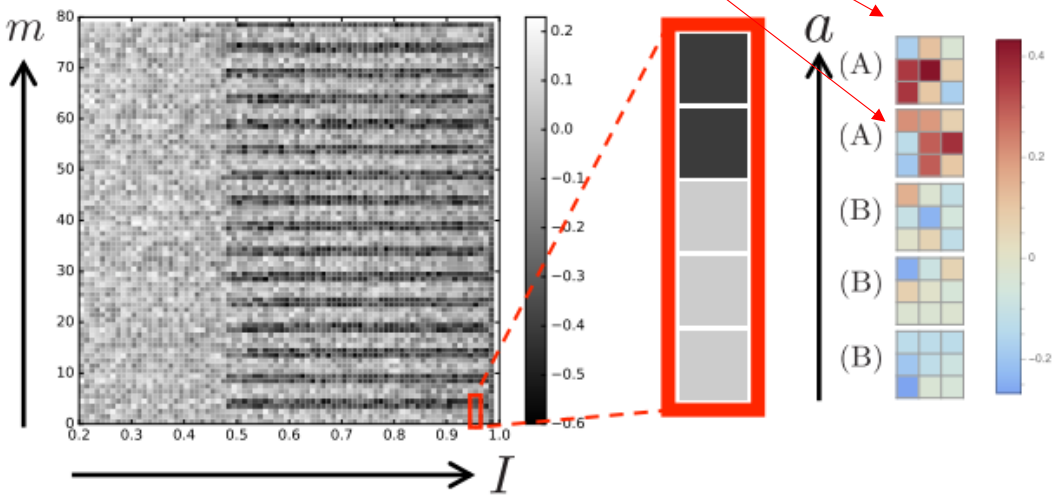
(I personally heard that many layers shows feature of phase trans. WITHOUT CNN)



Using CNN: one filter case (C=1)

The filter has a function of
“real space renormalization transformation”,
which gives average of spin = magnetization

Using CNN: 5 filter case (C=5)



Shows feature of phase transition!

“Order parameter” in neural network

- $W_{\mathrm{sum}}(\beta) = \sum W_I^m \mathrm{cl}_N^I(\beta),$
- $\tilde{W}_{\mathrm{sum}}(\beta) = a \tanh(c(\beta - \beta_{\mathrm{CNN}})) - b,$

Predicted by ML

…Motivated by Ising model in mean field approx.

W_{sum} is well fit by $\widetilde{W}_{sum}$

⇒W_{sum} can be regarded as order parameter!

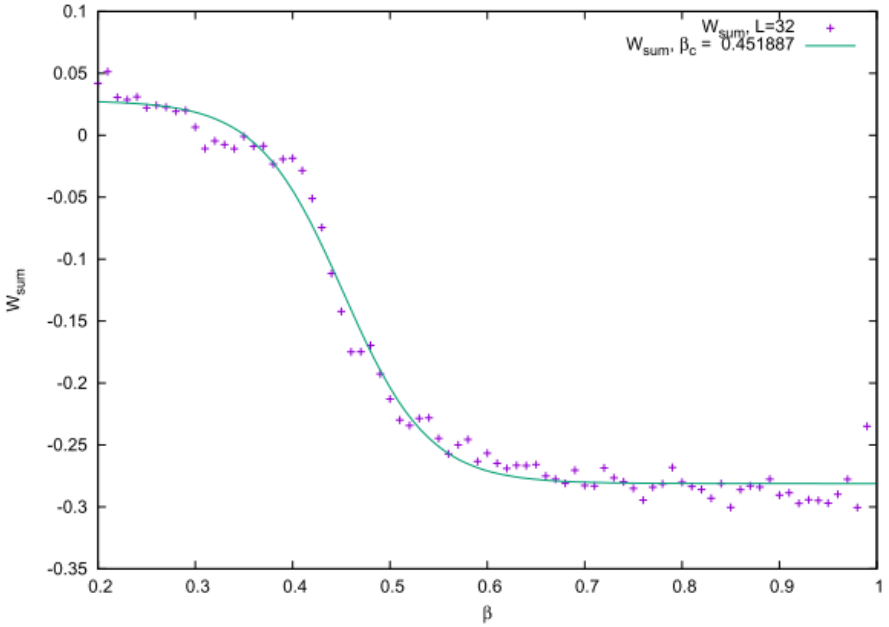


FIG. 7: $W_{\mathrm{sum}}(\beta)$ for $L = 32$. The horizontal axis is the inverse temperature β which is translated using (6).

Kai Zhou,Gergely Endrődi and Long-Gang Pang

“Regressive and generative neural networks for scalar field theory”

arXiv:1810.12879 [hep-lat]

Dual description of scalar field theory at finite density

-Original action: $\int_0^L dx_1 \int_0^{1/T} dx_2 [(D_\nu \phi)^* (D_\nu \phi) + m^2 \phi^* \phi + \lambda (\phi\phi)^2] \qquad D_\nu = \partial_\nu + i\mu\delta_{\nu,2}$

-Dual description: $Z = \sum_{\{k,l\}} \exp(-S^{\mathrm{lat}}[k,l]) = \sum_{\{k,l\}} \Pi_x Z^{k,l}(x)$

$$Z^{k,l}(x) = e^{\mu k_t(x)} W[s(k,l;x)] \delta[\nabla \cdot k(x)] \Pi_\nu A[k_\nu(x), l_\nu(x)]$$

$$W[s] = \int_0^\infty dr r^{s+1} e^{-(4+m^2)^2 - \lambda r^4}$$

$$s(k,l;x) = \sum_\nu [\,|k_\nu(x)]\,| + |k_\nu(x-\hat{\nu})| + 2(l_\nu(x) + l_\nu(x-\hat{\nu}))]$$

$$\nabla \cdot k(x) = \sum_\nu [k_\nu(x) - k_\nu(x-\hat{\nu})]$$

$$A[k_\nu(x), l_\nu(x)] = \frac{1}{(l_\nu(x) + |k_\nu(x)|)! l_\nu(x)!}$$

… this expression is real, no sign factor

Purpose of the study

- Detect phases of complex scalar field theory at finite temperature and density by using Deep Learning from bare configurations
- Predict the various observables using DL from bare configurations

Distribution of probability of condensation P

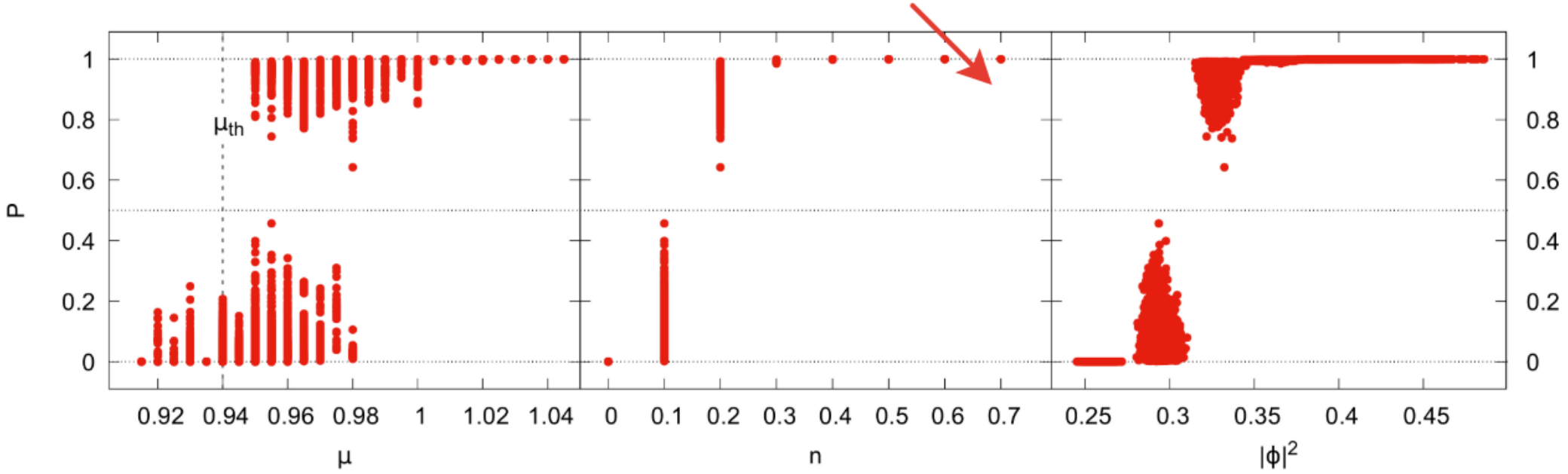
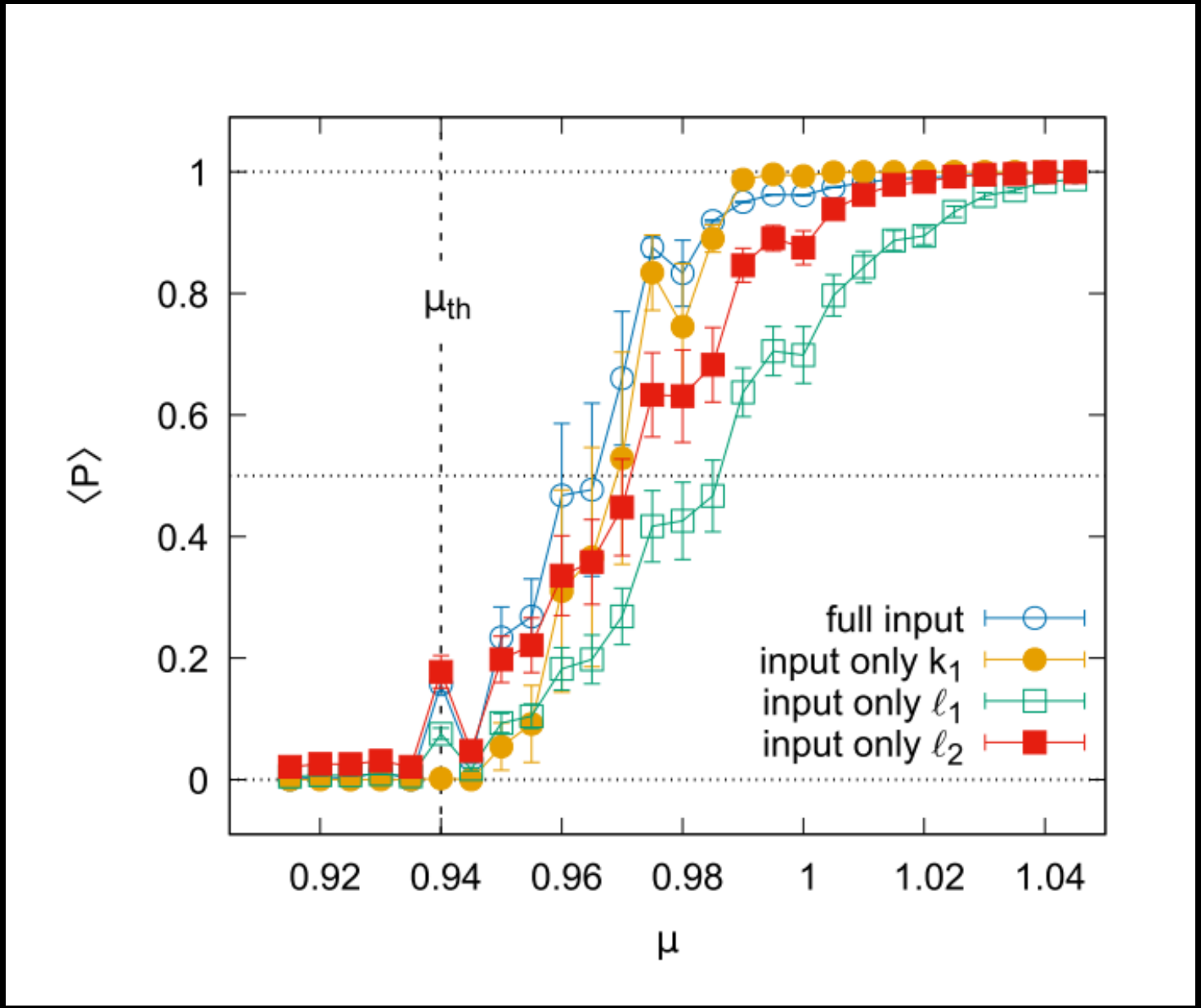


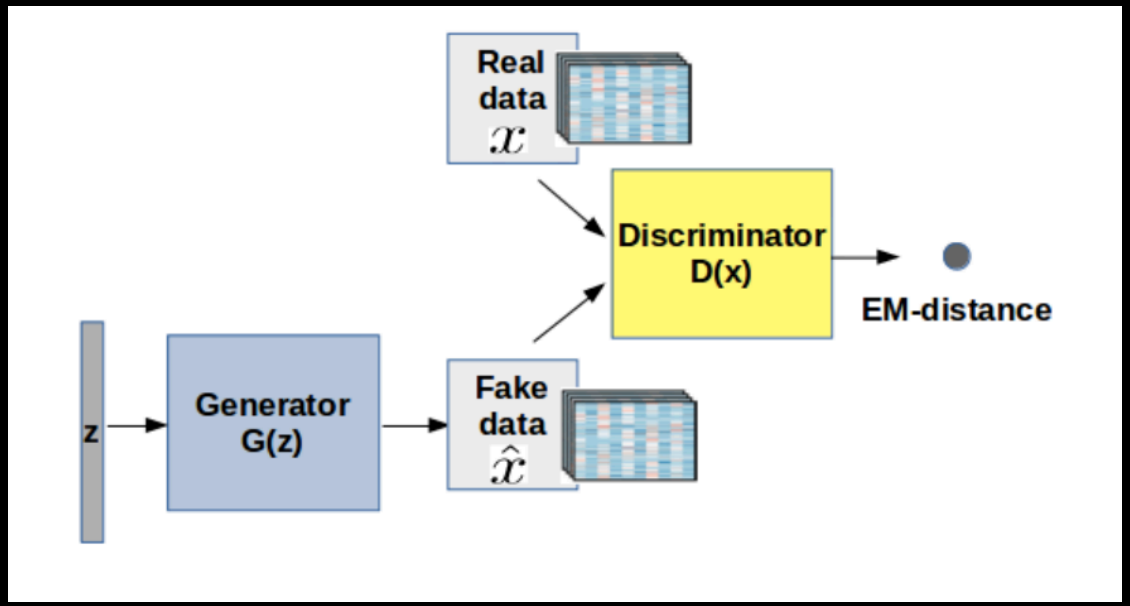
FIG. 3. The network predicted condensation probability P as a function of the chemical potential (left), the particle number density (middle) and the squared field (right). The dashed vertical line indicates the threshold chemical potential $\mu_{\rm th}$. Each point in the plot represents one configuration. As pointed out in Sec. II, n only assumes values that are integer multiples of 0.1, visible in the middle panel.

Probability of condensation from ML



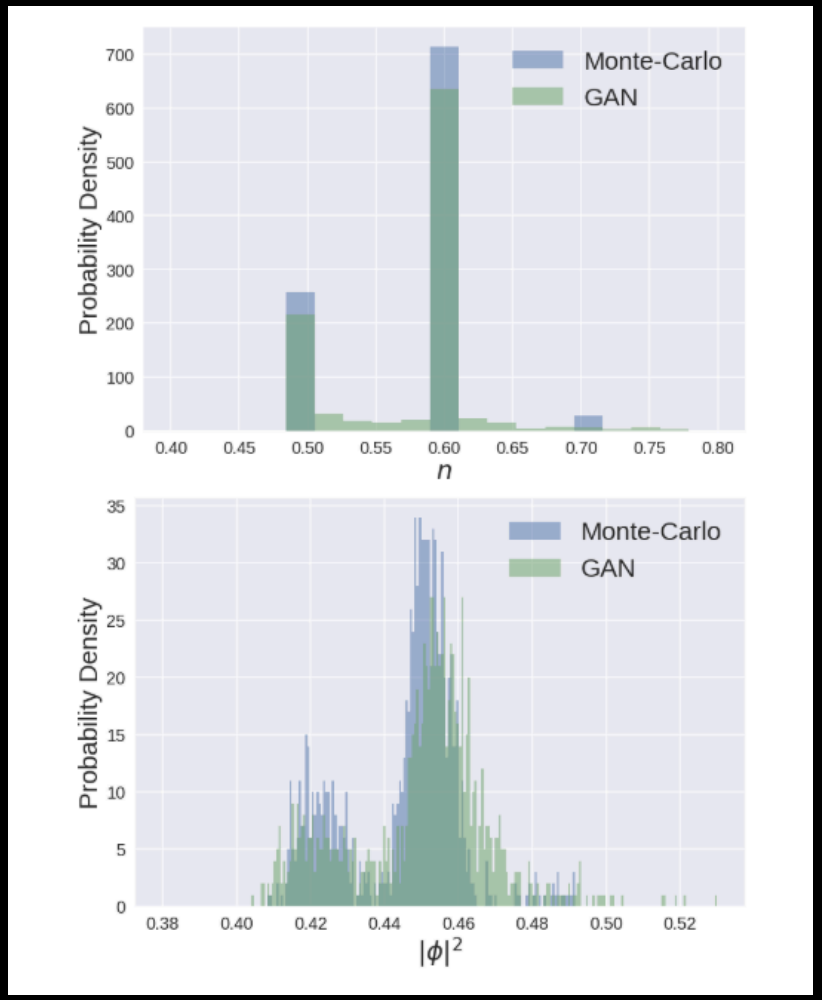
-They investigated the correlations btw observables & k_1, k_2, l_1, l_2

Generator of configurations



Generative Adversarial Network (GAN)
input : noise z
output: configurations

Competition between generator and discriminator
-> generator makes high quality fake data



Not only average, but also distribution are well reproduced

Y.Mori, K.Kashiwa and A.Ohnishi

“Application of a neural network to the sign problem via the path optimization method”

Prog. Theor. Exp. Phys. 2018, 023B04

- **Lefschetz Thimble method**… a method to circumvent sign problem
  Find a complex path on which imaginary part of integrand does not oscillate

$$\frac{d}{dt} z_i = \frac{\partial \bar{S}[\bar{z}]}{\partial \bar{z}_i}$$

t→∞: $\left. \frac{\partial S[z]}{\partial \bar{z}_i} \right|_{z=z_\sigma} = 0 \quad \cdots \mathcal{J}_\sigma, \mathcal{K}_\sigma$

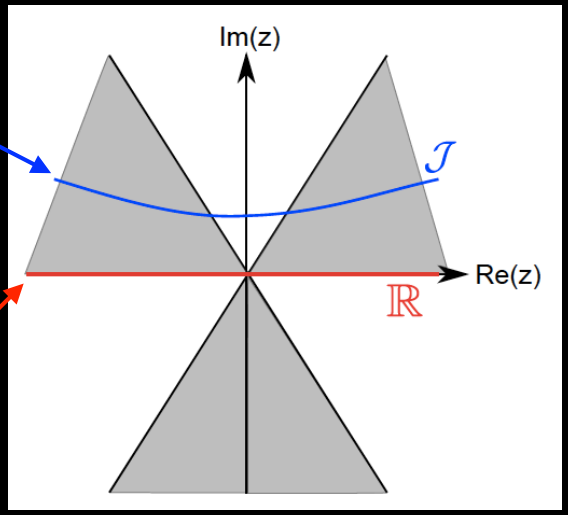
Thimbles
steepest descent (stable)
& ascent path (unstable)

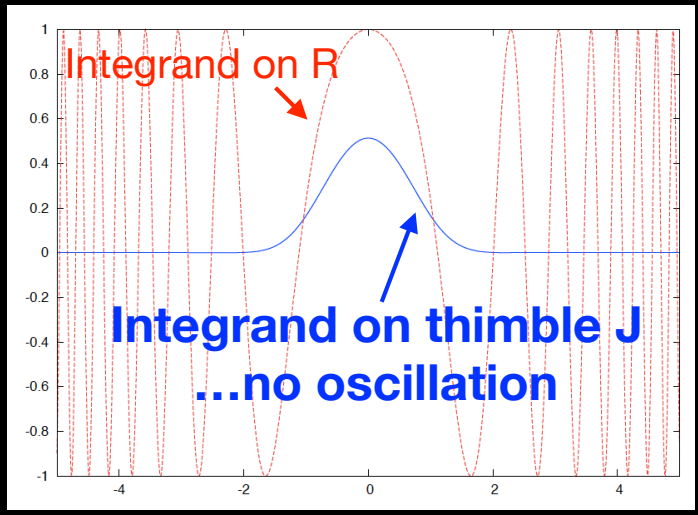
ex) Airy function

$$\mathrm{Ai}(a) = \int_R \frac{dx}{2\pi} \exp i\left(\frac{x^3}{3} + ax\right) = \sum_\sigma n_\sigma \int_{\mathcal{J}_\sigma} \frac{dz}{2\pi} \exp i\left(\frac{z^3}{3} + az\right)$$

Thimble

Original contour





2 thimbles

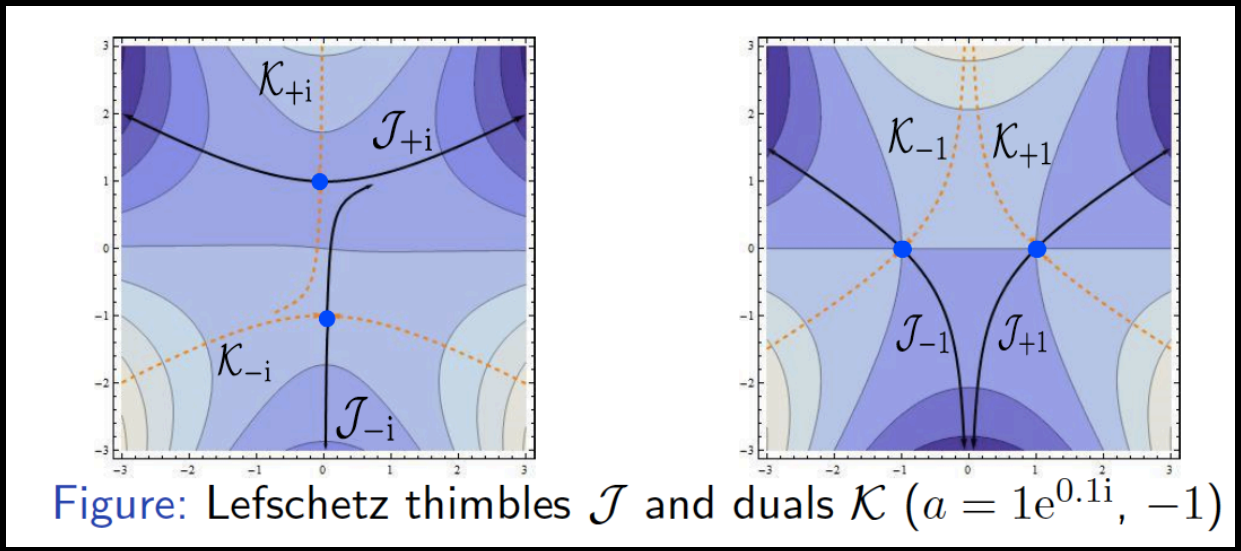


Figure: Lefschetz thimbles $\mathcal{J}$ and duals $\mathcal{K}$ $(a = 1\mathrm{e}^{0.1\mathrm{i}}, -1)$

Yuya Tanizaki
XQCD 2016

Purpose of the study

- Find “thimble” by using Machine Learning with feed-foward neural network

  path omtimization method

- examine the usefulness of the method in 2D complex $\lambda\phi^4$ theory at finite chemical potetial …where sign problem exists

Try to find the complexified path $z(t)$ which minimize the following loss function $\mathscr{F}(z(t))$ (=cost function in their paper)

$$\begin{aligned}\mathscr{F} &= \frac{1}{2}\int d^n t \,|\, e^{i\theta(t)} - e^{i\theta_0} \,|^2 \times |\, J(t) e^{-S(z(t))} \,| \\ &= \int d^n t \,|\, J(t) e^{-S(z(t))} \,| - \left| \int d^n J(t) e^{-S(z(t))} \right| \\ &= |Z| \left[\left| e^{i\theta}_{\mathrm{pq}} \right|^{-1} - 1 \right]\end{aligned}$$

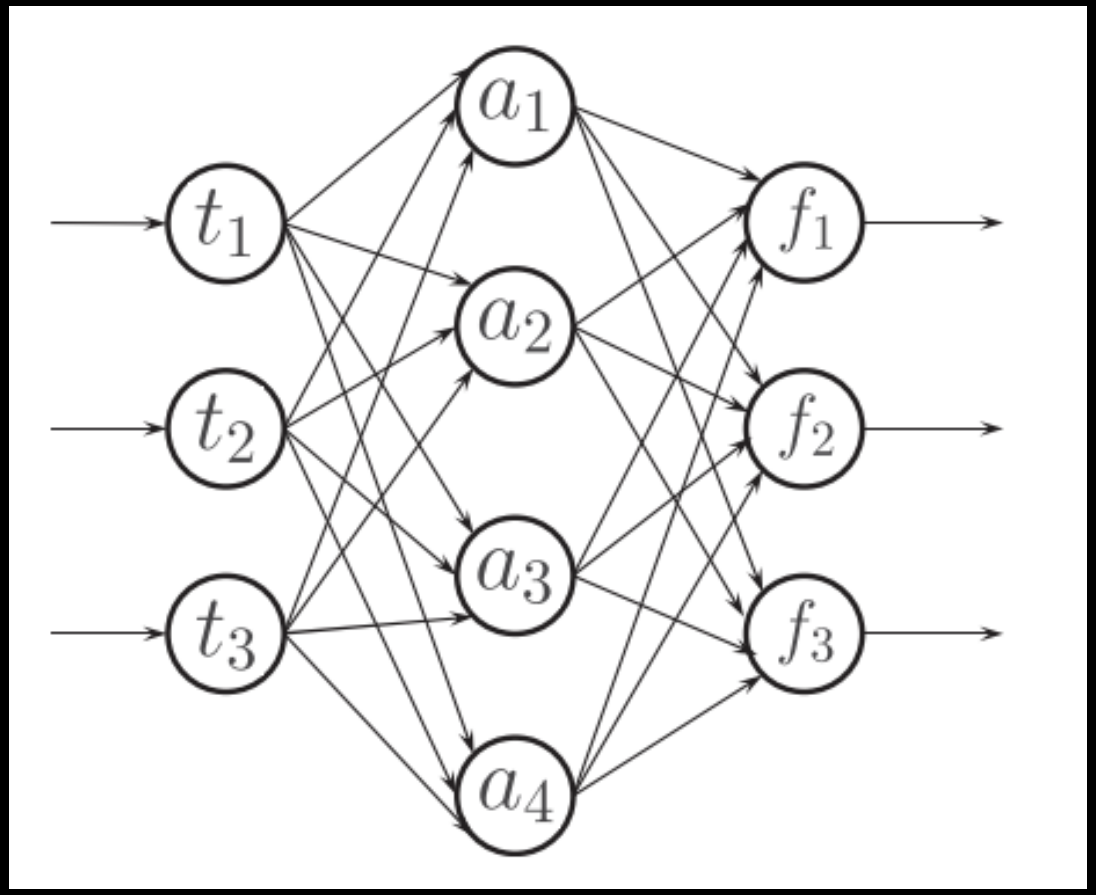
$$J(t) = \det(\partial z_i / \partial t_j) \text{ :Jacobian} \qquad Z = \int d^n t J(t) e^{-S(z(t)},$$

$$\theta(t) = \arg(J(t) e^{-S(z(t))}), \theta_0 = \arg(Z),$$

(pq…phase quenched: $\langle \mathcal{O} \rangle_{\mathrm{pq}} = \dfrac{\int d^n t \mathcal{O}(z(t)) \,|\, J(t) e^{-S(z(t))} \,|}{\int d^n t \,|\, J(t) e^{-S(z(t))} \,|} \qquad \langle \mathcal{O} \rangle = \dfrac{\langle \mathcal{O} e^{i\theta} \rangle_{\mathrm{pq}}}{\langle e^{i\theta} \rangle_{\mathrm{pq}}}$ **)**

Parametrization of path

$$z_i(t) = t_i + i(\alpha_i f_i(t) + \beta_i)$$



One hidden layer is enough

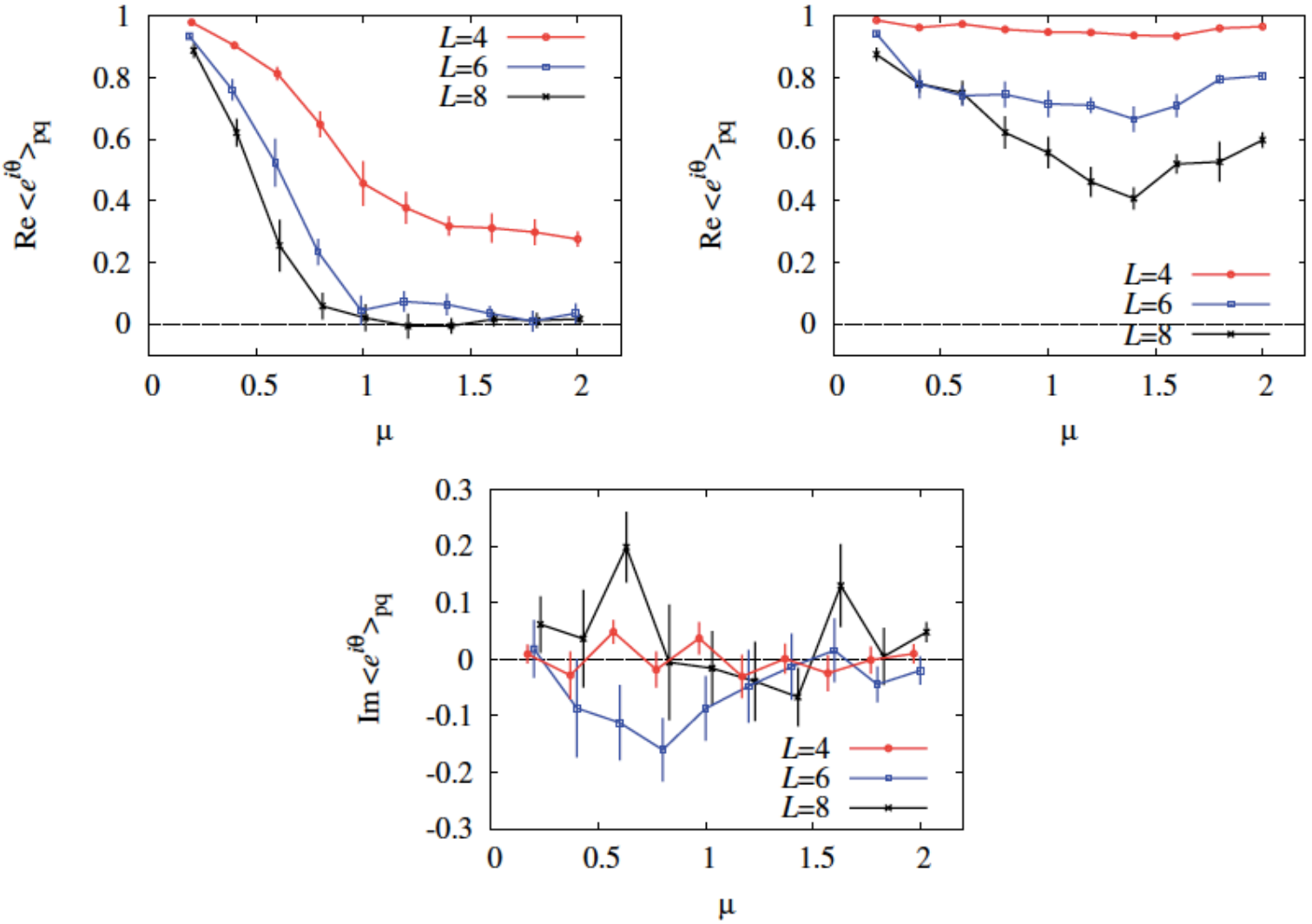


Fig. 2. The top left and top right panels show the real part of the average phase factor without and with the optimization as a function of μ. The bottom panel shows the imaginary part of the average phase factor with the optimization. Circles, squares, and crosses are results for $L = 4$, 6, and 8, respectively.

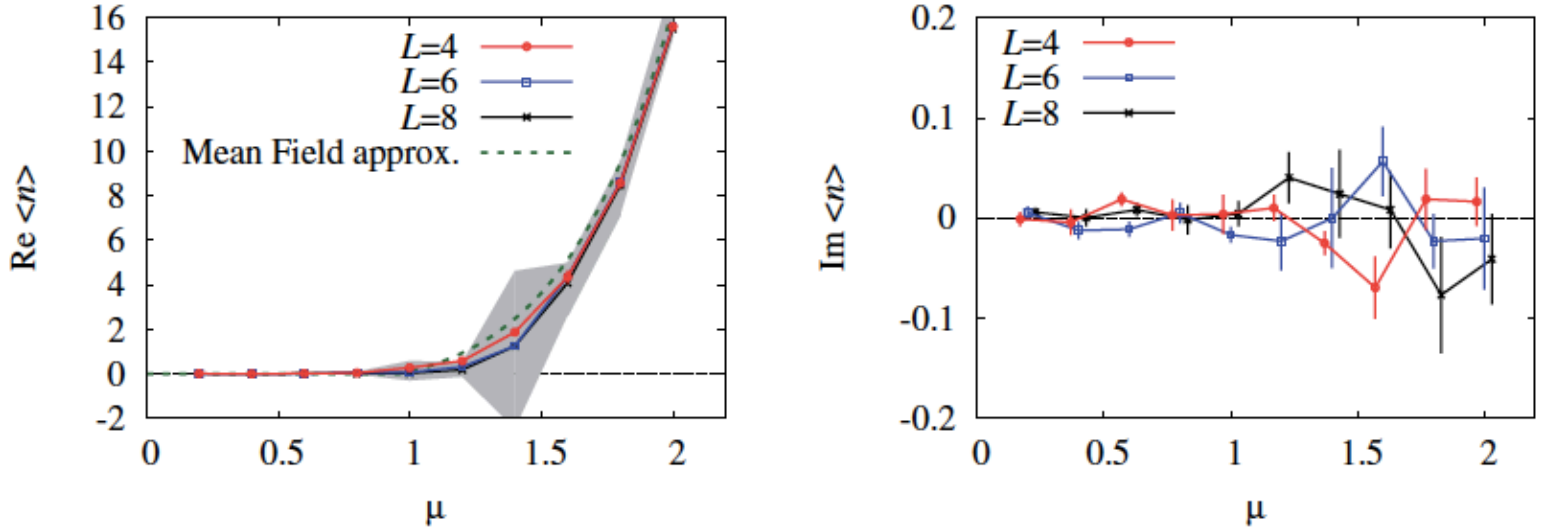


Fig. 3. The expectation value of the number density in the Monte Carlo calculations (solid lines with symbols) and in the mean field approximation (dashed line). The shaded area shows the expectation value without the path optimization method at $L = 8$.

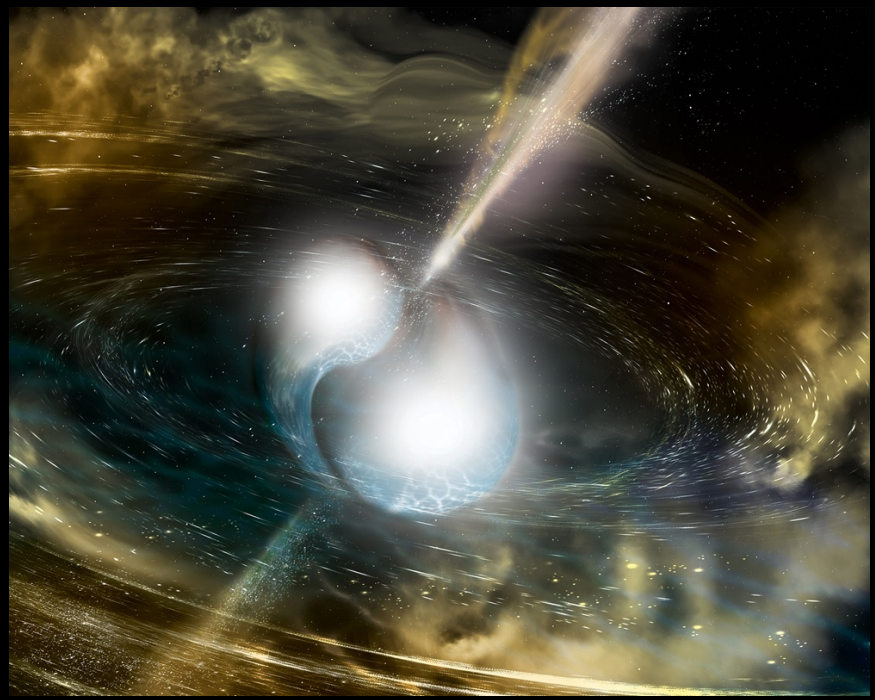
Y.Fujimoto, K,Fukushima and K.Murase

“Methodology study of machine learning for the neutron star equation of state”

Phys. Rev. D98, 023019 (2018)

Neutron star

- **Radius** about 10km
- **Mass** roughly 1 solar mass (1.4-2.0(?))
- Recently **gravitational wave**
  from collision of two neutron stars is observed

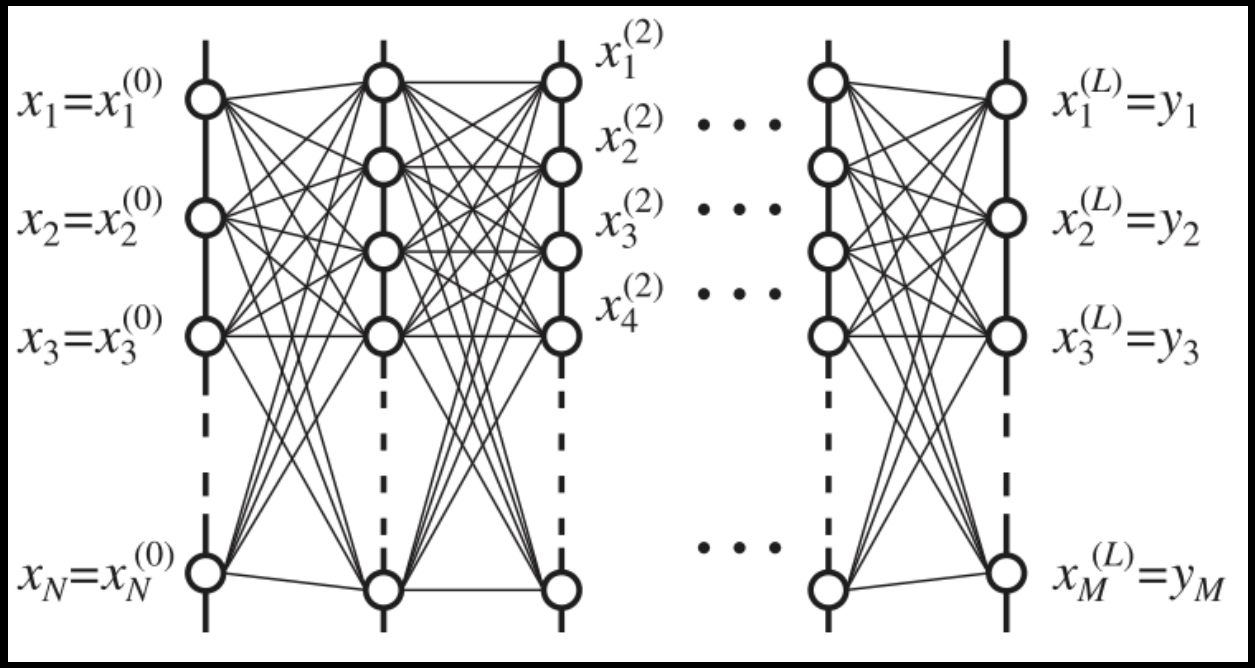



Size of neutron star
Google マップを検索する
移動時間、交通状況、付近の場所を表示
ウラジオストク
Владивосток
ルースキー
Русский
ルースキー島
Русский
航空写真
Google
地図データ ©2019 Google ロシア 利用規約 フィードバックの送信 10 km

Purpose of the study

- Discuss a methodology to construct Equation of State (EoS) with neural network from observational data, the relation of M-R (mass-radius relation of NS)
  - Bayesian analysis is one of the effective methodology based on a certain prior distribution of EoS
    … have to investigate dependence of prior distribution
  - This study is a complementary to Bayesian analysis

Feed Forward Neural Network



[Input]
"Observational" data

(M_i, R_i)

(i=1-15)

NN

[Output]
Sound velocity

$$c_{s^2,i} = \frac{dp_i}{d\rho_i}$$

(p, ρ) relation is fixed

→ **EoS is determined**

Making input data

- **Making EoS:**
  - **[0, ρ_0] :** conventional nuclear EoS
  - **[ρ_0, $8\rho_0$] :**
    - equally partitioned into 5 segments in logarithmic scale
    - randomly assign c_s to the five segments according to the uniform distribution $0.02<c_s<0.98$
    - determine pressure p at the segment boundary
    - interpolation is performed assuming $p \propto \rho^{\Gamma}$

- **Solve Tolman-Oppenheimer-Volkov equation (TOV eq.)**

$$\frac{dP}{dr} = -\frac{Gm}{r^2}\rho\left(1+\frac{P}{\rho c^2}\right)\left(1+\frac{4\pi r^3 P}{mc^2}\right)\left(1-\frac{2Gm}{rc^2}\right)^{-1}$$

$$\frac{dm}{dr} = 4\pi r^2 \rho$$ **(M,R) relation is obtained**

- **Adding errorbars (normal distribution)**
- **Loss function is defined by**

$$L[f_{NN}] = \left\langle \sum_{i=1}^{N_L}\left(\log\frac{c_{s,i}^2}{f_{NN,i}(\{M_j,R_j\})}\right)^2 \right\rangle$$

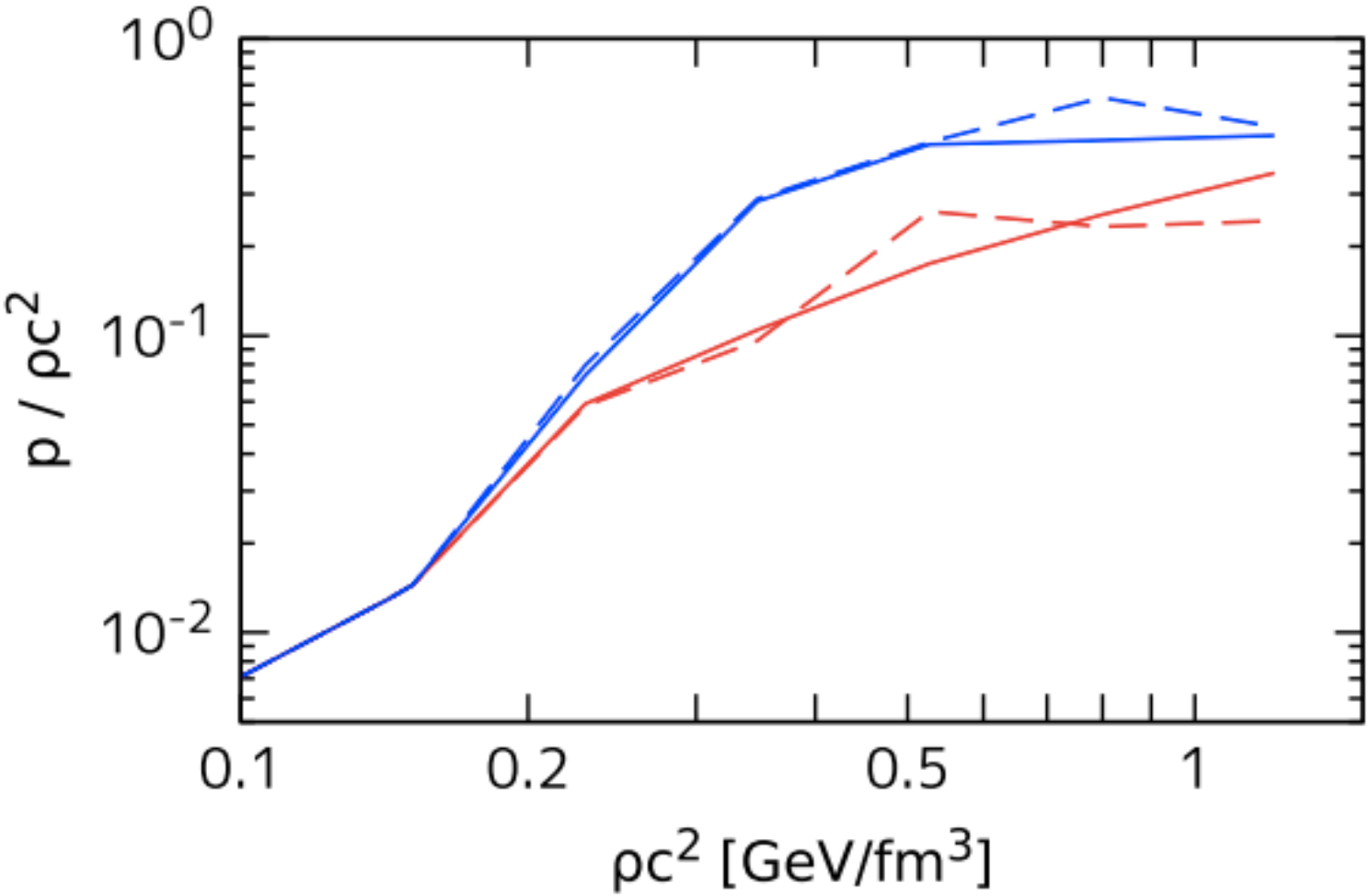


FIG. 3. Two examples of the randomly generated EoSs (dashed lines) and the machine learning outputs (solid lines) reconstructed from one observation of 15 M-R points [see Fig. 4 for actual (M_i, R_i)].

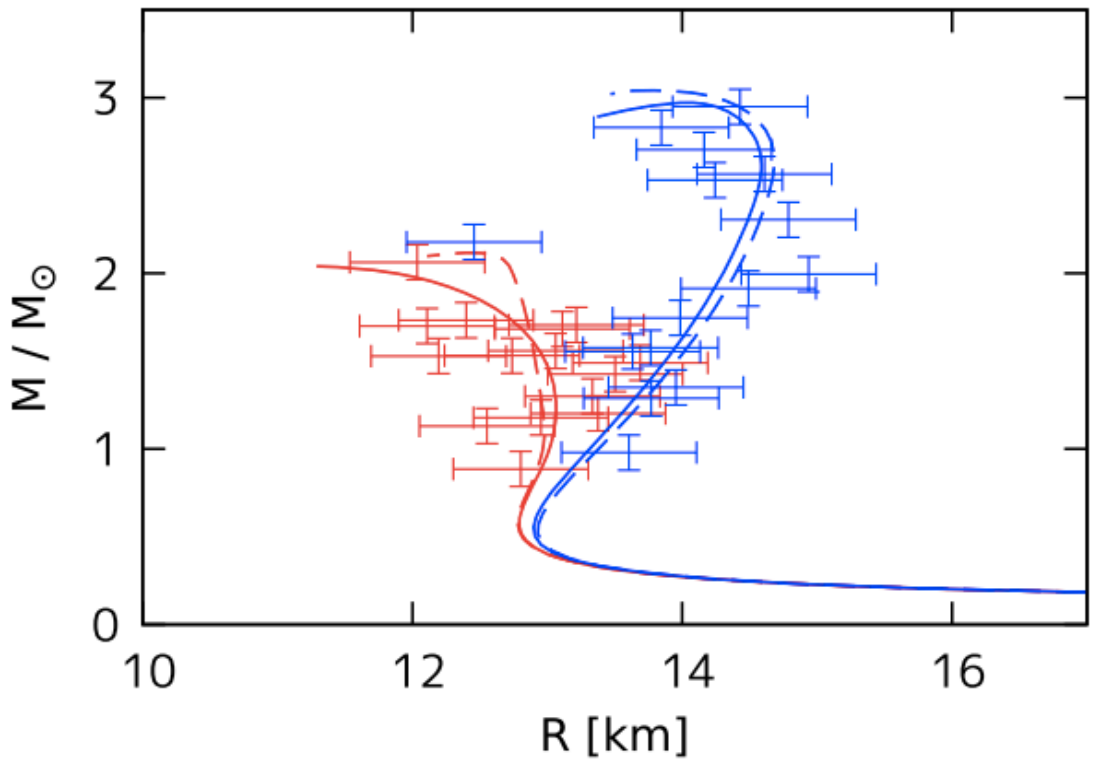


FIG. 4. Randomly sampled 15 data points and the *M*-*R* relations with the reconstructed EoS (solid lines) and the original EoS (dashed lines). The red and blue colors correspond to two EoSs shown with the same color in Fig. 3.

TABLE II. Root mean square of radius deviations for fixed masses.

Mass ($M_\odot$)	0.6	0.8	1.0	1.2	1.4	1.6	1.8
RMS (km)	0.16	0.12	0.10	0.099	0.11	0.11	0.12

Activity of our group on Machine Learning

- Sine-Gordon model in 1+1dim.

$$S[\phi] = \frac{1}{t}\int d^2x \left\{ \frac{1}{2}[\partial_\mu \phi(x)]^2 - g\cos\phi(x) \right\}$$

Sine-Gordon system has Berezinskii-Kosterlitz-Thouless phase transition
… thermodynamic quantities are smooth, but the correlation function has power damping under the phase transition temperature

cf) XY model in 2d does NOT have the phase of long-range order due to strong fluctuation in low dimension (Mermin-Wagner theorem)

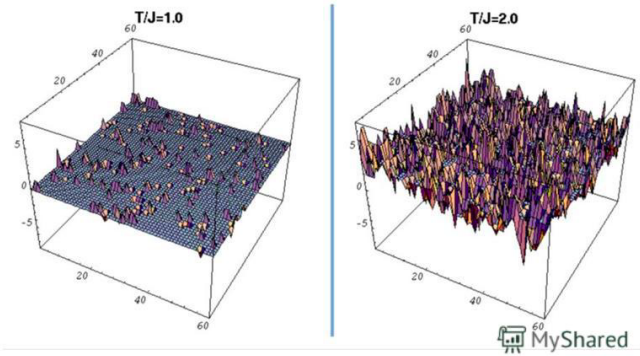
Roughening transition in lattice Sine-Gordon model

$$Z = \sum_{\{n(r)\}} \exp\left(-\sum_{r,\mu} \frac{1}{2\beta J}(n(r) - n(r-\mu))^2 \right) \quad \ldots(\bigstar)$$

… describing “roughness” of surface of crystal

- this is a dual theory of XY model
  (β & J are the parameters of
  inverse temperature & coupling in XY model)
- In rarefied gas limit, (★) becomes Sine-Gordon model

⇒ Roughening transition is related to BKT transition



Phase of roughening transition:

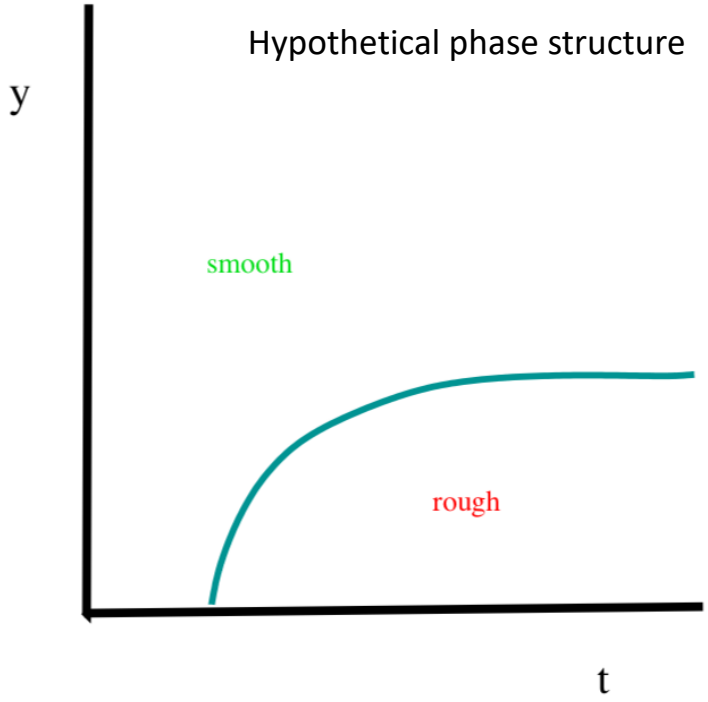
$$S[\phi] = \frac{1}{t}\int d^2x \left\{\frac{1}{2}[\partial_\mu \phi(x)]^2 - g\cos\phi(x)\right\}$$

$$y = g/2t$$

surface thickness:

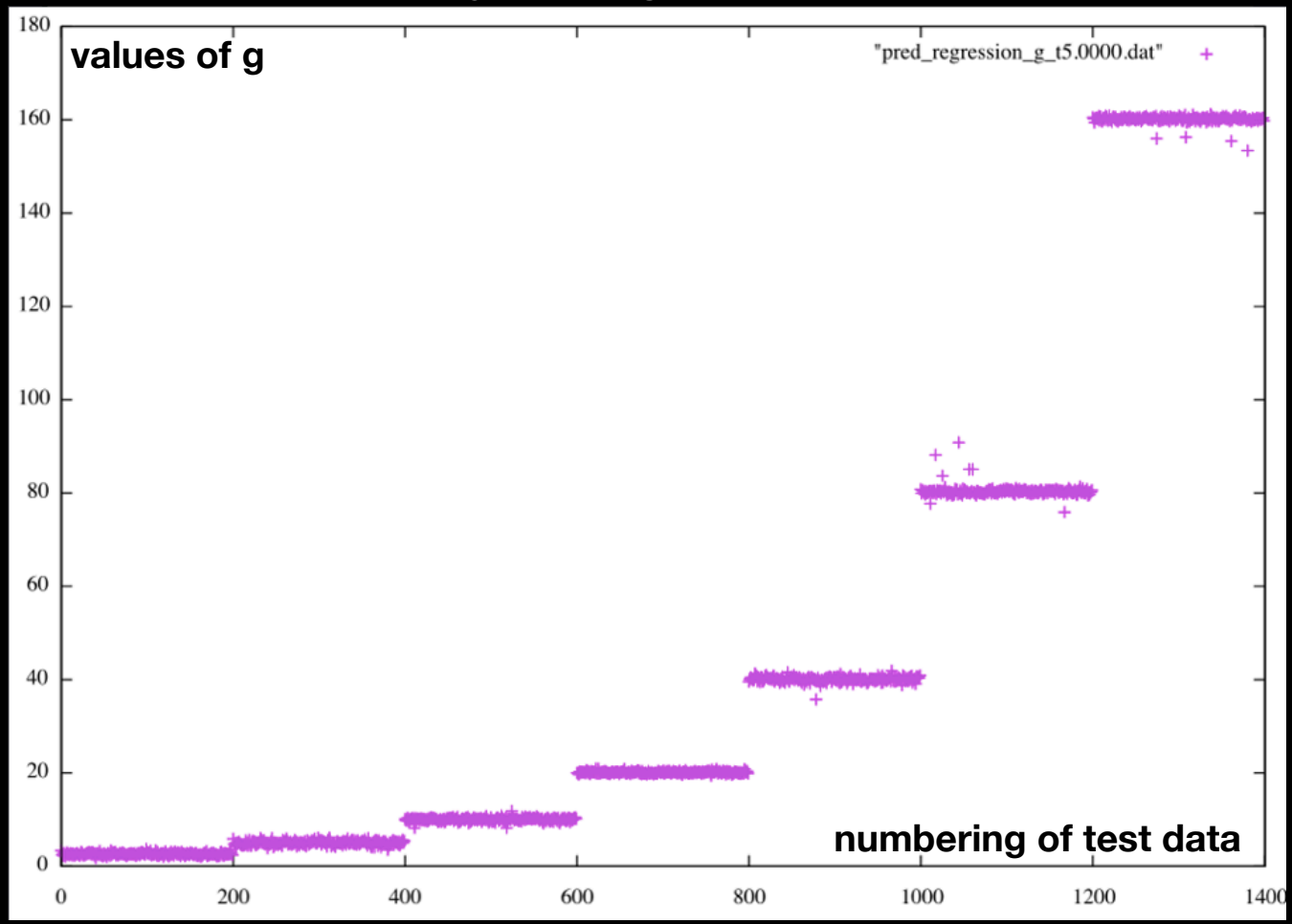
$$\sigma^2 = \frac{1}{V^2}\sum_{x,y}\langle(\phi(x)-\phi(y))^2\rangle$$

$$\sigma^2 \simeq \begin{cases} \frac{t}{\pi}\ln L + \text{const.}, & t > t_c \\ \text{const.} & t < t_c \end{cases}$$



cx) Inference of g (t=5) from bare configurations

- g=2.5, 5, 10, 20, 40, 80, 160, Nt x Nx=40 x 40
- Network: CNN & 2 layers, regression



- Good agreement
- What is the feature to infer the values of g?

  Can we extract “order parameter” of SG system?

cf) Tanaka & Tomiya … made o.p. from parameters of NN

Activity of our group on Machine Learning

- SU(2) Yang-Mills…talk by N.Gerasimenyuk
- Speed Radar (Speed Gun) on smartphone
  …talk by S.Lyubimov

Enjoy their talks!
Thank you for your attention!