Snake net with a neural network for detecting multiple phases in the phase diagram

Xiaodong Sun,[1] Huijiong Yang,[2] Nan Wu,[1,3] T. C. Scott,[4] Jie Zhang,[1] and Wanzhou Zhang[1,5,6,*]

[1]*College of Physics and Optoelectronics, Taiyuan University of Technology, Shanxi 030024, China*
[2]*College of Data Science, Taiyuan University of Technology, Shanxi 030024, China*
[3]*School for Physical Sciences, University of Science and Technology of China, Hefei 230026, China*
[4]*Institut für Physikalische Chemie, RWTH Aachen University, Aachen 52056, Germany*
[5]*Hefei National Laboratory for Physical Sciences at the Microscale and Department of Modern Physics, University of Science and Technology of China, Hefei 230026, China*
[6]*CAS Key Laboratory for Theoretical Physics, Institute of Theoretical Physics, Chinese Academy of Sciences, Beijing 100190, China*

(Received 10 September 2022; accepted 26 April 2023; published 22 June 2023)

Unsupervised machine learning applied to the study of phase transitions is an ongoing and interesting research direction. The active contour model, also called the snake model, was initially proposed for target contour extraction in two-dimensional images. In order to obtain a physical phase diagram, the snake model with an artificial neural network is applied in an unsupervised learning way by the authors of [Phys. Rev. Lett. **120**, 176401 (2018)]. It guesses the phase boundary as an initial snake and then drives the snake to convergence with forces estimated by the artificial neural network. In this work we extend this unsupervised learning method with one contour to a snake net with multiple contours for the purpose of obtaining several phase boundaries in a phase diagram. For the classical Blume-Capel model, the phase diagram containing three and four phases is obtained. Moreover, a balloon force is introduced, which helps the snake to leave a wrong initial position and thus may allow for greater freedom in the initialization of the snake. Our method is helpful in determining the phase diagram with multiple phases using just snapshots of configurations from cold atoms or other experiments without knowledge of the phases.

DOI: 10.1103/PhysRevE.107.065303

I. INTRODUCTION

Exploring the phases and phase diagrams of the matter is a long-standing task in physics [1]. Commonly found in life, such as water, there are three phases: solid-liquid-gas. In addition, states of matter exist at very low temperatures such as superconductors [2], superfluids [3], and, at very high temperatures, plasma states [4]. The study of the distribution of these phases in the phase diagram and the phase transition boundaries between them is very helpful for one to understand the natural world.

With the development of machine learning methods and their integration into various disciplines, machine learning methods are used to study the phases of matter [1,5]. Unsupervised machine learning does not require real labels for the data and is therefore more appreciated by researchers when studying unknown questions. Commonly used unsupervised machine learning methods are principal component analysis [6–8], t-distributed stochastic neighbor embedding [9,10], and diffusion maps [11–14]. Recent work also proposed a quantum algorithm to achieve a quantum computational speedup of diffusion maps [15].

In 2018 a simple snake model with a neural network was proposed to search for the phase boundaries between the two phases in the two-dimensional parameter space [16]. The snake model, also known as the active contour model, was originally proposed by Kass *et al.* in the 1980s for target contour extraction in two-dimensional images in the field of computer vision [17]. The method is useful because it replaces image processing with the active contour energy minimization problem. The movement of the active contour is driven by the well-defined image force $\mathbf{F}_{\text{img}}$.

For detecting phase transitions rather than images processing, the authors of Ref. [16] proposed the neural network discriminative cooperative network (DCN), which consists of a learner network $\mathcal{N}$ and a guesser network $\mathcal{G}$, and the two networks work in cooperation with each other. The DCN replaces the image force $\mathbf{F}_{\text{img}}$ of the snake model for images with the derivative of the cross-entropy cost function $\mathcal{S}$ between the outputs of $\mathcal{G}$ and $\mathcal{N}$ with respect to the position λ_g of snakes, i.e., $-\frac{\partial \mathcal{S}}{\partial \lambda_g}$. Using the method developed, the boundary between the superfluid and insulating phases can be obtained [16]. However, the simple snake model, i.e., only one contour with a DCN, still encounters challenges in the study of phase diagrams, especially for more than two distinct phases [16].

Here we propose to combine the snake net (SN) and the DCN together to find multiple boundaries between phases. The topology-preserving SN was developed by Butenuth *et al.* in 2012 for image contour extraction [18,19]. The SN has multiple snakes connected by common nodes. By updating the positions of the snakes, the SN model can realize the contours of images containing multiple colors, and the images can be cells, roads, and so on. In our SN-DCN method, the snakes in the SN are expected to converge to the real phase boundaries.

[*] Corresponding author: zhangwanzhou@tyut.edu.cn

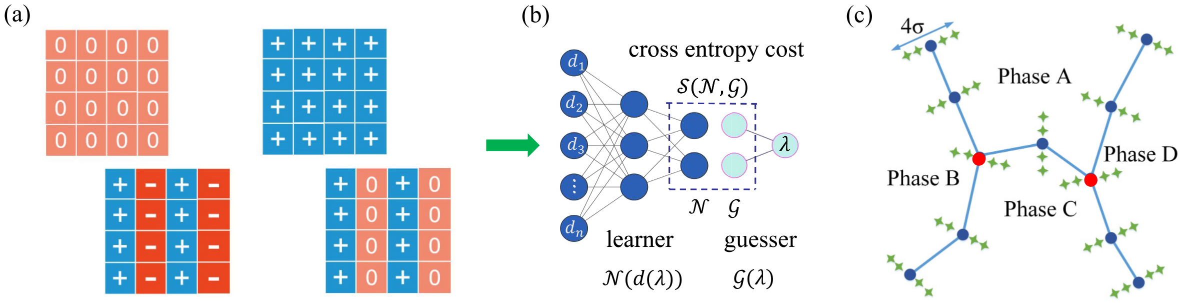

FIG. 1. The basic idea of using the SN-DCN to locate the phase boundaries. (a) The typical input configurations of the BC model. (b) The DCN tool processes the data. The DCN contains a learner network $\mathcal{N}$ and a guesser network $\mathcal{G}$. (c) The phase boundaries detected by the SN-DCN. Blue circles mark the nodes of the snake model, and the red circle indicates a special kind of node where the three snakes intersect. The unit width of the snake is indicated by σ (green diamond) and its value is dynamic. The line segments between nodes indicate the phase boundaries to be detected.

On the other hand, if the initial position of the snake is far from the true boundary, in such a case the initial snake converges very slowly and does not even get to the correct position because the snake does not feel enough force. Therefore we introduce the balloon force (BF) [20] to the snake model, originally proposed by Cohen *et al.* to help locate contours over image processing. The BF reduces the problematic requirement that the initial snake must be set near the true phase transition boundary. The BF can also speed up the movement of the snake and help decrease the number of iterative steps.

The basic idea is shown in Fig. 1. One first obtains the dataset of the system, such as the configuration of the spin systems. Then one initializes the position of the SN according to the necessary prior knowledge and applies a DCN containing $\mathcal{G}$ and $\mathcal{N}$ to drive the initial snakes. Eventually, the snakes locate at the phase boundaries, as shown in Fig. 1(c).

This work builds an SN-DCN to obtain the phase boundaries of the physical systems of interest. Here there are two versions of the SN-DCN. The simplest SN has only three snakes with a common node to get three-phase boundaries. The other extended version of the SN, with five snakes, yields five phase boundaries between the phases. The additional BF introduced here can also be selectively added to the snake net. We also test the hyperparameters of simple snake models with the DCN and find that the BF acceleration is most effective.

The outline of this paper is as follows. In Sec. II we present the SN-DCN method. In Sec. III the method is applied to the Blume-Capel (BC) model with three phases and four phases, respectively. Different topologies of the SN are also discussed. In Sec. IV the BF-SN with the DCN are presented and applied to the phase diagram for the quantum Bose-Hubbard (BH) model and the BC model. The conclusion and discussion are presented in Sec. V. In Appendix A a detailed description of the snake model and the iteration matrix of our models are presented.

II. THE SN-DCN METHOD

A. Input data

The input data type depends on the specific model and parameter range. For the BC model [21], the input data are the spin configurations obtained from Metropolis Monte Carlo simulations [22], as shown in Fig. 1(a). The symbols "+," "−," and "0" correspond to the values taken by the spins. The stripelike pattern corresponds to the states $\substack{+-\\+-}$ or $\substack{+0\\+0}$. The uniform pattern corresponds to the states $\substack{++\\++}$ and $\substack{00\\00}$, respectively. The data comes from a lattice with the size of 16 × 16. For the Bose-Hubbard model [23], the input data are wave functions in the mean-field framework and are expressed as the square of the expansion coefficients.

In real simulations, the physical parameters can be temperature or different types of interaction labeled by λ, which usually has two components in the physical parameter space (λ^x, λ^y).

As shown in Fig. 2, each snake in the net has 50 nodes marked by blue circles, i.e., λ_g^i, $i = 1, 2, \ldots, 50$. They also represent the position of the guessed phase transition points. The sampled parameters are denoted as λ_j, $j = 1, 2, \ldots, 1500$. For each node a line is drawn perpendicular to the snake with the node as the center point, and 30 sampling parameters are taken at uniform intervals on the line, whose length is restricted to $[-2\sigma, 2\sigma]$ and 4σ is also the width of the snakes. By simulating the BC model or the BH model with parameter λ, one can get an (average) configuration or wave function $d(\lambda)$. The input to $\mathcal{G}$ is the value of λ, and the input to $\mathcal{N}$ is $d(\lambda)$, as shown in Fig. 1.

B. DCN

Figure 1(b) shows the structure of the DCN, which includes a learner network $\mathcal{N}$ and a guesser network $\mathcal{G}$. The learner

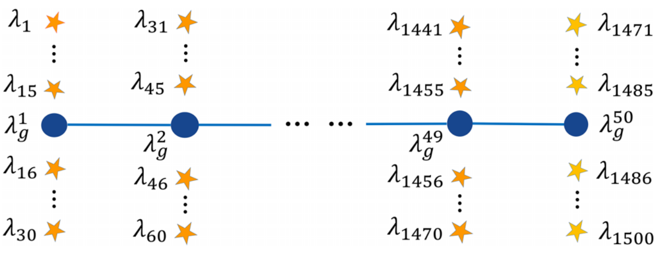

FIG. 2. The way DCN gets input parameters. The blue circles represent the nodes λ_g^i, $i = 1, \ldots, 50$. The orange pentagrams represent the sampled parameters collected.

network $\mathcal{N}$ is a fully connected network that absorbs the classical configurations of the BC model or wave functions of the BH model, labeled by $d(\lambda)$, and outputs their classifications $p_A^{\mathcal{N}}$ and $p_B^{\mathcal{N}}$, i.e., the probability that $d(\lambda)$ belongs to phases A and B, respectively. The neurons in the hidden layer read $y_H = f(d(\lambda)\cdot \mathbf{W_1} + \mathbf{b}_1)$, and the neurons in the output layer yield $\mathcal{N}(d(\lambda)) = (p_A^{\mathcal{N}}, p_B^{\mathcal{N}}) = f(y_H \cdot \mathbf{W_2} + \mathbf{b_2})$. Here $\mathbf{W_1}$ and $\mathbf{W_2}$ are the weight matrices and $\mathbf{b_1}$ and $\mathbf{b_2}$ are the bias vectors. The activation function for the neurons in the hidden and output layer is a sigmoid function.

As shown in Fig. 1(b), the guesser $\mathcal{G}$ absorbs λ and outputs two labels through the sigmoid function to determine the probability that λ belongs to phase A or B, defined as

$$\mathcal{G}_{A,B}(\lambda) = \left(p_A^{\mathcal{G}}, p_B^{\mathcal{G}}\right) = sigmoid[s_{A,B}(\lambda - \lambda_g)/\sigma], \tag{1}$$

where $s_{A,B} = -, +$. The cross-entropy cost function between $\mathcal{N}$ and $\mathcal{G}$ is defined as

$$\mathcal{S}(\mathcal{N}, \mathcal{G}) = -\mathcal{G}\cdot \log \mathcal{N} - (1-\mathcal{G})\cdot \log(1-\mathcal{N}), \tag{2}$$

and the smallest $\mathcal{S}$ indicates the best match between the guessed boundary and the true boundary.

Similarly to the theory of generative adversarial networks [24], the DCN simultaneously optimizes $\mathcal{N}$ and $\mathcal{G}$ to obtain the minimum $\mathcal{S}$. $\mathcal{N}$ gets better learning results by updating the parameters $\mathbf{W}_{\mathcal{N}}$ ($\mathbf{W_1}$, $\mathbf{W_2}$, $\mathbf{b_1}$, and $\mathbf{b_2}$), and $\mathcal{G}$ gets better guessing results by updating λ_g and σ. These parameters cooperate to achieve the purpose of discriminating between the two phases. The dynamics of both networks can be defined as

$$\Delta \mathbf{W}_{\mathcal{N}} = -\alpha_{\mathcal{N}} \partial \mathcal{S}/\partial \mathbf{W}_{\mathcal{N}}, \tag{3a}$$

$$\Delta \lambda_g = -\alpha_{\lambda_g} \partial \mathcal{S}/\partial \lambda_g, \tag{3b}$$

$$\Delta \sigma = -\alpha_\sigma \partial \mathcal{S}/\partial \sigma, \tag{3c}$$

where $\alpha_{\mathcal{N}}$, α_{λ_g}, and α_σ are the learning rates. The partial derivatives of the above equations are expressed as

$$\frac{\partial \mathcal{S}}{\partial \mathcal{G}} = -\log \mathcal{N} + \log(1-\mathcal{N}), \tag{4a}$$

$$\frac{\partial \mathcal{G}_{A,B}}{\partial \lambda_g} = -\frac{s_{A,B}}{4\sigma \cosh^2[(\lambda - \lambda_g)/2\sigma]}, \tag{4b}$$

$$\frac{\partial \mathcal{G}}{\partial \sigma} = \frac{\lambda - \lambda_g}{\sigma}\frac{\partial \mathcal{G}}{\partial \lambda_g}. \tag{4c}$$

Figure 3 gives the flowchart of the update of the position of one node. Starting from an initial input λ_g, then one gets the parameters labeled as $\lambda_1, \ldots, \lambda_{30}$. For each λ_i, there is a $\mathcal{S}_i$ obtained by evaluation of cross entropy. By averaging the 30 cross entropies, $\Delta\lambda_g$ can be obtained and used to update λ_g for the next round of iterations.

For detecting the phase boundaries of multiple phases, many nodes are usually required. Our aim is to initialize the nodes in the parameter plane and drive them all close to the real phase transition boundary using an active contour method or snake model introduced in the next section.

C. The SN model

1. The simple snake model

The snake model is defined as a parametric contour,

$$C(s,t) = [x(s,t), y(s,t)], \tag{5}$$

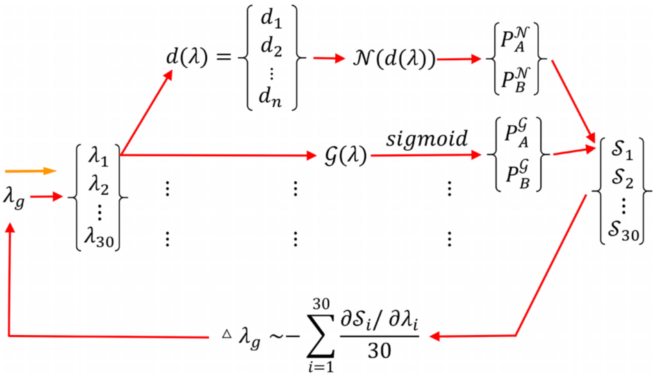

FIG. 3. The process of updating the nodes (λ_g) in the DCN. The orange arrow is at the beginning of the iterations.

where $s \in [0, 1]$ is a parameter, and $C(0,t) = C(1,t)$ for a closed contour where the boundary is periodic. t is the number of iterations. For the images, $x(y)$ refers to the real position of the nodes. For the physical phase diagram to be studied, $x(y)$ represents the value of physical parameters such as temperature, or the interactions.

The total energy E is composed of the internal energy E_{int} and the external energy E_{ext} or the image energy E_{img}. The snake in the image has total energy given by

$$E(C) = \int_0^1 [E_{\text{int}}(C(s)) + E_{\text{ext}}(C(s))]ds, \tag{6}$$

where the internal energy reads

$$E_{\text{int}} = \tfrac{1}{2}[\alpha(s)|C'(s)|^2 + \beta(s)|C''(s)|^2]. \tag{7}$$

In the equation above, $C'(s)$ and $C''(s)$ are the first derivative and the second derivative of $C(s)$ with respect to s. The parameters $\alpha(s)$ and $\beta(s)$ are adjustable and control the continuity and smoothness of the curve. Here the external energy is restricted to the image energy:

$$E_{\text{img}}(C(s)) = -|\partial G_\sigma(C)[I(C)]|^2, \tag{8}$$

where $I(C)$ is the value of pixels, ∂ is a gradient operator, and G_σ is a two-dimensional Gaussian kernel. The process of minimizing the total energy of the snake will allow the position of the snake to coincide with the boundary of the target object. The snake is driven by the image force:

$$\mathbf{F}_{x(y)} = -\frac{\partial E_{\text{img}}}{\partial x(y)}. \tag{9}$$

In the framework of the DCN, E_{img} is replaced by $\mathcal{S}$, and the node coordinates (x, y) are replaced with the physical parameter $(\lambda_g^x, \lambda_g^y)$.

2. The snake net model

In Fig. 4 the five snakes are separated by red circles. The snakes are denoted by C_i^j, where $j = A, B, C, D, E$ means five snakes and $i = 0, \ldots, n-1$ denotes the nodes of the snakes. In total, there are three kinds of nodes, characterized by their degrees $\rho(C)$. Specifically, $\rho(C) = 1, 2$ denote the outer endpoints and the inner nodes, respectively, which can be driven by the force similar to those in simple

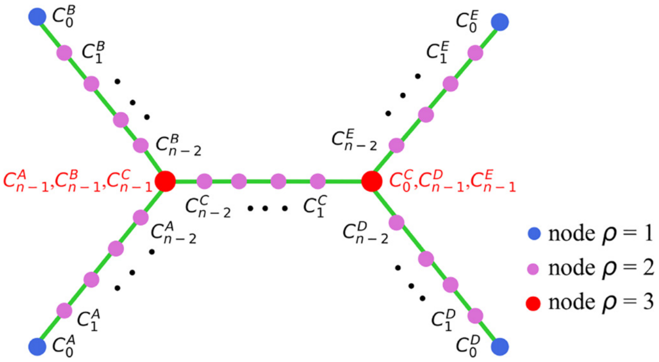

FIG. 4. Topology of the extended SN model. The SN model contain five snakes $C_i^A - C_i^E$. These five snakes have three different nodes $\rho = 1$, $\rho = 2$, and $\rho = 3$, respectively.

snakes. $\rho(C) = 3$ denotes the common node of the different snakes [18].

For a pure image, by the minimization of the energy functional Eq. (6), the different kinds of nodes satisfy the following differential equation:

$$-\alpha C'' + \beta C'''' + \frac{-\partial|\partial G_\sigma(C)[I(C)]|^2}{\partial C} = 0, \qquad (10)$$

whose detailed description is given in Appendix A. The above equation gives the best description of effects of internal forces $\mathbf{F}_{\text{int}} = \alpha C'' - \beta C''''$ and external forces $\mathbf{F}_{\text{ext}} = -\nabla E_{\text{img}}$ on each node of a snake. Switching from the pure images to the physical systems, the external forces have to be replaced by $\Delta\lambda_g$.

For a closed, i.e., periodic boundary condition, a finite difference operation on the Eq. (10) yields

$$\begin{aligned}\frac{\partial E_{\text{img}}}{\partial C_i} &+ \alpha[(C_i - C_{i-1}) - (C_{i+1} - C_i)] \\ &+ \beta(C_{i-2} - 2C_{i-1} + C_i) - 2\beta(C_{i-1} - 2C_i + C_{i+1}) \\ &+ \beta(C_i - 2C_{i+1} + C_{i+2}) = 0. \qquad (11)\end{aligned}$$

However, for the SN model, each snake has a common node C_{n-1} marked in red, which obeys

$$\frac{\partial E_{\text{img}}}{\partial C_{n-1}} + \xi[(C_{n-1} - C_{n-2}) - (C_{n-2} - C_{n-3})] = 0, \qquad (12)$$

where only the first term of Eq. (11) with β as a coefficient is retained, and the other terms, for example, C_n and C_{n+1}, are not present at the ending points. Here ξ is another parameter to be controlled [18].

By combining the set of equations for all nodes together (see Appendix A 2), the following iterative equation can be obtained:

$$\mathbf{A}C + \eta f(C) = 0, \qquad (13)$$

where $\mathbf{A}$ is a pentadiagonal band matrix, which only depends on the parameters α, β, and ξ. $\eta f(C) = \frac{\partial E_{\text{img}}}{\partial C}$, where η is an additional parameter to control the weight between internal and external energy. The iteration steps for the snakes between C_{next} and C_{current} are

$$C_{\text{next}} = (\mathbf{A} + \gamma\mathbf{I})^{-1}[\gamma C_{\text{current}} - \eta f(C_{\text{current}})], \qquad (14)$$

where $\mathbf{I}$ is the identity matrix, and γ is the step size of the snakes.

Figure 4 only shows the extended SN. Sometimes a simple SN can be used with three snakes. The difference between the simple SN and the extended SN is the number of common nodes. The former has only one common node and the latter has more than one common node. Moreover, mathematically, the iteration of Eq. (14) can be different. In Appendix A 2, three types of matrices $\mathbf{A}$ are shown for (i) a closed snake, (ii) a snake with a fixed node at one end and a common node at the other end, and (iii) a snake with both endpoints as common nodes.

For image segmentation [18], a big matrix $\mathbf{A}$ can contain the elements for all snakes. The couplings between different snakes are defined in $\mathbf{A}$. For physical systems we separate the big matrix into several small matrices for each snake, and the coupling between them is realized by passing the positions of the common nodes.

III. THE APPLICATION OF THE SN-DCN METHOD

A. The BC model

We choose the BC model [21] to test our method. The BC model on the square lattice is defined by the following Hamiltonian:

$$H = -J_x \sum_{\langle i,j\rangle_x} S_i S_j - J_y \sum_{\langle i,j\rangle_y} S_i S_j + D \sum_i S_i^2 - h \sum_i S_i, \qquad (15)$$

where $S_i = \pm 1, 0, i = 1, 2, \ldots, N$, N represents the total number of sites, and $J_{x(y)}$ is the exchange interaction between sites along the two directions. D is a single-spin anisotropy parameter, and h is an external magnetic field. Figure 5(a) shows the ground-state phase diagram of the BC model. The temperature parameter T/J_y is as low as 0.1. The color characterizing different phases is obtained by the value of $\sum S_i/N + |S_1 - S_2|$.

In the next sections we use different classical phases to test our SN-DCN method for the $h = 0$ BC model and the extended SN-DCN method with two common nodes for the $h \neq 0$ BC model, respectively.

B. The SN-DCN method

A simple SN contains three snakes, which have a common node, as shown in Fig. 5(a). The initial snakes are represented by dashed lines, and the final snakes are marked by solid lines. Snakes A, B, and C are marked in red, green, and purple, respectively. The topology of the SN model is consistent with the boundaries between the three phases, which are the ferromagnetic, super-antiferromagnetic, and paramagnetic phases. The configurations in a four-site cell are $\begin{smallmatrix}++\\++\end{smallmatrix}$, $\begin{smallmatrix}+-\\+-\end{smallmatrix}$, and $\begin{smallmatrix}00\\00\end{smallmatrix}$, respectively. The parameters for the SN-DCN are listed in Appendix B 1.

According to Eq. (3c), during the updating process the unit width σ of the snake is obtained. The cross-entropy cost $\mathcal{S}$, the internal energy E_{int}, and the external energy E_{ext} of the snakes are also recorded separately to ensure that the snakes meet the mechanical balance.

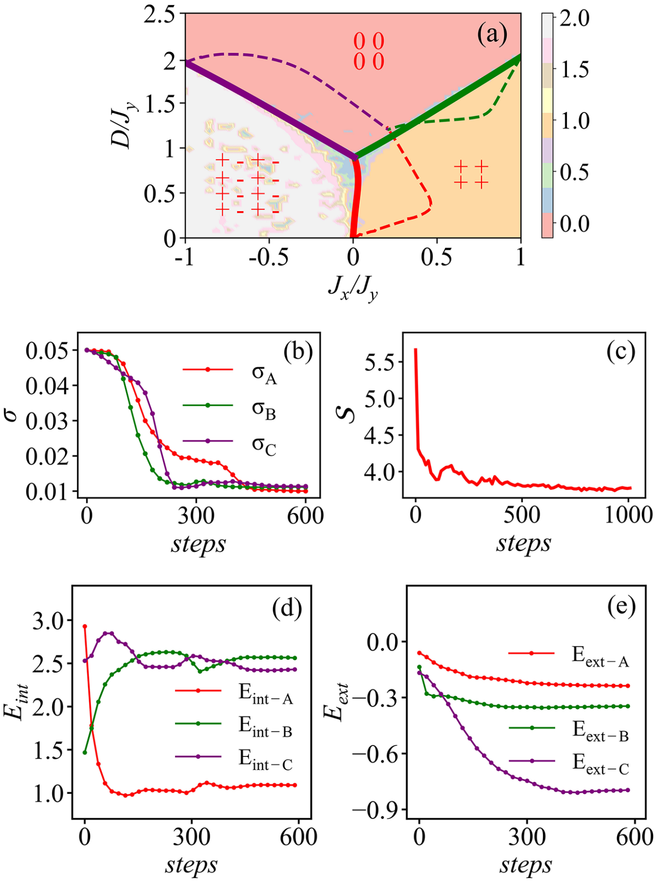

FIG. 5. The results of applying the SN-DCN method. (a) The phase diagram of the BC model with $h = 0$, initial snakes (dashed lines), and finial snakes (solid lines). (b) The average σ vs iteration steps. (c) Total $\mathcal{S}$ vs steps. (d, e) E_{int} and E_{ext} over steps. These quantities converge very well.

In Fig. 5(b) the unit widths σ_A, σ_B, σ_C of three snakes are shown. The width for each node is updated independently, so the average width of all nodes for each snake is given here, and these values converge from a value of 0.05 to about 0.01.

In Fig. 5(c) the total cross-entropy cost $\mathcal{S}$ is convergent. $\mathcal{S}$ is a function of the guessed boundary λ_g and the unit width σ of the snake. The DCN is used to find the minimum $\mathcal{S}$ and the corresponding λ_g and σ using the gradient descent method. Meanwhile, the values of $E_{\text{int}}^{A,B,C}$ and $E_{\text{ext}}^{A,B,C}$ also converge as shown in Figs. 5(d) and 5(e). The stability of the E_{int} indicates that the shape of snakes no longer changes. According to Eq. (8), the value of pixel $I(C)$ is replaced by order parameters, i.e., the colors shown in Fig. 5(a). The closer the snake is to the real phase boundary, the smaller the external energy. These results show that the SN-DCN can be applied to a phase diagram with three phases.

C. The distinct initial topology of SN-DCN

In the previous section the topology of the initial SN is consistent with the true phase boundaries. Here we discuss the correct results that are obtained with the wrong initial topology, i.e., the topology of the initial SN is different from the topology of the real boundaries.

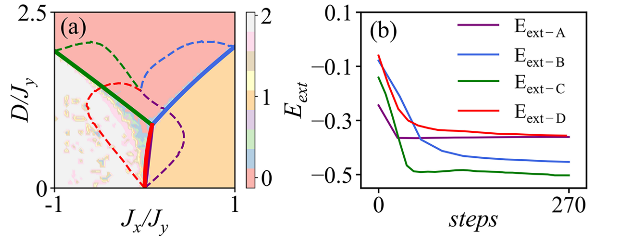

FIG. 6. (a) The SN-DCN containing four snakes is used to detect three boundaries. The initial SN are dashed lines and the final are solid lines. (b) The external energy of the two snakes overlaps finally.

In Fig. 6(a), initially, there are four snakes (dashed lines), but there are only three boundaries. Snakes A, B, C, and D are marked in purple, blue, green, and red, respectively. After updating, eventually the two snakes (red and purple) overlap to a single true phase boundary. This shows that even with an extra snake, our SN-DCN can still find the true phase boundaries correctly. To further check whether or not other properties overlap when the two snakes' positions converge, the energy of $snake_A$ and $snake_D$ overlap, as shown in Fig. 6(b).

D. The extended SN-DCN method

To test the generality of the SN-DCN method, the two movable common nodes are introduced. It can detect boundaries between four phases. Figure 7(a) shows the ground-state phase diagram of the BC model with $J_x/J_y = -1$. It contains four phases whose configurations are $\begin{smallmatrix}++\\++\end{smallmatrix}$, $\begin{smallmatrix}+-\\+-\end{smallmatrix}$, $\begin{smallmatrix}+0\\+0\end{smallmatrix}$, and $\begin{smallmatrix}00\\00\end{smallmatrix}$. The color is obtained by $\sum_{n=1}^{4} |S_i - S_{\text{in}}|/4$, where S_{in} are the spins located at the neighboring lattice sites.

The initial snakes are illustrated by the dashed lines, and then the phase boundaries are detected by the final snakes marked by solid lines in different colors. During the updating process, the quantities σ, $\mathcal{S}$, E_{int}, and E_{ext} are shown in Figs. 7(b)–7(e), and all of them can be convergent, which means that the SN model is *extendable*. The parameters for obtaining Fig. 7 are listed in the Appendix B 2.

IV. THE BALLOON FORCE AND ITS APPLICATION

A. The motivation for introducing the balloon force

The BF is inspired by the field of computer image processing [20]. It is used to solve the problem that when the initial snake is far from the target contour, the snake cannot feel the image force and cannot move. For physical systems, the pixel points in the image are replaced by physical configurations. It is not clear whether BF can help detect the boundary of the phase.

We use the phase diagram of the BH model to illustrate the effect of the BF. The Hamiltonian of the BH model is expressed as [23]

$$H = -J \sum_{\langle i,j \rangle} (b_i^{\dagger} b_j + b_j^{\dagger} b_i) + \sum_i \left(\frac{U n_i (n_i - 1)}{2} - \mu n_i \right), \tag{16}$$

where μ is the chemical potential, and J and U are the boson hopping energy and on-site interaction, respectively. b and

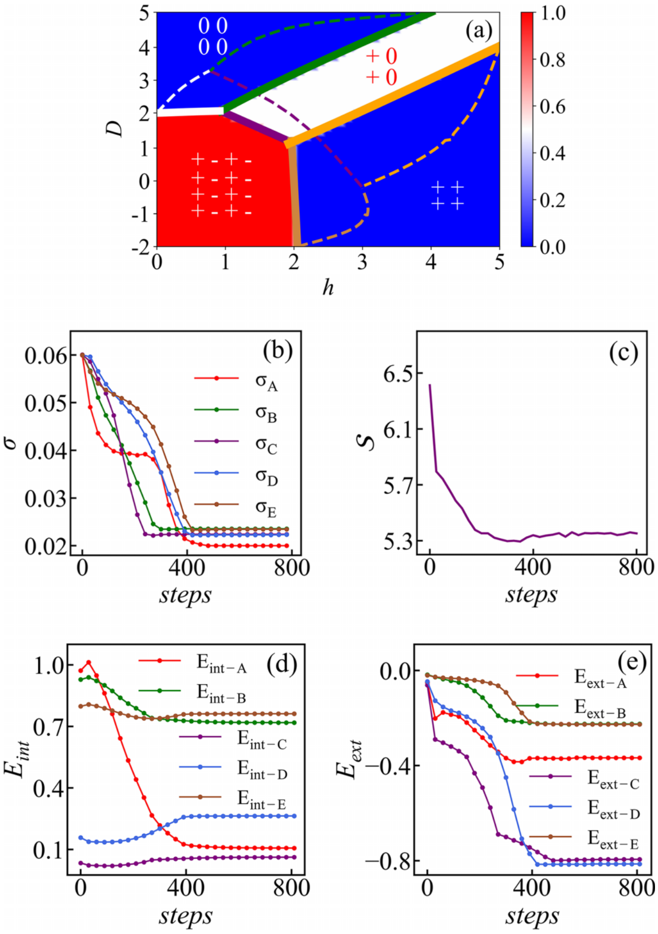

FIG. 7. The results of applying the extended SN-DCN method. (a) The ground-state phase diagram of the BC model in the plane D for h for $J_x/J_y = -1$, initial snakes (dashed lines), and finial snakes (solid lines). (b) The average σ vs iteration steps. (c) Total $\mathcal{S}$ vs steps. (d, e) E_{int} and E_{ext} over steps. These quantities converge very well.

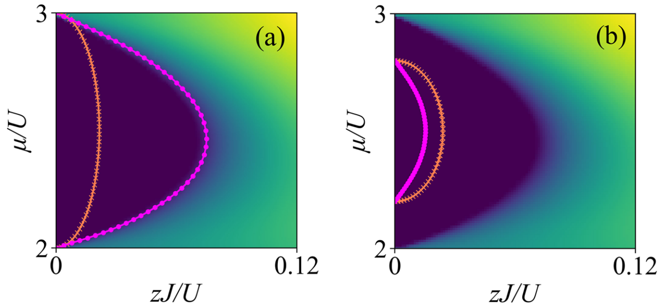

FIG. 8. (a) Without the balloon force, the DCN can help find the correct phase boundary (purple) if the initial snake (red) is set at the correct initial position. (b) The DCN fails to find the correct phase boundary (purple) if the initial snake (red) is set at the wrong initial position.

$b^\dagger$ are the boson creation and annihilation operators, respectively. n_i represents the particle number operator of site i. Using the mean-field approximation [25], the order parameter $\psi = \langle b_i^\dagger \rangle = \langle b_i \rangle$ can be introduced to describe the superfluid and insulated phases.

In Ref. [16] the initial snake first encloses the target contour and then it gradually shrinks to the target contour. Here, as shown in Fig. 8(a), the snake is initialized at a different location marked by the red symbols, i.e., within the target contour. The ending points of the initial snake are fixed at the ends of the axis $zJ/U = 0$. The purple line represents the snake after convergence.

However, in Fig. 8(b) the initial snake is fully immersed in the insulating phase marked in blue, and the snake is finally located near the initial position, only the shape made a small change under the action of internal forces. The reason is that the snake hardly feels the external force which is provided by cross-entropy cost $\mathcal{S}$. According to Eq. (2), the sampled data $d(\lambda)$ from the parameter marked with green symbols in Fig. 1(c) is from the same phase, $\mathcal{S}$ is a constant because the outputs $p_{A,B}^{\mathcal{G}}$ and $p_{A,B}^{\mathcal{N}}$ of $\mathcal{G}$ and $\mathcal{N}$ do not change. According to Eq. (3b), the nodes of the snake cannot move.

When the initial position of the snake is poorly chosen, the snake can get stuck in a certain region of the parameter space and fail to converge to the phase boundary. Therefore, an additional force, the balloon force, is introduced and it pushes the snake toward the phase boundary in such cases. The DCN with the balloon force method is named the BF-DCN method. Here a *decaying* BF is defined as

$$\mathbf{F}_{\text{balloon}} = \kappa \mathbf{n}(C), \tag{17}$$

where κ decays with the iteration steps $\kappa = \kappa_0 a^{-\text{steps}}$, and $\mathbf{n}(C)$ means the normal direction of the snake. The direction $\mathbf{n}(C)$ of the $\mathbf{F}_{\text{balloon}}$ outward along the normal direction is positive.

To visualize the iterative process more clearly, for the ith node the distance d_i between its positions at time t and the final time $t \longrightarrow \infty$ is defined as

$$d_i = \sqrt{\left|x_i^t - x_i^\infty\right|^2 + \left|y_i^t - y_i^\infty\right|^2}\ , \tag{18}$$

where (x_i^t, y_i^t) and (x_i^∞, y_i^∞) are the coordinates of the ith node at two different steps. Then the average distance is obtained as

$$D = \frac{\sum_{i=1}^{N} d_i}{N}, \tag{19}$$

where N represents the number of nodes in each snake. For convenience we follow the custom of Ref. [16] and normalize the range of coordinates in the physical parameter space.

B. Force analysis of snake nodes

To understand how the BF works, a force analysis of the snake is performed here. The total force $\mathbf{F}_{\text{tot}}$ is composed of the internal force $\mathbf{F}_{\text{int}}$ and external force $\mathbf{F}_{\text{ext}}$, i.e.,

$$\mathbf{F}_{\text{tot}} = \mathbf{F}_{\text{int}} + \mathbf{F}_{\text{ext}}, \quad \mathbf{F}_{\text{ext}} = \mathbf{F}_{\text{cost}} + \mathbf{F}_{\text{balloon}}, \tag{20}$$

where $\mathbf{F}_{\text{cost}}$ is the force introduced by the gradient descent method to find the minimum cross-entropy cost, and its direction is the direction normal to the snake. $\mathbf{F}_{\text{int}}$ depends on parameters such as α, β, etc., and only changes the appearance characteristics such as whether the snake is smooth or not, but not the overall position. Therefore the magnitude of $\mathbf{F}_{\text{int}}$ is not analyzed here.

In Fig. 9(a1) the forces on four snakes are shown. The BF is marked by white arrows and labeled as F_{b}, i.e., $\mathbf{F}_{\text{balloon}}$. When

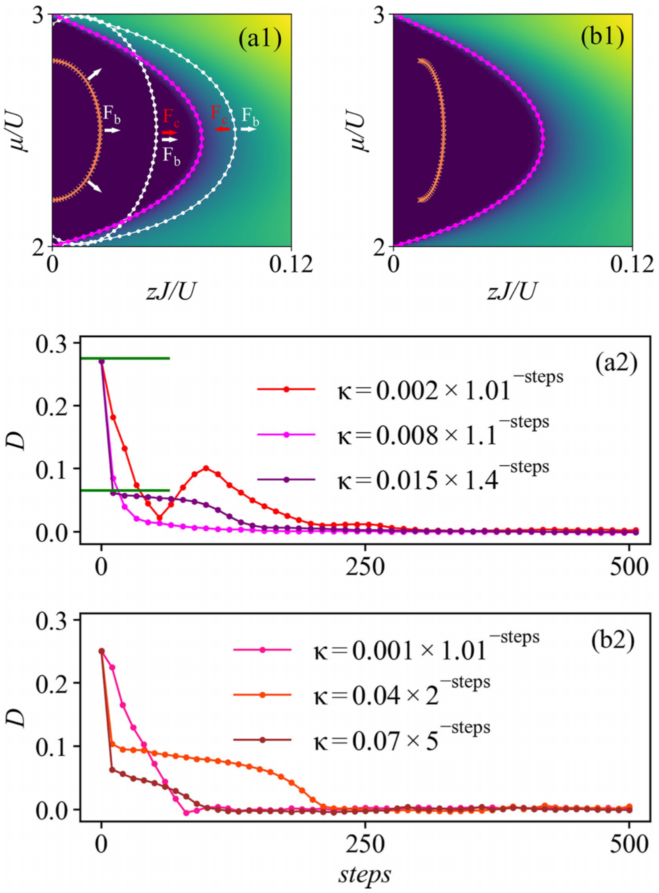

FIG. 9. Force analysis of snakes with balloon force. (a1) The initial snake (red) with only balloon force F_b, and the final snake (purple). Two possible positions of snakes (white lines) with forces. $\mathbf{F}_{\text{cost}}$ and $\mathbf{F}_{\text{balloon}}$, simplified as F_c and F_b. (a2) D vs steps for (a1). D is the average distance between the corresponding nodes of the initial and final snakes, as shown in Eq. (19). (b1) The initial snake is completely immersed in the insulated phase. (b2) D vs steps for (b1).

using the BF, the position of the real phase boundary relative to the initial snake needs to be known. The sign of $\mathbf{F}_{\text{balloon}}$ cannot be varied in our approach. Under the action of $\mathbf{F}_{\text{balloon}}$, the initial snake marked in red begins to expand gradually to the right.

The snakes located at other possible locations are also shown, where the snake marked with the white line on the left side perceives $\mathbf{F}_{\text{balloon}}$ and $\mathbf{F}_{\text{cost}}$ in the same direction, while the snake marked with the white line on the right side perceives $\mathbf{F}_{\text{balloon}}$ and $\mathbf{F}_{\text{cost}}$ in the opposite direction. The combined effect of the two forces confines the snake to the real phase boundary. In Fig. 9(a2), between the two green lines the snake is moving fast. This is because, in the early stages, the BF has not decayed as much. In the final stages, $D \approx 0$ means the snakes converge to the true phase boundary. In Fig. 9(b1), with the help of $\mathbf{F}_{\text{balloon}}$, at an initial position with $\mathbf{F}_{\text{cost}} = 0$ the snake is still able to iterate to the target position. The quantity D is shown in Fig. 9(b2).

Here we provide a short argument why the snake is guaranteed to converge in with a decaying force. As shown in Eq. (20), the external force $\mathbf{F}_{\text{ext}}$ includes $\mathbf{F}_{\text{cost}}$ and $\mathbf{F}_{\text{balloon}}$. The function of $\mathbf{F}_{\text{cost}}$ is similar to the restoring force of a spring, dedicated to pulling the nodes of the snake back to the equilibrium position, or the true phase boundary, where $\mathbf{F}_{\text{cost}} = 0$. The BF should also be close to 0 at the phase boundary; otherwise, if the BF is a nonzero constant the snake can go beyond the true phase boundary.

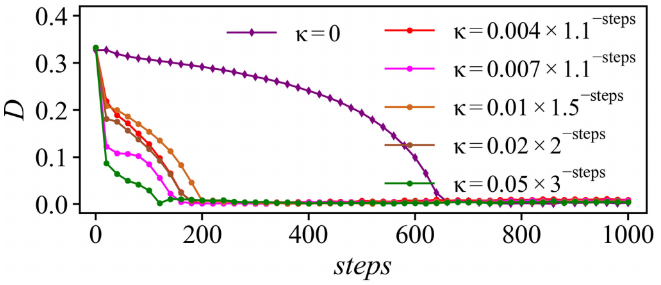

FIG. 10. D vs steps with ($\kappa \neq 0$) and without ($\kappa = 0$) balloon force. D is the average distance between the corresponding nodes of the initial and final snakes, as shown in Eq. (19). Clearly, the balloon force helps accelerate the convergence.

C. The superparameters and the improvement by the balloon force

In Fig. 10 the data of D vs $steps$ show that $\mathbf{F}_{\text{balloon}}$ reduces number of training steps. For comparison purposes, the initial positions corresponding to the different data lines are the same. $\kappa = 0$, i.e., $\mathbf{F}_{\text{balloon}} = 0$, results in a slow convergence effect with convergence steps to over 600. Other parameters are set to $\alpha = 0.001$, $\beta = 2$, and $\gamma = 0.15$. These parameters correspond to the fastest convergence with $\mathbf{F}_{\text{balloon}} = 0$. So we choose this set of data as a comparison for $\mathbf{F}_{\text{balloon}} \neq 0$.

We also show the effects of other parameters. We adjust many values of α, β, and γ, and none of them are found to accelerate the convergence of D more easily than the BF. Without the BF, the distances D are shown vs iteration steps with different values of α, β, and γ. In Fig. 11(a) α varies from 0.1 to 0.0001. The fastest parameter is an intermediate value of 0.001. The reason is that large α makes the snake straight and hinders bending. Small α leads to the curve being too easy to bend without being rigid. It has notorious difficulty in determining the weights α, β, and γ associated with the smoothness constraint, reported in a review reference [26]. Similarly, the results of modifying β and γ are shown in Figs. 11(b) and 11(c).

D. The balloon force applied to multiple phases

For physical systems, the fluctuation of data near phase boundaries is maximum. Especially, multiple phase boundaries meet and are more difficult to handle. Here we discuss whether or not BF works in phase diagrams containing more than two phases.

In Fig. 12(a) the BF is added to an initial snake that is immersed in the $^{00}_{00}$ phase, and this snake eventually converges to the phase boundary. Meanwhile, in Fig. 12(b) E_{int} and E_{ext} are also shown to verify the results.

In Fig. 13(a) the initial snakes are dashed lines and the final snakes are solid lines. Snakes A, B, and C are marked

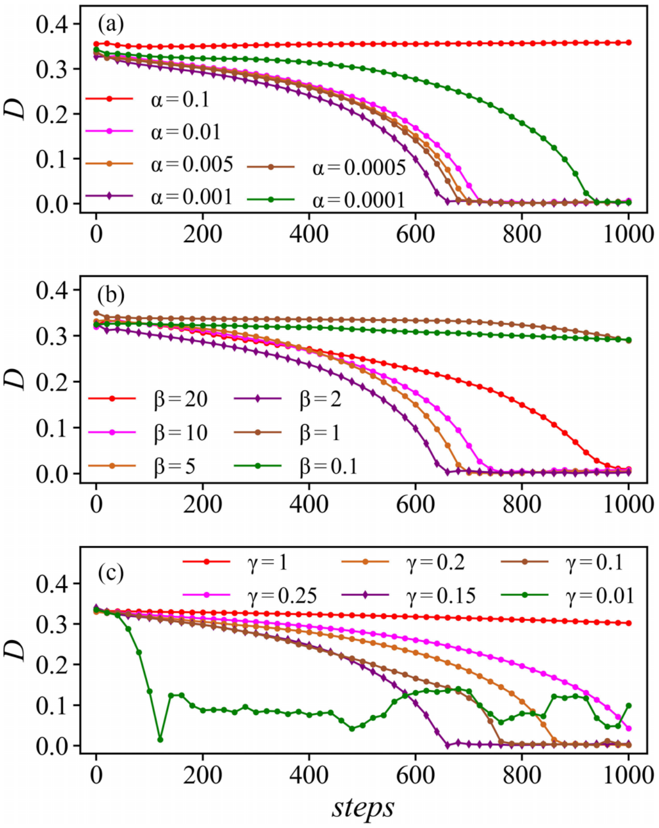

FIG. 11. Without balloon force, D vs steps for various parameters α, β, and γ. D is the average distance between the corresponding nodes of the initial and final snakes, as shown in Eq. (19). (a) Adjusting only α, (b) adjusting only β, and (c) adjusting only γ.

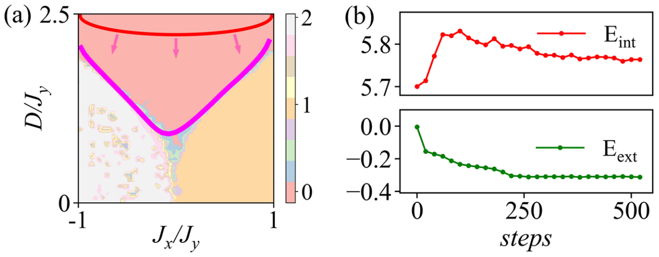

FIG. 12. The DCN with balloon force is applied to a phase diagram containing three phases. (a) The initial snake (red), the final snake (pink line), and the balloon force (pink arrows). (b) E_{int} and E_{ext} vs steps.

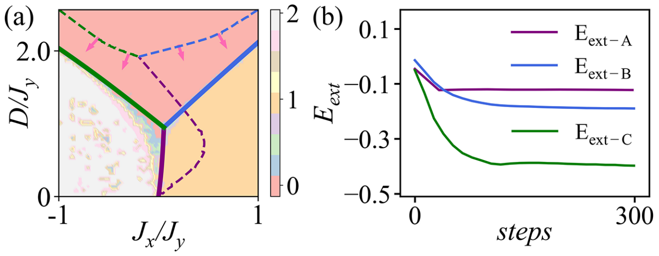

FIG. 13. Introducing balloon force to the SN-DCN. (a) The initial snakes (dashed lines), the final snakes (solid lines), the arrows represent the balloon force. (b) E_{ext} vs steps.

in purple, blue, and green, respectively. By applying BF to $snake_B$ and $snake_C$, respectively, we still get the correct result. In Fig. 13(b) the E_{ext} converge to a minimum value, indicating that the snakes stably stay at the true phase boundary. The results represent that it is feasible to select one or two of the snakes in the SN-DCN to add extra BF.

In short summary, the BF can somewhat reduce the restriction that the initial position of the snake model must be close to the real boundary; moreover, the BF can accelerate the movement of the snake with the DCN.

V. CONCLUSION AND DISCUSSION

In conclusion, we extend the DCN with a simple snake model by altering the topology to a snake net. This potentially allows one to map out two-dimensional phase diagrams featuring more than two distinct phases, which is a limitation of the original approach [16]. Moreover, we introduce an additional external force (balloon force) which may help the snake to leave its initial position more quickly or leave a wrong initial position and thus may allow for greater freedom in the initialization of the snake.

Our idea to introduce BF is from Ref. [16], which presents the difficulty of initialization of the snake and gives an idea to solve the problem: scale or move the snake. There are several ways to change the position of the snake, including translation, rotation, and expansion. BF is a dynamic way to move or expand the snake, which helps the snake to have overlapping parts with real phase boundaries. BF can be applied to simple snakes with open and periodic boundaries, as well as to snake nets. BF gives the snake some flexibility.

In this paper the BF does not completely solve the problem of limiting the initial position of the snake; it only makes the snake more flexible to some extent. When applying BF, *a priori* information about the relative position of the true phase boundary with respect to the initial position of the snake is useful. As shown in Fig. 9(b1), the snake expands from the left side of the phase diagram to the right side. Considering that the SN-DCN method is proposed in the framework of unsupervised learning, perhaps we can initialize two independently updated snakes, one moving in the direction to the left and the other to the right, finally dropping the snake that overlaps with the parameter axis. The direction in the expansion is considered a hyperparameter.

It should be remembered that if the possible location of the real phase boundary is known, it is simpler to adjust the position of the initial snake appropriately without using BF. However, through testing, it is found that the BF helps to make the initial snake position more flexible and provides the possibility to find unknown phase boundaries.

Although machine learning of phase transitions has been proposed for many years [1,5], unsupervised machine learning for a phase transition is still very active, including the interpretability of the machine learning [27,28], studying the topological phase [29,30], as well as the experimental implementation of the unsupervised learning method [31]. It is also expected that machine learning methods, in the future, will solve difficulties that are difficult to solve by traditional methods.

ACKNOWLEDGMENTS

W.Z. would like to thank Junyi Xu for his help and feedback, and Huijiong Yang and Nan Wu for their contributions while they were studying in the office during their undergraduate years. This work was supported by the Hefei National Research Center for Physical Sciences at the Microscale (KF2021002), and Project 12047503 supported by NSFC. J.Z. is supported by the Natural Science Foundation of Shanxi Province (Grant No. 20210302122405l). X.S. is supported by Local Science and Technology Development Project Guided by Central Government (YDZJSX2021A007).

APPENDIX A: DETAILS ABOUT THE SNAKE MODEL

In this Appendix we give a slightly more detailed derivation of the snake model, as well as the iteration matrix for different snakes.

1. Energy minimization condition

In order to get the best position of the snake, we need to calculate the minimum energy. Our derivation is slightly different from Ref. [18]. We call $C(s)$ the true contour, the one we are trying to find. We take some trial contour

$$C_{\mathrm{trial}}(s) = C(s) + \varepsilon\delta(s), \tag{A1}$$

which differs from the true contour by $\varepsilon\delta(s)$, where ε is a small quantity and $\delta(s)$ is an arbitrary function. If we write Eq. (6) as the following form,

$$\mathcal{F} = \int_0^1 \mathcal{L}[C(s), C'(s), C''(s)]ds, \tag{A2}$$

where $\mathcal{L}(C, C', C'') = E_{\mathrm{int}} + E_{\mathrm{ext}}$ is the energy per ds, where $[C(s), C'(s), C''(s)]$ are considered to be independent variables, then we get

$$\left.\frac{d\mathcal{F}(\varepsilon)}{d\varepsilon}\right|_{\varepsilon=0} = 0. \tag{A3}$$

Since we've already chosen the true contour, this makes the total energy minimum according to the principle of *calculus of variations*.

Putting Eq. (A2) in Eq. (A3), one gets

$$\begin{aligned}
\left.\frac{d\mathcal{F}(\varepsilon)}{d\varepsilon}\right|_{\varepsilon=0} &= \int_0^1 \left[\frac{\partial\mathcal{L}}{\partial(C+\varepsilon\delta)}\frac{d(C+\varepsilon\delta)}{d\varepsilon} + \frac{\partial\mathcal{L}}{\partial(C'+\varepsilon\delta')}\frac{d(C'+\varepsilon\delta')}{d\varepsilon} + \frac{\partial\mathcal{L}}{\partial(C''+\varepsilon\delta'')}\frac{d(C''+\varepsilon\delta'')}{d\varepsilon}\right]ds \\
&= \int_0^1 \left[\frac{\partial\mathcal{L}}{\partial C}\delta + \frac{\partial\mathcal{L}}{\partial C'}\delta' + \frac{\partial\mathcal{L}}{\partial C''}\delta''\right]ds \\
&= \int_0^1 \frac{\partial\mathcal{L}}{\partial C}\delta(s)ds + \frac{\partial\mathcal{L}}{\partial C'}d\delta(s) + \frac{\partial\mathcal{L}}{\partial C''}d\delta'(s) \\
&= \left.\frac{\partial\mathcal{L}}{\partial C'}\delta(s)\right|_0^1 + \left.\frac{\partial\mathcal{L}}{\partial C''}\delta'(s)\right|_0^1 + \int_0^1 \left[\frac{\partial\mathcal{L}}{\partial C}\delta(s) - \left(\frac{\partial\mathcal{L}}{\partial C'}\right)'\delta(s) - \left(\frac{\partial\mathcal{L}}{\partial C''}\right)'\delta'(s)\right]ds \\
&= \int_0^1 \left[\frac{\partial\mathcal{L}}{\partial C}\delta(s) - \left(\frac{\partial\mathcal{L}}{\partial C'}\right)'\delta(s) - \left(\frac{\partial\mathcal{L}}{\partial C''}\right)'\delta'(s)\right]ds \\
&= \int_0^1 \left[\frac{\partial\mathcal{L}}{\partial C} - \left(\frac{\partial\mathcal{L}}{\partial C'}\right)' + \left(\frac{\partial\mathcal{L}}{\partial C''}\right)''\right]\delta(s)ds = 0.
\end{aligned}$$

For the arbitrary function $\delta(s)$, the generalized Euler-Lagrange equation can be obtained as

$$\frac{\partial\mathcal{L}}{\partial C} - \left(\frac{\partial\mathcal{L}}{\partial C'}\right)' + \left(\frac{\partial\mathcal{L}}{\partial C''}\right)'' = 0, \tag{A4}$$

for solving the minimum value of the energy functional. Equation (A4) is formally a second-order Euler-Lagrange equation, i.e., a Jacobi-Ostrogradsky formulation, which has a number of applications in fundamental physics (e.g., Refs. [32–34]).

2. Iteration matrix A

By putting the product function Eq. (6) into Eq. (A4), we get the differential equation

$$-\alpha C'' + \beta C'''' + \frac{\partial E_{\mathrm{img}}}{\partial C} = 0. \tag{A5}$$

Here we can consider $\alpha C'' - \beta C''''$ and -∇E_{img} as internal forces $\mathbf{F}_{\mathrm{int}}$ and external forces $\mathbf{F}_{\mathrm{ext}}$ on the snake, respectively. The snake satisfies the mechanical balance $\mathbf{F}_{\mathrm{int}} + \mathbf{F}_{\mathrm{ext}} = 0$. For simplicity, $E_{\mathrm{img}} = E_{\mathrm{ext}}$ is assumed here, i.e., there is no other external force except the image force. Since E_{img} is not available as an expression of C for general images, Eq. (A5) has no analytical solution. Moreover, analytical solutions of higher-order differential equations are known to generate spurious or unstable solutions as evidenced by the Ostrogradsky instability. However, this equation can be reliably solved by the finite difference numerical method and reads

$$\begin{aligned}
&\frac{\partial E_{\mathrm{img}}}{\partial C} + \alpha[(C_i - C_{i-1}) - (C_{i+1} - C_i)] \\
&\quad + \beta(C_{i-2} - 2C_{i-1} + C_i) - 2\beta(C_{i-1} - 2C_i + C_{i+1}) \\
&\quad + \beta(C_i - 2C_{i+1} + C_{i+2}) = 0.
\end{aligned} \tag{A6}$$

Equation (A6) is the mechanical equation satisfied by node i and its neighborhood nodes. By combining the set of equations for all nodes together, the following equation can be

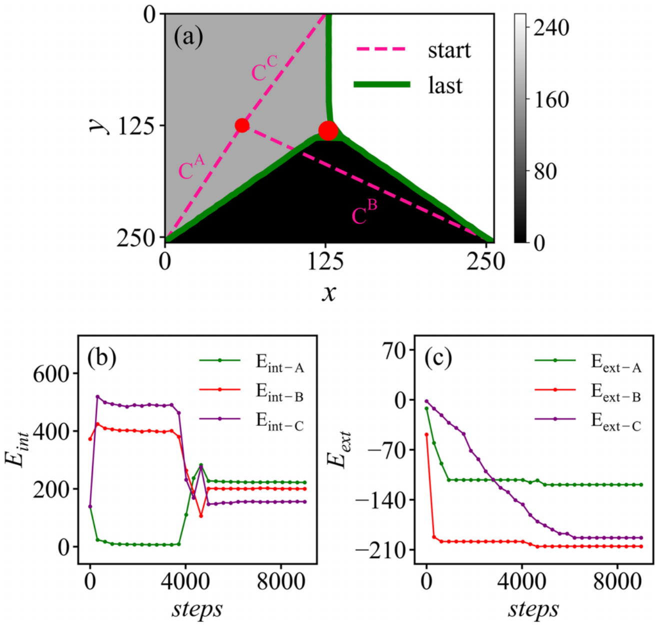

FIG. 14. The results of applying the SN without the DCN to detect contour in the image with three different colors. (a) The gray image, the initial snakes are in pink, and the final snakes are in green. The red dot represents the common node where the three snakes intersect. (b) E_{int}^{A} - E_{int}^{C} vs steps. (c) E_{ext}^{A} - E_{ext}^{C} vs steps.

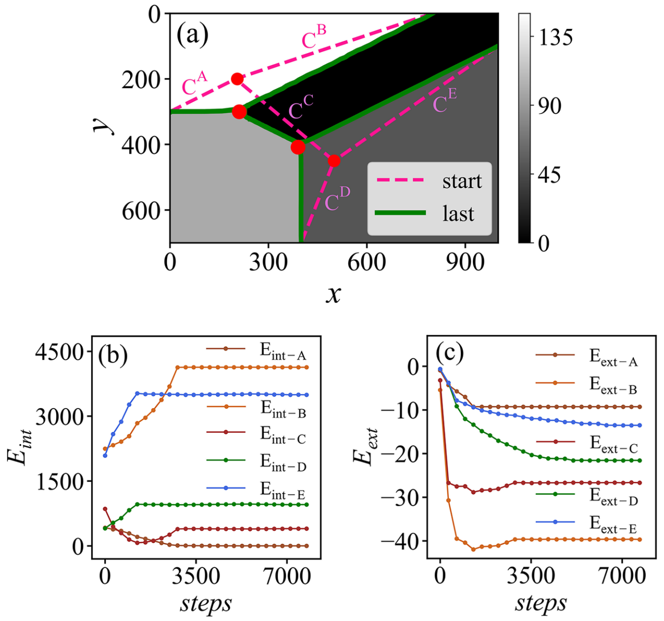

FIG. 15. The results of applying the SN without the DCN to detect contour in the image with four different colors. (a) The gray image, initial snakes (pink), final snakes (green), and the common nodes (red dots). (b) E_{int}^{A} - E_{int}^{E} vs steps. (c) E_{ext}^{A} - E_{ext}^{E} vs steps.

obtained,

$$\mathbf{A}C + \frac{\partial E_{\mathrm{img}}}{\partial C} = 0, \tag{A7}$$

where **A** is a pentadiagonal banded matrix, which only depends on the parameters α and β, and $C = [C_0, C_1, \ldots, C_{N-1}]^{\mathrm{Transpose}}$.

For a snake with *periodic* boundary conditions, the matrix is defined as

$$\mathbf{A} = \begin{bmatrix} c_0 & b_1 & a_2 & & & & a_{N-2} & b_{N-1} \\ b_0 & c_1 & b_2 & a_3 & & & & a_{N-1} \\ a_0 & b_1 & c_2 & b_3 & a_4 & & & \\ & a_1 & b_2 & c_3 & b_4 & a_5 & & \\ & & & & \cdots & & & \\ & & & a_{N-5} & b_{N-4} & c_{N-3} & b_{N-2} & a_{N-1} \\ a_0 & & & & a_{N-4} & b_{N-3} & c_{N-2} & b_{N-1} \\ b_0 & a_1 & & & & a_{N-3} & b_{N-2} & c_{N-1} \end{bmatrix}, \tag{A8}$$

where the values of a_i, b_i, c_i are as follows:

$$a_i = \beta, \tag{A9a}$$

$$b_i = -\alpha - 4\beta, \tag{A9b}$$

$$c_i = 2\alpha + 6\beta. \tag{A9c}$$

Pentadiagonal matrices are sparse band matrices and therefore useful for numerical analysis.

For the simple SN model, such as C^A with one fixed node at the end and one common node at the other end, as shown in Fig. 14(a), the matrix **A** is modified as

$$\mathbf{A} = \begin{bmatrix} d_0 & e_1 & b_2 & a_3 & & & & a_{N-1} \\ a_0 & b_1 & c_2 & b_3 & a_4 & & & \\ & a_1 & b_2 & c_3 & b_4 & a_5 & & \\ & & & & \cdots & & & \\ & & & a_{N-5} & b_{N-4} & c_{N-3} & b_{N-2} & a_{N-1} \\ a_0 & & & & a_{N-4} & b_{N-3} & e_{N-2} & d_{N-1} \\ & & & & & f_{N-3} & g_{N-2} & f_{N-1} \end{bmatrix}, \tag{A10}$$

where the elements of the first row are 0 due to the first node being fixed. Unlike the periodic boundary snake, there are many boundary-related elements that have been revised and the values of a_i, b_i, c_i, d_i, e_i, f_i, g_i are as follows:

$$a_i = \beta, \tag{A11a}$$

$$b_i = -\alpha - 4\beta, \tag{A11b}$$

$$c_i = 2\alpha + 6\beta, \tag{A11c}$$

$$d_i = -\alpha - 2\beta, \tag{A11d}$$

$$e_i = 2\alpha + 5\beta, \tag{A11e}$$

$$f_i = \xi, \tag{A11f}$$

$$g_i = -2\xi. \tag{A11g}$$

Similarly, for the snake which has two movable endpoints such as C^C as shown in Fig. 15(a), the matrix is

$$\mathbf{A} = \begin{bmatrix} f_0 & g_1 & f_2 & & & & & \\ d_0 & e_1 & b_2 & a_3 & & & & a_{N-1} \\ a_0 & b_1 & c_2 & b_3 & a_4 & & & \\ & a_1 & b_2 & c_3 & b_4 & a_5 & & \\ & & & & \cdots & & & \\ & & & a_{N-5} & b_{N-4} & c_{N-3} & b_{N-2} & a_{N-1} \\ a_0 & & & & a_{N-4} & b_{N-3} & e_{N-2} & d_{N-1} \\ & & & & & f_{N-3} & g_{N-2} & f_{N-1} \end{bmatrix}. \tag{A12}$$

Reference [18] defines a total matrix **A** which contains the elements for all snakes. Here the matrix **A** we defined is for each snake. The coupling between each snake can be achieved by passing the coordinates of common nodes from one snake to other snakes.

3. The iteration equation

Equation (A7) is a static mechanical equilibrium equation without considering the damping force of the deformed contour. To describe a dynamic contour, the time parameter t and the damping force $\mathbf{F}_{\text{dump}}(C_t) = -\gamma \frac{\partial C_t}{\partial t}$ and the inertia term $\mu \frac{\partial^2 C_t}{\partial t^2}$ are introduced, and then the equation

$$\mu \frac{\partial^2 C_t}{\partial t^2} = \mathbf{F}_{\text{dump}}(C_t) + \mathbf{F}_{\text{int}}(C_t) + \mathbf{F}_{\text{img}}(C_t) \tag{A13}$$

is obtained, where

$$-\mathbf{F}_{\text{int}} = -\alpha C_t'' + \beta C_t'''' = \mathbf{A}C, \tag{A14a}$$

$$-\mathbf{F}_{\text{img}} = \frac{\partial E_{\text{img}}}{\partial C} = \eta f(C_t), \tag{A14b}$$

and where η is an additional parameter in order to control the weight between internal and image energy. The inertia term $\mu \frac{\partial^2 C_t}{\partial t^2}$ is set to zero because the inertia term can cause the snake to cross the target boundary. This dynamic process becomes a quasistatic process as

$$\gamma \frac{\partial C_t}{\partial t} = \mathbf{F}_{\text{int}} + \mathbf{F}_{\text{img}}. \tag{A15}$$

The equation of the discrete snake becomes

$$-\frac{\gamma(C_{t+1} - C_t)}{\Delta t} = \mathbf{A}C_{t+1} + \eta f(C_t). \tag{A16}$$

Taking the time step as $\Delta t = 1$, one gets

$$-\gamma(C_{t+1} - C_t) = \mathbf{A}C_{t+1} + \eta f(C_t). \tag{A17}$$

Finally, the iteration equation of the snake is obtained as

$$C_{t+1} = (\mathbf{A} + \gamma \mathbf{I})^{-1}[\gamma C_t - \eta f(C_t)], \tag{A18}$$

where **I** is the identity matrix.

4. Segmentation of images

In Fig. 14(a) a gray image with three different values of the pixels are shown as white, gray, and black. The dashed lines are the initial snakes, and the solid lines are final snakes. It is clear that, for the pure image, the SN model can reach the boundaries between the different color blocks.

In Figs. 14(b) and 14(c) the internal and external energies of each snake have been given with the number of iterative steps. All the quantities converge very well. The internal energy converges, and this means the shapes of the snakes are no longer changing and the external energy converges to a minimum value, implying that the snakes move to the boundary to be found. The stabilization of both internal and external energies indicates that the snake stays steadily at the boundary to be sought. In Figs. 15(a)–15(c) similar results for a more general SN are shown. Although this task of contour extraction belongs to the field of computer vision and image processing, it is helpful to understand the SN-DCN.

APPENDIX B: PARAMETERS OF THE NEURAL NETWORKS

1. The parameters for obtaining Fig. 5

The parameters of the neural network $\mathcal{N}$ are set as follows: minibatch size $N_b = 1500$, initial learning rate $\alpha_{\mathcal{N}} = 0.01$, learning rate decay $= 0.999$, input layer 256, hidden layer 160, output layer 2, and optimizer = ‘ADAM’. Here the “ADAM” optimizer is implemented with the TENSOR FLOW library [35]. The parameters α, β, γ of $snake_A - snake_C$ are set to $\alpha = 0.05, 0.2, 0.2$, $\beta = 10, 5, 5$, and $\gamma = 0.6, 0.1, 0.1$, respectively. $\xi = 0.1$. The dynamic unit width σ of each node is initialized to 0.05 and is limited in the range from 0.07 to 0.01.

2. The parameters for obtaining Fig. 7

The parameters of the neural network $\mathcal{N}$ are set as follows: minibatch size $N_b = 1500$, initial learning rate $\alpha_{\mathcal{N}} = 0.01$, learning rate decay = 0.998, input layer 256, hidden layer 160, output layer 2, and optimizer = ‘ADAM’. The parameters α, β, γ of $snake_A$ are set to $\alpha = 3$, $\beta = 5$, and $\gamma = 0.2$. The parameters of $snake_B - snake_E$ are set to $\alpha = 2$, $\beta = 0.4$, and $\gamma = 0.22$. $\xi = 0.2$. The dynamic unit width σ of each node is initialized to 0.06 and is limited in the range from 0.08 to 0.02.

[1] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, Machine learning and the physical sciences, Rev. Mod. Phys. **91**, 045002 (2019).

[2] M. A. Biondi, A. T. Forrester, M. P. Garfunkel, and C. B. Satterthwaite, Experimental Evidence for an Energy Gap in Superconductors, Rev. Mod. Phys. **30**, 1109 (1958).

[3] K. K. Darrow, Helium the Superfluid, Rev. Mod. Phys. **12**, 257 (1940).

[4] R. C. Knechtli and J. Y. Wada, Generation and Measurement of Highly Ionized Quiescent Plasmas in Steady State, Phys. Rev. Lett. **6**, 215 (1961).

[5] J. Carrasquilla and R. G. Melko, Machine learning phases of matter, Nat. Phys. **13**, 431 (2017).

[6] L. Wang, Discovering phase transitions with unsupervised learning, Phys. Rev. B **94**, 195105 (2016).

[7] S. J. Wetzel, Unsupervised learning of phase transitions: From principal component analysis to variational autoencoders, Phys. Rev. E **96**, 022140 (2017).

[8] W. Hu, R. R. P. Singh, and R. T. Scalettar, Discovering phases, phase transitions, and crossovers through unsupervised machine learning: A critical examination, Phys. Rev. E **95**, 062122 (2017).

[9] K. Ch'ng, N. Vazquez, and E. Khatami, Unsupervised machine learning account of magnetic transitions in the Hubbard model, Phys. Rev. E **97**, 013306 (2018).

[10] Y. Yang, Z.-Z. Sun, S.-J. Ran, and G. Su, Visualizing quantum phases and identifying quantum phase transitions by nonlinear dimensional reduction, Phys. Rev. B **103**, 075106 (2021).

[11] J. F. Rodriguez-Nieva and M. S. Scheurer, Identifying topological order through unsupervised machine learning, Nat. Phys. **15**, 790 (2019).

[12] M. S. Scheurer and R.-J. Slager, Unsupervised Machine Learning and Band Topology, Phys. Rev. Lett. **124**, 226401 (2020).

[13] A. Lidiak and Z. Gong, Unsupervised Machine Learning of Quantum Phase Transitions Using Diffusion Maps, Phys. Rev. Lett. **125**, 225701 (2020).

[14] J. Wang, W. Zhang, T. Hua, and T.-C. Wei, Unsupervised learning of topological phase transitions using the Calinski-Harabaz index, Phys. Rev. Res. **3**, 013074 (2021).

[15] A. Sornsaeng, N. Dangniam, P. Palittapongarnpim, and T. Chotibut, Quantum diffusion map for nonlinear dimensionality reduction, Phys. Rev. A **104**, 052410 (2021).

[16] Y.-H. Liu and E. P. L. van Nieuwenburg, Discriminative Cooperative Networks for Detecting Phase Transitions, Phys. Rev. Lett. **120**, 176401 (2018).

[17] M. Kass, A. Witkin, and D. Terzopoulos, Snakes: Active contour models, Int. J. Comput. Vision **1**, 321 (1988).

[18] M. Butenuth and C. Heipke, Network snakes: graph-based object delineation with active contour models, Mach. Vision Appl. **23**, 91 (2012).

[19] M. Butenuth and F. Jetzek, Network snakes for the segmentation of adjacent cells in confocal images, in *Bildverarbeitung für die Medizin 2007, Algorithmen, Systeme, Anwendungen, Proceedings des Workshops vom 25.-27. März 2007 in München*, Informatik Aktuell, edited by A. Horsch, T. M. Deserno, H. Handels, H.-P. Meinzer, and T. Tolxdorff (Springer, Berlin, 2007), pp. 247–251, doi: 10.1007/978-3-540-71091-2_50.

[20] L. D. Cohen, On active contour models and balloons, CVGIP: Image Understanding **53**, 211 (1991).

[21] A. I. Guerrero and D. A. Stariolo, The Blume-Capel model in a square lattice with $J_x = -J_y$ interactions in an external field, Physica A **532**, 121839 (2019).

[22] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, Equation of State Calculations by Fast Computing Machines, J. Chem. Phys. **21**, 1087 (1953).

[23] D. Jaksch, C. Bruder, J. I. Cirac, C. W. Gardiner, and P. Zoller, Cold Bosonic Atoms in Optical Lattices, Phys. Rev. Lett. **81**, 3108 (1998).

[24] J. Heaton, I. Goodfellow, Y. Bengio, and A. Courville, Deep learning, Genet. Program Evolvable Mach. **19**, 305 (2018).

[25] D. van Oosten, P. van der Straten, and H. T. C. Stoof, Quantum phases in an optical lattice, Phys. Rev. A **63**, 053601 (2001).

[26] C. W. Bong, C. C. Liew, and H. Y. Lam, Ground-glass opacity nodules detection and segmentation using the snake model, in *Bio-Inspired Computation and Applications in Image Processing*, edited by X.-S. Yang and J. P. Papa (Academic Press, New York, 2016), Chap. 5, pp. 87–104.

[27] J. Arnold, F. Schäfer, M. Žonda, and A. U. J. Lode, Interpretable and unsupervised phase classification, Phys. Rev. Res. **3**, 033052 (2021).

[28] J. Arnold and F. Schäfer, Replacing Neural Networks by Optimal Analytical Predictors for the Detection of Phase Transitions, Phys. Rev. X **12**, 031044 (2022).

[29] S. Tibaldi, G. Magnifico, D. Vodola, and E. Ercolessi, Unsupervised and supervised learning of interacting topological phases from single-particle correlation functions, SciPost Phys. **14**, 005 (2023).

[30] L.-W. Yu, S.-Y. Zhang, P.-X. Shen, and D.-L. Deng, Unsupervised learning of interacting topological phases from experimental observables, Fundam. Res. (2023), doi: 10.1016/j.fmre.2022.12.016.

[31] Y. Yu, L.-W. Yu, W. Zhang, H. Zhang, X. Ouyang, Y. Liu, D.-L. Deng, and L.-M. Duan, Experimental unsupervised learning of non-Hermitian knotted phases with solid-state spins, npj Quantum Inf. **8**, 116 (2022).

[32] R. A. Moore and T. C. Scott, Quantization of second-order Lagrangians: Model problem, Phys. Rev. A **44**, 1477 (1991).

[33] R. A. Moore, Formal quantization of a chaotic model problem, Can. J. Phys. **77**, 221 (1999).

[34] T. C. Scott and D. Andrae, Quantum nonlocality and conservation of momentum, Phys. Essays **28**, 374 (2015).

[35] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, and M. Devin *et al.*, TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems, arXiv:1603.04467.