Machine learning of phase transitions in the percolation and XY models

Wanzhou Zhang,[1,2] Jiayu Liu,[1] and Tzu-Chieh Wei[2]
[1]*College of Physics and Optoelectronics, Taiyuan University of Technology, Shanxi 030024, China*
[2]*C. N. Yang Institute for Theoretical Physics and Department of Physics and Astronomy, State University of New York at Stony Brook, Stony Brook, New York 11794-3840, USA*

(Received 16 May 2018; revised manuscript received 18 January 2019; published 29 March 2019)

In this paper, we apply machine learning methods to study phase transitions in certain statistical mechanical models on the two-dimensional lattices, whose transitions involve nonlocal or topological properties, including site and bond percolations, the XY model, and the generalized XY model. We find that using just one hidden layer in a fully connected neural network, the percolation transition can be learned and the data collapse by using the average output layer gives correct estimate of the critical exponent ν. We also study the Berezinskii-Kosterlitz-Thouless transition, which involves binding and unbinding of topological defects, vortices and antivortices, in the classical XY model. The generalized XY model contains richer phases, such as the nematic phase, the paramagnetic and the quasi-long-range ferromagnetic phases, and we also apply machine learning method to it. We obtain a consistent phase diagram from the network trained with only data along the temperature axis at two particular parameter Δ values, where Δ is the relative weight of pure XY coupling. Aside from using the spin configurations (either angles or spin components) as the input information in a convolutional neural network, we devise a feature engineering approach using the histograms of the spin orientations in order to train the network to learn the three phases in the generalized XY model and demonstrate that it indeed works. The trained network by using system size $L \times L$ can be used to the phase diagram for other sizes ($L' \times L'$, where $L' \neq L$) without any further training.

DOI: 10.1103/PhysRevE.99.032142

I. INTRODUCTION

Recent advancement in computer science and technology has enabled processing big data and artificial intelligence. Machine learning (ML) has been successfully and increasingly applied to everyday life, such as digital recognition, computer vision, news feeds, and even autonomous vehicles [1]. Aside from that, ML methods have also been recently adopted to various fields of science and engineering and, in particular, in the context of phases of matter and phase transitions in physics [2–5]. The main tools are roughly divided into (I) supervised machine learning (classification or regression with labeled training data) and (II) unsupervised machine learning (clustering of unlabeled data) [6].

Amongst the earliest development in this direction of phases of matter, it was used in the study of the thermodynamical phase transitions of the classical Ising model and its gauge variant by supervised machine learning methods [2]. In addition, unsupervised learning was also applied to the Ising model and the XY model mainly by the principal component analysis (PCA) method and the autoencoder (an artificial neural network) [3–5,7]. Other unsupervised methods, such as random trees embedding and t-distributed stochastic neighboring ensemble (SNE), have also been used (see, e.g., [8]). Instead of just learning the transition, a learning scheme called confusion method was invented to predict the phase transitions [9], and similarly a moving-window method was also shown to be useful [10]. Beyond classical physics, the quantum many-body problems have also been studied with artificial neural networks in the description of equilibrium and dynamical properties [11]. For example, the strongly correlated Fermi systems were studied using, e.g., connected networks [12], self-learning methods [13], and even with ML methods beyond limitation of the sign problems [14]. Other systems have also been studied successfully, such as topological phases [15–19], disorder systems [20], quantum percolation model [21], nonequilibrium models [22,23], and many others [24–31]. The machine learning has also been discussed in the context of tensor networks [32,33], and is helpful to accelerate the Monte Carlo sampling and reduce the autocorrelation time [34–36]. Attempts have been made to understand theoretically by mapping it to the renormalization group [37].

Here, we focus on using mostly supervised machine learning methods to study two types of classical statistical models, whose transitions involve nonlocal or topological properties, including site and bond percolations, the XY model and the generalized XY (GXY) models. In order to apply supervised learning to calculate phase boundaries, one needs to prepare Monte Carlo configurations in three regimes [2]: (i) below the suspected T_c, (ii) above the suspected T_c, and (iii) an intermediate regime in a region containing T_c. The first two regimes are used for training: by applying the trained algorithm to configurations in the vicinity of T_c, one can infer an accurate T_c. However, if the purpose is to learn (instead of predicting) the phase transition, configurations from all three regimes are used for both training and testing, with the latter being used to verify that the network indeed can learn the transition and the distinct phases with high confidence [2].

When the model under study has a local order parameter (OP) such as the magnetization, the optimized fully connected network (FCN) actually can recognize phase transitions by essentially averaging over local spins [2]. For the phase transition characterized by the nonlocal order parameters, such as the topological phase of the Ising gauge model [2] or the classical XY model [38], the convolutional neural network (CNN) is a better tool than the FCN as it encodes spatial information. It was demonstrated in the classical two-dimensional (2D) Ising gauge model that the optimized CNN essentially uses violation of local energetic constraints to distinguish the low-temperature from the high-temperature phase [2].

In percolation, nonlocal information such as the wrapping or spanning of a cluster is needed to characterize the phases and the transition in-between. Percolation is one of the simplest statistical physical models that exhibits a continuous phase transition [39–41], and the system is characterized by a single parameter, the occupation probability p of a site or bond, instead of temperature. If a spanning cluster exists in a randomly occupied lattice, then the configuration percolates [42]. Can the neural network be trained to recognize such nonlocal information and learn or even predict the correct critical point? If so, can it be used to reveal other properties of the continuous transition, such as any of the critical exponents? This motivates us to study percolation using machine learning methods.

We first use the unsupervised t-SNE method to characterize configurations randomly generated for various occupation probability p and find that it gives clear separation of configurations away from the percolation threshold p_c. This indicates that other machine learning methods such as supervised ones will likely work, which we also employ. We find that both the FCN and the CNN works for learning the percolation transition, and the networks trained with configurations labeled with information of whether they are generated with $p > p_c$ or $p < p_c$ can result in a data collapse for different system sizes, giving the critical exponent of the correlation length. We alternatively train the neural network to learn the existence of a spanning cluster and with this the percolation transition can be identified (without supplying labels of $p > p_c$ or $p < p_c$).

Our interest in the XY model originates from its topological properties, vortices, and how ML methods can be used to learn the Berezinskii-Kosterlitz-Thouless (BKT) transition. The XY model in the two- and three-dimensional lattices was studied by the unsupervised PCA method [4,5] and generative model [43]. In Ref. [5], various choices of input were considered, e.g., spin configurations (i.e., components of spins), local vortices, and their square into the PCA method. It was concluded that learning the vortex-antivortex unbinding to predict the transition might be difficult in the PCA. In a very recent work by Beach, Golubeva, and Melko [38], it was shown that the CNN works better than FCN to learn the BKT transition. Furthermore, the advantage of using vortices rather than spin configurations only shows up for large system sizes, e.g., $L \gtrsim 32$, but for smaller sizes using spin configurations may work better. Here, for small sizes we use either spin orientations or their components as input to a CNN and verify that both give successful learning of the BKT transition in the XY model. Additionally, we find that using the histograms of the spin orientations also works for the XY model and can be applied efficiently to larger system sizes Additionally, we find that using the histograms of the spin orientations also works for the XY model and the training can be done efficiently for larger system sizes. To go beyond the XY model, we find it interesting to apply the ML to the generalized XY (GXY) model [44,45] as it contains more complex configurations such as half-vortices linked by strings (domain walls) and an additional nematic phase. We find the use of spin configurations (either angles or spin components) and histograms both works. The advantage of the latter approach is that the training can be done for larger system sizes and, moreover, trained network for one system size can be applied to other system sizes.

The outline of this work is as follows. In Sec. II, we introduce models to be studied, i.e., site and bond percolations, the XY model and the GXY models. In Sec. III A, we show classifications via the t-SNE approach on the high-dimensional configurations of site and bond percolation on both square and triangular lattices. Then, supervised learning using the fully connected neural network is illustrated in Sec. III B. In Sec. III C, the CNN structure is introduced and the learning results of percolations by using the CNN are described, and a different way of labeling the configuration (using the existence of a spanning cluster) is used. No labeling regarding $p > p_c$ or $p < p_c$ is used there, but the transition point can be obtained. In Sec. IV we use the CNN to study the XY model and its generalized versions. For the GXY model containing a nematic phase, we construct the histograms of the spin orientations and then use them as images for the network to learn. This way of feature engineering can result in better learning of the transitions. We conclude in Sec. V.

II. MODELS

In this paper we study two types of classical statistical physical models, include (I) site and bond percolations, and (II) the XY model and (III) the GXY models. Let us introduce them as follows.

(Ia) *Site percolation*. The site percolation can be defined by the partition function

$$Z = \sum_{\{\sigma\}} p_s^{n_s^\sigma} (1 - p_s)^{N - n_s^\sigma}, \tag{1}$$

where, e.g., on the square lattice with $N = L \times L$ sites, p_s is the probability of site occupation, n_s^σ is the number of sites being occupied in the configuration labeled by σ. In order to obtain the critical phase transition points by Monte Carlo simulations, usually the wrapping probability R is defined [42] in the case of the periodic boundary condition. With open boundaries, a cluster growing large enough could touch the two opposite boundaries and hence it is referred to a spanning cluster. The wrapping cluster is defined as a cluster that connects opposite sides that would be in the otherwise open boundary condition, as illustrated in Figs. 1(a) and 1(b). A cluster forming along either the x or y direction can contribute to R. In ML method, we do not need to measure this observable directly and a naive labeling of each configuration is given according to how it is generated according to the occupation probability p and p's relation with the critical value (the percolation threshold) p_c, i.e., whether $p > p_c$ (say labeled

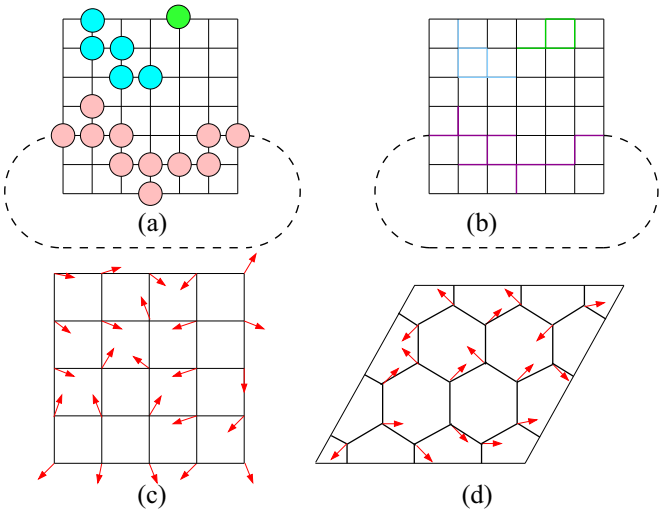

FIG. 1. (a) A site percolation configuration on the square lattice with periodic boundary condition. With the periodic boundary condition, the dashed line connects the opposite sides of the largest cluster and it demonstrates the "wrapping." (b) A bond percolation configuration on a square lattice. (c) Configuration of the XY model on the square lattice. (d) Configuration of the XY model on the honeycomb lattice.

as "1") or $p < p_c$ (say labeled as "0"). This resembles the scenario in the Ising model whether configurations are labeled according to whether they are generated above or below the transition temperature T_c. But, such a topological property of wrapping (or percolating) can be used as an alternative labeling for training the neural network (see Sec. III C 3).

(Ib) *Bond percolation*. The bond percolation partition function can be defined as

$$Z = \sum_{\{\sigma\}} p_b^{n_b^\sigma} (1 - p_b)^{N - n_b^\sigma}, \tag{2}$$

where p_b is the probability of occupying a bond on the lattices, n_b is the number of bonds being occupied in the bond configuration σ. The wrapping or spanning cluster is defined in a similar way as in the site percolation.

(II) *XY model*. The Hamiltonian of the classical XY model [46] is given by

$$H = -J \sum_{\langle i,j \rangle} \vec{s}_i \cdot \vec{s}_j = -J \sum_{\langle i,j \rangle} \cos(\theta_i - \theta_j), \tag{3}$$

where $\vec{s}_i$ is unit vector with two real components and $\langle i, j \rangle$ denotes a nearest-neighbor pair of sites i and j, and θ_i in $(0, 2\pi]$ is a classical variable defined at each site. The sum in the Hamiltonian is over nearest-neighbor pairs or bonds on the square lattice ($L \times L$) with the periodic boundary condition; other lattices can be also considered.

(III) *Generalized XY model*. The Hamiltonian of the classical GXY models is given by

$$H = -\sum [\Delta \cos(\theta_i - \theta_j) + (1 - \Delta)\cos(q\theta_i - q\theta_j)], \tag{4}$$

where Δ is the the relative weight of the pure XY model, and q is another integer parameter that could drive the system to form a nematic phase. For both $\Delta = 0$ and 1 the model reduces to the pure XY model (redefining $q\theta$ as θ in the first case), and hence the transition temperature is identical to that of the pure XY model. However, such a redefinition is not possible with $\Delta \neq 1$. The phase diagrams of the GXY models (4) depend on the integer parameter q, and they have been explored extensively [44,45].

III. PERCOLATIONS

Even though we mainly use supervised learning methods, we will begin the study of percolation using an unsupervised method, the t-distributed stochastic neighbor embedding (t-SNE). We shall see that it can characterize configurations randomly generated in percolation to two distinct groups with high and low probabilities p of occupation as well as a belt containing configurations generated around the percolation threshold $p = p_c$. This gives us confidence to proceed with supervised methods such as FCN and CNN.

A. Learning percolation by t-SNE

The t-SNE is an unsupervised machine learning algorithm for dimensionality reduction via an iteration procedure [47]. By using a nonlinear technique (unlike the PCA), it projects high-dimensional data (e.g., M-dimensional objects $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$) into a two-dimensional space, which can then be visualized in a scatter plot, where similar (or dissimilar) objects are modeled by nearby (or distant) points. (Here, the bold symbols mean that each $\boldsymbol{x}$ is an M-component vector.) For example, it has been successfully used to analyze the Ising configurations and project the data into two-dimensional scattering figures [2].

We show, in Fig. 2(A), for site percolation on the square lattice with size $L = 32$, such a scatter plot, produced by using $M = 11\,000$ site configurations in the t-SNE procedure, where each configuration $\boldsymbol{x}$ contains $32 \times 32 = 1024$ elements 0 or 1. Figure 2(A) is the distribution obtained after only one step of iteration in the t-SNE method. Clearly, after the first iteration, the data for both labels are still mixed together and there is no separation into distinct groups. However, after 2000 iterations, as shown in Fig. 2(B), the data converge into three distinct groups, with two concentrated clusters and a wide "belt." The concentrated cluster with yellow solid circles indicates data generated from nonpercolating (or subcritical) phase, i.e., $p < p_c$, while the purple cluster indicates data from the percolating (or supercritical) phase, i.e., $p > p_c$. In addition to the two distinct clusters, the belt contains the data around the percolation transition point p_c (roughly between 0.2 and 0.8). Similar behavior in the t-SNE analysis of the distribution is also obtained in the percolation study on the triangular lattice, as shown in Figs. 2(a) and 2(b).

We remarked that there are only two colors used in Figs. 2(A), 2(a), 2(B) and 2(b), indicating only above or below p_c, but a continuous hue between pink (dark shade) and yellow (light shade) was used in Figs. 2(C) and 2(c) to denote the occupation probability p. In Figs. 2(C) and 2(c), we show the embeddings for occupation probability p of the *bond* percolation on the square lattice and on the triangular lattice, respectively, both with $L = 16$. The behavior is similar to that of site percolation. The upshot is that the t-SNE method can

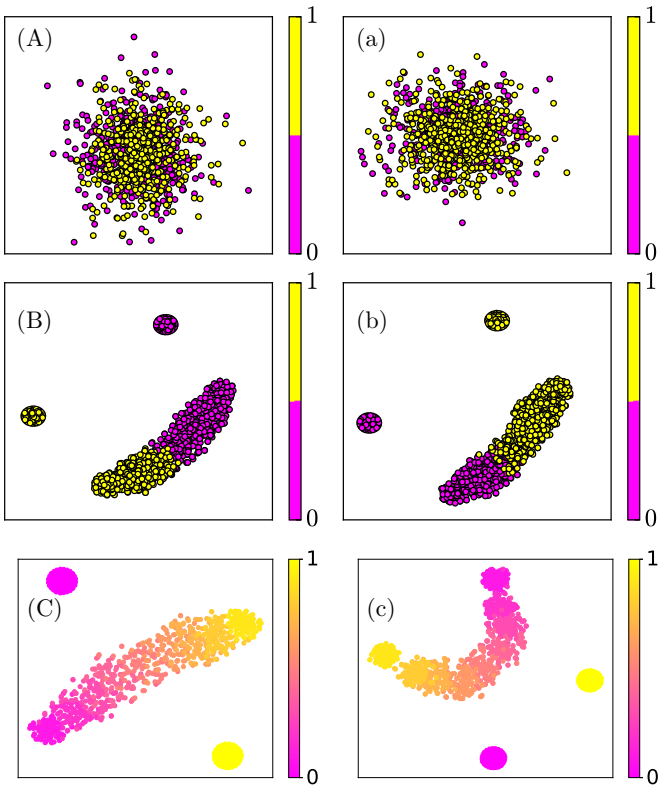

FIG. 2. The t-SNE distribution of site percolation on the square lattice with $L = 32$ for (A) one step of iteration, and (B) 2000 steps of iterations. (C) The result for bond percolation on the square lattice with size $L = 16$ after 1100 iterations. The t-SNE distribution of site percolation on the triangular lattice with $L = 32$ for (a) only one step of iteration and (b) 2000 steps of iteration. (c) The result for the bond percolation on triangular lattice at $L = 16$ after 1100 iterations. Note that in (A), (a), (B), and (b), we only use two colors pink (dark shade) and yellow (light shade), but in (C) and (c) the samples are colored according to the probability p of occupation, from pink (dark shade) to yellow (light shade).

characterize percolation configurations into different phases and near the transition. In order to obtain the transition point p_c, one can probably divide the belt into two halves and the probability value p at the cut can be used as an estimation of the percolation threshold. But, we do not do that here, as we will use supervised learning below to learn the transition more accurately.

B. Learning percolation by FCN

In Fig. 3, we show the structure of the FCN (which we implemented with the TensorFlow library [48]), which consists of one input layer, one hidden layer, and one output layer of neurons. The network between two neighboring layers is fully connected, i.e., each neuron in one layer is connected to every neuron in the previous and next layers. The layers are interconnected by sets of correlation weights (usually denoted by a matrix $\boldsymbol{W}$) and there are biases (denoted by a vector $\boldsymbol{b}$ associated with neurons at each layer, except the input one). The input layer accepts the data sets of images or configurations and then the network processes the data according to the weights and biases, as well as some activation

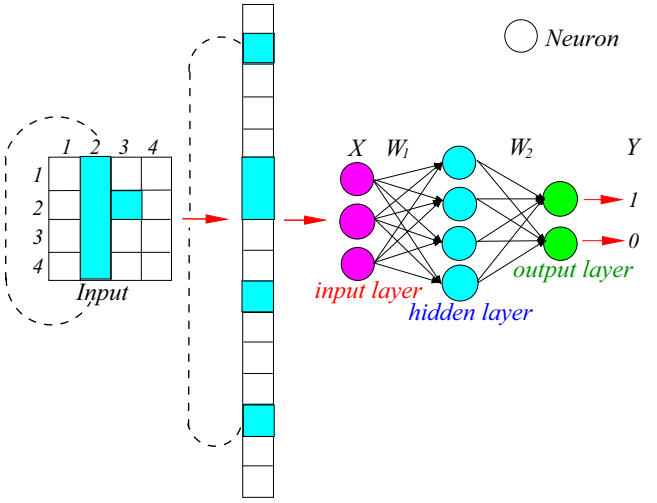

FIG. 3. Architecture of the fully connected neural network used to obtain p_c for the site percolation model. The input is a set of = configurations in a two-dimensional lattice with periodic lattice with size $L = 4$. A percolating configuration is illustrated, where the blue squares are the occupied sites while the empty squares are not occupied.

function for each neuron. The optimization is performed to minimize some cost function, e.g., the cross entropy function in Eq. (8) via the stochastic gradient decent method. The number of neurons in the input layer should be equal to the total number of lattice sites, i.e., $L \times L$ in the site percolation (or the number of spins in the Ising model), or the total number of bonds in the bond percolation. The values of the neurons are denoted as the input vector $\boldsymbol{x}$.

The hidden layer is to transform the inputs into something that the output layer can use and one can employ as many such hidden layers (but we will focus on just one hidden layer in FCN). It determines the mapping relationships. For example, as illustrated in Fig. 3, we denote $\boldsymbol{W_1}$ as the weight matrix from the input to the hidden layer and $\boldsymbol{b_1}$ the bias vector for the hidden layer. The number of neurons in the hidden layers generally is chosen to be approximately of the same order as the size of the input layer or less. Assume the activation function for the neurons in the hidden layer is f. Then, the neurons in the hidden layer will output $\boldsymbol{y_H} = f(\boldsymbol{x} \cdot \boldsymbol{W_1} + \boldsymbol{b_1})$. Denote $\boldsymbol{W_2}$ as the weight matrix from the hidden to the output layer and $\boldsymbol{b_2}$ the bias vector for the output layer. Assume the activation function for the neurons in the hidden layer is g. Then, the neurons in the output layer will have states described by $\boldsymbol{y} = g(\boldsymbol{y_H} \cdot \boldsymbol{W_2} + \boldsymbol{b_2})$. One choice usually used as the activation is the so-called *sigmoid* function

$$\sigma(z) \equiv \frac{1}{1 + e^{-z}}, \tag{5}$$

related to the Fermi-Dirac distribution in physics. Another is the so-called *softmax* function that, when applied to a vector with components z_j, gives another vector with components

$$a_j = \frac{e^{z_j}}{\sum_k e^{z_k}}, \tag{6}$$

and it is related to the Boltzmann weight in physics. Yet another choice that has become popular recently is the rectifier, defined as $f_{\rm rec}(z) = \max(0, z)$, and one unit that uses this activation function is called a rectified linear unit. In principle, one can employ as many hidden layers in the network. It is the universality of the neural network, i.e., it can approximate any given function, that makes machine learning powerful.

These weights $\boldsymbol{W}$'s and biases $\boldsymbol{b}$'s need to be optimized at the training stage. The inputs consist of pairs of $\{\boldsymbol{x} : \boldsymbol{y_T}\}$, where each configuration is represented by a vector $\boldsymbol{x}$ of $L \times L$ components of value being 1 (whether a site is occupied) or 0 (not occupied) and a corresponding label $\boldsymbol{y_T}$ indicating whether the configuration $\boldsymbol{x}$ is generated above or below the percolation threshold p_c. This label can be described by one single binary number, e.g., 1 representing $p > p_c$ and 0 representing $p < p_c$. The cost function can be chosen as (i) the average two-norm between the label vectors $\boldsymbol{y_T}$ and the output layer vector $\boldsymbol{y}$ (resulting from input $\boldsymbol{x}$) over many such pairs,

$$L_2 \equiv \frac{1}{N} \sum_{\boldsymbol{x}} |\boldsymbol{y}(\boldsymbol{x}) - \boldsymbol{y_T}(\boldsymbol{x})|^2, \tag{7}$$

or (ii) the average cross entropy between such pairs

$$C_E \equiv -\frac{1}{N} \sum_{\boldsymbol{x}} \sum_{j} [\boldsymbol{y_T}_j \log \boldsymbol{y}_j + (1 - \boldsymbol{y_T}_j) \log(1 - \boldsymbol{y}_j)]. \tag{8}$$

An additional term called regularization, such as $\lambda/(2N) \sum_{\boldsymbol{i}} |\boldsymbol{W_i}|^2$, is introduced to the cost function in order to prevent overfitting. The optimization is done with stochastic gradient descent using the TensorFlow library.

Once the network is optimized after the training stage, we use as input different and independently generated configurations with possibly different sets of p values, and use the average values of the output layer $\boldsymbol{y}$ (sometimes referred to as the average output layer) from the network to estimate the transition. This is referred to as the test stage. The two components in $\boldsymbol{y}$ are in the range [0,1], and the larger the value of the component (associated with a neuron) gives the more probable prediction the neuron makes. Usually, we plot such average numbers for both output neurons (one is associated label 0 and the other 1). This results in two curves as a function of the probability p, as illustrated in Fig. 4. The value of p at which the two curves cross is used as an estimate of the percolation threshold. Note that, as seen below in Sec. IV, when we encounter three or more phases to identify, then the number of neurons in the output layer will be accordingly three or more.

In Fig. 4(A), we show the two average values y_1 and y_2 in the output layer, obtained by testing the network with site percolation configurations in the range of $0.562 < p < 0.625$ for different system sizes. The FCN was trained using data from $0 < p < 1$ with 2500 samples per value of p. To reduce the statistical errors, 20 000 samples per p are used for the test data to obtain two neurons $y_1(p)$ and $y_2(p)$, i.e., the average values of the output layer. We find that the two corresponding lines cross, and the crossing in Fig. 4(B) as function of $1/L$ is used to estimate the transition point in the thermodynamic limit $L \to \infty$. From this, p_c is estimated to be 0.594 ± 0.002. This agrees with the phase transition from the Monte Carlo

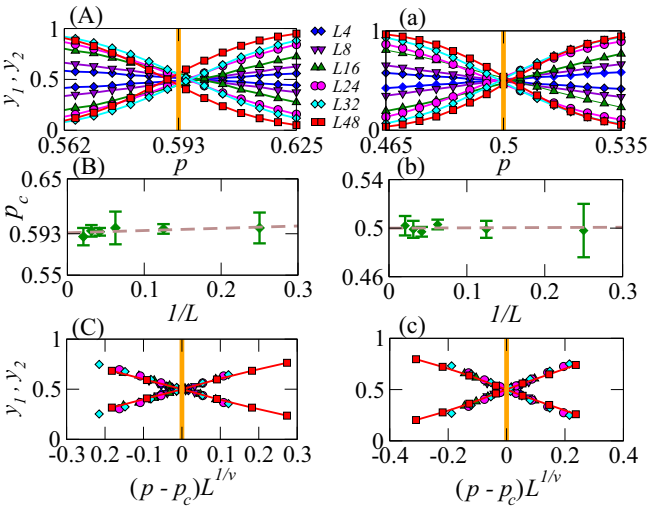

FIG. 4. The two values y_1 and y_2 of the output layer of site percolation on the square lattice (left columns) and triangular lattice (right columns) with the system sizes $L = 4, 8, 16, 24, 32, 48$ using the fully connected network. The second and last rows are, respectively, the average results of the output layer (y_1, y_2), and the finite size scaling for extracting the critical point, the data collapse of y_1 and y_2.

methods within error bars [49]. In Fig. 4(C), we show the data collapse of the two average output layers. Due to the finite size effects, the intersects between different sizes $L = 4, 8, 16, 24, 32$, and 48, are slightly shifted away from the exact p_c. By taking into account this and by rescaling the horizontal axis with $(p - p_c)L^{1/\nu}$, the data collapse very well to a single curve for each neuron output, with the use of the exponent $\nu = \frac{4}{3}$ from percolation theory [50].

During the stage of training, the labeling $\boldsymbol{y_T}$ gives the information whether a configuration is generated at the probability p greater or less than p_c. The information about whether the individual configuration is percolating or not is not known. However, one would expect that for the configurations generated sufficiently away from $p = p_c$, the neural network is learning such a property. But close to p_c even if a configuration is generated at $p < p_c$ it can still be percolating and vice versa even if a configuration is generated at $p > p_c$ it may not be percolating. There are a lot of fluctuations near p_c; see also Figs. 9(e) and 9(f). In Sec. III C 3, we will use the alternative labeling by giving the information of whether a configuration is percolating or not.

We also apply the FCN to site percolation on other lattices. For example, on the triangular lattice, the percolation threshold $p_c = 0.5$ is also exactly known [51]. Similar to the square lattices, we generate the configuration with randomly occupied sites with a statistically independent probability p and then label the configurations according to whether $p > p_c$ or $p < p_c$. After training the FCN, we test FCN with different data sets, the average values of the output layer cross at the phase transition point $p_c = 0.5$ [51]. The data collapse and the finite size scaling are shown in Figs. 4(b) and 4(c), respectively.

Similarly, we use the FCN to study bond percolation, and the results are shown in Figs. 5(A) and 5(B) give the average output layer for the square lattice (with sizes $L =$

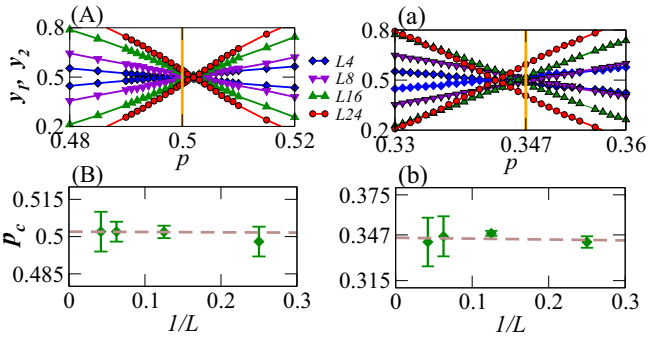

FIG. 5. The two values y_1 and y_2 of the output layer of the machine learning for bond percolation on the square lattice (left columns) and triangular lattice (right columns) with the system sizes $L = 4, 8, 16, 24$ using the fully connected network. The first and the second rows show the average results in the output layer y_1 and y_2 and the finite size scaling for the critical points, respectively.

8, 16, 24, 32), and finite size scaling (yielding the critical point at $p_c = 0.5$), respectively. Figures 5(a) and 5(b) show the results of bond percolation on the triangular lattices. The bond percolation threshold [51] is at $2\sin(\pi/18) \approx 0.347$ and our result agrees with it. We remark that in the training, each input configuration has $2L \times L$ bond variables for the square lattice and $3L \times L$ bond variables for triangular lattices.

Site percolation on the Bethe lattice was studied analytically and exact transition was known, and thus it is interesting to apply the neural network. In Fig. 6(a) we illustrate the Bethe lattice with four shells, indicated by the green dashed lines. Different from the square or triangular lattices, the Bethe lattice with coordination number z (here $z = 3$) has a topological tree structure that expands from a central site out to infinity [52]. Each site has one neighboring site pointing toward the central site and $z - 1$ sites going away from it. The total number of sites in Kth shell is $N_K = z(z-1)^{(K-1)}$. Checking whether the configuration percolates or not by machine learning is interesting because the path connecting any two sites of different trees has to go through the central site. The exact critical probability [52] is known to be at $p_c = 0.5$ and our learning results in Fig. 6(b) show that the FCN can recognize the phase transition after training the network. The results are obtained using $K = 3$ and 5 and the total size is $N = 1 + \sum_1^K N_K$, i.e., 22 and 94.

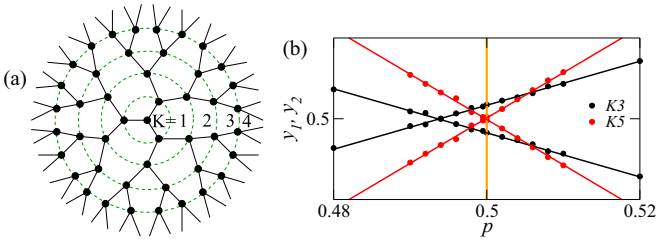

FIG. 6. (a) Bethe lattice with coordination number $z = 3$. The lattice sites are represented by solid circles at different shells $K = 0, 1, 2, \ldots$. (b) The average values y_1 and y_2 in the output layer of site percolation on the Bethe lattice for $K = 3$ and 5.

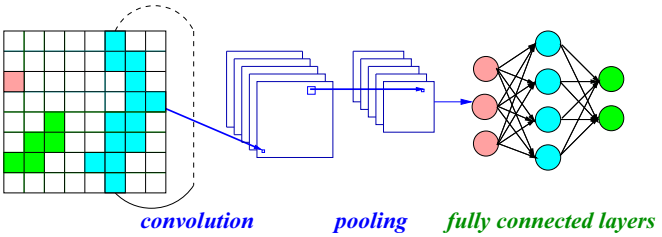

FIG. 7. Architecture of the fully connected neural network used to obtain p_c for the site percolation model. The input is the configurations in a two-dimensional lattice with periodic boundary condition, illustrated with size $L = 4$. The colored or shaded squares are the occupied sites while the white squares are not occupied.

C. Learning percolation by CNN

We have seen in the previous section that the FCN works well in learning percolation transition. However, the information about the lattice structure is not explicitly used, but rather it might be inferred during optimization. For problems that have such natural spatial structure, the CNN is naturally suited and can yield better results.

1. CNN structure

We first begin by discussing the structure of the CNN shown in Fig. 7, where the input is a two-dimensional array or an image. There is a filter with a small size such as 5×5 also called a local receptive field, that processes information of a small region. This same local receptive field moves along the lattice to give a coarse-grained version of the original 2D array or image. We can move the filter not one lattice site but a few (which is usually called *stride* to the next region). We can also pad the outer regions with columns or rows of zeros so as to maintain the same size of the filtered array, which is referred to as *padding*. This results in a filtered or generally coarse-grained hidden layer. We can use many different local receptive fields to obtain many such layers, usually referred to as *kernels*. Roughly speaking, the original image is converted to small ones with different features. For each kernel, a further processing called *maxpooling* is done on nonoverlapping small patches, e.g., 2×2 regions, that further coarse grain the arrays. Other pooling methods can be used. One can repeat such convolution+pooling layers a few times, but we will use one such combination layer in the percolation and two in the later part of the XY and the GXY models. After this, there is a fully connected layer of neurons, as in the FCN. This can be repeated a few more layers, but we will only use one such layer here. Finally, the fully connected layer is connected to a final output layer, and the number of neurons depends on the output type; for example, to distinguish digits 0 to 9, there are 10 output neurons. For distinguishing between two phases, there are two output neurons. Our optimization of the CNN again takes the advantage of the TensorFlow library.

2. Site and bond percolations

We repeat the same study of percolation on square and triangular lattices with CNN. In the CNN structure, we use two combination layers (i.e., convolution+maxpool). In Figs. 8(A) and 8(B), we show the results of site percolation

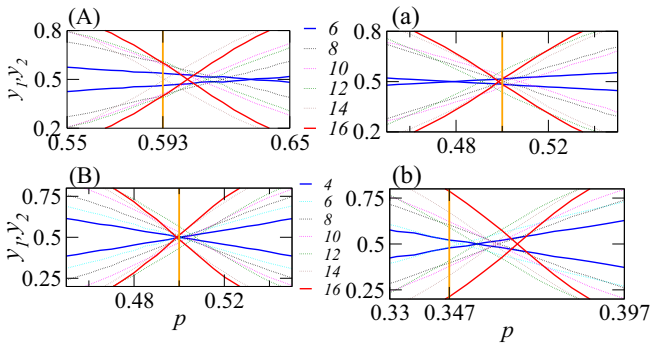

FIG. 8. The two values y_1 and y_2 of the output layer, using the CNN for site [(A) and (a)] and bond percolation [(B) and (b)] on the square (left) and triangular (right) lattices, respectively.

and bond percolation results on the square lattice, respectively. The critical point is converged to 0.593 and 0.5, respectively. During the training, for the site percolation on, e.g., the square lattice, the $L \times L$ site occupation configuration is used as input. For the bond percolation, the $2L \times L$ bond configuration is used. As expected, the CNN works well in learning the phase in percolation. For the site and bond percolation on the triangular lattices, the results are also shown in Figs. 8(a) and 8(b). We note that some slight improvement in the learning of the transition point can be made by careful choosing of p's; see Appendix C. We also comment that since the Bethe lattices are not regular and we do not use CNN for the corresponding percolation problem. But, it should be possible in principle. Moreover, the FCN result of the percolation on the Bethe lattices is good enough.

3. *Labeling by cluster identifying algorithm*

In this section, we would like to show a different ways of training the data using the nonlocal property of whether the configuration is percolating or not as the label $\boldsymbol{y}$. In the previous works for thermodynamic phase transitions [2,5,9], for a configuration $\boldsymbol{x}$ at parameter such as T, the corresponding label y is set to be 0 if $T < T_c$, which means that the configuration is belonging to one phase. If $T > T_c$, the configurations belong to another phase, and the label is set to be 1. Because the configurations around the percolation threshold p_c have fluctuations and the configurations at $p < p_c$ may also be percolating and some configurations at $p > p_c$ may not be percolating, as illustrated in Figs. 9(e) and 9(f). Therefore, it is interesting to label the configuration according to whether or not the configuration has a spanning or wrapping cluster, instead of the relationship between occupation probability p and p_c.

Using the new labeling scheme, we show the results in Figs. 9(a) and 9(c) with sizes $L = 8, 16, 32, 48$ by using FCN and 9(b) and 9(d) with sizes $L = 8, 12, 20$ by using CNN. Here, no information about whether $p > p_c$ or $p < p_c$ is given to the network, and the labels of the configurations are obtained by cluster-identifying algorithm [53]. However, the crossing obtained from the average values of the two output neurons gives the prediction of the percolation transition. They agree with the known results very well.

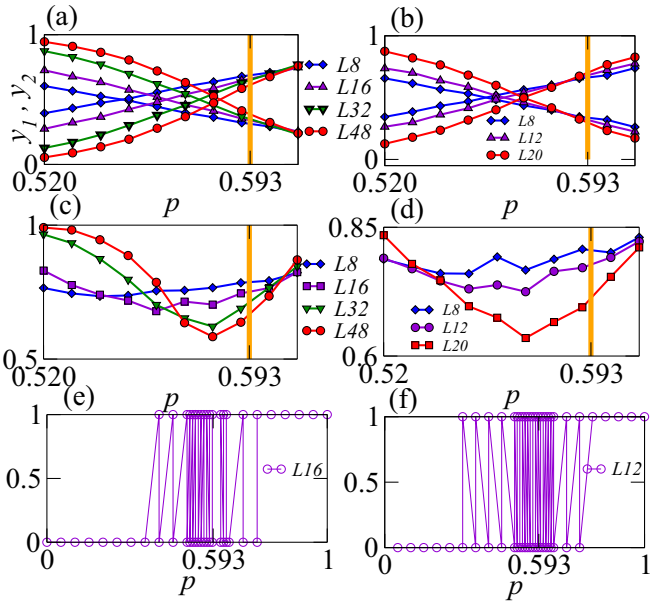

FIG. 9. The two values y_1 and y_2 of the output layer, using the cluster identifying method to label the configuration, with (a) the FCN and (b) the CNN. The respective accuracy curves of (a) and (b) are shown in (c) with FCN and in (d) with CNN. Labels {0, 1} for percolation or not are shown for randomly generated configurations with (e) $L = 16$ and (f) $L = 12$. Notice the fluctuations near the transition.

IV. *XY* AND *GXY* MODELS

We now turn to the second type of models that we are interested in, which includes the pure XY model and the GXY model. As remarked in the Introduction, we shall first use as input to the network either projections of the spin vector onto x axis and y axis, i.e.,

$$\boldsymbol{x} = (\cos\theta_1, \sin\theta_1, \ldots, \cos\theta_N, \sin\theta_N), \tag{9}$$

or the spin orientations $\{\theta_i\}$. The vortices are defined as the winding numbers, i.e., a collection ± 1 for vortices and antivortices, but we shall not use those as input, for the reason remarked earlier due to the results in Ref. [38]. The results in Ref. [38] show that the detection of vortices does not necessarily result in the best classification accuracy, especially for lattices of less than approximately 1000 spins. The advantage may show up for larger sizes, but the training becomes more costly. Here, we limit ourselves to smaller sizes for using spin orientations (or their components) in the training, but later introduce a different approach in feature engineering that can be efficiently applied to larger systems. In terms of the two-dimensional image for the CNN, when we use the spin components, the $L \times L$ sites need to be effectively doubled to $L \times 2L$ that gives the same information of $\boldsymbol{x}$.

In the following, we will focus on the pure XY model on both the square and the honeycomb lattices and the GXY model with $q = 2$, 3 and $q = 8$ on the square lattices.

A. Pure *XY* model

The configurations of XY model on the square and honeycomb lattices are obtained by the classical Monte Carlo

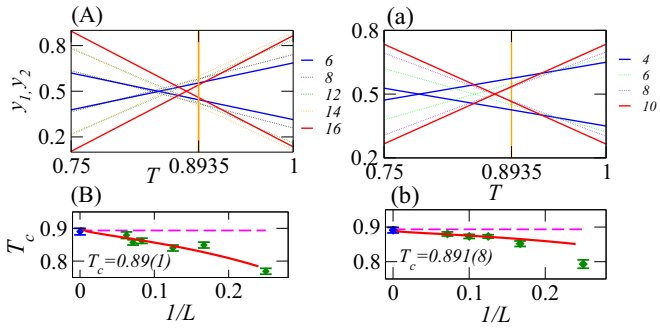

FIG. 10. The two values y_1 and y_2 of the output layer using the CNN for XY model by inputting (A) $\{\theta_i\}$ with sizes $L = 6, 8, 12, 14, 16$ and (a) $\{\sin\theta_i, \cos\theta_i\}$ with sizes $L = 4, 6, 8, 10$ on the square lattices. (B), (b) Finite size scaling of the critical points; the results obtained are consistent with $T_c = 0.8935$ within error bars for both types of the inputs.

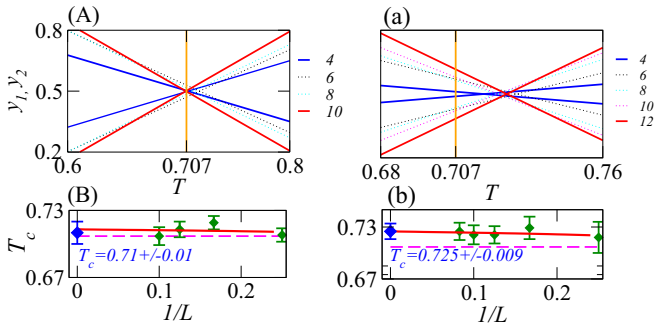

FIG. 11. The two values y_1 and y_2 of the output layer using the CNN for XY model by inputting (A) $\{\theta_i\}$ with sizes $L = 4, 6, 8, 10$ and (a) $\{\sin\theta_i, \cos\theta_i\}$ with sizes $L = 4, 6, 8, 10, 12$ on the honeycomb lattices. (B), (b) Finite size scaling of the critical points; the results obtained are consistent with $T_c = 0.707$ using both types of the inputs.

method; see, e.g., Refs. [54,55]. In the zero temperature limit, the spin at each site will point to the same orientation as the model is ferromagnetic. However, at a finite and small temperature T less than T_{KT}, the spin orientations of spins point almost to the same direction with some fluctuation, but there are excitations in the form of bound vortex-anitvortex pairs. Above the T_{KT}, the spin orientations will become disordered as the vortex-anitvortex pairs unbind.

Since the phase transition points are already known for the XY model on both lattices [46,56,57], i.e., $T_c^{\text{square}} = 0.8935(1)$, and $T_c^{\text{honeycomb}} = \sqrt{2}/2$, respectively, it is thus interesting to see whether or not machine learning could recognize the phase transition of XY model. Although the BKT transition of the XY model has been studied by machine learning methods [38], it is our motivation to go beyond and study the GXY model.

In Figs. 10(A) and 10(a), we show the learning results using the raw spin configurations (i.e., both spin directions $\{\theta_i\}$ and spin components $\{\sin\theta_i, \cos\theta_i\}$) on the square lattices with sizes $L = 4$ to 16. The lines y_1 and y_2 correspond to the average values of the two output neurons. Due to the fact that the performance of the network is lowest near the critical points, we use 25 000 Monte Carlo samples of the training data and of test data for each temperature T. In this way, we obtain results with low standard deviations around the critical point in the range $0.75 < T < 1$.

Figures 10(B) and 10(b) show the dependence on the lattice size L of the estimated critical points T_c. The green symbols are obtained from the intersections. The red curves are fitted by the result from renormalization group [58]:

$$T_c(L) = T_c + \frac{b}{[\log(L)]^2}, \tag{10}$$

where the coefficient $b = \pi^2/4c$ and c is a parameter. In the thermodynamic limit, the estimated transition temperatures are $T_c = 0.89 \pm 0.01$ and $T_c = 0.891 \pm 0.008$ for the pure XY model by using as the input $\{\theta_i\}$ and $\{\cos\theta_i, \sin\theta_i\}$, respectively. The results agree with the result of $T_c = 0.8935$. The CNN indeed works well in learning the BKT transition for the XY model, as previously demonstrated in Ref. [38] on the square lattice, so the success here comes with no surprise.

Using the spin configurations on the honeycomb lattices as the input to the CNN gives good learning results as shown in Fig. 11. We note that the unit cell of the honeycomb has two sites. Each configuration thus has N elements, where $N = 2L \times L$ for using $\{\theta_i\}$ and $N = 4L \times L$ for using $\{\cos\theta_i, \sin\theta_i\}$. Our result agrees decently with the theoretical value $T_c = \sqrt{2}/2 \approx 0.707$. (Some slight improvement can be made by choosing training data generated at temperatures symmetric about T_c and without including those at T_c; see Appendix C and Fig. 20.) In the next section we will study the GXY model to extend the machine learning beyond the XY model.

B. Generalized XY models

1. $q = 2$ and 3

The phase diagram of GXY models [44,45] is rich and two examples with $q = 2$ and 3 are shown, respectively, in Figs. 12(a) and 12(b). In the lower temperature and $\Delta = 0$ limit, the system is in the generalized nematic phase that has q preferred spin orientations. The statistical distribution for spin orientations in the nematic phase displays q peaks as shown in Fig. 12(c). In the $\Delta = 1$ limit, the system is quasi-long-range ferromagnetic (broken reflection symmetry) in low-temperature limit and the distribution of the spin orientations is shown in Fig. 12(d). In the higher temperature, the system becomes disordered, and is in a paramagnetic phase. The distribution for the paramagnetic phase is also shown in Figs. 12(e) and 12(f) for $q = 2$ and 3, respectively. Clearly, the distributions spread through a very wide arrange of angles due to strong thermal fluctuations.

We use the configuration of (1) $\{\theta_i\}$ and (2) $\{\cos\theta_i, \cos\theta_i\}$ as input to the CNN with two convolutional layers, and the network indeed can learn the transition. For both $\Delta = 1$ (pure XY model) and 0 (the GXY model) the phase transition is located at $T/J = 0.8935$. As remarked earlier, this is because the $\Delta = 0$ GXY model isomorphic to the usual XY model [59] by changing the variable $q\theta_i$ as $\bar{\theta}_i$. For both $\Delta = 0$ and 1, using either (1) or (2) as the input works well and the CNN can distinguish, respectively, the quasi-long-range ferromagnetic phase and the nematic phase, from the high-temperature disordered paramagnetic phase. Figure 13(A) displays the

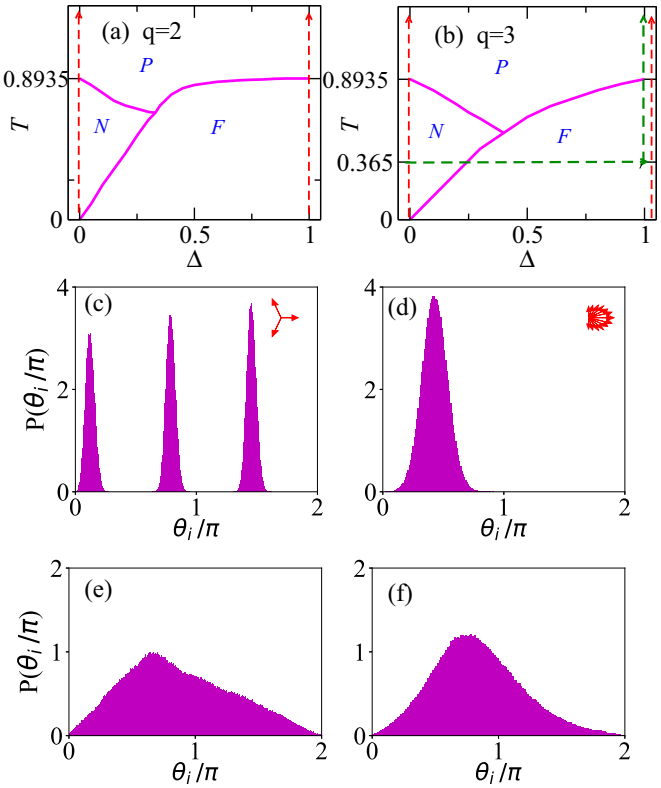

FIG. 12. Phase diagrams of the GXY model for (a) $q = 2$ and (b) $q = 3$, the data are from Refs. [44,45]. The symbols N, P, and F represent the nematic, ferromagnetic, and paramagnetic phases, respectively. The dashed lines show the parameter paths to be scanned. (c) Histograms of the configurations $\{\theta_i\}$ for the nematic phase at $q = 3$, $\Delta = 0$, and $T = 0.2$, the three independent peaks show three preferred orientations. (d) Histogram for the spin orientations in the (quasi-long-range) ferromagnetic phase at $q = 3$, $\Delta = 1$, $T = 0.2$, where the system prefers one spin angle with some fluctuations. (e), (f) Histogram for the paramagnetic phase at $q = 2$, $\Delta = 0$, $T = 2$ (e), and the paramagnetic phase at $q = 3$, $\Delta = 0$, $T = 2$ (f), respectively. Note that for illustrations, these distributions are obtained by over 2000 samples. But for the input to the neural network, we use histograms each derived from a very small number of samples, such as 20.

average values of the output layer and the performance of learning by using $\{\theta_i\}$ as input for $q = 2$ at $\Delta = 0$, with $L = 6, 8, 12, 14, 16$, while that in Fig. 13(a) is obtained from using $\{\sin\theta_i, \cos\theta_i\}$ as input. The resulting two curves y_1 and y_2 from the two neurons in the output layer can distinguish different phases and their crossing gives the transition. In the thermodynamic limit, the critical points are estimated to be $T_c = 0.92 \pm 0.03$ and $T_c = 0.92 \pm 0.05$, respectively, using the two different types of input.

Here, in the network there are 32 and 64 kernels in the first and second layers, respectively. We use the Wolff-cluster algorithm to generate configurations for the GXY model. To obtain enough equilibrated states, we throw out the configurations during the first 10 000 Monte Carlo steps. To avoid the correlations between configurations, we pick up configurations with intervals of 2–5 Monte Carlo steps at each temperature.

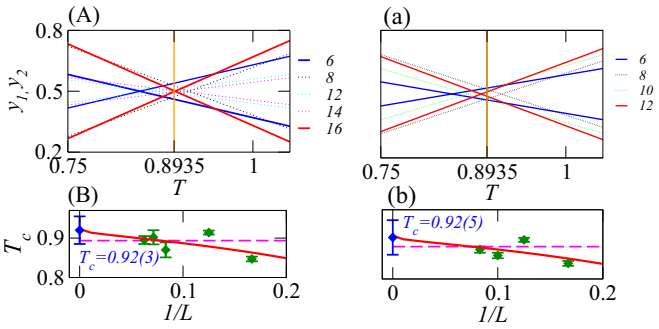

FIG. 13. The two values y_1 and y_2 of the output layer of learning of (A) $\{\theta_i\}$ and (a) $\{\sin\theta_i, \cos\theta_i\}$ for (a) $q = 2$, $\Delta = 0$ with various lattice sizes. (B), (b) The estimated critical points from the dependence on lattices sizes $1/L$. In the thermodynamical limit, $T_c = 0.92(3)$ and $T_c = 0.92(5)$ are estimated.

Inspired by the main difference between the above three phases being the shape of the histograms, we investigate whether using such feature engineering (i.e., histograms) can help the learning better. In principle, spin configurations from each sample in the Monte Carlo algorithm generate a histogram. However, to make the histogram smooth, we use multiple configurations to average (e.g., 20) for small system sizes. Employing Wolff-cluster algorithm allows us to access larger lattices, we can use just one single configuration to generate a histogram. After obtaining the histogram, we segment the images into a 32×32 matrix of black and white pixels, in which the white area is set to 0 and colorful area is set to 1. These matrices of pixels are our engineered feature and are used as the input to the CNN for training.

We directly recognize the histograms and obtain results as accurate as other kinds of inputs, using the two-layer-convolution CNN as shown in Figs. 14(a)–14(d) along the four red dashed lines in Figs. 12(a) and 12(b). For both $\Delta = 0$ and 1, the GXY model has the phase transition located at $T/J = 0.8935$. The results of using histograms as the engineered feature make the CNN able to recognize different phases in the generalized XY model and the associated transitions in these two limits of Δ. For completeness, we also scan a path in the phase diagram of $q = 3$ model by first varying Δ at $T/J = 0.365$ (for which the $\Delta_c = 0.25$ [45]), and the results shown in Figs. 14(e) and 14(f) demonstrate that the neural network, in particular, can also distinguish the nematic phase from the ferromagnetic phase and learn the transition point. The approach of using histograms helps us to access larger lattices without any further training.

Armed with the success of learning two distinct phases, we move on to test whether our method can be used to distinguish three phases. In particular, we scan the phase diagram in Fig. 12(b) along the green dashed lines, combining the previous horizontal path varying Δ (at $T/J = 0.365$) and then the vertical path by varying T/J (at $\Delta = 1$); this path cuts through the three phases: N, F, and P. To do this, we need to use three neurons in the output layer. During the training of the network, the configurations in the three phases are labeled as 0, 1, and 2, respectively. The sigmoid function maps the output layer located between the range [0, 1]. In the test stage, the first neuron in the output layer (representing the N phase)

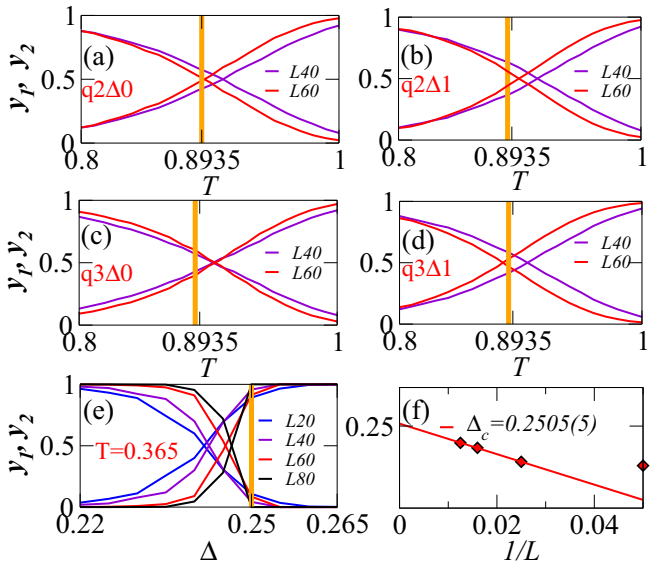

FIG. 14. The two values y_1 and y_2 of the output layer in the learning of histograms of $\{\theta_i\}$ for (a) $q = 2$, $\Delta = 0$; (b) $q = 2$, $\Delta = 1$; (c) $q = 3$, $\Delta = 0$; (d) $q = 3$, $\Delta = 1$. (e) The average results in the output layer and (f) the finite size scaling of the critical points by scanning the parameter Δ, and critical point Δ_c is estimated to be 0.2505(5).

is 1 for $\Delta < 0.25$ and becomes zero at other two phases (the F and P phases). The output of the other two neurons shows converse behavior as shown in Fig. 15(a). There are three curves corresponding to the three neurons in the output layer. The accuracy is shown in Fig. 15(b) and it equals to 100% at noncritical regimes and decreases near the two critical points around ($\Delta = 0.25$, $T = 0.365$) and ($\Delta = 1$, $T = 0.89$). In short, we have demonstrated successful learning of three phases (N, P, and F) and the transitions.

As a final test, we use a semisupervised method to retrieve the global phase diagram of GXY model with $q = 2$ and 3 on the square lattice of 12×12 sites. What we mean by the semisupervised method is that, only the limited data that are not generic are used in the training, for example,

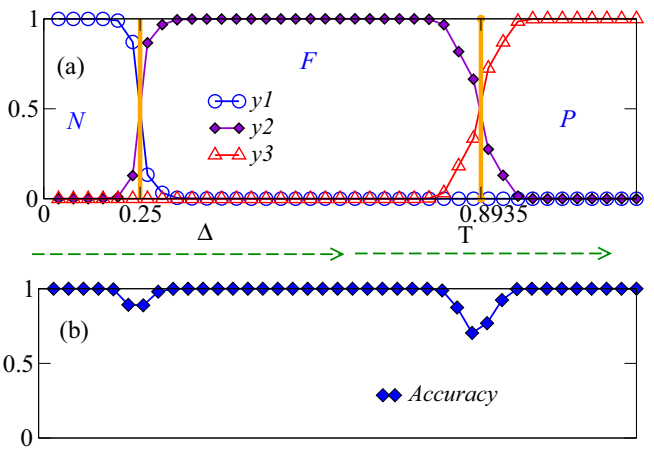

FIG. 15. (a) The results in averaging outcomes separately in the three neurons of the output layer by scanning the green dashed lines in Fig. 12(b). (b) The accuracy of the learning.

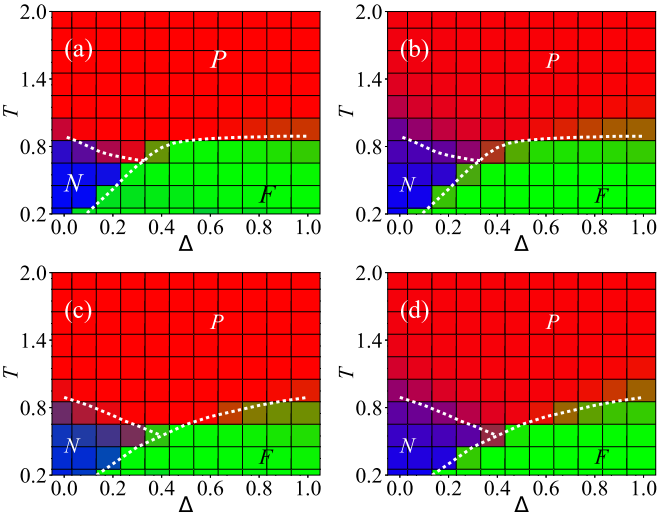

FIG. 16. The phase diagram of the GXY model, which contains nematic phase (N, blue), ferromagnetic phase (F, green), and paramagnetic phase (P, red), obtained by the semisupervised method discussed in the text. The neural network is trained only for data from $\Delta = 0$ and 1. (a) The $q = 2$ model: the input data for training are the histogram of spin orientations $\{\theta_i\}$. (b) The $q = 2$ model: the input data for training are the spin orientations. (c) The $q = 3$ model: the input data for training are the histogram of spin orientations. (d) The $q = 3$ model: the input data for training are spin orientations θ_i. These agree with those (from Monte Carlo simulations) in Figs. 12(a) and 12(b). The phase boundaries represented by white lines by Monte Carlo methods are also shown.

those along $\Delta = 0$ and 1, i.e., the two vertical paths in Figs. 14(a) and 14(b). These are nongeneric, as they only give very limited representations in the phase diagram. Once the network is trained and optimized, we use the neural network to predict the phases, using the configurations generated from Monte Carlo in the whole phase diagram, with parameters (Δ, T) covering the range $\Delta = 0, 0.1, 0.2, \ldots, 1$ and $T = 0.2, 0.4, \ldots, 1.8, 2$. We use both the spin orientation and the histogram as the input, and the phase diagrams thus obtained, as shown in Fig. 16, agree well with those in Figs. 12(a) and 12(b).

2. $q = 8$

For $q > 3$, the results in Refs. [60,61] suggest the existence of new phases at intermediate values of Δ that do not appear for either $\Delta = 0$ or 1. Since the existence and/or nature of some of those transitions are still disputed, it would be interesting to see the outcome of the neural networks in those cases.

Without loss of generality, a typical value $q = 8$ is chosen. In Fig. 17(a), the phase diagram, containing N, P, and F phases, is shown. The blue dashed lines are the data from Refs. [60,61]. The color of the small squares is the mapping of the average values in the output layer y_N, y_{F_2}, y_P, and y_F of the two-layer CNN. The color is mapped via $z = y_P + 20 * y_N + 40 * y_{F_2} + 60 * y_F$ and its normalized expression $z = [z - \min(z)]/\max(z)$. Aside from the previous phases of the $q = 3$ model, a new phase F_2 phases emerges, consistent with Refs. [60,61].

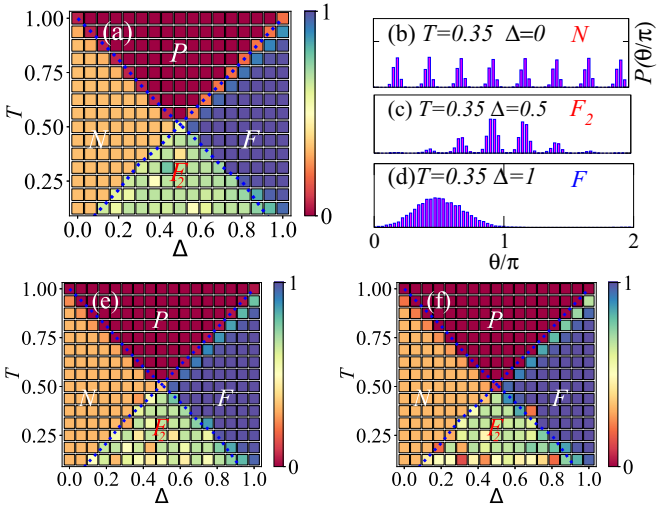

FIG. 17. (a) The phase diagram of the GXY model with $q = 8$, containing N, P, F, and F_2 phases from Ref. [60]. The network is trained from the spin histograms of size $L = 100$. (b) Illustration of the distributions of spins directions in the nematic phase show 8 peaks. (c) The distributions of spins directions in F_2 phase. (d) The distributions of spins directions in F phase. Using the trained network of (a), we obtain the phase diagrams of systems with sizes $L = 150$ (e) and $L = 200$ (f).

Here, we comment on the way we train the CNN. First, the lattice size should be large enough such as $L = 100$. By fixing the temperatures at $T = 0.35$ and 0.8, we scan the parameter Δ from 0 to 1 in the phase diagram with an interval of 0.05. For each parameter point, 2500 histogram samples for training are used and 25 000 samples of histograms for testing, and four labels for the different phases are used. We can distinguish the new emerged phase F_2 by the nonzero labels. At the same time, the boundaries between other phases are also obtained, consistent with the known results (indicated by the blue dotted lines).

In Figs. 17(b)–17(d), the distributions of spins directions $P(\theta/\pi)$ for each phase are shown according to configurations, generated by the Wolff-cluster Monte Carlo method. Different from the case of $q = 3$, $P(\theta/\pi)$ in the $q = 8$ nematic phase has 8 peaks at $T = 0.35$, $\Delta = 0$. In the F_2 phase, as shown Fig. 17(c), some spin directions (peaks) dominate the configuration at $T = 0.35$, $\Delta = 0.5$. In the F phase, all of the directions for the spins are restricted in a half-plane $0 < \theta < \pi$.

In order to have efficient machine learning algorithms with large data sets, the representation of the data (e.g., configurations from Monte Carlo simulations) can affect both efficiency and accuracy. In this paper, one idea is to use distribution of spin orientations as the input to the neural network, instead of the complete spin configurations. In the usual machine learning, the size of the latter ($L \times L$ spins) can be so large that the training time can take too long. However, transforming the data into distributions can reduce the cost of training, as the distribution of the data is a much smaller set.

Furthermore, the neural network trained by histograms of spin orientations from a system of size $L \times L$ can be used to test the distributions from other system sizes without further training. In Figs. 17(e) and 17(f), the phase diagrams, consistent with the previous one, are obtained by inputting the test data from systems with sizes 150×150 and 200×200, respectively. The network used for the test is from that training using data from the 100×100 system size. We note that the size of our histogram images was chosen and fixed more or less arbitrarily at, e.g., 32×32, and that other (larger) sizes can be used to increase accuracy. Using previously trained network avoids training the network repeatedly for other system sizes.

V. CONCLUSION

In summary, we have used machine learning methods to study the percolation, the XY model, and the GXY model in the two-dimensional lattices. For the percolation phase transition, the unsupervised t-SNE can map the high-dimensional data sets of configurations into a two-dimensional image with classifiable data. Using the FCN even without explicitly giving the two-dimensional spatial structure still allows to recognize the percolation phase transition. By feeding the information about the existence or not of the spanning cluster, the transition can be predicted without any training information on whether the configurations are generated with $p > p_c$ or $p < p_c$. The percolation exponent ν was obtained correctly using results from the output neurons. We have also demonstrated that the CNN method works well for percolation, but there is no substantial advantage using CNN. The advantage of the CNN against the FCN arises in the study of the XY model and the generalized XY models.

The pure XY model on the square and honeycomb lattice in our study was learned by inputting the spin configurations $\{\cos\theta_i, \sin\theta_i\}$, $\{\theta_{i,j}\}$ and even with small sizes such as $L = 4, \ldots, 16$, the critical point could be obtained by performing the finite size scaling. For the generalized XY model with $q = 2$, 3, and 8, the global phase diagrams were obtained by a semisupervised method, i.e., with a network trained by learning just some limited set of the data. Specifically, for $q = 8$, the new phase F_2 in the range of $0 < \Delta < 1$ is also be confirmed through the perspective of machine learning.

The use of spin configurations as the naive input works for training the network to recognize phases in the XY model. One key difference between the phases is the probability distribution of the spin orientations. We have devised a feature engineering using the histograms of the spin orientations instead, and this has resulted in successful learning of various phases in the generalized XY model beyond the XY model. Moreover, the trained network with system size $L \times L$ can also be used for testing data from other system sizes ($L' \times L'$, where $L' \neq L$), saving additional training effort. The use of machine learning in phases of matter in general may still need the ingenuity of appropriate features for the neural network to learn, but can become a useful tool.

Note added. We recently learned of a very interesting paper by Suchsland and Wessel [62]. Even though it is beyond the scope of this paper, but as a future direction, it will be interesting to employ the methods presented there in percolation and the generalized XY models. (They have already analyzed the XY model.)

ACKNOWLEDGMENTS

W.Z. is grateful for the valuable discussion with C. X. Ding on Monte Carlo simulations and gratefully acknowledges financial support from China Scholarship Council and the NSFC under Grant No. 11305113. T.-C.W. is supported by the National Science Foundation under Grants No. PHY 1314748 and No. PHY 1620252.

APPENDIX A: t-SNE METHOD

Here, we summarize for convenience the t-distributed stochastic neighbor embedding (t-SNE) method [47], which is an improvement from the stochastic neighbor embedding (SNE) method [63]. The main idea of these methods is to, from a set of high-dimensional data represented by high-dimensional vectors $\{x_i\}$, obtain a corresponding set of low-dimensional (e.g., two- or three-dimensional) data, represented by a set of vectors $\{y_i\}$ such that the latter maintains key features of the former. In the t-SNE method, the pairwise similarity q_{ij}'s for a pair of low-dimensional vectors y_i and y_j is defined as

$$q_{i,j} = \frac{(1+||y_i - y_j||^2)^{-1}}{\sum_{k\neq l}(1+||y_k - y_l||^2)^{-1}}, \tag{A1}$$

whereas that for the high-dimensional vectors is defined as $p_{i,j} \equiv (p_{j|i} + p_{i|j})/(2n)$, where n is the total number of vectors and

$$p_{j|i} = \frac{\exp\left(-||x_i - x_j||^2/2\sigma_i^2\right)}{\sum_{k\neq l}\exp\left(-||x_k - x_l||^2/2\sigma_i^2\right)}. \tag{A2}$$

The lower-dimensional vectors y's are obtained by using a gradient descent approach by minimizing the cost function

$$C = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}, \tag{A3}$$

where the gradient by varying y_i's is given by

$$\frac{\delta C}{\delta y_i} = 4\sum_j (p_{ij} - q_{ij})(y_i - y_j)(1+||y_i - y_j||^2)^{-1}. \tag{A4}$$

In the above, the standard deviation σ_i's are obtained by fixing the so-called perplexity (supplied by the user)

$$\mathrm{Perp}(P_i) = 2^{-\sum_j p_{j|i}\log_2 p_{j|i}}, \tag{A5}$$

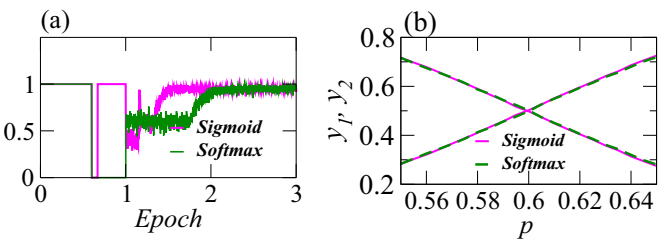

FIG. 18. (a) The training accuracy by using the sigmoid and softmax activation, respectively. (b) For the site percolation on the square lattices, we find the average output layers are the same using enough samples (25 000).

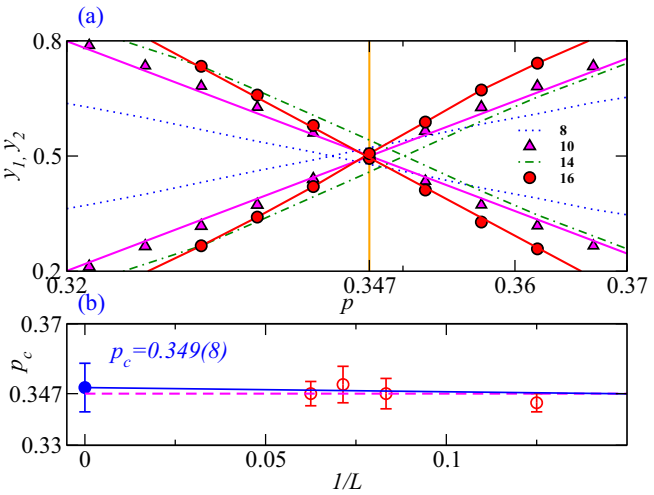

FIG. 19. The result of bond percolation on the triangular lattice via training the CNN with configurations generated symmetrically with respect to p_c. This is to be compared with Fig. 8(b), and the results here show some improvement.

which is typically chosen between 5 and 50, according to the performance. Here, we set it to be 20.

The initial low-dimensional vectors y_i's are generated randomly. Iterating the gradient descent procedure will yield an improved approximation consecutively, until the gradient is very small.

APPENDIX B: ACTIVATION FUNCTIONS OF SIGMOID AND SOFTMAX

There are many activation functions to be used in the neural network; here we use the sigmoid function for the FCN. Taking the site percolation as an example for test, we output

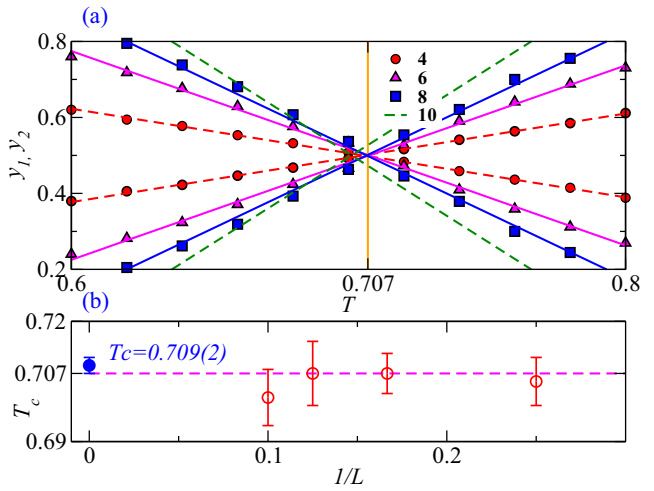

FIG. 20. (a) Averaged values of the two neurons in the output layer, y_1 and y_2, for the XY model on the honeycomb lattice by choosing the training data $\{\cos\theta_i, \sin\theta_i\}$ below T_c and above T_c symmetrically with respect to T_c but not including it. (b) The finite size scaling of T_c and the value in the thermodynamic limit is estimated to be 0.709(2).

the results with 25 000 samples per occupation probability. In Fig. 18(a), the training accuracy of using the sigmoid function reaches an equilibrated stage faster than that by the softmax function. It is also found that the results of using softmax and sigmoid function are almost the same as shown in Fig. 18(b). Therefore, enough samples can avoid systematic error induced by different activation functions.

APPENDIX C: CHOOSING TRAINING DATA

There are several factors that could affect the trained network. In this Appendix, we compare two choices of the training data and the resultant learned transition points, i.e., intersections of y_1 and y_2. The first approach for the training data is to use configurations corresponding to probabilities p's uniformly, as we have used in most of the plots in the main text, such as the percolation in Fig. 8. The second choice is to use configurations generated at p's symmetric with respect to p_c (but not including those at p_c), which is demonstrated in Fig. 19.

We remark that the training data used in Fig. 8 were obtained at p's that are uniform. By judiciously making the choice such that these p's are symmetric with respect to the transition point (excluding the transition point), better learning of the transition can be made.

We also test this for the XY model on the honeycomb. In Fig. 20(a), y_1 and y_2 for the two neurons in the output layer are plotted and the result $T_c = 0.709(2)$ from finite size scaling in Fig. 20(b) appears more accurate than $T_c = 0.729(8)$ in Fig. 11(b).

[1] M. Jordan and T. Mitchell, Machine learning: Trends, perspectives, and prospects, Science **349**, 255 (2015).

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

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

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

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

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, Cambridge, MA, 2016).

[7] C. Wang and H. Zhai, Unsupervised learning of frustrated classical spin models I: Principle component analysis, Phys. Rev. B **96**, 144432 (2017).

[8] 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).

[9] E. van Nieuwenburg, Y. Liu, and S. Huber, Learning phase transitions by confusion, Nat. Phys. **13**, 435 (2017).

[10] P. Broecker, F. Assaad, and S. Trebst, Quantum phase recognition via unsupervised machine learning, arXiv:1707.00663.

[11] G. Carleo and M. Troyer, Solving the quantum many-body problem with artificial neural networks, Science **355**, 602 (2017).

[12] K. Ch'ng, J. Carrasquilla, R. Melko, and E. Khatami, Machine Learning Phases of Strongly Correlated Fermions, Phys. Rev. X **7**, 031038 (2017).

[13] J. Liu, H. Shen, Y. Qi, Z. Meng, and L. Fu, Self-learning Monte Carlo method in fermion systems, Phys. Rev. B **95**, 241104 (2017).

[14] P. Broecker, J. Carrasquilla, R. Melko, and S. Trebst, Machine learning quantum phases of matter beyond the fermion sign problem, Sci. Rep. **7**, 8823 (2017).

[15] G. Torlai and R. Melko, A Neural Decoder for Topological Codes, Phys. Rev. Lett. **119**, 030501 (2017).

[16] Y. Zhang and E. A. Kim, Quantum Loop Topography for Machine Learning, Phys. Rev. Lett. **118**, 216401 (2017).

[17] D. Deng, X. Li, and S. Sarma, Exact machine learning topological states, Phys. Rev. B **96**, 195145 (2017).

[18] D. Deng, X. Li, and S. Sarma, Quantum Entanglement in Neural Network States, Phys. Rev. X **7**, 021021 (2017).

[19] P. Zhang, H. Shen, and H. Zhai, Machine Learning Topological Invariants with Neural Networks, Phys. Rev. Lett. **120**, 066401 (2018).

[20] F. Schindler, N. Regnault, and T. Neupert, Probing many-body localization with neural networks, Phys. Rev. B **95**, 245134 (2017).

[21] T. Mano and T. Ohtsuki, Phase diagrams of three-dimensional Anderson and quantum percolation models using deep three-dimensional Convolutional Neural Network, J. Phys. Soc. Jpn. **86**, 113704 (2017).

[22] M. Bukov, A. Day, D. Sels, P. Weinberg, A. Polkovnikov, and P. Mehta, Reinforcement Learning in Different Phases of Quantum Control, Phys. Rev. X **8**, 031086 (2018).

[23] J. Venderley, V. Khemani, and E. Kim, Machine Learning Out-of-Equilibrium Phases of Matter, Phys. Rev. Lett. **120**, 257204 (2018).

[24] G. Torlai, G. Mazzola, J. Carrasquilla, M. Troyer, R. Melko, and G. Carleo, Neural-network quantum state tomography, Nat. Phys. **14**, 447 (2018).

[25] L. Li, T. E. Baker, S. R. White, and K. Burke, Pure density functional for strong correlations and the thermodynamic limit from machine learning, Phys. Rev. B **94**, 245129 (2016).

[26] D. Crawford, A. Levit, N. Ghadermarzy, J. Oberoi, and P. Ronagh, Reinforcement Learning Using Quantum Boltzmann Machines, Quantum Inf. Comput. **18**, 0051 (2016).

[27] K. I. Aoki and T. Kobayashi, Restricted Boltzmann machines for the long range Ising models, Mod. Phys. Lett. B **30**, 1650401 (2016).

[28] C. Li, D. Tan, and F. Jiang, Applications of neural networks to the studies of phase transitions of two-dimensional Potts models, Ann. Phys. **391**, 312 (2018).

[29] L. Arsenault, A. Lopez-Bezanilla, O. Lilienfeld, and A. Millis, Machine learning for many-body physics: The case of the Anderson impurity model, Phys. Rev. B **90**, 155136 (2014).

[30] G. Torlai and R. G. Melko, Learning thermodynamics with Boltzmann machines, Phys. Rev. B **94**, 165134 (2016).

[31] X. Gao and L. Duan, Efficient representation of quantum many-body states with deep neural networks, Nat. Commun. **8**, 662 (2017).

[32] E. M. Stoudenmire and D. J. Schwab, Supervised learning with quantum-inspired tensor networks, Adv. Neural Inf. Process. Syst. **29**, 4799 (2016).

[33] J. Chen, S. Cheng, H. Xie, L. Wang, and T. Xiang, On the equivalence of restricted Boltzmann machines and tensor network states, Phys. Rev. B **97**, 085104 (2018).

[34] L. Huang and L. Wang, Accelerate Monte Carlo simulations with restricted Boltzmann machines, Phys. Rev. B **95**, 035105 (2017); L. Wang, Exploring cluster Monte Carlo updates with Boltzmann machines, Phys. Rev. E **96**, 051301(R) (2017).

[35] J. Liu, Y. Qi, Z. Meng, and L. Fu, Self-learning Monte Carlo method, Phys. Rev. B **95**, 041101 (2017).

[36] N. Portman and I. Tamblyn, Sampling algorithms for validation of supervised learning models for Ising-like systems, J. Comput. Phys. **350**, 871 (2017).

[37] P. Mehta and D. Schwab, An exact mapping between the variational renormalization group and deep learning, arXiv:1410.3831.

[38] M. Beach, A. Golubeva, and R. Melko, Machine learning vortices at the Kosterlitz-Thouless transition, Phys. Rev. B **97**, 045207 (2018).

[39] S. R. Broadbend and J. M. Hammersley, Percolation processes. I. Crystals and Mazes, Proc. Cambridge Philos. Soc. **53**, 629 (1957).

[40] J. M. Hammersley, in *Percolation Structure and Process*, edited by G. Deutscher, R. Zallen, and J. Adler (Adam Hilger, Bristol, 1983).

[41] G. Grimmett, *Percolation* (Springer, New York, 1989).

[42] J. P. Hovi and A. Aharony, Scaling and universality in the spanning probability for percolation, Phys. Rev. E **53**, 235 (1996).

[43] M. Cristoforetti, G. Jurman, A. I. Nardelli, and C. Furlanello, Towards meaningful physics from generative models, arXiv:1705.09524.

[44] D. Hübscher and S. Wessel, Stiffness jump in the generalized XY model on the square lattice, Phys. Rev. E **87**, 062112 (2013).

[45] F. Poderoso, J. Arenzon, and Y. Levin, New Ordered Phases in a Class of Generalized XY Models, Phys. Rev. Lett. **106**, 067202 (2011).

[46] S. Chung, Essential finite-size effect in the 2D XY model, Phys. Rev. B **60**, 11761 (1999); P. Olsson, Monte Carlo analysis of the two-dimensional XY model. II. Comparison with the Kosterlitz renormalization-group equations, *ibid.* **52**, 4526 (1995).

[47] L. van der Maaten, Accelerating t-SNE using tree-based algorithms, J. Mach. Learn. Res. **15**, 3221 (2014); L. van der Maaten and G. Hinton, Visualizing non-metric similarities in multiple maps, Machine Learning **87**, 33 (2012); Visualizing high-dimensional data using t-SNE, J. Mach. Learn. Res. **9**, 2579 (2008).

[48] M. Abadi, A. Agarwal, P. Barham *et al.*, TensorFlow: Large-scale machine learning on heterogeneous distributed systems, arXiv:1603.04467.

[49] M. Newman and R. Ziff, Efficient Monte Carlo Algorithm and High-Precision Results for Percolation, Phys. Rev. Lett. **85**, 4104 (2000).

[50] M. Levenshteĭn, B. Shklovskiĭ, M. Shur, and A. Ĕfros, The relation between the critical exponents of percolation theory, Zh. Eksp. Teor. Fiz. **69**, 386 (1975) [Sov. Phys. JETP **42**, 197 (1976)].

[51] M. Sykes and J. Essam, Exact critical percolation probabilities for site and bond problems in two dimensions, J. Math. Phys. **5**, 1117 (1964).

[52] A. Saberi, Recent advances in percolation theory and its applications, Phys. Rep. **578**, 1 (2015).

[53] M. E. J. Newman and R. M. Ziff, Fast Monte Carlo algorithm for site or bond percolation, Phys. Rev. E **64**, 016706 (2001).

[54] R. Swendsen and J. Wang, Nonuniversal Critical Dynamics in Monte Carlo Simulations, Phys. Rev. Lett. **58**, 86 (1987).

[55] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, Equations of state calculations by fast computing machines, J. Chem. Phys. **21**, 1087 (1953).

[56] Y. Hsieh, Y. J. Kao, and A. W. Sandvik, Finite-size scaling method for the Berezinskii-Kosterlitz-Thouless transition, J. Stat. Mech.: Theory Exp. (2013) P09001.

[57] B. Nienhuis, Exact Critical Point and Critical Exponents of O(n) Models in Two Dimensions, Phys. Rev. Lett. **49**, 1062 (1982); Y. Deng, T. Garoni, W. Guo, H. Blöte, and A. Sokal, Cluster Simulations of Loop Models on Two-Dimensional Lattices, *ibid.* **98**, 120601 (2007).

[58] D. R. Nelson and J. M. Kosterlitz, Universal Jump in the Superfluid Density of Two-Dimensional Superfluids, Phys. Rev. Lett. **39**, 1201 (1977).

[59] G. A. Canova, F. Poderoso, J. J. Arenzon, and Y. Levin, Reply to "Incommensurate vortices and phase transitions in two-dimensional XY models with interaction having auxiliary minima" by S. E. Korshunov, arXiv:1207.3447.

[60] G. A. Canova, Y. Levin, and J. J. Arenzon, Competing nematic interactions in a generalized XY model in two and three dimensions, Phys. Rev. E **94**, 032140 (2016).

[61] G. A. Canova, Y. Levin, and J. J. Arenzon, Kosterlitz-Thouless and Potts transitions in a generalized XY model, Phys. Rev. E **89**, 012126 (2014).

[62] P. Suchsland and S. Wessel, Parameter diagnostics of phases and phase transition learning by neural networks, Phys. Rev. B **97**, 174435 (2018).

[63] G. E. Hinton and S. T. Roweis, Stochasitic neighbor embedding, *Advances in Neural Information Processing Systems* (The MIT Press, Cambridge, MA 2002), Vol. 15, pp. 833–840.