Efficient machine-learning representations of a surface code with boundaries, defects, domain walls, and twists

Zhih-Ahn Jia,[1,2,3,4,*] Yuan-Hang Zhang,[5] Yu-Chun Wu,[3,4,†] Liang Kong,[6,‡] Guang-Can Guo,[3,4] and Guo-Ping Guo[3,4]

[1]*Microsoft Station Q, University of California, Santa Barbara, California 93106-6105, USA*

[2]*Department of Mathematics, University of California, Santa Barbara, California 93106-6105, USA*

[3]*Key Laboratory of Quantum Information, Chinese Academy of Sciences, School of Physics, University of Science and Technology of China, Hefei, Anhui 230026, People's Republic of China*

[4]*Synergetic Innovation Center of Quantum Information and Quantum Physics, University of Science and Technology of China, Hefei, Anhui 230026, People's Republic of China*

[5]*School of the Gifted Young, University of Science and Technology of China, Hefei, Anhui 230026, People's Republic of China*

[6]*Shenzhen Institute for Quantum Science and Engineering, and Department of Physics, Southern University of Science and Technology, Shenzhen 518055, People's Republic of China*

(Received 2 May 2018; revised manuscript received 29 September 2018; published 4 January 2019)

Machine-learning representations of many-body quantum states have recently been introduced as an ansatz to describe the ground states and unitary evolutions of many-body quantum systems. We investigate one of the most important representations, the restricted Boltzmann machine (RBM), in the stabilizer formalism. A general method to construct RBM representations for stabilizer code states is given, and exact RBM representations for several types of stabilizer groups with the number of hidden neurons equal to or less than the number of visible neurons are presented. The result indicates that the representation is extremely efficient. Then we analyze a surface code with boundaries, defects, domain walls, and twists in full detail and find that almost all the models can be efficiently represented via the RBM ansatz: the RBM parameters of the perfect case, boundary case, and defect case are constructed analytically using the method we provide in the stabilizer formalism, and the domain wall and twist case is studied numerically. In addition, the case for Kitaev's $D(\mathbb{Z}_d)$ model, which is a generalized model of the surface code, is also investigated.

DOI: 10.1103/PhysRevA.99.012307

I. INTRODUCTION

As the leading proposal for achieving fault-tolerant quantum computation, surface codes have attracted researchers' attention in recent years. Since Kitaev [1] made the ingenious step of transforming a quantum error correction code (QECC) into a many-body interacting quantum system (more precisely, he constructed a Hamiltonian, now known as a toric code, which is a gapped anyon system and whose ground-state space is exactly the code space; the encoded information is protected by the topological properties of the system), the surface code model has been extensively investigated from both the QECC perspective and condensed matter perspective. Studies on the surface code cross-fertilize both areas. Suppose that we are encoding information with n physical bits, i.e., with the Hilbert space $\mathcal{H} = (\mathbb{C}^2)^{\otimes n}$. The code space $\mathcal{C}$ is a subspace of $\mathcal{H}$. This subspace can be regarded as the ground-state space of the corresponding surface code Hamiltonian H, which is just the negative of the summation of all stabilizer generators.

From a condensed matter perspective, to conquer the challenge of efficiently representing the many-body states with exponential complexity, a neural network, one of the most important tools in machine learning [2,3], is introduced to efficiently represent the ground states of strongly correlated many-body systems [4], which are beyond the mean-field paradigm, for which the density matrices are of the tensor product form in the thermodynamic limit. The mean-field approach is very successful in bosonic systems (the quantum de Finetti theorem) but fails for other strongly correlated systems. There are also some other approaches: the quantum Monte Carlo method [5–9], which suffers from the sign problem, and the tensor network representation [10], whose special form, matrix product states (MPS), has great success in $1d$ systems [11,12], but for the $2d$ case, it is unknown whether the corresponding projected entangled pair states (PEPS) are enough and extracting information is $\#P$-hard in general; the best known approximation algorithm still spends superpolynomial time under assumptions [13,14]. The connections between the machine-learning representation and other representations are also extensively exploited [15–18].

Machine learning as a method for analyzing data has been prevalent in many scientific areas [2,3,19], including computer vision, speech recognition, and chemical synthesis, among which the artificial neural network plays an important role in recognizing or even discovering particular patterns of the input data. Quantum machine learning (QML) [20], which is an emerging interdisciplinary scientific area at the

*giannnjia@foxmail.com; zajia@math.ucsb.edu

†wuyuchun@ustc.edu.cn

‡kongl@sustc.edu.cn

intersection of quantum mechanics and machine learning, has recently attracted much attention [4,17,21–26]. There are two crucial branches of QML: the first one is to develop new quantum algorithms which share some features of machine learning and behave faster and better than their classical counterparts [21–23]; the second one, which is also the focus of this work, is to use the classical machine-learning methods to assist the study of quantum systems. Machine-learning methods are so powerful that they can be used for distinguishing phases [24], quantum control [27], error-correcting of topological codes [28], quantum tomography [29,30], and efficiently representing quantum many-body states [4,17,25,26,31–33]. Among all of them, using neural networks as variational wave functions to approximate the ground state of many-body quantum systems has received much attention recently. Many different neural network architectures have been tested and the most successful one is the restricted Boltzmann machine (RBM) [4,17,25,26]. It has been shown that the RBM can efficiently represent ground states of several many-body models, including the Ising model [4], toric code model [25,26], and graph states [17].

In this work, we study the RBM representation in a stabilizer formalism and we provide some more systematic analyses. It is shown that for many stabilizer groups, the RBM representations are extremely efficient: the number of hidden neurons approximates the number of visible neurons. We take the surface code with boundaries, defects, domain walls, and twists as some concrete examples, and we find that all these models can be represented by the RBM. The exact solution is given for the boundary and defect cases. We also analyze Kitaev's $D(G)$ model for the $G = \mathbb{Z}_d$ case. Our results can be useful for building the RBM neural network when analyzing the anyon model or QECC in the stabilizer formalism.

The work is organized as follows. In Sec. II we provide an elaborate description of the stabilizer formalism, Kitaev $D(G)$ model, $\mathbb{Z}_2$ surface code model, and the properties when these models are regarded as anyon models. Then we construct the surface code models with boundaries, defects, domain walls, and twists, and give the precise stabilizer operators and Hamiltonians of these models. In Sec. III, we give a brief review of RBM representations of states. Then, in Sec. IV, the RBM representations in the stabilizer formalism are worked out and many explicit solutions of stabilizer states are constructed. In Sec. V, using the results developed in Sec. IV, we provide a detailed analysis of RBM representations of a surface code with boundaries, defects, domain walls, and twists. And the general Kitaev $D(G)$ case is done in Sec. VI. Finally, in Sec. VII we give our discussion and conclusions.

II. SURFACE CODE MODEL WITH BOUNDARIES, DEFECTS, DOMAIN WALLS, AND TWISTS

In this section, we give a brief review of the basics of a surface code in the stabilizer formalism, and the corresponding surface code Hamiltonian is an anyon model. For simplicity of illustration, we will assume hereinafter that the lattice is a square lattice placed on a plane, but all our results can be extended to general cases similarly. We will analyze the boundaries, defects, domain walls, and twists in the surface code from an anyon theoretic perspective.

A. Stabilizer formalism

QECCs are commonly expressed in the stabilizer formalism [34,35]. To prevent the encoded information from noise, the logical quantum states are encoded redundantly in a k-dimensional subspace $\mathcal{C}$ of the n-qubit physical space $\mathcal{H} = (\mathbb{C}^2)^{\otimes n}$. The stabilizer group $\mathbf{S}$ for $\mathcal{C}$ is an Abelian subgroup of the Pauli group $\mathbf{P}_n = \{I, \sigma_x, \sigma_y, \sigma_z\}^{\otimes n} \times \{\pm 1, \pm i\}$; more precisely, $\mathcal{C}$ is the invariant subspace for $\mathbf{S}$ acting on $\mathcal{H}$. Since each operator T_j in $\mathbf{S}$ is a Hermitian operator and $[T_i, T_j] = 0$ for all i, j, the code states are the common eigenstates of all elements T_j in $\mathbf{S}$, i.e.,

$$T_j|\Psi\rangle = +1|\Psi\rangle, \ \forall j. \tag{1}$$

Suppose $\mathbf{S}$ is generated by m independent operators $\{T_1, \ldots, T_m\}$. Note that $T_j^2 = I$ for all $j = 1, \ldots, m$; then any $T \in \mathbf{S}$ can be uniquely expressed as $T = T_1^{\alpha_1} T_2^{\alpha_2} \cdots T_m^{\alpha_m}$ where $\alpha_j \in \{0, 1\}$; thus the order of stabilizer group $\mathbf{S}$ is 2^m. The numbers of physical qubits n, generators of stabilizer group m, and encoded logical qubits k are related by a simple formula $n = m + k$.

To construct logical operators $\bar{L}$ which leave the code space invariant and transform the logical states into each other, notice that any pair of Pauli operators must commute or anticommute. Any Pauli operator that anticommutes with elements in $\mathbf{S}$ cannot leave the code space invariant and logical gates must not be in $\mathbf{S}$ or they cannot achieve the logical transformation. Therefore, the logical gate operator must live in the centralizer $\mathbf{C} \subset \mathbf{P}_n$ of the stabilizer group. It is worth mentioning that the representation of the logical operator is not unique. Two logical operators $\bar{L}$ and $\bar{L}' = \bar{L}T$ with $T \in \mathbf{S}$ satisfy $\bar{L}'|\Psi\rangle = \bar{L}|\Psi\rangle$ for all code states $|\Psi\rangle$.

Another important quantity to characterize the stabilizer code is the code distance d. It is defined as the smallest set of qubits which supports one nontrivial logical operator of the code. The stabilizer code with n physical qubits, k encoded logical qubits, and code distance d is denoted as $[[n, k, d]]$.

B. Lattice model on a surface

The anyon model of the surface code is a $D(\mathbb{Z}_2)$ quantum double model [1]. For a given surface Σ, consider its cellulation $\mathcal{C}(\Sigma)$ which is the set of all cells; we denote the set of 2-cells (i.e., plaquettes) as $\mathcal{C}^2(\Sigma)$, 1-cells (i.e., edges) $\mathcal{C}^1(\Sigma)$, and 0-cells (i.e., vertices) $\mathcal{C}^0(\Sigma)$. We can attach a physical space $\mathcal{H}_{e_i}$ on each edge e_i of the lattice, the basis is chosen as $\{|g\rangle : g \in G\}$ labeled by elements in G, and the whole space is then $\mathcal{H} = \bigotimes_{e_i \in \mathcal{C}^1(\Sigma)} \mathcal{H}_{e_i}$. The quantum double model $D(G)$ for the general finite group G can be defined on a general two-dimensional lattice, but here, for convenience, we only employ the square lattice and the group G is chosen as the Abelian group $\mathbb{Z}_2$. To proceed we define the operators $L_\pm^g$ which are associated with the vertices of the lattice $\mathcal{C}^1(\Sigma)$ and $T_\pm^h$, which are associated with plaquettes of the lattice $\mathcal{C}^2(\Sigma)$,

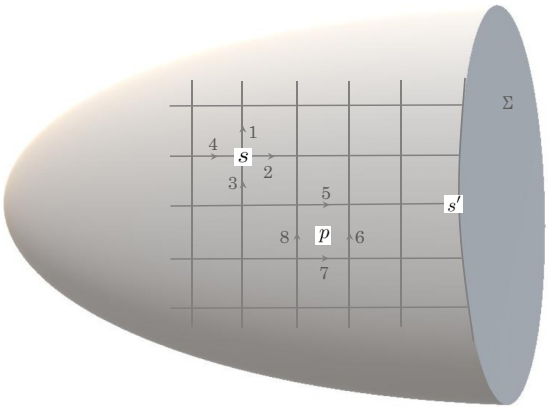

FIG. 1. Surface code model on a surface Σ.

such that

$$L_+^g|z\rangle = |gz\rangle, \quad L_-^g|z\rangle = |zg^{-1}\rangle,$$
$$T_+^h|z\rangle = \delta_{h,z}|z\rangle, \quad T_-^h|z\rangle = \delta_{h^{-1},z}|z\rangle.$$

It is easy to check that these operators satisfy the following relations:

$$L_+^g T_+^h = T_+^{gh} L_+^g, \quad L_+^g T_-^h = T_-^{hg^{-1}} L_+^g,$$
$$L_-^g T_+^h = T_+^{hg^{-1}} L_+^g, \quad L_-^g T_-^h = T_-^{gh} L_-^g.$$

Now consider the orientable surface Σ as in Fig. 1, where a square lattice is placed on it. To consistently define the Hamiltonian we give each edge an orientation; here we take the vertical edges upward and horizontal edges rightward. Changing the orientation of an edge corresponds to changing $|g\rangle$ to $|g^{-1}\rangle$. For the $\mathbb{Z}_2$ case, $0^{-1} = 0$ and $1^{-1} = 1$; thus the orientation is not necessary for the $D(\mathbb{Z}_2)$ model. We now define two types of operators: the star operators defined on vertices (see s vertex as in Fig. 1)

$$A(s) = \frac{1}{|G|} \sum_{g \in G} L_{-,1}^g L_{-,2}^g L_{+,3}^g L_{+,4}^g, \tag{2}$$

where for edges pointing to s we assign L_+^g, otherwise we assign L_-^g, and plaquette operators defined as (see p plaquette as in Fig. 1)

$$B(p) = \sum_{h_5 h_6 h_7 h_8 = 1_G} T_{-,5}^{h_5} T_{+,6}^{h_6} T_{+,7}^{h_7} T_{-,8}^{h_8}, \tag{3}$$

where if p is on the left of the edge we assign T_+^h to the edge; otherwise we assign T_-^h. $A(s)$, $A(s')$, $B(p)$, $B(p')$ commute with each other for all $s, s' \in \mathcal{C}^0(\Sigma)$ and $p, p' \in \mathcal{C}^2(\Sigma)$.

Note that for group $\mathbb{Z}_2$, $0^{-1} = 0$ and $1^{-1} = 1$; thus $L_+^0 = L_-^0 = I$ and $L_+^1 = L_-^1 = \sigma_x$, and $T_+^0 = T_-^0 = \Pi_{|0\rangle\langle 0|}$ and $L_+^1 = L_-^1 = \Pi_{|1\rangle\langle 1|}$. We introduce new operators A_s and B_p. For each vertex (star) s and each plaquette p, construct the following vertex and plaquette operators:

$$A_s = \Pi_{i \in \mathrm{star}(s)} \sigma_x^i, \quad B_p = \Pi_{i \in \partial p} \sigma_z^i, \tag{4}$$

where we use ∂p to represent the boundary edges of the plaquette p. A_s and B_p are stabilizer operators in the stabilizer QECC formalism and they commute with each other, i.e., $[A_s, A_{s'}] = [B_p, B_{p'}] = [A_s, B_p] = 0$ for all vertices s, s' and plaquettes p, p'. And all these operators are Hermitian with eigenvalues ± 1. Notice that $A(s) = \frac{1}{2}(I + A_s)$ and $B(p) = \frac{1}{2}(I + B_p)$. Here for simplicity we construct the following Hamiltonian:

$$H_\Sigma = -\sum_s A_s - \sum_p B_p, \tag{5}$$

which will be referred to as the surface code Hamiltonian. H_Σ is the negative summation of all stabilizer generators; thus the ground states for it correspond to the solution $A_s|\Omega\rangle = |\Omega\rangle$, $B_p|\Omega\rangle = |\Omega\rangle$ for all $s \in \mathcal{C}^0(\Sigma)$ and $p \in \mathcal{C}^2(\Sigma)$, which turn out to be code states in the stabilizer formalism.

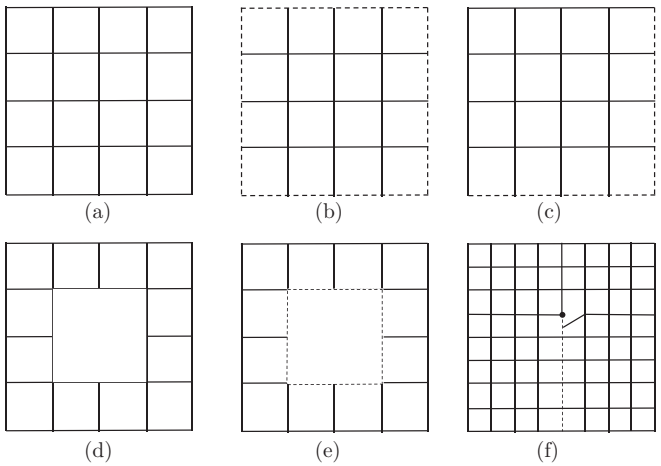

FIG. 2. The surface code with boundaries, defects, and twists. (a) Smooth boundary; (b) rough boundary; (c) mixed boundary; (d) smooth defect; (e) rough defect; (f) twist Q and domain wall W.

C. Boundaries, defects, and twists

Real samples of quantum matter have boundaries and defects, so it is also important to analyze the quantum double model on a lattice with boundaries and defects. As discussed in Refs. [36–39], we can construct boundary and defect Hamiltonians. For convenience, we denote H_{bulk}, H_{bondary}, H_{defect}, and H_{twist} the Hamiltonians of bulk, boundaries, defects, and twists, respectively.

1. Gapped boundaries of the surface code

In general, the gapped boundary of the quantum double model $D(G)$ is determined by the subgroup $K \in G$ (up to conjugation) and a 2-cocycle in $H^2(K, \mathbb{C}^\times)$ [37]. To define a Hamiltonian for gapped boundaries, we first need to give the orientation of the boundaries and then introduce the local terms of each star and plaquette near the boundary (which depend on a subgroup K of G). Here, we will focus on the simplest $\mathbb{Z}_2$ case and we take K to be $\mathbb{Z}_2$ itself.

There are two types of boundaries for a planar code: a smooth one and a rough one [40,41]. Let us now define some new star and plaquette operators [see Figs. 2(a)–2(c)]. For smooth boundaries, we can see that the plaquettes do not change near the boundary, but the star operators change. We need to introduce two kinds of operators: a corner star

operator C_s and a boundary star operator D_s [see Fig. 2(a)]:

$$C_s = \sigma_x^1\sigma_x^2, \quad D_s = \sigma_x^3\sigma_x^4\sigma_x^5.$$

The smooth boundary Hamiltonian then reads

$$H_{\text{sboundary}} = -\sum_s C_s - \sum_s D_s,$$

in which all terms are commutative with each other; thus $H_{\text{sboundary}}$ is a gapped Hamiltonian.

For the rough boundaries, the star operators near the boundary remain unchanged but the plaquette operators change, and we similarly introduce two kinds of operators: a corner plaquette operator E_p and a boundary plaquette operator F_p [see Fig. 2(b)]:

$$E_p = \sigma_z^1\sigma_z^2, \quad F_p = \sigma_z^3\sigma_z^4\sigma_z^5.$$

Similarly, we have the gapped Hamiltonian for rough boundaries:

$$H_{\text{rboundary}} = -\sum_p E_p - \sum_p F_p.$$

We can also introduce the mixed boundaries, which are the mixed case of smooth and rough boundaries [see Fig. 2(c)]. The boundary Hamiltonian then reads

$$H_{\text{mboundary}} = -\sum_s C_s - \sum_s D_s - \sum_p E_p - \sum_p F_p.$$

When an e particle moves to the rough boundary, it will condense into the vacuum of the boundary. Similarly, the m particle will condense in the smooth boundary. Thus the boundary phase is condensed from the bulk phase. Conversely, the bulk phase can also be recovered from the boundary phase via the half loop of the m and e particles. This is the content of the famous boundary-bulk duality.

2. Defects of the surface code

Let us now consider the case where we punch several holes $\mathfrak{h}_1, \ldots, \mathfrak{h}_k$ on the lattice. To describe the holes, we need to specify k subgroups $K_1, \ldots, K_k$ of G. Here, we still assume that all K_i are equal to $G = \mathbb{Z}_2$. Then the hole Hamiltonian will be

$$H_{\text{defect}} = \sum_i H_{\mathfrak{h}_i}.$$

Like the case for boundaries, there are two typical types of holes: a smooth one and a rough one; see Figs. 2(d) and 2(e). The main difference is that we do not need to introduce the corner star operator for the smooth hole, and do not need to introduce corner plaquette operators for a rough hole. Therefore, we have the Hamiltonians for holes as

$$H_{s\mathfrak{h}} = -\sum_s D_s, \quad H_{r\mathfrak{h}} = -\sum_p F_p.$$

3. Twists of the surface code

As depicted in Fig. 2(f), there is a dislocation in the lattice, along a line W (referred to as a one-dimensional domain wall). Plaquettes are shifted such that the plaquette in the vicinity of W is changed. W can be regarded as a mixed one-dimensional defect, and the point between smooth and rough $1d$ defects is also a special kind of defect named a twist defect [36,42]. A twist defect is a zero-dimensional defect, which has many interesting properties.

The plaquette operators near the domain wall will change, for example $W_p = \sigma_z^5\sigma_z^6\sigma_z^7\sigma_x^4$ as depicted in Fig. 2(f). Besides, we must introduce a new stabilizer operator $Q = \sigma_x^5\sigma_y^1\sigma_z^2\sigma_z^3\sigma_z^4$ as depicted in Fig. 2(f). It is easy to check that each W_p and Q are commutative with bulk vertex operators and plaquette operators. Therefore, we have the following twist Hamiltonian:

$$H_{\text{twist}} = -\sum_p W_p - Q. \tag{6}$$

We see that a geometric change of lattice implies significant change of the Hamiltonian. If we move one e particle around the point Q, it becomes an m particle, and similarly for an m particle around Q. The m particle will condense to vacuum as moving into the smooth part of W, the e particle will condense as moving into the rough part W, but both the e and m particles will condense into vacuum as moving into the twist point Q.

III. NEURAL NETWORK ANSATZ

The restricted Boltzmann machine (RBM), a shallow generative stochastic artificial neural network that can learn a probability distribution over its set of inputs, was initially invented by Smolensky [43] in 1986. It is a particular kind of Boltzmann machine [44,45]. It has recently been introduced in many-body physics to efficiently represent the ground state of gapped many-body quantum systems [4]. The approach based on the RBM, the counterpart of the deep neural network representation, is also developed later [17].

We now briefly introduce the machine-learning representation of a state based on the restricted Boltzmann architecture. Consider an n-spin physical system $\mathcal{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_n\}$. An RBM neural network contains two layers: a visible layer and a hidden layer (see Fig. 4); we place n spin variables $\{v_1, \ldots, v_n\}$ in a fixed basis $\{|\mathbf{v}\rangle = |v_1, \ldots, v_n\rangle\}$ on n corresponding neurons in the visible layer. There are m auxiliary variables $\{h_1, \ldots, h_m\}$ where $v_i, h_j \in \pm 1$ in the hidden layer. The neurons in the visible layer are connected with the neurons in the hidden layer, but there are no intralayer connections. The weights for visible neurons v_i and hidden neurons h_j are denoted as a_i and b_j, respectively, and the weight on the edge between h_j and v_i is denoted as W_{ji}. Note that $\Omega = \{\mathbf{a} = (a_1, \ldots, a_n), \mathbf{b} = (b_1, \ldots, b_m), W = (W_{ij})\}$ are the parameters need to be trained which completely determine the corresponding RBM construction. An RBM state (up to some normalization constant) is then of the form

$$|\Psi\rangle_{\text{RBM}} = \sum_{\mathbf{v}} \Psi(\mathbf{v}, \Omega)|\mathbf{v}\rangle, \tag{7}$$

where $\{\mathbf{v}\}$ is the chosen basis and the coefficient $\Psi(\mathbf{v}, \Omega)$ is obtained by tracing out the hidden neuron

variables [4]:

$$\Psi(\mathbf{v},\Omega) = \sum_{\mathbf{h}} e^{\mathbf{a}^T\mathbf{v}+\mathbf{b}^T\mathbf{h}+\mathbf{h}^T W\mathbf{v}},$$
$$= \sum_{\mathbf{h}} e^{\sum_i a_i v_i+\sum_j b_j h_j+\sum_{i,j} h_j W_{ji} v_i},$$
$$= e^{\sum_i a_i v_i}\prod_{j=1}^{m} 2\cosh\left(b_j+\sum_i W_{ji} v_i\right). \quad (8)$$

Hereinafter, we will choose the σ_z basis for each spin space, and $|+1\rangle$ and $|-1\rangle$ are two basis states such that $\sigma_z|+1\rangle = +1|+1\rangle$ and $\sigma_z|-1\rangle = -1|-1\rangle$, i.e., $\sigma_z^i|v_i\rangle = v_i|v_i\rangle$. Similarly, we have $\sigma_x^i|v_i\rangle = |-v_i\rangle$ and $\sigma_y^i|v_i\rangle = i\,v_i|-v_i\rangle$.

One of the most central problems of the RBM representation of quantum many-body states is its representational power. The mathematical foundation of the neural network representations originates from the representation theorem developed by Kolmogorov [46,47] and Arnold [48], and Le Roux and Bengio's work [49], which stresses the case for the RBM. It has been shown that the RBM can efficiently represent toric code states [26], $1d$ symmetry protected topological states [26], and graph states [17]. The connection between neural network states and tensor network states is also extensively explored [15,17,18,31]. See Ref. [50] for a review of the quantum neural network states.

IV. NEURAL NETWORK REPRESENTATION OF STATES IN THE STABILIZER FORMALISM

It is believed that the RBM can represent the ground state of local gapped systems. Here, we analyze RBM representations in the stabilizer formalism in a much more general way. As we will see, since there is no intralayer connection in the RBM, the concept of locality does not emerge. Even for some nonlocal stabilizer group, the corresponding ground state can be efficiently represented using the RBM.

To begin with, we introduce our general methodology of constructing RBM representations of stabilizer states. Suppose that the stabilizer group is generated by $\{T_1,\ldots,T_m\}$. Since all other operators are just products of the generator operators, to give the stabilizer state, we only need to restrict

$$T_k|\Psi\rangle = +1|\Psi\rangle. \quad (9)$$

T_k is the product of Pauli operators; thus we suppose that

$$T_k|\mathbf{v}_k,\tilde{\mathbf{v}}\rangle = \lambda_k|\mathbf{v}_k',\tilde{\mathbf{v}}\rangle, \quad (10)$$

where $\mathbf{v}_k$ are the spins that T_k acts nontrivially on, $\tilde{\mathbf{v}}$ are the rest of the spins, and λ_k is the possible phase shift caused by T_k.

Using Eqs. (9) and (10) and plugging in $|\Psi\rangle = \sum_{\mathbf{v}}\Psi(\mathbf{v};\Omega)|\mathbf{v}\rangle$, we have

$$T_k|\Psi\rangle = \sum_{\mathbf{v}}\Psi(\mathbf{v}_k,\tilde{\mathbf{v}};\Omega)T_k|\mathbf{v}_k,\tilde{\mathbf{v}}\rangle$$
$$= \sum_{\mathbf{v}}\Psi(\mathbf{v}_k,\tilde{\mathbf{v}};\Omega)\lambda_k|\mathbf{v}_k',\tilde{\mathbf{v}}\rangle. \quad (11)$$

Meanwhile,

$$|\Psi\rangle = \sum_{\mathbf{v}}\Psi(\mathbf{v}_k',\tilde{\mathbf{v}};\Omega)|\mathbf{v}_k',\tilde{\mathbf{v}}\rangle. \quad (12)$$

From Eqs. (11) and (12) we conclude that

$$\lambda_k\Psi(\mathbf{v}_k,\tilde{\mathbf{v}};\Omega) = \Psi(\mathbf{v}_k',\tilde{\mathbf{v}};\Omega). \quad (13)$$

Equation (13) must hold for all spin configurations, and is almost impossible to solve directly for large systems. To tackle this problem, we employ the Jastrow wave function [51] as a trial function and introduce the restriction that the wave function takes the form

$$\Psi(\mathbf{v};\Omega) = \prod_k f_k(\mathbf{v}_k), \quad (14)$$

where each $f_k(\mathbf{v}_k)$ is the function of several local spins in the big system. We will first work out the values of $f_k(\mathbf{v}_k)$, then use hidden neuron connections to represent them.

Plugging Eq. (14) into Eq. (13), we have

$$\lambda_k\Psi(\mathbf{v}_k,\tilde{\mathbf{v}};\Omega) = \lambda_k\prod_k f_k(\mathbf{v}_k) = \lambda_k f_k(\mathbf{v}_k)\prod_{l\neq k} f_l(\mathbf{v}_l)$$

and

$$\Psi(\mathbf{v}_k',\tilde{\mathbf{v}};\Omega) = f_k(\mathbf{v}_k')\prod_{l\neq k} f_l(\mathbf{v}_l').$$

Thus,

$$\lambda_k f_k(\mathbf{v}_k)\prod_{l\neq k} f_l(\mathbf{v}_l) = f_k(\mathbf{v}_k')\prod_{l\neq k} f_l(\mathbf{v}_l'). \quad (15)$$

Finally, to find a solution to Eq. (15), we set the corresponding terms equal to each other:

$$\lambda_k f_k(\mathbf{v}_k) = f_k(\mathbf{v}_k'), \quad (16)$$

$$f_l(\mathbf{v}_l) = f_l(\mathbf{v}_l'). \quad (17)$$

Given a set of stabilizers, we can find a set of $f_k(\mathbf{v}_k)$ that satisfies Eqs. (16) and (17), and then find the RBM parameters corresponding to each $f_k(\mathbf{v}_k)$.

Here, we argue that the most important thing is the stabilizer's configuration which determines the architecture of the neural network. To begin with, we divide groups of stabilizer generators into several types: $\mathbf{S}_X$, $\mathbf{S}_Y$, $\mathbf{S}_Z$, which only contain tensor products of σ_x, σ_y, and σ_z, respectively; $\mathbf{S}_{XY}$, $\mathbf{S}_{YZ}$, $\mathbf{S}_{XZ}$, which contain tensor products of σ_x and σ_y, of σ_y and σ_z, and of σ_x and σ_z; and $\mathbf{S}_{XYZ}$, which only contains tensor products of σ_x, σ_y, and σ_z. We will use the notation $\mathbf{S}_X \sqcup \mathbf{S}_Z$ to mean that the generators of the stabilizer group only involve elements of $\mathbf{S}_X$ and $\mathbf{S}_Z$ type, and similarly for others.

We will prove that all code states in the $\mathbf{S}_X$ (resp. $\mathbf{S}_Y$, $\mathbf{S}_Z$) stabilizer formalism can be exactly and efficiently represented by RBM. Specifically, we can assign one hidden neuron to each stabilizer operator which only connects with visible neurons it acts nontrivially on [corresponding to one $f_k(\mathbf{v}_k)$ in Eq. (14)]. As for code states in the $\mathbf{S}_{XZ}$ (resp. $\mathbf{S}_{XY}$, $\mathbf{S}_{YZ}$) stabilizer formalism, using machine-learning

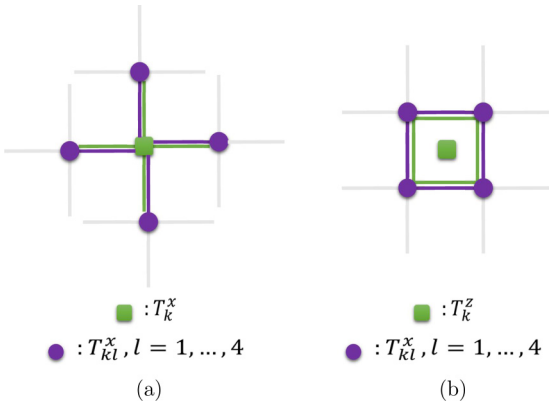

FIG. 3. (a) A vertex and (b) a plaquette taken from the lattice of the toric code model. Each edge corresponds to one spin, and different circles on the vertices and faces correspond to different stabilizer operators. The highlighted edges show the spins that the stabilizer acts on.

techniques, we can give efficient RBM representations with high accuracy.

A. $\mathbf{S}_X$, $\mathbf{S}_Y$, and $\mathbf{S}_Z$

Equations (16) and (17) suggest that we should treat each $f_k(\mathbf{v}_k)$ (i.e., each stabilizer operator) individually. To begin with, we draw the spins $\mathbf{v}_k$ from the whole system and analyze this subsystem. In general, there will be multiple stabilizers acting on the subsystem $\mathbf{v}_k$. We will call T_k the "major stabilizer" in the subsystem $\mathbf{v}_k$. The equation $T_k|\Psi\rangle = +1|\Psi\rangle$ simply corresponds to Eq. (16), but for other stabilizers T_l, the equation $T_l|\Psi\rangle = +1|\Psi\rangle$ does not correspond to Eq. (17). Namely, the effect of T_l on $|\Psi\rangle$ is split into two parts: the possible phase shift λ_l, which is only shown in Eq. (16), and the possible spin flip changing $\mathbf{v}_l$ into $\mathbf{v}_l'$, which is shown in both Eq. (16) and Eq. (17). Thus, when analyzing the subsystem $\mathbf{v}_k$, the nonmajor stabilizers can only flip spins and cannot affect the phase.

Returning to our analysis of different types of stabilizers, as a nonmajor stabilizer in the subsystem, $T^z \in \mathbf{S}_Z$ has no effect on the subsystem, while $T^x \in \mathbf{S}_X$ and $T^y \in \mathbf{S}_Y$ only have the effect of flipping spins. When analyzing the subsystem $\mathbf{v}_k$, we can ignore all nonmajor T^z, while regarding all nonmajor T^y as T^x.

In conclusion, when concerning the subsystem $\mathbf{v}_k$ only, there is one major stabilizer T_k and multiple nonmajor stabilizers $T_{l_{\mathbf{v}_k}} \in \mathbf{S}_X$ acting on them. $f_k(\mathbf{v}_k)$ describes the common eigenstate of $\{T_k, T_{l_{\mathbf{v}_k}}\}$. Since the size of the subsystem is small in general, $f_k(\mathbf{v}_k)$ can be easily found by solving Eq. (1). Treating every stabilizer T_k in the same way, we can get a set of functions $\{f_k(\mathbf{v}_k)\}$, and the wave function is given by Eq. (14).

We take the Kitaev toric code state as an example, and give a much simpler and more intuitive construction compared to [26].

Figure 3(a) shows a vertex taken from the lattice. When concerning the four spins connected to the vertex only, there are five stabilizer operators acting on them (we ignored the four T_z because of the reason stated above), with four of them independent of each other (with the relationship $T_{k1}^x T_{k2}^x T_{k3}^x T_{k4}^x = T_k^x$). We can check that $T_k^x \Psi(\mathbf{v}_k, \tilde{\mathbf{v}}; \Omega) = \Psi(-\mathbf{v}_k, \tilde{\mathbf{v}}; \Omega)$ corresponds to Eq. (16), and the equations for $T_{k1}^x, \ldots, T_{k4}^x$ correspond to Eq. (17). The dimension $\mathcal{L}$ for this subsystem is $2^{4-4} = 1$, and it is easy to find that the stabilizer state is $|++++\rangle$. For the plaquette shown in Fig. 3(b), similar results can be obtained in the same way.

Therefore, if we can construct the RBM representation of each vertex and plaquette, we will get the RBM representation of the toric code state. With the functions $f_k(\mathbf{v}_k)$, the RBM representation is easy to find. Attaching one hidden neuron to each vertex and plaquette, we can check that one solution is

$$a_k = \frac{i\pi}{4}, \quad b_p = -i\pi, \quad W_{pp_k} = \frac{i\pi}{4}, \quad \text{for plaquettes},$$
$$a_k = 0, \quad b_q = 0, \quad W_{qq_k} = 0, \quad \text{for vertices}.$$

For the general solution and details in calculation (in a less intuitive way), see the Appendix.

This result is a special case for the general $\mathbf{S}_X \sqcup \mathbf{S}_Z$ stabilizer formalism that we will discuss here. For $T_p^z \in \mathbf{S}_Z$ and $T_q^x \in \mathbf{S}_X$, the commutation relation between them tells us that they can only share an even number of spins. To begin with, we take out the spins that T_p^z acts on and try to construct the function $f_p(\mathbf{v}_p)$. The equation $T_p^z \Psi(\mathbf{v}_p, \tilde{\mathbf{v}}; \Omega) = \prod_p v_p \Psi(\mathbf{v}_p, \tilde{\mathbf{v}}; \Omega)$ corresponds to Eq. (16), and for the most general case, there exists a T_q^x for any pair of spins in $\mathbf{v}_p$, which correspond to Eq. (17). Suppose there are l spins that T_p^z acts on; then the number of independent stabilizers is also l (one T_p^z, and $l-1$ independent T_q^x), so the dimension $\mathcal{L} = 2^{l-l} = 1$. Therefore, we can find the unique stabilizer state of this subsystem, and express it using the function $f_p(\mathbf{v}_p)$.

Similarly, we can take out the s spins that T_q^x acts on and analyze this subsystem. Let T_p^z and $T_{q'}^x$ be the stabilizers other than T_q^x that act on part of the spins in this subsystem. T_p^z do not flip spins; thus they have no effect on Eq. (17), and we can ignore them in the analysis. Since T_q^x commutes with every other $T_{q'}^x$, in the most general case there can exist a $T_{q'}^x$ for every spin in this subsystem, and the number of independent stabilizers is s (one for each spin). Therefore the dimension of this subsystem is also 1, and we can also express the ground state of the subsystem using a function $f_q(\mathbf{v}_q)$.

Attaching one hidden neuron to each stabilizer, we can get the RBM representation of the $\mathbf{S}_X \sqcup \mathbf{S}_Z$ stabilizer formalism similarly to the Kitaev toric code model. One solution is

$$a_k = \frac{i\pi}{4}, \quad b_p = -l\frac{i\pi}{4}, \quad W_{pp_k} = \frac{i\pi}{4}, \quad \text{for } T_p^z,$$
$$a_k = 0, \quad b_q = 0, \quad W_{qq_k} = 0, \quad \text{for } T_q^x.$$

We can check that these solutions satisfy Eqs. (16) and (17). Hidden neurons with $b = 0$ and $W = 0$ have no contribution in the wave function; therefore we can remove the hidden neurons corresponding to T_q^x. The details in calculation can also be found in the Appendix.

Algorithm 1 Constructing RBM representation for $\mathbf{S}_X \sqcup \mathbf{S}_Z$, $\mathbf{S}_Y \sqcup \mathbf{S}_Z$, and $\mathbf{S}_X \sqcup \mathbf{S}_Y$ stabilizer states

Input: The group of stabilizer generators
$G = \{T_1, T_2, \ldots, T_m\}$
Output: The RBM parameters $\{a_i, b_j, W_{ij}\}$
Begin with no hidden neurons and all weights set to 0.
if $G \in \mathbf{S}_Y \sqcup \mathbf{S}_Z$ **then**
  for v_i in $\mathbf{v}$ **do**
    $a_i = a_i - \frac{i\pi}{4}$
    Replace all $\mathbf{S}_Y$ with $\mathbf{S}_X$.
  end for
else if $G \in \mathbf{S}_X \sqcup \mathbf{S}_Y$ **then**
  Change to σ_y basis.
end if ▷ Do nothing when $G \in \mathbf{S}_X \sqcup \mathbf{S}_Z$
for $j = 1$ to m **do**
  if $T_j \in \mathbf{S}_Z$ **then**
    Add a hidden neuron h_j
    for $v_i \in \mathbf{v}_j$ **do**
      $a_i = a_i + \frac{i\pi}{4}, b_j = b_j - \frac{i\pi}{4}, W_{ij} = \frac{i\pi}{4}$
    end for
  else if $T_j \in \mathbf{S}_X$ **then**
    continue
  end if
end for

However, this method fails if we try to generalize it to the situation where $T_r^y \in \mathbf{S}_Y$. Since T_r^y commutes with every other $T_{r'}^y$, the number of spins shared by them can be odd.

However, our general methodology tells us that for the subsystem $\mathbf{v}_r$, the major stabilizer is T_r^y, while the nonmajor stabilizers are actually $T_{r'_{\mathbf{v}_r}}^x \in \mathbf{S}_X$, which may not commute with T_r^y. Therefore there may not exist a common eigenstate for the stabilizers in this subsystem, and this method fails.

The RBM representation of T_r^y can be deduced from the T_q^x case. Since $\sigma_x|v\rangle = |-v\rangle$ and $\sigma_y|v\rangle = iv|-v\rangle = \exp(\frac{i\pi}{2}v)|-v\rangle$, using the function $T_r^y|\Psi\rangle = +1|\Psi\rangle$, we have

$$\begin{aligned} &T_r^y \Psi(\mathbf{v}_r, \tilde{\mathbf{v}}; \Omega)|\mathbf{v}_r, \tilde{\mathbf{v}}\rangle \\ &= \exp\left(\frac{i\pi}{2}\sum_r v_r\right) T_r^x \Psi(\mathbf{v}_r, \tilde{\mathbf{v}}; \Omega)|\mathbf{v}_r, \tilde{\mathbf{v}}\rangle \\ &= \Psi(-\mathbf{v}_r, \tilde{\mathbf{v}}; \Omega)|-\mathbf{v}_r, \tilde{\mathbf{v}}\rangle \end{aligned}$$

or

$$\begin{aligned} &T_r^x \left[\exp\left(\frac{i\pi}{4}\sum_r v_r\right)\Psi(\mathbf{v}_r, \tilde{\mathbf{v}}; \Omega)\right]|\mathbf{v}_r, \tilde{\mathbf{v}}\rangle \\ &= \left\{\exp\left[\frac{i\pi}{4}\sum_r (-v_r)\right]\Psi(-\mathbf{v}_r, \tilde{\mathbf{v}}; \Omega)\right\}|-\mathbf{v}_r, \tilde{\mathbf{v}}\rangle. \end{aligned}$$

Suppose the eigenstate for T_r^x with eigenvalue 1 is $|\Psi'\rangle$; then we have

$$\Psi(\mathbf{v}_r, \tilde{\mathbf{v}}; \Omega) = \exp\left(-\frac{i\pi}{4}\sum_r v_r\right)\Psi'(\mathbf{v}_r, \tilde{\mathbf{v}}; \Omega).$$

Therefore, we conclude that for T_r^y, the RBM parameters are

$$a_k = a'_k - \frac{i\pi}{4}, \quad b_r = b'_r, \quad W_{rr_k} = W'_{rr_k}, \tag{18}$$

where the parameters with a prime denote the parameters for the corresponding T_r^x case. Note that the result here is for the whole spin system, while the results we get for T_p^z and T_q^x earlier are for the subsystems taken out from the big system.

Using Eq. (18) and our previous result for $\mathbf{S}_X \sqcup \mathbf{S}_Z$, we can directly get the RBM representation for the $\mathbf{S}_Y \sqcup \mathbf{S}_Z$ cases. Furthermore, as for the $\mathbf{S}_X \sqcup \mathbf{S}_Y$ cases, if we use the σ_y basis instead of the σ_z basis, the matrix form of the Pauli operators under the new basis reads

$$\sigma_x' = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \to \sigma_y, \quad \sigma_y' = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \to \sigma_z,$$
$$\sigma_z' = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \to \sigma_x.$$

In this way we convert $\mathbf{S}_X \sqcup \mathbf{S}_Y$ to $\mathbf{S}_Y \sqcup \mathbf{S}_Z$, which we have already solved.

To conclude, we have summarized our construction in Algorithm 1. However, one must note that this algorithm only specifies one code state in the entire code space. To convert between different code states, we utilize the RBM representation of string operators. See Sec. V D for details.

B. $\mathbf{S}_{XY}$, $\mathbf{S}_{XZ}$, $\mathbf{S}_{YZ}$, and $\mathbf{S}_{XYZ}$

When generalizing our method in Sec. IV A to more complicated cases, problems arise. When $G \in \mathbf{S}_X \sqcup \mathbf{S}_Z$, for each stabilizer T_j, we can always take $\mathbf{v}_j$ as a subsystem, and the solution is guaranteed to exist; however, when $G \in \mathbf{S}_{XZ}$, there exist cases where there are more independent stabilizers than spins in each subsystem, so there does not exist a common eigenstate, and our method fails. Under such circumstances, merging the neighboring subsystems into bigger subsystems may help, but no general merging rules have been found, and in the worst case the size of a subsystem might reach the size of the whole system. Therefore, we resort to numerical methods instead.

Reference [52] proved the existence of efficient RBM representations for stabilizer states, confirming that a numerical method would work. Therefore, we try to construct a fully connected RBM for each subsystem, and obtain the RBM representation of the state via Eq. (14). By appropriately choosing subsystems, the system size we have to deal with will be much smaller.

Here we briefly introduce the method of training the RBM. Given the set of stabilizers $\{T_1, \ldots, T_m\}$, the Hamiltonian is defined as $H = -\sum_j T_j$, and the code states are ground states of the Hamiltonian. This problem is already solved in [4], in which was adopted reinforcement learning to minimize the energy. But for small subsystems, higher accuracy can be achieved by first calculating the full state vector $|\Psi_j\rangle$ for the subsystem, then minimizing the distance function

$$d = \arccos\sqrt{\frac{\langle\Psi_j|\Psi_{\mathrm{RBM}}\rangle\langle\Psi_{\mathrm{RBM}}|\Psi_j\rangle}{\langle\Psi_j|\Psi_j\rangle\langle\Psi_{\mathrm{RBM}}|\Psi_{\mathrm{RBM}}\rangle}}.$$

The cost for exactly computing $|\Psi_j\rangle$ and $|\Psi_{\mathrm{RBM}}\rangle$ grows exponentially, but is tractable for subsystems with moderate size. A small trick to reduce complexity by a constant factor is to evaluate the RBM coefficients in the sequence of the gray code.

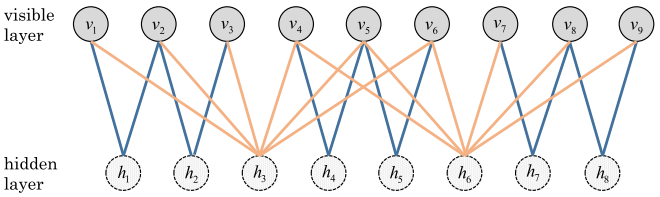

FIG. 4. Restricted Boltzmann machine representation of Shor's [[9,1,3]] code state in stabilizer formalism.

So far, we have given the relatively complete construction of the RBM state in the stabilizer formalism. Later in this paper we will give examples for different situations.

Let us take Shor's [[9,1,3]] code [53] as an example to illustrate our general approach for constructing stabilizer states. The stabilizer generators $\mathbf{S}_{[[9,1,3]]}$ for Shor's code are

$$\begin{aligned} T_1 &= \sigma_z^1\sigma_z^2 IIIIIII, \\ T_2 &= I\sigma_z^2\sigma_z^3 IIIIII, \\ T_3 &= \sigma_x^1\sigma_x^2\sigma_x^3\sigma_x^4\sigma_x^5\sigma_x^6 III, \\ T_4 &= III\sigma_z^4\sigma_z^5 III, \\ T_5 &= IIII\sigma_z^5\sigma_z^6 II, \\ T_6 &= III\sigma_x^4\sigma_x^5\sigma_x^6\sigma_x^7\sigma_x^8\sigma_x^9, \\ T_7 &= IIIIII\sigma_z^7\sigma_z^8 I, \\ T_8 &= IIIIIII\sigma_z^8\sigma_z^9. \end{aligned} \tag{19}$$

As depicted in Fig. 4, we assign a hidden neuron h_k to each stabilizer T_k such that it is only connected with the qubits (visible neurons) which T_k acts on nontrivially. Note that $\mathbf{S}_{[[9,1,3]]} = \mathbf{S}_X \sqcup \mathbf{S}_Z$; $T_1, T_2, T_4, T_5, T_7, T_8$ are of $\mathbf{S}_Z$ type with each of them acting on $l = 2$ qubits nontrivially. T_3 and T_6 are of $\mathbf{S}_X$ type, and among the qubits they act on nontrivially, the number of T_p^z that act on $v_1, v_3, v_4, v_6, v_7, v_9$ is 1, and that act on v_2, v_5, v_8 is 2. Thus the RBM parameters $\Omega_{[[9,1,3]]}$ for $|\Psi_{[[9,1,3]]}\rangle$ are

$$\begin{aligned} &a_k = i\frac{\pi}{4}, \quad k = 1,3,4,6,7,9, \\ &a_k = i\frac{\pi}{2}, \quad k = 2,5,8, \\ &b_p = -i\frac{\pi}{2}, \quad W_{pp_k} = i\frac{\pi}{4}, \quad p = 1,2,4,5,7,8, \\ &b_q = 0, \quad W_{qq_k} = 0, \quad q = 3,6. \end{aligned}$$

V. EFFICIENT NEURAL NETWORK REPRESENTATION OF THE SURFACE CODE

Using the general result obtained above, now we explicitly construct the RBM representation of the defect surface code.

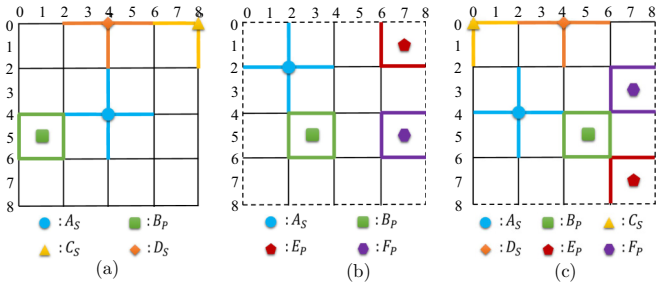

FIG. 5. Planar code with boundaries. (a) Smooth boundary; (b) rough boundary; (c) mixed boundary. The different types of stabilizers and the corresponding qubits they act on nontrivially are highlighted using different colors.

A. Planar code with boundaries

There are two types of boundaries for the planar code—smooth ones and rough ones, as shown in Fig. 5—and we will construct RBM representations for both cases.

1. Smooth boundaries

We take the 4×4 square lattice as a concrete example. We use $0, 1, 2, \ldots$ to label the rows and columns, and X_{ij} for the star (plaquette) operator on the vertex (face) (i, j) when both i and j are even (odd), as shown in Fig. 5. Similarly, v_{ij} denotes the qubit attached to the edge (i, j), where i and j have different parity. There are four types of stabilizers:

$$A_{ij} = \prod_{(m,n)\in \mathrm{star}(i,j)} \sigma_x^{mn}, \quad B_{ij} = \prod_{(m,n)\in \partial(i,j)} \sigma_z^{mn},$$

$$C_{ij} = \prod_{(m,n)\in \mathrm{star}(i,j) \text{ on the corner}} \sigma_x^{mn},$$

$$D_{ij} = \prod_{(m,n)\in \mathrm{star}(i,j) \text{ on the boundary}} \sigma_x^{mn},$$

where C_{ij} and D_{ij} denote the star operators on the corner and boundary, respectively. If (i, j) is not on the boundary or corner, then $\mathrm{star}(i, j)$ contains the 4 adjacent edges of the vertex (i, j), or $\mathrm{star}(i, j) = \{(i-1, j), (i+1, j), (i, j-1), (i, j+1)\}$. Otherwise it contains only 3 or 2 adjacent edges, as depicted in Fig. 5(a). $\partial(i, j)$ has the same expression except that (i, j) denotes a face instead of a vertex. As an example, the highlighted operators in Fig. 5(a) can be written as

$$\begin{aligned} &A_{44} = \sigma_x^{34}\sigma_x^{54}\sigma_x^{43}\sigma_x^{45}, \quad B_{51} = \sigma_z^{41}\sigma_z^{61}\sigma_z^{50}\sigma_z^{52}, \\ &C_{08} = \sigma_x^{07}\sigma_x^{18}, \quad D_{04} = \sigma_x^{14}\sigma_x^{03}\sigma_x^{05}. \end{aligned}$$

Using our conclusion above, we can connect a hidden neuron to each B_{ij}, and the RBM parameters are

$$a_{ij} = \begin{cases} \frac{i\pi}{2}, & i, j \in \{1, 2, \ldots, 7\}, \\ \frac{i\pi}{4}, & i \in \{0, 8\} \text{ or } j \in \{0, 8\}, \end{cases}$$

$$b_{B_{ij}} = -i\pi, \quad W_{B_{ij},k} = \frac{i\pi}{4}.$$

2. Rough boundaries

We also take the 4×4 square lattice as an example. There are four types of stabilizers, as shown in Fig. 5(b):

$$A_{ij} = \prod_{(m,n)\in \mathrm{star}(i,j)} \sigma_x^{mn},$$

$$B_{ij} = \prod_{(m,n)\in \partial(i,j)} \sigma_z^{mn},$$

$$E_{ij} = \prod_{(m,n)\in \partial(i,j)(i,j) \text{ on the corner}} \sigma_z^{mn},$$

$$F_{ij} = \prod_{(m,n)\in \partial(i,j)(i,j) \text{ on the boundary}} \sigma_z^{mn}.$$

As an example, the highlighted stabilizer operators are written as

$$A_{22} = \sigma_x^{12}\sigma_x^{32}\sigma_x^{21}\sigma_x^{23}, \quad B_{53} = \sigma_z^{43}\sigma_z^{63}\sigma_z^{52}\sigma_z^{54},$$
$$E_{17} = \sigma_z^{27}\sigma_z^{16}, \quad F_{57} = \sigma_z^{47}\sigma_z^{67}\sigma_z^{56}.$$

Using our conclusion above, the RBM parameters for this case are

$$a_{ij} = \frac{i\pi}{2}, \quad b_{B_{ij}} = -i\pi, \quad b_{E_{ij}} = -\frac{i\pi}{2}, \quad b_{F_{ij}} = -\frac{3i\pi}{4},$$
$$W_{X_{ij},k} = \frac{i\pi}{4}, \quad X \in \{B, E, F\}.$$

3. Mixed boundaries

In this example, the upper and left-hand sides of the lattice have smooth boundaries, while the lower and right-hand sides have rough boundaries. Therefore all six types of stabilizers appear in this example, as shown in Fig. 5(c). We can calculate the RBM parameters in this case, which are

$$a_{ij} = \begin{cases} \frac{i\pi}{4}, & i = 0 \text{ or } j = 0, \\ \frac{i\pi}{2}, & \text{otherwise}, \end{cases}$$
$$b_{B_{ij}} = -i\pi, \; b_{E_{ij}} = -\frac{i\pi}{2}, \quad b_{F_{ij}} = -\frac{3i\pi}{4},$$
$$W_{X_{ij},k} = \frac{i\pi}{4}, \quad X \in \{B, E, F\}.$$

B. Planar code with defects

In this section, we will discuss the RBM representation of smooth and rough defects in the planar code.

1. Smooth defect

As Fig. 6(a) shows, the smooth defect causes the change in the four highlighted stabilizers, which are

$$D_{24} = \sigma_x^{14}\sigma_x^{23}\sigma_x^{25}, \quad D_{42} = \sigma_x^{32}\sigma_x^{52}\sigma_x^{41},$$
$$D_{46} = \sigma_x^{36}\sigma_x^{56}\sigma_x^{47}, \quad D_{64} = \sigma_x^{63}\sigma_x^{65}\sigma_x^{74}.$$

And the vertices (2,2),(2,6),(6,2),(6,6) have no operators defined on them. Using our conclusions above, the RBM parameters in this case are

$$a_{ij} = \begin{cases} \frac{i\pi}{4}, & (i, j) \text{ on the boundary of the defect}, \\ \frac{i\pi}{2}, & \text{otherwise}, \end{cases}$$
$$b_{B_{ij}} = -i\pi, \quad W_{B_{ij},k} = \frac{i\pi}{4}.$$

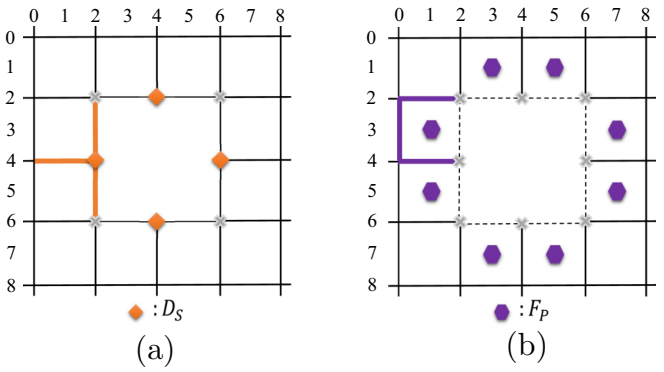

FIG. 6. Planar code with defects. (a) Smooth defect; (b) rough defect. The stabilizers affected by the defect are highlighted in the graph, and there are no operators defined on the vertices labeled with a cross.

2. Rough defect

As Fig. 6(b) shows, the rough defect causes the change in the eight highlighted stabilizers, where $F_{31} = \sigma_z^{21}\sigma_z^{41}\sigma_z^{30}$, and similarly for the others. The eight vertices labeled with a cross have no stabilizers defined on them. In this case, the RBM parameters are

$$a_{ij} = \frac{i\pi}{2}, \quad b_{B_{ij}} = -i\pi, \quad b_{F_{ij}} = -\frac{3i\pi}{4},$$
$$W_{B_{ij},k} = \frac{i\pi}{4}, \quad W_{F_{ij},k} = \frac{i\pi}{4}.$$

C. Planar code with twists and typical machine-learning procedure for complicated cases

The domain wall and twist have already been described in Sec. V C. We introduced a new twist operator $Q = \sigma_x^5\sigma_y^1\sigma_z^2\sigma_z^3\sigma_z^4$, and the plaquette operators near the domain wall W also changed, such as $W_p = \sigma_z^5\sigma_z^6\sigma_z^7\sigma_x^4$. Since $Q \in \mathbf{S}_{XYZ}$ and $W_p \in \mathbf{S}_{XZ}$, which we have not obtained a general result yet, in this section we explicitly construct the RBM representation of the planar code with twists using machine-learning techniques.

Figure 7 shows the planar code with a domain wall and twist. As described in Sec. IV B, we need to find a minimal subsystem in which the number of independent stabilizers is at most the same as the number of spins. It turns out that we need to include all the spins near the domain wall in the subsystem, and in this case the subsystem is the 13 highlighted spins, with 13 independent stabilizers acting on them. Therefore the dimension of this subsystem is $2^{13-13} = 1$, so that we can find a unique ground state for it.

Then we construct a local fully connected RBM for the 13 spins, with 13 hidden neurons. The target state Φ is the ground state of the subsystem, and the RBM state is denoted as Φ'.

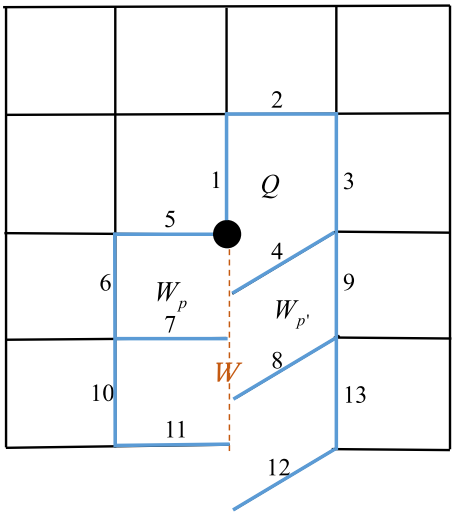

FIG. 7. Planar code with a domain wall and twist.

In the training process, we use an optimization procedure to minimize the distance function

$$d = \arccos \sqrt{\frac{\langle \Phi' | \Phi \rangle \langle \Phi | \Phi' \rangle}{\langle \Phi' | \Phi' \rangle \langle \Phi | \Phi \rangle}}.$$

Since this system is small, we can calculate the target state Φ exactly. We used the MATLAB Optimization Toolbox, which applies the sequential quadratic programming (SQP) algorithm, an iterative method for nonlinear optimization, to minimize the distance function d and to find a set of RBM parameters $\{a_i, b_j, W_{ij}\}$. Figure 8 shows the typical optimization procedure, in which the final value of d is 0.007, indicating the fidelity is 0.99995. We can see that the distance function converges smoothly to 0.

D. Topological excitations

The RBM representation of excited states in the Kitaev toric code model has already been constructed by Deng *et al.* in [26]. For the completeness of our paper, we quote their results and show that edge excitation can also be represented in similar ways.

There are two types of excitations: electric excitation created by the string operator $S^z(t) = \prod_{j \in t} \sigma_z^j$, and magnetic excitation created by the string operator $S^x(t') = \prod_{j \in t'} \sigma_x^j$. Reference [26] showed that acting the operator $S^z(t) = \prod_{j \in t} \sigma_z^j$ on the ground state corresponds to connecting a hidden neuron h_j to each v_j that $S^z(t)$ acts on, with parameters $b_j = -\frac{i\pi}{2}$, $W_j = \frac{i\pi}{2}$. After this operation, we have

$$\begin{aligned}
&\Psi'(\mathbf{v}_j, \tilde{\mathbf{v}})|\mathbf{v}_j, \tilde{\mathbf{v}}\rangle \\
&= \prod_j \left\{ \cosh\left[\frac{i\pi}{2}(v_j - 1) \right] \right\} \Psi(\mathbf{v}_j, \tilde{\mathbf{v}})|\mathbf{v}_j, \tilde{\mathbf{v}}\rangle \\
&= \left(\prod_j \sigma_z^j \right) \Psi(\mathbf{v}_j, \tilde{\mathbf{v}})|\mathbf{v}_j, \tilde{\mathbf{v}}\rangle,
\end{aligned}$$

and a pair of e particles are created. Meanwhile, acting the operator $S^x(t') = \prod_{j \in t'} \sigma_x^j$ on the ground state corresponds to flipping all the signs of the parameters associated with v_j. In this way,

$$\begin{aligned}
\Psi'(\mathbf{v}_j, \tilde{\mathbf{v}})|\mathbf{v}_j, \tilde{\mathbf{v}}\rangle &= \Psi(-\mathbf{v}_j, \tilde{\mathbf{v}})|\mathbf{v}_j, \tilde{\mathbf{v}}\rangle \\
&= \left(\prod_j \sigma_x^j \right) \Psi(-\mathbf{v}_j, \tilde{\mathbf{v}})|-\mathbf{v}_j, \tilde{\mathbf{v}}\rangle,
\end{aligned}$$

and a pair of m particles are created.

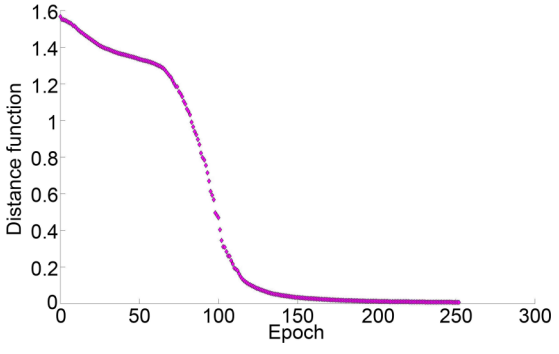

FIG. 8. The typical training procedure of a full connected RBM. The distance function converges smoothly to 0.

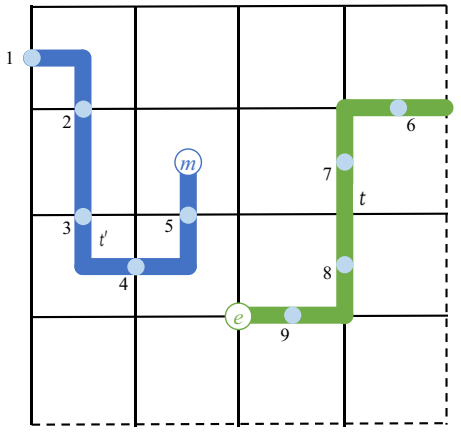

FIG. 9. String operators $S^z(t)$ and $S^x(t')$. Since one end of the string operator is on the boundary, only one e (or m) particle is created.

Figure 9 shows the two types of string operators. The string operator $S^z(t) = \sigma_z^6 \sigma_z^7 \sigma_z^8 \sigma_z^9$ should have created a pair of e particles, but since it has one end on the rough boundary, one e particle condensed into vacuum as it moved into the rough boundary. Similarly, $S^x(t') = \sigma_x^1 \sigma_x^2 \sigma_x^3 \sigma_x^4 \sigma_x^5$ has one end on the smooth boundary, so that it only creates one m particle on the other end. With the RBM representation of string operators $S^z(t)$ and $S^x(t')$, such physical process can be exactly and efficiently represented in RBM language.

VI. RBM REPRESENTATION FOR THE GENERAL $D(G)$ KITAEV MODEL

Consider a lattice with square geometry and assign d-level spins on each edge of the lattice. By labeling spin states with

the group elements $|0\rangle, \ldots, |d-1\rangle$, we then can introduce the generalized Pauli operators

$$X = \sum_{h\in\mathbb{Z}_d} |h+1(\mathrm{mod}d)\rangle\langle h|, \quad Z = \sum_{h\in\mathbb{Z}_d} \omega^h |h\rangle\langle h|, \tag{20}$$

where $\omega = e^{2\pi i/d}$ is the dth root of unity. For the $d=2$ case, we get the usual Pauli operators σ_x and σ_z, and they are anticommutative. In general, we have the commutation relation

$$ZX = \omega XZ. \tag{21}$$

Since X only displaces the label of basis by unity, it is easy to check that the eigenstates of X are of the form

$$|x\rangle = \frac{1}{\sqrt{d}} \sum_{h\in\mathbb{Z}_d} \omega^{xh} |h\rangle, \tag{22}$$

with the corresponding eigenvalue ω^{-x} for each $x \in \mathbb{Z}_d$.

Then we can define the star operators and plaquette operator as (see Fig. 1)

$$A_s = X_1 X_2 X_3^\dagger X_4^\dagger, \quad B_p = Z_5^\dagger Z_6 Z_7 Z_8^\dagger. \tag{23}$$

Note that now the lattice is a directed graph; thus the different directions are distinguished by operators and their Hermitian conjugates. All eigenvalues of A_v and B_p are of the form ω^g for some $g \in \mathbb{Z}_d$.

The Hamiltonian of the $D(\mathbb{Z}_d)$ model is then

$$H = -\sum_s \sum_{h\in\mathbb{Z}_d} (A_s)^h - \sum_p \sum_{h\in\mathbb{Z}_d} (B_p)^h. \tag{24}$$

Now, we try to construct the RBM representation for the general $D(G)$ Kitaev model. Since the spins can take d different values, we need to generalize the traditional two-value RBM to d-value cases. Specifically, for the generalized RBM, the visible-layer variables $\{v_1, \ldots, v_n\}$ can have d different values, while the hidden-layer variables $\{h_1, \ldots, h_m\}$ are still two-valued, where $v_i \in \{0, 1, \ldots, d-1\}$ and $h_j \in \{+1, -1\}$. The RBM ansatz takes the same form as Eq. (8), except that v_i becomes d-valued.

To begin with, consider the equation $B_p|\Psi\rangle = +1|\Psi\rangle$. Using $|\Psi\rangle = \sum_{\mathbf{v}} \Psi(\mathbf{v};\Omega)|\mathbf{v}\rangle$, we have

$$\begin{aligned} B_p \Psi(\mathbf{v};\Omega)|\mathbf{v}\rangle &= \exp\left(\frac{2\pi i}{d} \sum_k v_{p_k}^*\right) \Psi(\mathbf{v};\Omega)|\mathbf{v}\rangle \\ &= \Psi(\mathbf{v};\Omega)|\mathbf{v}\rangle, \end{aligned} \tag{25}$$

where $v_{p_k}^* = \pm v_{p_k}$, in which the plus sign is taken for the edge pointing in the positive direction (with respect to the plaquette), and the minus sign for the negative direction. To make Eq. (25) hold, we only need to restrict

$$\sum_k v_{p_k}^* = nd, \tag{26}$$

where n is an integer. To this end, we connect $d-1$ hidden neurons $h_l, l \in \{1, \ldots, d-1\}$, to $\{v_{p_1}, \ldots, v_{p_k}\}$, with $W_{p_l,p_k}^* = \frac{i\pi}{d}$ and $b_{p_l} = \frac{i\pi l}{d} - \frac{i\pi}{2}$. In this way, we have

$$\begin{aligned} \Psi(\mathbf{v};\Omega) &= \prod_p \left\{ 2^{d-1} \prod_l \cosh\left[\left(l + \sum_k v_{p_k}^* \right) \frac{i\pi}{d} - \frac{i\pi}{2} \right] \right\} \\ &= \prod_p \left\{ 2^{d-1} \prod_l \sin\left[\left(l + \sum_k v_{p_k}^* \right) \frac{\pi}{d} \right] \right\}. \end{aligned}$$

Since

$$\prod_l \sin\left[\left(l + \sum_k v_{p_k}^* \right) \frac{\pi}{d} \right] = \begin{cases} \pm \sin\frac{\pi}{d} \sin\frac{2\pi}{d} \cdots \sin\frac{(d-1)\pi}{d}, & \sum_k v_{p_k}^* = nd, \\ 0, & \text{otherwise}, \end{cases}$$

we can see that this set of parameters meets our requirement.

Then let us consider the equation $A_s|\Psi\rangle = +1|\Psi\rangle$. Since X is the shifting operator and each A_s acts on two adjacent spins in a plaquette, we can check that if both edges point in the positive (or negative) direction (with respect to the plaquette), A_s will raise one spin while lowering the other; otherwise A_s will raise or lower both spins. In both cases, the operator A_s conserves the sum ($\sum_k v_{p_k}^* \mod d$).

In most cases, the restriction $A_s|\Psi\rangle = +1|\Psi\rangle$ is automatically satisfied because the quantity $\sum_k v_{p_k}^*$ does not change after applying the operator A_s. However, $\sum_k v_{p_k}^*$ can also change by d, and we would have an extra -1 in the wave function. To make the restriction hold, we add an extra term $\exp(\frac{i\pi}{d}\sum_k v_{p_k}^*)$ to the wave function, which also adds an additional -1 to the wave function when $\sum_k v_{p_k}$ changes by d, and does not change when $\sum_k v_{p_k}^*$ does not change. We can check that $A_s|\Psi\rangle = +1|\Psi\rangle$ holds for this new wave function.

In conclusion, to represent the $D(G)$ Kitaev model in RBM language, we can connect $d-1$ hidden neurons to each plaquette, with RBM parameters

$$a_{p_k} = \pm\frac{i\pi}{d}, \quad b_{p_l} = \frac{i\pi l}{d} - \frac{i\pi}{2}, \quad W_{p_l,p_k} = \pm\frac{i\pi}{d},$$

where for a_{p_k} and W_{p_l,p_k}, the plus sign is taken for the edge pointing in the positive direction, and minus for the negative direction. For $d=2$, this model becomes the regular toric code model, and the RBM representation is equivalent to what we have constructed in Sec. IV A except that we use 0 and 1 to label spins here.

VII. CONCLUSIONS AND DISCUSSION

We have provided a systematic analysis of the RBM representation in the stabilizer formalism, and we find that for many crucial stabilizer groups, the exact RBM solutions exist and the number of hidden neurons is almost equal to the visible neurons. The developed results then enable us to analyze a surface code model with boundaries, defects, domain walls, and twists, and we also investigate the Kitaev $D(\mathbb{Z}_d)$ model in the form of an RBM that can be optimized using the variational Monte Carlo method, with the exact solution provided. Our result sheds light on the representational power of neural network states and gives guidance when building the RBM neural network in the stabilizer formalism. We also mention that in Ref. [52], we have shown that all stabilizers can be

reduced to the stabilizer groups which we studied in this work, see also [54] for the construction from the view point of topologically ordered states. Thus it is of central importance to construct the RBM representation in the stabilizer formalism. Many directions can been exploited further, such as providing the exact RBM solution of Kitaev's $D(G)$ model for non-Abelian group G and developing an algorithm to create the RBM solution in the stabilizer formalism. All these are left for future study.

ACKNOWLEDGMENTS

We acknowledge comments from anonymous referees which have substantially improved the paper. We also acknowledge Rui Zhai and Yan-Jun He for many helpful discussions. This work was supported by the National Key Research and Development Program of China (Grant No. 2016YFA0301700), the National Natural Science Foundation of China (Grants No. 11275182 and No. 11625419), and the Anhui Initiative in Quantum Information Technologies (Grant No. AHY080000).

Z.-A.J. and Y.-H.Z. contributed equally to this work.

APPENDIX: RBM REPRESENTATION IN THE STABILIZER FORMALISM

In this appendix, we give the detailed calculation as a supplement to Sec. IV A.

For $T_p^z = \sigma_z^{p_1}\sigma_z^{p_2}\cdots\sigma_z^{p_l} \in \mathbf{S}_Z$, T_p^z only flips the phases of spins $v_{p_1}, v_{p_2}, \ldots, v_{p_l}$, i.e., $T_p^z|v_1, v_2, \ldots, v_n\rangle = (\prod_{k=1}^{l} v_{p_k})|v_1, v_2, \ldots, v_n\rangle$. Therefore the constraint $T_p^z|\Psi\rangle = +1|\Psi\rangle$ can be represented in the RBM form as

$$\begin{aligned} &T_p^z\Psi(\mathbf{v};\Omega)|\mathbf{v}\rangle \\ &= \left(\prod_{k=1}^{l} v_{p_k}\right) e^{\sum_i a_i v_i} \prod_{j=1}^{m} 2\cosh\left(b_j + \sum_i W_{ji} v_i\right)|\mathbf{v}\rangle \\ &= e^{\sum_i a_i v_i} \prod_{j=1}^{m} 2\cosh\left(b_j + \sum_i W_{ji} v_i\right)|\mathbf{v}\rangle. \end{aligned}$$

By canceling the terms which are unrelated with the hidden neuron corresponding to T_p, we will get that

$$\begin{aligned} &\left(\prod_{k=1}^{l} v_{p_k}\right)\cosh\left(b_p + \sum_k W_{pp_k} v_{p_k}\right) \\ &\quad = \cosh\left(b_p + \sum_k W_{pp_k} v_{p_k}\right), \end{aligned}$$

where we use p_k to label the l visible neurons which are connected with h_p. Now if the number of -1 among v_{p_k} is $0, 2, 4, \ldots$, then we further have $\cosh(b_p + \sum_k W_{pp_k} v_{p_k}) = \cosh(b_p + \sum_k W_{pp_k} v_{p_k})$ which is obviously true; if the number of -1 among v_{p_k} is $1, 3, 5, \ldots$, then we have $-\cosh(b_p + \sum_k W_{pp_k} v_{p_k}) = \cosh(b_p + \sum_k W_{pp_k} v_{p_k})$, from which we know that $\cosh(b_p + \sum_k W_{pp_k} v_{p_k})$ must be zero. To this end, we restrict ourselves to

$$b_p + \sum_k W_{pp_k} v_{p_k} = i\frac{2m+1}{2}\pi \tag{A1}$$

when the number of -1 among v_{p_k} is odd, with m an integer.

There are many solutions of Eq. (A1); we need to adjust the b_p and W_{pp_k} to fit our need. Here we provide a solution, where we take W_{pp_k} the same for all v_{p_k}. It is easy to check that the weights related to the hidden neuron h_p (which corresponds to T_p) can be

$$\begin{aligned} &b_p = -i\frac{\pi}{4}, \quad W_{pp_k} = i\frac{\pi}{4}; \quad l = 1, 5, 9, 13, \ldots, \\ &b_p = -i\frac{\pi}{2}, \quad W_{pp_k} = i\frac{\pi}{4}; \quad l = 2, 6, 10, 14, \ldots, \\ &b_p = i\frac{\pi}{4}, \quad W_{pp_k} = i\frac{\pi}{4}; \quad l = 3, 7, 11, 15, \ldots, \\ &b_p = i\frac{\pi}{2}, \quad W_{pp_k} = i\frac{\pi}{4}; \quad l = 4, 8, 12, 16, \ldots. \end{aligned} \tag{A2}$$

From Eq. (A1), we can see that adding $ni\pi$ to b_p will not change the result, where n can be an arbitrary integer. As we only need one solution, we can rewrite Eq. (A2) in a more compact form:

$$b_p = -i\frac{l\pi}{4}, \quad W_{pp_k} = i\frac{\pi}{4}. \tag{A3}$$

However, we must point out that this result only holds when there only exists one type of stabilizer T_p^z. More general cases will be discussed later.

The case for $T_q^x = \sigma_x^{q_1}\sigma_x^{q_2}\cdots\sigma_x^{q_s} \in \mathbf{S}_X$ is more complicated. T_q^x will flip the spins of $v_{q_1}, v_{q_2}, \ldots, v_{q_l}$, i.e., $T_q^x|v_{q_1}, \ldots, v_{q_s}, \ldots\rangle = |-v_{q_1}, \ldots, -v_{q_s}, \ldots\rangle$. Therefore the constraint $T_q^x|\Psi\rangle = +1|\Psi\rangle$ can be represented in RBM form as

$$T_q^x\Psi(\mathbf{v}_q, \tilde{\mathbf{v}}; \Omega) = \Psi(-\mathbf{v}_q, \tilde{\mathbf{v}}; \Omega).$$

More precisely, if we cancel the terms unrelated to visible neurons $v_{q_k}, k = 1, \ldots, s$, we have

$$\begin{aligned} &e^{\sum_k a_{q_k}(-v_{q_k})}\cosh\left[b_q + \sum_k W_{qq_k}(-v_{q_k})\right] \\ &\quad\times \prod_{q',\langle q'q\rangle} \cosh\left[b_{q'} + \sum_k W_{q'q_k}(-v_{q_k}) + \sum_{q_k'\neq q_k} W_{q'q_k'} v_{q_k'}\right] \\ &= e^{\sum_k a_{q_k} v_{q_k}}\cosh\left(b_q + \sum_k W_{qq_k} v_{q_k}\right) \\ &\quad\times \prod_{q',\langle q'q\rangle} \cosh\left(b_{q'} + \sum_k W_{q'q_k} v_{q_k} + \sum_{q_k'\neq q_k} W_{q'q_k'} v_{q_k'}\right), \end{aligned}$$

where by $\langle q'q\rangle$ we mean that T_q and $T_{q'}$ share some visible neurons. To solve the equation directly is very difficult; now to illustrate the validity of our architecture, we only give one special solution to this equation, where we let the corresponding terms on each side of the equation equal each other. The solution can be chosen as

$$a_{q_k} = ni\pi, \ W_{qq_k} = 0,$$

and b_q can take any value. Specifically, we choose

$$a_{q_k} = 0, \ b_q = 0, \ W_{qq_k} = 0.$$

We need to explain this result here, for it seems that we only obtained a trivial solution, since we supposed that $T_q^x \in \mathbf{S}_X$,

which means that all stabilizer generators only flip the spins without adding a phase factor, or that all the involved spin configurations have the same coefficient in the wave function, which is exactly the same with our result above. Therefore, we can remove the hidden neuron corresponding to T_q^x. Again we must emphasize that this result only holds for $T_q^x \in \mathbf{S}_X$, without any other types of stabilizers.

Now let us consider what will happen if we combine two sets of constraints together. To begin with we consider the case where $\{T_p^z, T_q^x\} \in \mathbf{S}_X \sqcup \mathbf{S}_Z$. T_p^z does not involve spin flips, and the constraint $T_p^z|\Psi\rangle = +1|\Psi\rangle$ still needs to be satisfied. Thus the hidden neuron corresponding to T_p^z remains unchanged, with the weights

$$b_p = -i\frac{l\pi}{4}, \quad W_{pp_k} = i\frac{\pi}{4}. \tag{A4}$$

However, T_q^x will flip spins that T_p^z acts on, and the result is different. After canceling the terms unrelated to v_{q_k}, we have

$$\begin{aligned}
&e^{\sum_k a_{q_k} v_{q_k}} \cosh\left(b_q + \sum_k W_{qq_k} v_{q_k}\right) \prod_{q',\langle q'q\rangle} \cosh\left(b_{q'} + \sum_k W_{q'q_k} v_{q_k} + \sum_{q'_k \neq q_k} W_{q'q'_k} v_{q'_k}\right) \\
&\quad \times \prod_{p,\langle pq\rangle} \cosh\left(b_p + \sum_k W_{pq_k} v_{q_k} + \sum_{p_k \neq q_k} W_{pp_k} v_{p_k}\right) \\
&= e^{\sum_k a_{q_k}(-v_{q_k})} \cosh\left[b_q + \sum_k W_{qq_k}(-v_{q_k})\right] \prod_{q',\langle q'q\rangle} \cosh\left[b_{q'} + \sum_k W_{q'q_k}(-v_{q_k}) + \sum_{q'_k \neq q_k} W_{q'q'_k} v_{q'_k}\right] \\
&\quad \times \prod_{p,\langle pq\rangle} \cosh\left[b_p + \sum_k W_{pq_k}(-v_{q_k}) + \sum_{p_k \neq q_k} W_{pp_k} v_{p_k}\right].
\end{aligned}$$

In order to find a solution to this equation, we first analyze the last term:

$$\begin{aligned}
&\cosh\left[b_p + \sum_k W_{pq_k}(-v_{q_k}) + \sum_{p_k \neq q_k} W_{pp_k} v_{p_k}\right] \\
&= \cosh\left(b_p + \sum_k W_{pp_k} v_{p_k} - 2\sum_k W_{pq_k} v_{q_k}\right) = \cosh\left(b_p + \sum_k W_{pp_k} v_{p_k} - \frac{i\pi}{2}\sum_k v_{q_k}\right),
\end{aligned} \tag{A5}$$

where in the last equation we used the result $W_{pq_k} = \frac{i\pi}{4}$. Since T_p^z and T_q^x commute with each other, the number of visible neurons shared by T_p^z and T_q^x is even, or $\sum_{k,\langle pq\rangle} v_{q_k} = 2m$. Thus, we can further simplify Eq. (A5) into

$$\cosh\left(b_p + \sum_k W_{pp_k} v_{p_k} - \frac{i\pi}{2}\sum_k v_{q_k}\right) = \begin{cases} \cosh(b_p + \sum_k W_{pp_k} v_{p_k}), & \sum_{k,\langle pq\rangle} v_{q_k} = 4n, \\ -\cosh(b_p + \sum_k W_{pp_k} v_{p_k}), & \sum_{k,\langle pq\rangle} v_{q_k} = 4n+2, \end{cases}$$

or

$$\cosh\left(b_p + \sum_k W_{pp_k} v_{p_k} - \frac{i\pi}{2}\sum_k v_{q_k}\right) = e^{\frac{i\pi}{2}\sum_{k,\langle pq\rangle} v_{q_k}} \cosh\left(b_p + \sum_k W_{pp_k} v_{p_k}\right).$$

In this way, the equation following Eq. (A4) becomes

$$\begin{aligned}
&e^{\sum_k a_{q_k} v_{q_k}} \cosh\left(b_q + \sum_k W_{qq_k} v_{q_k}\right) \prod_{q',\langle q'q\rangle} \cosh\left(b_{q'} + \sum_k W_{q'q_k} v_{q_k} + \sum_{q'_k \neq q_k} W_{q'q'_k} v_{q'_k}\right) \\
&= e^{\sum_k a_{q_k}(-v_{q_k})} \cosh\left[b_q + \sum_k W_{qq_k}(-v_{q_k})\right] \prod_{q',\langle q'q\rangle} \cosh\left[b_{q'} + \sum_k W_{q'q_k}(-v_{q_k}) + \sum_{q'_k \neq q_k} W_{q'q'_k} v_{q'_k}\right] \\
&\quad \times \prod_{p,\langle pq\rangle} e^{\frac{i\pi}{2}\sum_{k,\langle pq\rangle} v_{q_k}}.
\end{aligned}$$

To find a solution, we let

$$e^{\sum_k a_{q_k} v_{q_k}} = e^{\sum_k a_{q_k}(-v_{q_k})} \prod_{p,\langle pq\rangle} e^{\frac{i\pi}{2}\sum_{k,\langle pq\rangle}(v_{q_k})}, \quad \cosh\left(b_q + \sum_k W_{qq_k} v_{q_k}\right) = \cosh\left[b_q + \sum_k W_{qq_k}(-v_{q_k})\right],$$

$$\cosh\left(b_{q'} + \sum_k W_{q'q_k} v_{q_k} + \sum_{q'_k \neq q_k} W_{q'q'_k} v_{q'_k}\right) = \cosh\left[b_{q'} + \sum_k W_{q'q_k}(-v_{q_k}) + \sum_{q'_k \neq q_k} W_{q'q'_k} v_{q'_k}\right].$$

Therefore the solution is

$$a_{q_k} = n_{p,q_k}\frac{i\pi}{4}, \quad b_q = 0, \quad W_{qq_k} = 0, \tag{A6}$$

where n_{p,q_k} denotes the number of T^z_p that acts on v_{q_k}. b_q can take any value, so we choose it to be 0 and remove the hidden neuron corresponding to T^x_q.

To better illustrate the physical meanings of these parameters, we rearrange Eqs. (A4) and (A6):

$$a_k = \frac{i\pi}{4}, \quad b_p = -l\frac{i\pi}{4}, \quad W_{pp_k} = \frac{i\pi}{4}, \quad \text{for } T^z_p,$$

$$a_k = 0, \quad b_q = 0, \quad W_{qq_k} = 0, \quad \text{for } T^x_q.$$

We reassigned the parameters a_k, and this is the result we give in Sec. IV A. In this way, the wave function becomes

$$\begin{aligned}\Psi(\mathbf{v};\Omega) &= \prod_p \left\{\exp\left(\frac{i\pi}{4}\sum_k v_{p_k}\right) \times 2\cosh\left[\frac{i\pi}{4}\sum_k (v_{p_k} - 1)\right]\right\} \\ &= \prod_p f_p(\mathbf{v}_p).\end{aligned}$$

We can check that $f_p(\mathbf{v}_p) = e^{l\frac{i\pi}{4}}$ is the same for all spin configurations with $\prod_k v_{p_k} = 1$. Or, the wave function remains unchanged after flipping an even number of spins, which meets our requirement. We can further check that every condition in Sec. IV A is satisfied.

[1] A. Kitaev, Fault-tolerant quantum computation by anyons, Ann. Phys. **303**, 2 (2003).

[2] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, Nature (London) **521**, 436 (2015).

[3] G. E. Hinton and R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science **313**, 504 (2006).

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

[5] D. J. Scalapino and R. L. Sugar, Method for Performing Monte Carlo Calculations for Systems with Fermions, Phys. Rev. Lett. **46**, 519 (1981).

[6] R. Blankenbecler, D. J. Scalapino, and R. L. Sugar, Monte Carlo calculations of coupled boson-fermion systems. I, Phys. Rev. D **24**, 2278 (1981).

[7] F. Fucito, G. Parisi, E. Marinari, and C. Rebbi, A proposal for Monte Carlo simulations of fermionic systems, Nucl. Phys. B **180**, 369 (1980).

[8] J. E. Hirsch, Two-dimensional Hubbard model: Numerical simulation study, Phys. Rev. B **31**, 4403 (1985).

[9] J. E. Hirsch, D. J. Scalapino, R. L. Sugar, and R. Blankenbecler, Efficient Monte Carlo Procedure for Systems with Fermions, Phys. Rev. Lett. **47**, 1628 (1981).

[10] R. Orús, A practical introduction to tensor networks: Matrix product states and projected entangled pair states, Ann. Phys. **349**, 117 (2014).

[11] Z. Landau, U. Vazirani, and T. Vidick, A polynomial time algorithm for the ground state of one-dimensional gapped local Hamiltonians, Nat. Phys. **11**, 566 (2015).

[12] I. Arad, Z. Landau, U. Vazirani, and T. Vidick, Rigorous RG algorithms and area laws for low energy eigenstates in 1D, Commun. Math. Phys. **356**, 65 (2017).

[13] N. Schuch, M. M. Wolf, F. Verstraete, and J. I. Cirac, Computational Complexity of Projected Entangled Pair States, Phys. Rev. Lett. **98**, 140506 (2007).

[14] A. Anshu, I. Arad, and A. Jain, How local is the information in tensor networks of matrix product states or projected entangled pairs states, Phys. Rev. B **94**, 195143 (2016).

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

[16] Y. Huang and J. E. Moore, Neural network representation of tensor network and chiral states, arXiv:1701.06246.

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

[18] I. Glasser, N. Pancotti, M. August, I. D. Rodriguez, and J. I. Cirac, Neural-Network Quantum States, String-Bond States, and Chiral Topological States, Phys. Rev. X **8**, 011006 (2018).

[19] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, Vol. 1 (MIT Press, Cambridge, MA, 1998).

[20] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, Nature (London) **549**, 195 (2017).

[21] P. Rebentrost, M. Mohseni, and S. Lloyd, Quantum Support Vector Machine for Big Data Classification, Phys. Rev. Lett. **113**, 130503 (2014).

[22] V. Dunjko, J. M. Taylor, and H. J. Briegel, Quantum-Enhanced Machine Learning, Phys. Rev. Lett. **117**, 130501 (2016).
[23] A. Monràs, G. Sentís, and P. Wittek, Inductive Supervised Quantum Learning, Phys. Rev. Lett. **118**, 190503 (2017).
[24] J. Carrasquilla and R. G. Melko, Machine learning phases of matter, Nat. Phys. **13**, 431 (2017).
[25] D.-L. Deng, X. Li, and S. Das Sarma, Quantum Entanglement in Neural Network States, Phys. Rev. X **7**, 021021 (2017).
[26] D.-L. Deng, X. Li, and S. Das Sarma, Machine learning topological states, Phys. Rev. B **96**, 195145 (2017).
[27] M. August and X. Ni, Using recurrent neural networks to optimize dynamical decoupling for quantum memory, Phys. Rev. A **95**, 012335 (2017).
[28] G. Torlai and R. G. Melko, Neural Decoder for Topological Codes, Phys. Rev. Lett. **119**, 030501 (2017).
[29] Y. Zhang and E.-A. Kim, Quantum Loop Topography for Machine Learning, Phys. Rev. Lett. **118**, 216401 (2017).
[30] G. Torlai, G. Mazzola, J. Carrasquilla, M. Troyer, R. Melko, and G. Carleo, Neural-network quantum state tomography, Nat. Phys. **14**, 447 (2018).
[31] Y. Huang and J. E. Moore, Neural network representation of tensor network and chiral states, arXiv:1701.06246.
[32] H. Saito, Solving the Bose-Hubbard model with machine learning, J. Phys. Soc. Jpn. **86**, 093001 (2017).
[33] Z. Cai and J. Liu, Approximating quantum many-body wave functions using artificial neural networks, Phys. Rev. B **97**, 035116 (2018).
[34] D. Gottesman, Stabilizer codes and quantum error correction, Ph.D. thesis, Caltech, 1997, arXiv:quant-ph/9705052.
[35] D. Gottesman, Theory of fault-tolerant quantum computation, Phys. Rev. A **57**, 127 (1998).
[36] A. Kitaev and L. Kong, Models for gapped boundaries and domain walls, Commun. Math. Phys. **313**, 351 (2012).
[37] S. Beigi, P. W. Shor, and D. Whalen, The quantum double model with boundary: Condensations and symmetries, Commun. Math. Phys. **306**, 663 (2011).
[38] H. Bombin and M. A. Martin-Delgado, Family of non-Abelian Kitaev models on a lattice: Topological condensation and confinement, Phys. Rev. B **78**, 115421 (2008).
[39] I. Cong, M. Cheng, and Z. Wang, Hamiltonian and algebraic theories of gapped boundaries in topological phases of matter, Commun. Math. Phys. **355**, 645 (2017).
[40] S. B. Bravyi and A. Y. Kitaev, Quantum codes on a lattice with boundary, arXiv:quant-ph/9811052.
[41] M. H. Freedman and D. A. Meyer, Projective plane and planar quantum codes, Found. Comput. Math. **1**, 325 (2001).
[42] H. Bombin, Topological Order with a Twist: Ising Anyons from an Abelian Model, Phys. Rev. Lett. **105**, 030403 (2010).
[43] P. Smolensky, Information processing in dynamical systems: Foundations of harmony theory, Tech. Rep., University of Colorado at Boulder, Department of Computer Science, 1986.
[44] G. E. Hinton and T. J. Sejnowski, Optimal perceptual inference, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, New York, 1983), pp. 448–453.
[45] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, A learning algorithm for Boltzmann machines, Cognit. Sci. **9**, 147 (1985).
[46] A. Kolmogorov, On the representation of continuous functions of several variables by superpositions of continuous functions of a smaller number of variables, Dokl. Akad. Nauk SSSR **108**, 179 (1956).
[47] A. N. Kolmogorov, On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition, Dokl. Akad. Nauk SSSR **114**, 953 (1957).
[48] V. I. Arnold, *Vladimir I. Arnold—Collected Works: Representations of Functions, Celestial Mechanics, and KAM Theory 1957-1965*, Vol. 1 (Springer Science & Business Media, 2009).
[49] N. L. Roux and Y. Bengio, Representational power of restricted Boltzmann machines and deep belief networks, Neural Comput. **20**, 1631 (2008).
[50] Z.-A. Jia, Y. Biao, Z. Rui, Y.-C. Wu, G.-C. Guo, and G.-P. Guo, Quantum neural network states, arXiv:1808.10601.
[51] R. Jastrow, Many-body problem with strong forces, Phys. Rev. **98**, 1479 (1955).
[52] Y.-H. Zhang, Z.-A. Jia, Y.-C. Wu, G.-C. Guo, and G.-P. Guo, An efficient algorithmic way to construct Boltzmann machine representations for arbitrary stabilizer code, arXiv:1809.08631.
[53] P. W. Shor, Scheme for reducing decoherence in quantum computer memory, Phys. Rev. A **52**, R2493 (1995).
[54] S. Lu, X. Gao, and L.-M. Duan, Efficient representation of topologically ordered states with restricted Boltzmann machines, arXiv:1810.02352.