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, NY 11794-3840, USA*

(Dated: April 10, 2018)

In this paper, we apply machine learning methods to study phase transitions in certain statistical mechanical models, whose transitions involve non-local 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 anti-vortices, in the classical XY model. As pointed out by M. Beach, A. Golubeva and R. Melko [Phys. Rev. B **97**, 045207 (2018)], using spin orientations (i.e. angles) rather than vortex configuration gives better results in learning the transition, we thus try alternatively the spin components as the input information in a convolutional neural network and verify that this indeed works for learning the transition in the XY model on both the square and honeycomb lattices. To go beyond the XY model, we apply also the machine learning methods to the generalized XY model, which consists of a nematic phase, in addition to the paramagnetic and the quasi-long-range ferromagnetic phase. We find that the use of spin configurations (either angles or spin components) works and, additionally, using the histograms of the spin orientations also works for both the XY model and the generalized XY model in learning the transition. We use such a feature engineering approach to train the network to learn the three phases in the generalized XY model. We obtain a consistent phase diagram from the network trained with only data along the temperature axis at two particular parameter values, i.e. $\Delta = 0$ and $\Delta = 1$.

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]. Besides 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 divided into supervised machine learning (training the neural network to recognize the aims based on the previous known results) and unsupervised machine learning (classifying the information and predicting the results) [6].

Amongst the earliest development in this direction of phases of matter, it was used in the study of the thermal 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) used for unsupervised learning [3–5, 7]. Instead of just learning the transition, a learning scheme called confusion method was invented to predict the phase transitions [8], and similarly a moving-window method was also proposed and shown to be useful [9]. Beyond classical physics, the quantum many-body problems have also been studied with artificial neural networks in the description of equilibrium and dynamical properties [10]. For example, the strongly correlated Fermi systems were studied using e.g. connected networks [11], self-learning methods [12], and even with ML methods beyond the sign problems [13]. Other systems have also been studied successfully, such as topological phases [14–18], disorder systems [19], non-equilibrium models [20, 21] and many others [22–29]. The machine learning has also been discussed in the context of tensor networks [30, 31], and is helpful to accelerate the Monte Carlo sampling and reduce the auto correlation time [32–34]. Attempts have been made to understand theoretically by mapping it to the renormalization group [35].

Here we focus on using mostly supervised machine learning methods to study two types of classical statistical models, whose transitions involve non-local or topological properties, including site and bond percolations, the XY model and the generalized XY models. The general idea of supervised machine learning in phase transitions is to input configurations, e.g., of Ising spins generated by Monte-Carlo sampling at various temperatures into the neural network, and training the network to recognize or distinguish phases (giving e.g. $T > T_c$ or $T < T_c$ as training labels associated with configurations), as done in the work of Carrasquilla and Melko [2]. Then other independently generated sets of configurations are used to test how well the network has learned the phases and the associated transition. If the model 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 non-local order parameters, such as topological phase of the gauge model [2] or the XY model [36], the convolutional neural network (CNN) is a better tool than the FCN as it encodes spatial information. It was demonstrated in the classical 2d Ising gauge model that optimized the CNN essentially uses violation of local energetic constraints to distinguish the low-temperature from the high-temperature phase.

In percolation, non-local information such as the wrapping or spanning of a cluster is needed to characterize the percolation phase transition. Percolation is one of the simplest statistical physical models that show a continuous phase transition [37–39], and the system is characterized by single a 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 [40]. Can the neutral network be trained to recognize such nonlocal information and learn or even predict the correct critical point, as well as reveal other properties of the continuous transition, such as any 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 other machine learning methods such as supervised ones will 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 neutral 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 BKT transition. The XY model in the two- and three-dimensional lattices were studied by the unsupervised PCA method [4, 5]. In Ref. [5], various choices of input were considered, e.g. spin configurations (i.e. components of spins), local vorticities, and their square into the PCA method. It was concluded that learning the vortex-antivortex unbinding transition might be difficult in the PCA. In a very recent work by Beach, Golubeva and Melko [36], it was shown that using spin orientations works better than vortex configurations in the learning of the transition and moreover, the CNN performs much better than the FCN. Motivated by these works, we use either spin orientations or their components as input to a CNN and verify that this gives successful learning of the BKT transition in the XY model. To go beyond the XY model, we find it interesting to apply the ML to the generalized XY (GXY) model [41, 42] 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) indeed works. Additionally, we find that that using the histograms of the spin orientations also works for both the XY model and the generalized XY model in learning the transition.

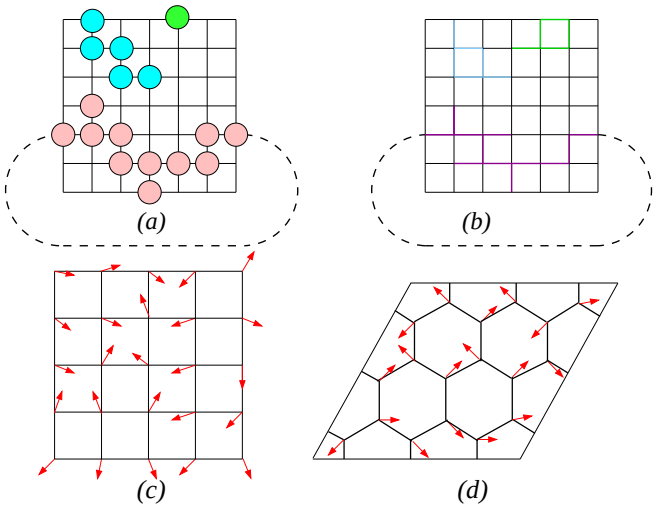

FIG. 1: (Color online) (a) A site percolation configuration on the square lattice with periodic boundary condition. Using the periodic boundary condition, the dashed line connecting the opposite sides of the largest cluster means "wrapping". (b) A bond percolation configuration on a square lattice. (c) Configuration of XY model on the square lattice. (d)Configuration of XY model on the honeycomb lattice.

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 generalized XY 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. Percolation on the Bethe lattice is also discussed in Sec. III C. In Sec. IV A, the CNN structure is introduced and the learning results of percolations by using the CNN are described in Sec. IV B, and a different way of labeling the configuration (using the existence of a spanning cluster) is used in Sec. IV C. No labeling regarding $p > p_c$ or $p < p_c$ is used there, but the transition point can be obtained. In Sec. V we use the CNN to study the XY model ands its generalized versions. For the generalized XY 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. VI.

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 generalized XY 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 numbers 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 [40] in the case of the periodic boundary condition. With open boundaries, a cluster growing large enough could touch the two opposite boundaries is hence called 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 Fig. 1 (a) and (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 if the label 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 as '1') or $p < p_c$ (say labeled as '0').

(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 [43] 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, 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 generalized XY 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 coupling strength between nearest-neighbor spins, and q is another integer parameter that could drive the system to form a nematic phase. The model will be reduced to the pure XY model if the coupling $\Delta = 1$ and the phase diagram has been explored extensively [41, 42].

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 close to 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

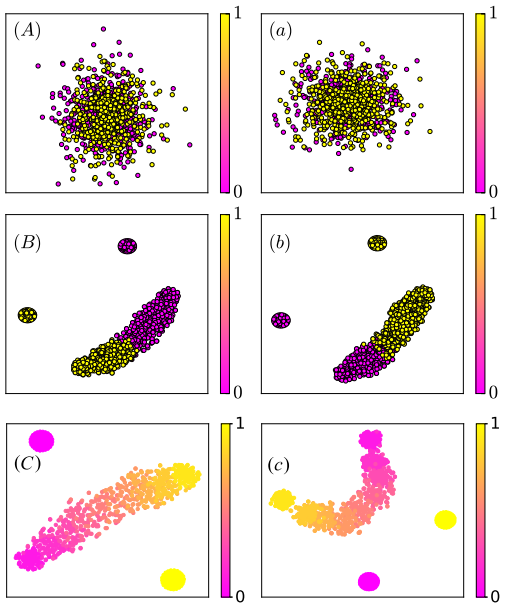

FIG. 2: (Color online) The t-SNE distribution of site percolation on the square lattice with $L = 32$ for (A) one time of iteration, and (B) 2000 times of iterations. (C) The result for bond percolation on the square lattice with size $L = 16$ after 1100 iterations. The t-SNE distribution of site percolation on the triangular lattice with $L = 32$ for only iteration (a) one time and (b) 2000 times 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, but in (C) and (c) the color indicates the occupation probability p.

The t-SNE is an unsupervised machine learning algorithm for dimensionality reduction via an iteration procedure [44]. 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 means that each $\boldsymbol{x}$ is a 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 inputting $M = 11000$ site configurations into the t-SNE toolbox, where each configuration $\boldsymbol{x}$ contains $32 \times 32 = 1024$ elements 0 or 1. It 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 converges into three distinct groups, with three concentrated clusters and a wide 'belt'. The condensed cluster by yellow solid circles indicates non-percolating (or subcritical) phase while the purple condensed cluster means the percolating (or supercritical) phase. The belt contains the data around the percolation transition point p_c. There are only two colors used in Figs. 2 (A) & (B). 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) & (b). In Fig. 2 (C), we show the occupation probability p of the bond percolation on the square lattice with size $L = 16$, where the color there instead denotes the occupation probability p ranging from 0 'continuously' to 1. Clearly, the data corresponding to percolation and non-percolation are concentrated at the two small balls. The belt corresponds to the data around the critical point p_c, and similarly for the bond percolation data on the triangular lattice shown in Fig. 2 (c). The upshot is that the t-SNE method can characterize percolation configurations into different phases and near the transition.

B. Learning percolation by FCN

In Fig. 3, we show the structure of the FCN (which we implemented with the TensorFlow library [45]), which consists of one input layer, one hidden layer and one output layer of neutrons. 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 layer. The layers are interconnected by sets of correlation weights (usually denoted by a matrix $\boldsymbol{W}$) and there are biases (denoted by a vectro $\boldsymbol{b}$ associated with neurons at each layer (except the input one). The input layer accepts the data sets of images or configurations and

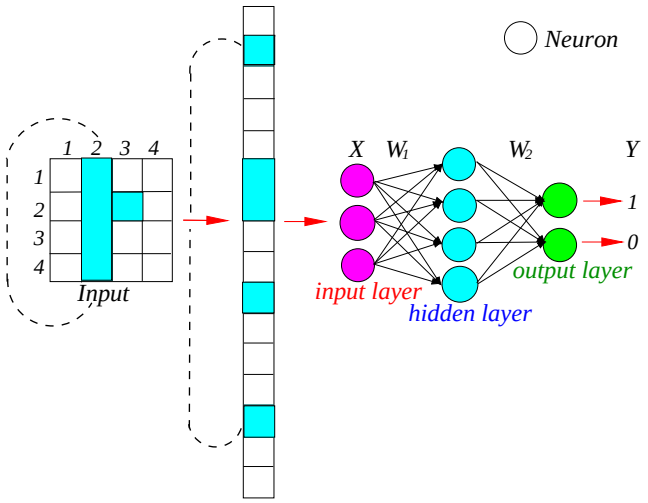

FIG. 3: (Color online) Architecture of the fully connected neural network used to obtain p_c for the site percolation model. The input are the configurations in a two-dimensional lattices with periodic lattice with size $L = 4$. The typical configuration is a percolation configuration, where the blue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

then the network will process the data according to the weights and biases, as well as some activation function for each neuron. The optimization is performed to minimize some cost function, e.g., the cross entropy function in Eq. (8) and 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 there can be 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 that is usually used as the activation include 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. In principle, one can 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.

It is these weights $\boldsymbol{W}$'s and biases $\boldsymbol{b}$'s that need to be optimized at the training stage. The inputs consists 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 expressed as a vector, e.g. the two-component vector $\boldsymbol{y_T} = (1, 0)$ representing $p > p_c$ and $\boldsymbol{y_T} = (0, 1)$ 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,

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

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 different and independently generated configurations with possibly different sets of p values, and use the average of the output layer $\boldsymbol{y}$ 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 (or neuron) gives the more probable prediction the neuron makes. Usually we plot such average numbers for both output neurons and result 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.

In Fig. 4 (A), we show the average of values in the output layer, obtained by testing the network with site percolation configurations in the range of $0<p<1$ for different system sizes. We find that the lines cross and the crossings, and in Figure 4 (C) the positions of intersection as function of $1/L$ is used to estimate the transition point in the thermal dynamical limit $L \to \infty$, for which

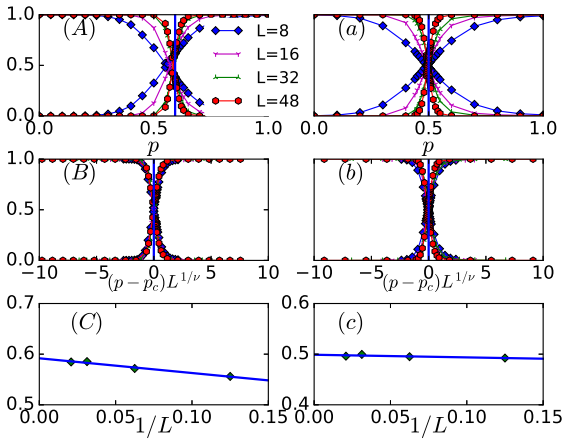

FIG. 4: (Color online) The results of site percolation on the square lattice (left columns) and triangular lattice (right columns) with the system sizes $L = 8, 16, 32, 48$ using the fully connected network. The first, the second and last rows are, respectively, the average results of the output layer,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

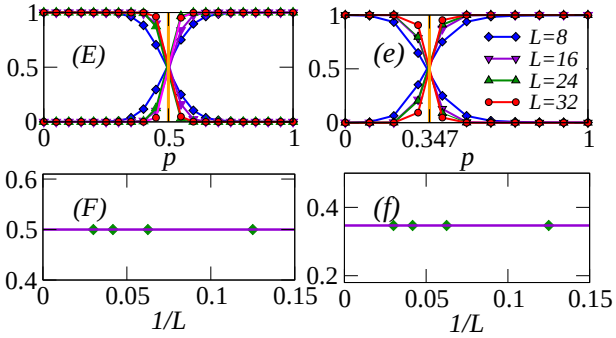

FIG. 5: (Color online) The results of the machine learning for bond percolation on the square lattice (left columns) and triangular lattice (right columns) with the system sizes $L = 8, 16, 32, 48$ using the fully connected network. The first and the second rows are the average results in the output layer and the finite size scaling of the critical points, respectively.

p_c is converged to 0.593. This agrees with the phase transition from the Monte-Carlo methods [46]. In Fig. 4 (B), we show the data collapse of the two average output layers. Due to the finite size effects, the intersects between different sizes $L = 8, 16, 32, 48$, are slightly shifted away from the exact p_c. By taking into accout of 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 $\nu = 4/3$ from percolation theory [47].

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 $p < p_c$ can be percolating and vice versa even if a configuration is generated at $p > p_c$ may not be percolating. There are a lot of fluctuations near p_c; see also Figs. 9 (e) and (f). In Sec. IV C, 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 [48]. 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 output layers cross at the phase transition point $p_c = 0.5$ [48]. The data collapse and the finite size scaling are shown in Figs. 4 (b) and (c), respectively.

Similarly, we use the FCN to study bond percolation, and the results are shown in Figs. 4 (E) and (F) give the average output layer for the square lattice (with sizes $L = 8, 16, 24, 32$), and finite size scaling (yielding the critical point at $p_c = 0.5$), respectively. Figs. 4 (e) and (f) show the results of bond percolation on the triangular lattices. The bond percolation threshold [48] is at $2\sin(\pi/18) \approx 0.347$ 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.

C. Site percolation on the Bethe 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, i.e., 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 a tree structure that expands from a central site out to infinity [49]. Each site has one neighboring site pointing towards the central site and $z-1$ sites going away from it. The total number of sites in K-th shell is $N_K = z(z-1)^{(K-1)}$. In 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 [49] 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 results are obtained using $K = 3, 4, 5$ and total sizes up to $N = 1 + \sum_1^K N_K$ and the sites are up to 22, 46, 94.

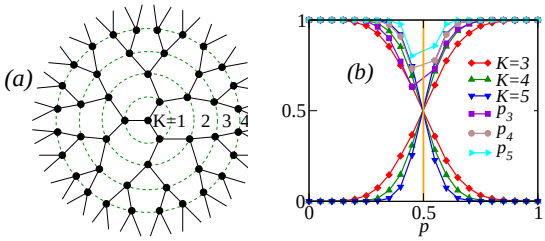

FIG. 6: (Color online) (a) Bethe lattice with coordination number $z = 3$. The lattice sites are represented by solid circles at different shells $K = 0, 1, 2, \cdots$. (b) Average results in the output layer (labeled by $K = 3, 4, 5$) and the network's performance (labeled by P_3, P_4, P_5) of site percolation on the Bethe lattice.

IV. LEARNING PERCOLATION BY CNN

We have seen in the previous section that the FCN works very 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.

A. CNN structure

We first begin by discussing the structure of the CNN shown in Fig. 7, where in the input is from a two-dimensional array or an image. There is a filter with a small size such as 5×5 or 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. 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

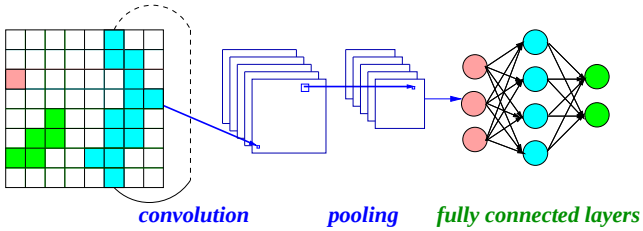

FIG. 7: (Color online) Architecture of the fully connected neural network used to obtain p_c for the site percolation model. The input is the configurations in a two-dimensional lattice with periodic boundary condition, illustrated with size $L = 4$. The the colored or shaded squares are the occupied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

different local receptive fields to obtain many such layers, usually referred to as *kernels*. Roughly speaking, the original image is converted small ones. For each kernel, a further processing called *max-pooling* is done on non-overlapping 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 generalized XY models. After this, then there is a fully connected layer of neurons, as in the FCN. This can be repeated a few more layers, but we will not do that 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 ten 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.

B. Site and bond percolations

We repeat the same study of percolation on square and triangular lattices with CNN. In the CNN structure, we just use one combination layer (convolution+maxpool) which contains 32 kernels. In Figs. 8 (A) and (B), we show the results of site percolation and bond percolation results on the square lattice, respectively. The critical point is converged to 0.593 and 0.5, respectively, as shown in Fig. 8 (C). 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 very well in learning the phases in percolation.

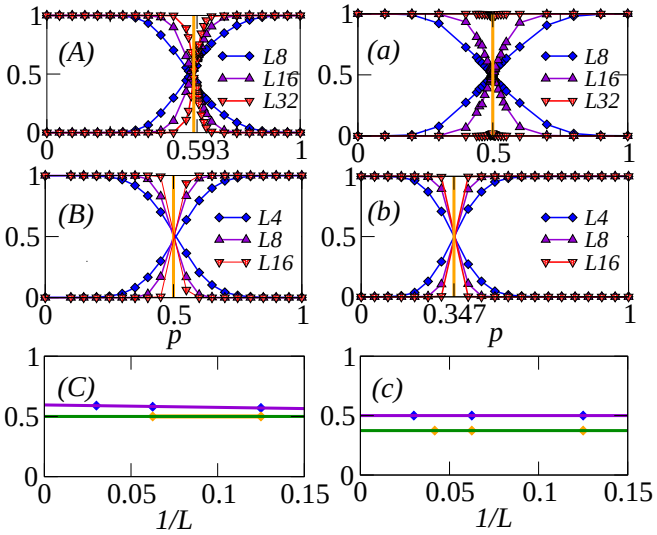

FIG. 8: (Color online) The results of using the CNN for site and bond percolation on the square (left) and triangular (right) lattices, respectively.

C. A different way of training

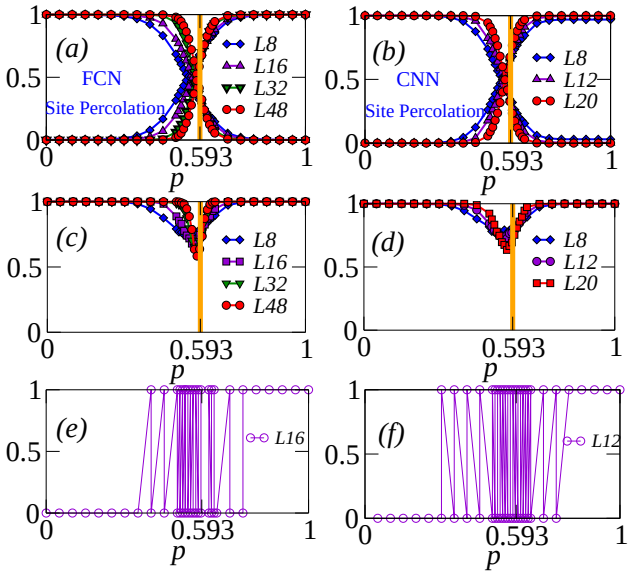

FIG. 9: (Color online) The average results of the output layer, using a different way of labeling the configuration, with (a) the FCN and (b) the CNN. The respective accuracy of (a) and (b) is shown, respectively, in (c) with FCN and (d) with CNN. Labels $\{0,1\}$ for (e) $L=16$ and (f) $L=12$ and notice the fluctuations near the transition.

In this subsection, we would like to show a different ways of training the data using the non-local property of whether the configuration is percolating or not as the label $\boldsymbol{y}$. In the previous works for thermal phase transitions [2, 5, 8], 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 are belong to another phase, and the label is set be 1. Actually, for percolation, because the configurations around the p_c has fluctuations and the configurations at $p<p_c$ may be percolating and some configurations at $p>p_c$ are not percolating, as illustrated in Fig. 9 (e) and (f). Clearly, there is a belt around $p_c=0.593$ where a lot of configurations at $p<p_c$ are percolating and a lot of configurations at $p>p_c$ are not percolating. 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 (c) with sizes $L=8,16,32,48$ by using FCN and (b) and (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. 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.

V. THE XY MODELS

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

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

or the spin orientations $\{\theta_i\}$. The vorticitices are defined as the winding numbers, i.e., a collection ± 1 for vortices and anti-vortices, but we shall not use those as input, for the reason remarked earlier due to the results in Ref. [36]. In term 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 generalized XY model with $q = 2$ and $q = 3$ on the square lattice.

A. The pure XY model

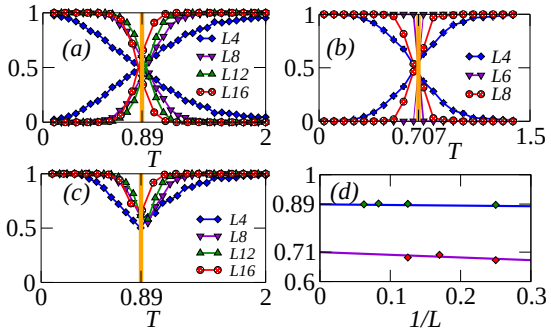

FIG. 10: (Color online) Average results of the neurons in the output layer using the CNN for XY model by inputting {sin $\theta_{i,j}$, cos $\theta_{i,j}$} on (a) the square and (b) honeycomb lattices. (c) The accuracy for learning the XY model on square lattice. (d) Finite-size scaling of the critical points; the results obtained are 0.89 and 0.71 respectively, for the square and honeycomb lattices.

The configurations of XY model on the square and honeycomb lattices are obtained by the classical Monte-Carlo method; see e.g. Refs [50, 51]. The spin at each site is described by its orientation θ_i or a unit vector (cosθ_i, sinθ_i), and therefore, each configuration in the latter case has $2N$ elements as in Eq. (9), where $N = L \times L$ for the square lattice and $N = 2L \times L$ for the honeycomb lattices, the latter due to that the unit cell of has two sites. In the zero temperature limit, the spin at each site will point to the same orientation because the model is ferromagnetic. However at a finite and small temperature T less than T_{KT}, the orientations of spins points almost to the same directions with some fluctuation, but there are excitations in the form of bound vortex-anitvortex pairs. Above the T_{KT}, the orientations will become disordered as the vortex-anitvortex pairs unbind.

Since the phase transition points are already known for the XY model on the both lattices[43, 52], i.e. , $T_c^{\rm Square} = 0.89213$, and $T_c^{\rm Honeycomb} = \sqrt{2}/2$, respectively, thus it is interesting to see whether or not machine learning could recognize the phase transition of XY model. The authors of Ref. [36] showed that the CNN works better than the FCN and that spin orientations work better than the vortex configurations in the machine learning for the XY model on the square lattice.

In Figs. 10 (a) and (b), we show the learning results using the raw spin configurations (i.e. spin componenets {sin$\theta_{i,j}$, cos$\theta_{i,j}$}) on the square and the honeycomb lattices. The learning accuracy for the square-lattice case is shown in Fig. 10 (c). We find the intersection of the two curves (corresponding to average of the two output neurons) under finite-size scaling converges well to the known values of T_{KT} in both cases, even for not very large sizes used, as shown in Fig. 10(d). The CNN indeed works well in learning the BKT transition for the XY model, as previously demonstrated in Ref. [36] on the square lattice, so the success comes with no surprise. We also verify that using the angles of the spins as the input to the CNN, as done in Ref. [36] gives good learning results for the transition on both square and honeycomb lattices as shown in Fig.11. To go beyond the XY model, in the next section, we will study the generalized XY model.

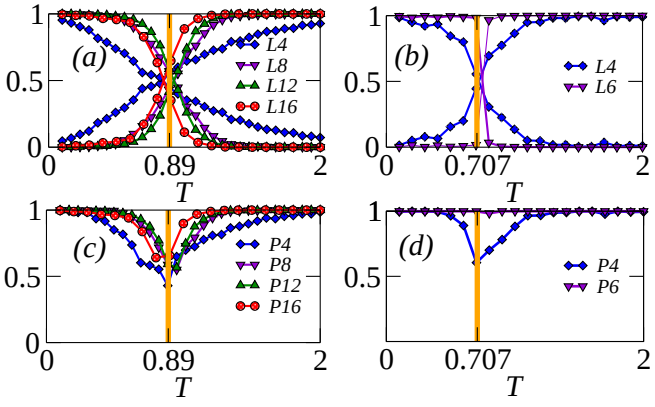

FIG. 11: (Color online) Average results of the neurons in the output layer using the CNN for XY model by inputting { $\theta_{i,j}$} on (a) the square lattices (b) the honeycomb lattices. (c) The accuracy for learning the XY model on square lattice (d) the honeycomb lattices.

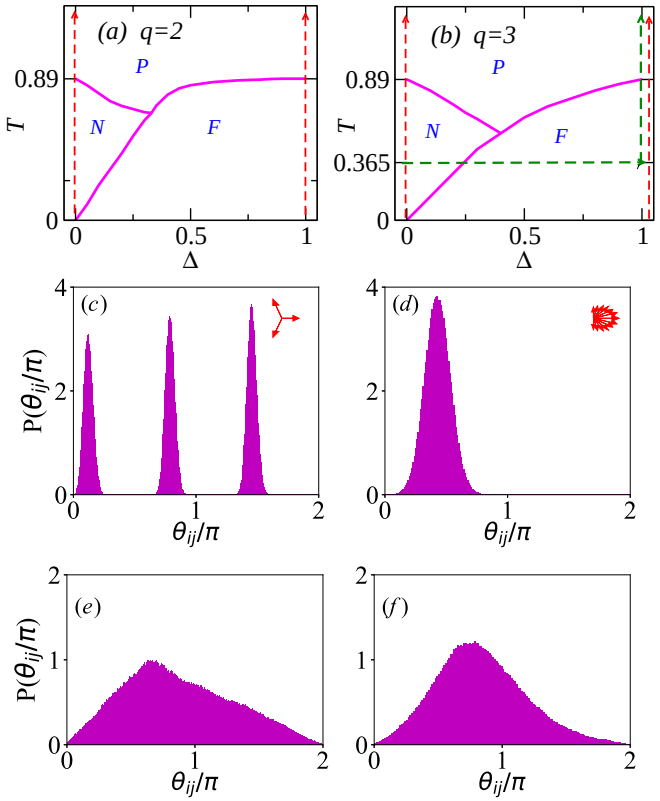

FIG. 12: Phase diagrams of the generalized XY model for (a) $q = 2$ and (b) $q = 3$, the data are from Refs[41, 42].The symbols N, P and F represent the nematic, ferromagnetic and paramagnetic phases. The dashed lines show the parameter paths to be scanned. (c) Histograms of the configurations $\theta_{i,j}$ 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 angles 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 will use histograms each derived from a very small number of samples, such as 20.

B. The generalized XY models

The phase diagram of generalized XY models [41, 42] is rich and two examples wth $q = 2$ and $q = 3$ are shown, respectively, in Figs. 12 (a) and (b). In the lower temperature and $\Delta = 0$ limit, the system is in the generalized nematic phase that has q preferred spin orientations. The statistically 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

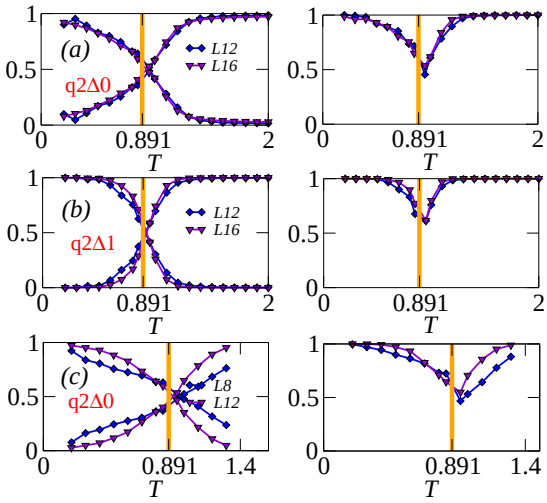

FIG. 13: (Color online) The results in the average output layer (left) and the performance (right) of learning of $\theta_{i,j}$ for (a) $q = 2$, $\Delta = 0$, $L = 12, 16$; (b) $q = 2$, $\Delta = 1$, $L = 12, 16$ The results in the average output layer (left) and performance (right) of learning of $\{\sin\theta_{i,j}, \cos\theta_{i,j}\}$ for (c) $q = 2$, $\Delta = 0$, $L = 8, 12$.

orientations is shown in Fig. 12(d). In the higher temperature, the system become disordered,i.e., and is in a paramagnetic phase. The distribution for the paramagnetic phase are also shown in Figs. 12(e) and (f) for $q = 2$ and 3, respectively. Clearly, the distributions spread the whole arrange due to strong thermal fluctuations.

We use the configuration of (1) $\theta_{i,j}$ and (2) $(\cos\theta_{i,j}, \cos\theta_{i,j})$ as input to the CNN with two convolutional layers, and the network can learn the transition. For both $\Delta = 0$ (pure XY model) and 1, the generalized XY model has the phase transition located at $T/J = 0.89$. For both $\Delta = 0$ and $\Delta = 1$, 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. From learning the configurations of angles $\{\theta_{i,j}\}$, Figs. 13 (a) shows the average output layer and performace of learning of $\theta_{i,j}$ for the thermal dynamical phase transition at $q = 2$, $\Delta = 0$, $L = 12, 16$ while in (b) for $\Delta = 1$. By inputing the configurations of $\{\sin\theta_{i,j}, \cos\theta_{i,j}\}$, the average output layer could also distinguish different phases as shown in Figs. 13 (c). Here in the network there are 32 and 64 kernels in the first and second layers, respectively. We use Metropolis algorithm to generate configurations for generalized XY model. Improvement can be made by using cluster algorithms. To obtain enough equilibrated states, we throw out the configurations during the first 10000 Monte-Carlo steps To avoid the correlations between configurations, we pick up configurations with interval of 50 Monte Carlo steps at each temperature. In the training, we also adjust the regularization parameter

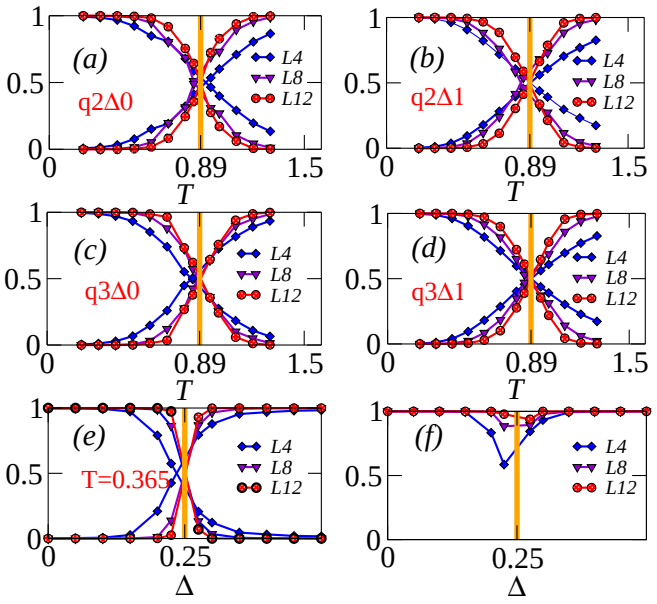

FIG. 14: (Color online) The results in the average output layer of learning of histograms of $\theta_{i,j}$ for (a) $q = 2$, $\Delta = 0$; (b) $q = 2$, $\Delta = 1$; (c) $q = 3$, $\Delta = 0$; (d) $q = 3$, $\Delta = 1$. (e) The average results in the output layer and (f) the accuracy of the learning for $q = 3$, $T = 0.365$ by scanning the parameter Δ.

λ and the number of hidden units appropriately.

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 generates a histogram. However, to make the histogram smooth, we use multiple configurations to average (e.g. 20). After that, 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. Then these matrices of pixels are used as the input to the CNN for training.

We directly recognize the histogram and obtain better results using the two-layer-convolution CNN as shown in Figs. 14 (a)-(d) along the four red dashed lines in Figs. 12 (a)-(b). For both $\Delta = 0$ and 1, the generalized XY model has the phase transition located at $T/J = 0.89$. 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 transition 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 $\Delta_c = 0.25$ [42]), and the results shown in Figs. 14 (e)-(f) demonstrate that the neural network, in particular, can distinguish the nematic phase from the ferromagnetic phase and learn the transition point. The results using the histograms seem to give slightly more accurate transition values than using the spin configurations or their components.

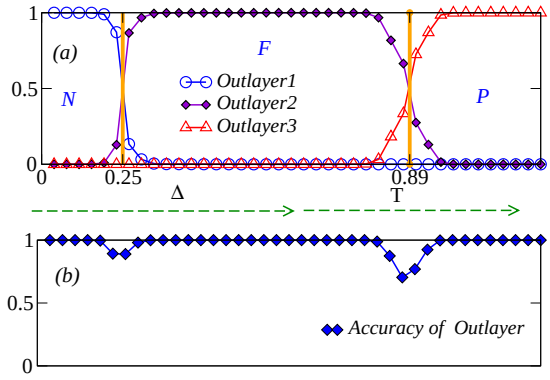

FIG. 15: (a) The results in averaging outcomes separately in the three neurons of the output layer by scanning the green dashed lines in Fig. 12(b). (b) The accuracy of the learning.

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 $\Delta_c = 0.25$ [42]), then varying T/J. The results of the CNN to learn the two different phases and transitions using the images of histograms are shown in Figs. 14 (e)-(f), and in this way the network, in particular, can distinguish the nematic phase from the ferromagnetic phase. Note that here there are only two neurons in the output layer.

Armed with the success of learning two distinct phases, we move on to test whether our method can be used to distinquish 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 *sigmod* function maps the outlayer located between the range $[0, 1]$. In the test stage, the first neuron in the output layer (representing the N phase) 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 obeys similar behavior as shown in Fig. 15 (a), where there are thus three curves corresponding to the three neurons in the output. The accuracy is shown in Fig. 15 (b) and it equals to 100% at non-critical 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 semi-supervised method to retrieve the global phase diagram of generalized XY model with $q = 2$ and $q = 3$ on the square lattice of 12×12 sites. What we mean by the semi-supervised method is that, only the limited data that are not generic are used in the training, for example, those along $\Delta = 0$ and $\Delta = 1$, i.e., the two vertical paths in Fig. 14(a) and (b). These are non-generic, as they only give very limited rep-

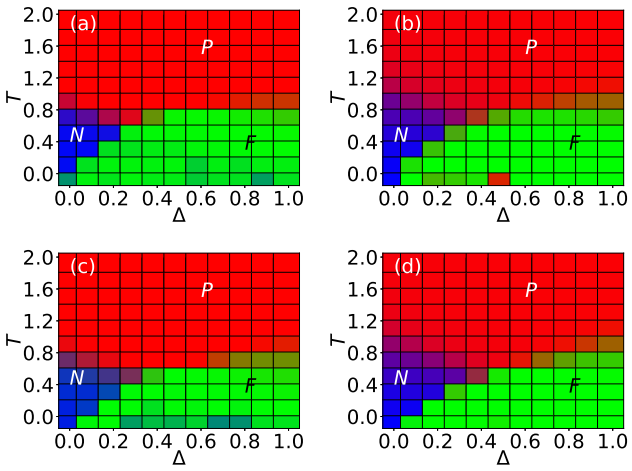

FIG. 16: The phase diagram of the generalized XY model, which contains, Nematic phase (N, blue), Ferromagnetic phase (F, green), and paramagnetic phase (P, red), obtained by the semi-supervised method discussed in the text. The neural network is trained only for data from $\Delta = 0$ and $\Delta = 1$. (a) The $q = 2$ model: the input data for training is the histogram of spin orientations. (b) The $q = 2$ model: the input data for training is the of spin orientations. (c) The $q = 3$ model: the input data for training is the histogram of spin orientations. (d) The $q = 3$ model: the input data for training is the of spin orientations. These agree with those (from Monte Carlo simulations) in Figs. 12(a)&(b).

resentations 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 $\Delta = 0, 0.1, 0.2, \cdots, 1$ and $T = 0, 0.2, 0.4, \cdots, 1.8, 2$. We use both the histogram and the spin orientation as the input, and the phase diagrams thus obtained, as shown in Fig. 16, agrees very well with those in Figs. 12 (a)&(b).

VI. CONCLUSION

In summary, we have used machine learning methods to study the percolation, the XY model and the generalized XY model. For the percolation phase transition, the unsupervised t-SNE can map the high dimensional data-sets of configurations into an 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 $L = 4, 6, 8$, the critical point could be obtained by performing the finite-size scaling. For the generalized XY model with $q = 2$ and 3, the main difference between the phases are the probability distributions of the spin orientations. The use of spin configurations as the naive input works. We devised the feature engineering using the histograms of the spin orientations, and this resulted in successful learning of various phases in the generalized XY model. The global phase diagram of the generalized XY model was otained by a semi-supervised method, i.e., with a network trained by learning just some limited set of the data. The use of machine learning in phases of matter may still need the ingenuity of appropriate features for the neural network to learn, but can become a useful tool.

Appendix A: t-SNE method

Here we summarize for convenience the t-Distributed Stochastic Neighbor Embedding (t-SNE) method [44], which is an improvement from the Stochastic Neibhgor Embedding (SNE) method [53]. 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. 2- or 3-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

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

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

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

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),

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

which is typically choosen between 5 and 50, according to the performace. 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.

Acknowledgments

WZ thanks for the valuable discussion with C. X. Ding on Monte-Carlo simulations and gratefully acknowledge financial support from China Scholarship Council and the NSFC under Grant No.11305113; TCW is supported by the National Science Foundation under Grants No. PHY 1314748 and No. PHY 1620252.

[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] J. Sebastian, 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, in *D*eep Learning (MIT Press, 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] E. van Nieuwenburg, Y. Liu and S. Huber, Learning phase transitions by confusion, Nature Phys **13**, 435 (2017).
[9] P. Broecker, F. Assaad and S. Trebst, Quantum phase recognition via unsupervised machine learning, arXiv:1707.00663
[10] G. Carleo and M. Troyer, Solving the Quantum Many-Body Problem with Artificial Neural Networks, Science **355**, 602 (2017).
[11] K. Ch'ng, J. Carrasquilla, R. Melko and E. Khatami, Machine Learning Phases of Strongly Correlated Fermions, Phys. Rev. X **7**, 031038 (2017).
[12] 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).
[13] 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).
[14] G. Torlai and R. Melko, A Neural Decoder for Topological Codes, Phys. Rev. Lett. **119**, 030501 (2017).
[15] Y. Zhang and E. A. Kim, Quantum Loop Topography for Machine Learning, Phys. Rev. Lett. **118**, 216401 (2017).
[16] D. Deng, X. Li and S. Sarma, Exact Machine Learning Topological States, Phys. Rev. B **96**, 195145 (2017).
[17] D. Deng, X. Li and S. Sarma, Quantum Entanglement in Neural Network States, Phys. Rev. X **7**, 021021, (2017).
[18] P. Zhang, H. Shen and H. Zhai, Machine Learning Topological Invariants with Neural Networks, Phys. Rev. Lett. **120**, 066401 (2018).
[19] F. Schindler, N. Regnault and T. Neupert, Probing many-body localization with neural networks, Phys. Rev. B **95**, 245134 (2017).
[20] M. Bukov, A. Day, D. Sels, P. Weinberg, A. Polkovnikov and P. Mehta, Reinforcement Learning in Different Phases of Quantum Control, arXiv:1705.00565.
[21] J. Venderley, V. Khemani and E. Kim, Machine learning out-of-equilibrium phases of matter,arXiv:1711.0002.
[22] G. Torlai, G. Mazzola, J. Carrasquilla, M. Troyer, R. Melko and G. Carleo, Many-body quantum state tomography with neural networks, arXiv:1703.05334.
[23] 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).
[24] D. Crawford, A. Levit, N. Ghadermarzy, J. Oberoi and P. Ronagh, Reinforcement Learning Using Quantum Boltzmann Machines, arXiv:1612.05695.
[25] K. I. Aoki, T. Kobayashi, Restricted Boltzmann Machines for the Long Range Ising Models, Mod. Phys. Lett. B **30**, 1651401 (2016).
[26] C. Li, D. Tan and F. Jiang, Applications of neural networks to the studies of phase transitions of two-dimensional Potts models, arXiv:1703.02369.
[27] 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).
[28] G. Torlai, R. G. Melko, Learning Thermodynamics with Boltzmann Machines, Phys. Rev. B **94**, 165134 (2016).
[29] X. Gao and L. Duan, Efficient Representation of Quantum Many-body States with Deep Neural Networks, Nat. Commun.**8**, 662 (2017).
[30] E. Stoudenmire and D. Schwab, Supervised Learning with Quantum-Inspired Tensor Networks, arXiv:1605.05775.
[31] J. Chen, S. Cheng, H. Xie, L. Wang and T. Xiang, On the Equivalence of Restricted Boltzmann Machines and Tensor Network States, arXiv:1701.04831.
[32] L. Huang, L. Wang, Accelerate Monte Carlo Simulations with Restricted Boltzmann Machines, Phys. Rev. B **95**, 035105 (2017);
Can Boltzmann Machines Discover Cluster Updates, Phys. Rev. E **96**, 051301 (2017)
[33] J. Liu, Y. Qi, Z. Meng, L. Fu, Self-Learning Monte Carlo Method, Phys. Rev. B **95**, 041101 (2017).
[34] N. Portman and I. Tamblyn, Sampling algorithms for val-

idation of supervised learning models for Ising-like systems, J. Comput. Phys. **350**, 871 (2017).

[35] P. Mehta and D. Schwab An exact mapping between the Variational Renormalization Group and Deep Learning, arXiv:1410.3831.

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

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

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

[39] G. Grimmett, in *Percolation* (Springer-Verlag, New York, 1989).

[40] J. Hovi and A. Aharony, Phys.Rev.E **53**, 235 (1996).

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

[42] F. Poderoso, J. Arenzon and Y. Levin, New ordered phases in a class of generalized XY models, Phys. Rev. Lett. **106**, 067202 (2011).

[43] 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, Phys. Rev. B **52**, 4526 (1995).

[44] 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); L. van der Maaten and G. Hinton, Visualizing High-Dimensional Data Using t-SNE, J Mach Learn Res. **9**, 2579 (2008).

[45] M. Abadi, A. Agarwal and P. Barham *et al*, TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems, arXiv:1603.04467.

[46] M. Newman and R. Ziff, Efficient Monte Carlo algorithm and high-precision results for percolation, Phys. Rev. Lett. **85**,4104 (2000).

[47] 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).

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

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

[50] R. Swendsen, and J. Wang, Nonuniversal critical dynamics in Monte Carlo simulations, Phys. Rev. Lett. **58** 86 (1987).

[51] 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).

[52] 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 Phys. Rev. Lett. **98** 120601 (2007).

[53] G. E. Hinton and S. T. Roweis, Stochasitic Neighbor Embedding. In *Advances in Neural Information Processing Systems*, volume **15**, pages 833-840 (The MIT Press, Cambridge, MA 2002).