Unsupervised Generative Modeling Using Matrix Product States

Zhao-Yu Han,[1] Jun Wang,[1] Heng Fan,[2,4] Lei Wang,[2,4,*] and Pan Zhang[3,†]

[1]*School of Physics, Peking University, Beijing 100871, China*
[2]*Institute of Physics, Chinese Academy of Sciences, Beijing 100190, China*
[3]*Key Laboratory of Theoretical Physics, Institute of Theoretical Physics,*
Chinese Academy of Sciences, Beijing 100190, China
[4]*CAS Center for Excellence in Topological Quantum Computation,*
University of Chinese Academy of Sciences, Beijing 100190, China

(Received 27 September 2017; revised manuscript received 23 April 2018; published 17 July 2018)

Generative modeling, which learns joint probability distribution from data and generates samples according to it, is an important task in machine learning and artificial intelligence. Inspired by probabilistic interpretation of quantum physics, we propose a generative model using matrix product states, which is a tensor network originally proposed for describing (particularly one-dimensional) entangled quantum states. Our model enjoys efficient learning analogous to the density matrix renormalization group method, which allows dynamically adjusting dimensions of the tensors and offers an efficient direct sampling approach for generative tasks. We apply our method to generative modeling of several standard data sets including the Bars and Stripes random binary patterns and the MNIST handwritten digits to illustrate the abilities, features, and drawbacks of our model over popular generative models such as the Hopfield model, Boltzmann machines, and generative adversarial networks. Our work sheds light on many interesting directions of future exploration in the development of quantum-inspired algorithms for unsupervised machine learning, which are promisingly possible to realize on quantum devices.

DOI: 10.1103/PhysRevX.8.031012

Subject Areas: Computational Physics,
Condensed Matter Physics,
Quantum Information

I. INTRODUCTION

Generative modeling, a typical example of unsupervised learning that makes use of a huge amount of unlabeled data, lies at the heart of the rapid development of modern machine learning techniques [1]. Different from discriminative tasks such as pattern recognition, the goal of generative modeling is to model the probability distribution of data and thus be able to generate *new* samples according to the distribution. At the research frontier of generative modeling, it is used for finding good data representation and dealing with tasks with missing data. Popular generative machine learning models include Boltzmann machines (BM) [2,3] and their generalizations [4], variational autoencoders (VAE) [5], autoregressive models [6,7], normalizing flows [8–10], and generative adversarial networks (GAN) [11]. For generative model design, one tries to balance the representational power and efficiency of learning and sampling.

There is a long history of the interplay between generative modeling and statistical physics. Some celebrated models, such as the Hopfield model [12] and Boltzmann machine [2,3], are closely related to the Ising model and its inverse version, which learns couplings in the model based on given training configurations [13,14].

The task of generative modeling also shares similarities with quantum physics in the sense that both of them try to model probability distributions in an immense space. Precisely speaking, it is the wave functions that are modeled in quantum physics, and probability distributions are given by their squared norm according to Born's statistical interpretation. Modeling probability distributions in this way is fundamentally different from the traditional statistical physics perspective. Hence, we may refer to probability models that exploit quantum state representations as "Born machines." Various *Ansätze* have been developed to express quantum states, such as the variational Monte Carlo [15], the tensor network (TN) states, and recently artificial neural networks [16]. In fact, physical systems like quantum circuits are also promising candidates for implementing Born machines.

In the past decades, tensor network states and algorithms have been shown to be an incredibly potent tool set for

[*] wanglei@iphy.ac.cn
[†] panzhang@itp.ac.cn

modeling many-body quantum states [17,18]. The success of TN description can be theoretically justified from a quantum information perspective [19,20]. In parallel to quantum physics applications, tensor decomposition and tensor networks have also been applied in a broader context by the machine learning community for feature extraction, dimensionality reduction, and analysis of the expressibility of deep neural networks [21–26].

In particular, the matrix product state (MPS) is a kind of TN where the tensors are arranged in a one-dimensional geometry [27]. The same representation is referred to as tensor train decomposition in the applied math community [28]. Despite its simple structure, the MPS can represent a large number of quantum states extremely well. MPS representation of ground states has been proven to be efficient for one-dimensional gapped local Hamiltonians [29]. In practice, optimization schemes for MPS such as the density-matrix renormalization group (DMRG) [30] have been successful even for some quantum systems in higher dimensions [31]. Some recent works extended the application of MPS to machine learning tasks like pattern recognition [32], classification [33], and language modeling [34]. Efforts also drew a connection between Boltzmann machines and tensor networks [35].

In this paper, building on the connection between unsupervised generative modeling and quantum physics, we employ MPS as a model to learn the probability distribution of given data with an algorithm that resembles DMRG [30]. Compared with statistical-physics-based models such as the Hopfield model [12] and the inverse Ising model, MPS exhibits a much stronger learning ability, which adaptively grows by increasing the bond dimensions of the MPS. The MPS model also enjoys a direct sampling method [36] much more efficient than that of the Boltzmann machines, which require a Markov chain Monte Carlo (MCMC) process for data generation. When compared with popular generative models such as GAN, our model offers a more efficient way to reconstruct and denoise from an initial (noisy) input using the direct sampling algorithm, as opposed to GAN, where mapping a noisy image to its input is not straightforward.

The rest of the paper is organized as follows. In Sec. II, we present our model, training algorithm, and direct sampling method. In Sec. III, we apply our model to three data sets: Bars and Stripes for a proof-of-principle demonstration, random binary patterns for capacity illustration, and the MNIST handwritten digits for showing the generalization ability of the MPS model in unsupervised tasks such as reconstruction of images. Finally, Sec. IV discusses future prospects of the generative modeling using more general tensor networks and quantum circuits.

II. MPS FOR UNSUPERVISED LEARNING

The goal of unsupervised generative modeling is to model the joint probability distribution of given data. With the trained model, one can then generate new samples from the learned probability distribution. Generative modeling finds wide applications such as dimensional reduction, feature detection, clustering, and recommendation systems [37]. In this paper, we consider a data set $\mathcal{T}$ consisting of binary strings $\boldsymbol{v} \in \mathcal{V} = \{0,1\}^{\otimes N}$, which are potentially repeated and can be mapped to basis vectors of a Hilbert space of dimension 2^N.

The probabilistic interpretation of quantum mechanics [38] naturally suggests modeling data distribution with a quantum state. Suppose we encode the probability distribution into a quantum wave function $\Psi(\boldsymbol{v})$; measurement will collapse it and generate a result $\boldsymbol{v} = (v_1, v_2, \dots, v_N)$, with a probability proportional to $|\Psi(\boldsymbol{v})|^2$. Inspired by the generative aspects of quantum mechanics, we represent the model probability distribution by

$$\mathbb{P}(\boldsymbol{v}) = \frac{|\Psi(\boldsymbol{v})|^2}{Z}, \tag{1}$$

where $Z = \sum_{\boldsymbol{v}\in\mathcal{V}} |\Psi(\boldsymbol{v})|^2$ is the normalization factor. We also refer to it as the "partition function" to draw an analogy with the energy-based models [39]. In general, the wave function $\Psi(\boldsymbol{v})$ can be complex valued, but in this work, we restrict it to be real valued. Representing probability density using the square of a function was also put forward by previous works [32,40,41]. These approaches ensure the positivity of probability and naturally admit a quantum mechanical interpretation.

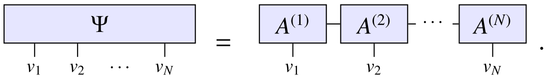
A. Matrix product states

Quantum physicists and chemists have developed many efficient classical representations of quantum wave functions. A number of these developed representations and algorithms can be adopted for efficient probabilistic modeling. Here, we parametrize the wave function using MPS:

$$\Psi(v_1, v_2, \dots, v_N) = \mathrm{Tr}(A^{(1)v_1} A^{(2)v_2} \cdots A^{(N)v_N}), \tag{2}$$

where each $A^{(k)v_k}$ is a $\mathcal{D}_{k-1}$ by $\mathcal{D}_k$ matrix, and $\mathcal{D}_0 = \mathcal{D}_N$ is demanded to close the trace. For the case considered here, there are $2\sum_{k=1}^{N} \mathcal{D}_{k-1}\mathcal{D}_k$ parameters on the right-hand side of Eq. (2). The representational power of MPS is related to von Neumann entanglement entropy of the quantum state, which is defined as $S = -\mathrm{Tr}(\rho_A \ln \rho_A)$. Here, we divide the variables into two groups $\boldsymbol{v} = (\boldsymbol{v}_A, \boldsymbol{v}_B)$, and $\rho_A = \sum_{\boldsymbol{v}_B} \Psi(\boldsymbol{v}_A, \boldsymbol{v}_B)\Psi(\boldsymbol{v}'_A, \boldsymbol{v}_B)$ is the reduced density matrix of a subsystem. The entanglement entropy sets a lower bound for the bond dimension at the division $S \le \ln(\mathcal{D}_k)$. Any probability distribution of an N-bit system can be described by a MPS, as long as its bond dimensions are free from any restriction. The inductive bias using MPS with limited bond dimensions comes from dropping off the minor components of the entanglement spectrum. Therefore, as the bond dimension increases, a MPS enhances its ability to parametrize complicated functions. See Refs. [17,18] for recent reviews on MPS and its applications on quantum many-body systems.

In practice, it is convenient to use MPS with $\mathcal{D}_0 = \mathcal{D}_N = 1$ and, consequently, reduce the leftmost and rightmost matrices to vectors [30]. In this case, Eq. (2) reads schematically



(3)

Here, the blocks denote the tensors and the connected lines indicate tensor contraction over virtual indices. The dangling vertical bonds denote physical indices. We refer to Refs. [17,18] for an introduction to these graphical notations of TN. Henceforth, we shall present formulas with more intuitive graphical notations wherever possible.

The MPS representation has gauge degrees of freedom, which allows one to restrict the tensors with canonical conditions. We remark that, in our setting of generative modeling, the canonical form significantly benefits from computing the *exact* partition function Z. More details about the canonical condition and the calculation of Z can be found in Appendix A.

B. Learning MPS from data

Once the MPS form of wave function $\Psi(\boldsymbol{v})$ is chosen, learning can be achieved by adjusting parameters of the wave function such that the distribution represented by Born's rule Eq. (1) is as close as possible to the data distribution. A standard learning method is called "maximum likelihood estimation," which defines a (negative) log-likelihood function and optimizes it by adjusting the parameters of the model. In our case, the negative log-likelihood (NLL) is defined as

$$\mathcal{L} = -\frac{1}{|\mathcal{T}|}\sum_{\boldsymbol{v}\in\mathcal{T}} \ln \mathbb{P}(\boldsymbol{v}), \tag{4}$$

where $|\mathcal{T}|$ denotes the size of the training set. Minimizing the NLL reduces the dissimilarity between the model probability distribution $\mathbb{P}(\boldsymbol{v})$ and the empirical distribution defined by the training set. It is well known that minimizing $\mathcal{L}$ is equivalent to minimizing the Kullback-Leibler divergence between the two distributions [42].

Armed with canonical form, we are able to differentiate the negative log-likelihood (4) with respect to the components of an order-4 tensor $A^{(k,k+1)}$, which is obtained by contracting two adjacent tensors $A^{(k)}$ and $A^{(k+1)}$. The gradient reads

$$\frac{\partial \mathcal{L}}{\partial A^{(k,k+1)w_k w_{k+1}}_{i_{k-1}i_{k+1}}} = \frac{Z'}{Z} - \frac{2}{|\mathcal{T}|}\sum_{\boldsymbol{v}\in\mathcal{T}} \frac{\Psi'(\boldsymbol{v})}{\Psi(\boldsymbol{v})}, \tag{5}$$

where $\Psi'(\boldsymbol{v})$ denotes the derivative of the MPS with respect to the tensor element of $A^{(k,k+1)}$, and $Z' = 2\sum_{\boldsymbol{v}\in\mathcal{V}} \Psi'(\boldsymbol{v})\Psi(\boldsymbol{v})$. Note that although Z and Z' involve summations over an exponentially large number of terms, they are tractable in the MPS model via efficient contraction schemes [17]. In particular, if the MPS is in the mixed-canonical form [17], Z' can be significantly simplified to $Z' = 2A^{(k,k+1)w_k w_{k+1}}_{i_{k-1}i_{k+1}}$. The calculation of the gradient, as well as variant techniques in gradient descent such as the stochastic gradient descent (SGD) and adaptive learning rate, are detailed in Appendix B. After gradient descent, the merged order-4 tensor is decomposed into two order-3 tensors, and then the procedure is repeated for each pair of adjacent tensors.

The derived algorithm is quite similar to the celebrated DMRG method with a two-site update, which allows us to adjust dynamically the bond dimensions during the optimization and to allocate computational resources to the important bonds that represent essential features of data. However, we emphasize that there are key differences between our algorithm and DMRG:

(i) The loss function of the classic DMRG method is usually the energy, while our loss function, the averaged NLL (4), is a function of data.

(ii) With a huge amount of data, the landscape of the loss function is typically very complicated, so that modern optimizers developed in the machine learning community, such as the stochastic gradient descent and learning rate adapting techniques [43], are important to our algorithm. Since the ultimate goal of learning is optimizing the performance on the test data, we do not really need to find the optimal parameters minimizing the loss on the training data. One usually stops training before reaching the actual minima to prevent overfitting.

(iii) Our algorithm is data oriented. It is straightforward to parallelize over the samples since the operations applied to them are identical and independent. In fact, it is a common practice in the modern deep learning framework to parallelize over this so-called "batch" dimension [37]. As a concrete example, the GPU implementation of our algorithm is at least 100 times faster than the CPU implementation on the full MNIST data set.

C. Generative sampling

After training, samples can be generated independently according to Eq. (1). In other popular generative models, especially an energy-based model such as a restricted Boltzmann machine (RBM) [3], generating new samples is often accomplished by running MCMC from an initial configuration, due to the intractability of the partition function. In our model, one convenience is that the partition function can be exactly computed with complexity linear in system size. Our model enjoys a direct sampling method, which generates a sample bit by bit from one end of the MPS to the other [36]. The detailed generating process is as follows.

It starts from one end, say, the Nth bit. One directly samples this bit from the marginal probability $\mathbb{P}(v_N) = \sum_{v_1,v_2,\ldots,v_{N-1}} \mathbb{P}(\boldsymbol{v})$. It is clear that this can be easily performed if we have gauged all the tensors except $A^{(N)}$ to be left canonical because $\mathbb{P}(v_N) = |\boldsymbol{x}^{v_N}|^2/Z$, where we define $x^{v_N}_{i_{N-1}} = A^{(N)v_N}_{i_{N-1}}$, and the normalization factor reads $Z = \sum_{v_N \in \{0,1\}} |\boldsymbol{x}^{v_N}|^2$. Given the value of the Nth bit, one can then move on to sample the $(N-1)$th bit. More generally, given the bit values v_k, $v_{k+1}, \ldots, v_N$, the $(k-1)$th bit is sampled according to the conditional probability

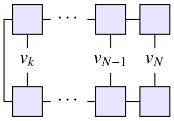
$$\mathbb{P}(v_{k-1}|v_k, v_{k+1}, \ldots, v_N) = \frac{\mathbb{P}(v_{k-1}, v_k, \ldots, v_N)}{\mathbb{P}(v_k, v_{k+1} \ldots, v_N)}. \tag{6}$$

As a result of the canonical condition, the marginal probability can be simply expressed as

$$\mathbb{P}(v_k, v_{k+1}, \ldots, v_N) = |\boldsymbol{x}^{v_k,v_{k+1},\ldots,v_N}|^2/Z. \tag{7}$$

$x^{v_k,v_{k+1},\ldots,v_N}_{i_{k-1}} = \sum_{i_k,i_{k+1},\ldots,i_{N-1}} A^{(k)v_k}_{i_{k-1}i_k} A^{(k+1)v_{k+1}}_{i_k i_{k+1}} \cdots A^{(N)v_N}_{i_{N-1}}$ has been settled since the kth bit is sampled. Schematically, its squared norm reads

$$|\boldsymbol{x}^{v_k,v_{k+1},\ldots,v_N}|^2 =$$



$$. \qquad (8)$$

Multiplying the matrix $A^{(k-1)v_{k-1}}$ from the left, and calculating the squared norm of the resulting vector $x^{v_{k-1},v_k,\ldots,v_N}_{i_{k-2}} = \sum_{i_{k-1}} A^{(k-1)v_{k-1}}_{i_{k-2}i_{k-1}} x^{v_k,v_{k-1},\ldots,v_N}_{i_{k-1}}$, one obtains

$$\mathbb{P}(v_{k-1}, v_k, \ldots, v_N) = |\boldsymbol{x}^{v_{k-1},v_k,\ldots,v_N}|^2/Z. \tag{9}$$

Combining Eqs. (7) and (9), one can compute the conditional probability (6) and sample the bit v_{k-1} accordingly. In this way, all the bit values are successively drawn from the conditional probabilities given all the bits on the right. This procedure gives a sample strictly obeying the probability distribution of the MPS.

This sampling approach is not limited to generating samples from scratch in a sequential order. It is also capable of inference tasks when part of the bits are given. In that case, the canonicalization trick may not help greatly if there is a segment of unknown bits sitting between given bits. Nevertheless, the marginal probabilities are still tractable because one can also contract ladder-shaped TN efficiently [17,18]. As will be shown in Sec. III, given these flexibilities of the sampling approach, MPS-based probabilistic modeling can be applied to image reconstruction and denoising.

D. Features of the model and algorithms

We highlight several salient features of the MPS generative model and compare it to other popular generative models. Most significantly, MPS has an explicit tractable probability density, while still allowing efficient learning and inference. For a system sized N, with prescribed maximal bond dimension $\mathcal{D}_{\max}$, the complexity of training on a data set of size $|\mathcal{T}|$ is $\mathcal{O}(|\mathcal{T}|N\mathcal{D}^3_{\max})$. The scaling of generative sampling from a canonical MPS is $\mathcal{O}(N\mathcal{D}^2_{\max})$ if all the bits to be sampled are connected to the boundaries; otherwise, given some segments, the conditional sampling scales as $\mathcal{O}(N\mathcal{D}^3_{\max})$.

1. Theoretical understanding of the expressive power

The expressibility of MPS was intensively studied in the context of quantum physics. The bond dimensions of MPS put an upper bound on its ability to capture entanglement entropy. These solid theoretical understandings of the representational power of MPS [17,18] make it an appealing model for generative tasks.

Considering the success of MPS for quantum systems, we expect a polynomial scaling of the computational resources for data sets with short-range correlations. Treating data sets of two-dimensional images using MPS is analogous to the application of DMRG to two-dimensional quantum systems [31]. Although, in principle, an exact representation of the image data set may require exponentially large bond dimensions as the image resolution increases, at computationally affordable bond dimensions, the MPS may already serve as a good approximation that captures dominant features of the distribution.

2. Adaptive adjustment of expressibility

Performing optimizations for the two-site tensor instead of for each tensor individually allows one to dynamically adjust the bond dimensions during the learning process. Since for realistic data sets the required bond dimensions are likely to be inhomogeneous, adjusting them dynamically allocates computational resources in an optimal manner. This situation will be illustrated clearly using the MNIST data set in Sec. III C and in Fig. 4.

Adjustment of the bond dimensions follows the distribution of singular values in Eq. (B4), which is related to the low entanglement inductive bias of the MPS representation. Adaptive adjustment of MPS is advantageous compared to most other generative models. Because in most cases, the architecture (which is the main limiting factor of the expressibility of the model) is fixed during the learning procedure, only the parameters are tuned. By adaptively tuning the bond dimensions, the representational power of MPS can grow as it gets more acquainted with the training data. In this sense, adaptive adjustment of expressibility is analogous to the structural learning of probabilistic graphical models, which is, however, a challenging task due to the discreteness of the structural information.

3. Efficient computation of exact gradients and log-likelihood

Another advantage of MPS compared to the standard energy-based model is that training can be done with high efficiency. The two terms contributing to the gradient in Eq. (5) are analogous to the negative and positive phases in the training of energy-based models [39], where the visible variables are unclamped and clamped, respectively. In the energy-based models, such as RBM, a typical evaluation of the first term requires approximated MCMC sampling [44] or sophisticated mean-field approximations, e.g., Thouless-Anderson-Palmer equations [45]. Fortunately, the normalization factor and its gradient can be calculated exactly and straightforwardly for MPS. The exact evaluation of gradients guarantees the associated stochastic gradient descent unbiased.

In addition to efficiency in computing gradients, the unbiased estimate of the log-likelihood and its gradients benefits significantly when compared with classic generative models such as RBM, where the gradients are approximated due to the intractability of the partition function. First, with MPS we can optimize the NLL directly, while with RBM, the approximate algorithms such as contrastive divergence (CD) are essentially optimizing a loss function other than NLL. This results in the fact that some region of configuration space could never be considered during training RBM and a subsequently poor performance on, e.g., denoising and reconstruction. Second, with MPS, we can monitor the training process easily using exact NLL instead of other quantities such as reconstruction error or pseudolikelihood for RBM, which introduce bias to monitoring [67].

4. Efficient direct sampling

The approach introduced in Sec. II C allows direct sampling from the learned probability distribution. This completely avoids the slowing mixing problem in the MCMC sampling of energy-based models. MCMC randomly flips the bits and compares the probability ratios for accepting and rejecting the samples. However, the random walks in the state space can get stuck in a local minimum, which may bring unexpected fluctuations of long-time correlation to the samples. Sometimes this raises issues with the samplings. As a concrete example, consider the case where all training samples are exactly memorized by both MPS and RBM. This is to say that NLL of both models are exactly $\ln|\mathcal{T}|$, and only training samples have finite probability in both models. Meanwhile, other samples, even with only one bit different, have zero probability. It is easy to check that our MPS model can generate samples, which is identical to one of the training samples using the approach introduced in Sec. II C. However, RBM will not work at all in generating samples, as there is no direction that MCMC could follow for increasing the probability of samplings.

It is known that when graphical models have an appropriate structure (such as a chain or a tree), the inference can be done efficiently [46,47], while these structural constraints also limit the application of graphical models with intractable partition functions. The MPS model, however, enjoys both the advantages of efficient direct sampling and a tractable partition function. The sampling algorithm is formally similar to the ones of autoregressive models [6,7]; however, being able to dynamically adjust its expressibility makes the MPS a more flexible generative model.

Unlike GAN [11] or VAE [5], the MPS can explicitly give tractable probability, which may enable more unsupervised learning tasks. Moreover, the sampling in MPS works with arbitrary prior information of samples, such as fixed bits, which supports applications like image reconstruction and denoising. We note that this offers an advantage over the popular GAN, which easily maps a random vector in the latent space to the image space, but having difficulties in the reverse direction—mapping a vector in the images space to the latent space as prior information to sampling.

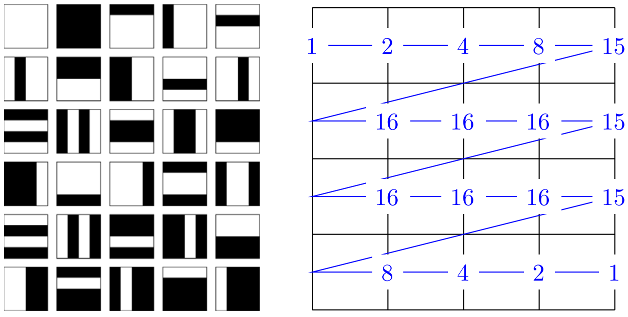
III. APPLICATIONS

In this section, to demonstrate the ability and features of the MPS generative modeling, we apply it to several standard data sets. As a proof of principle, we first apply our method to the toy data set of Bars and Stripes, where some properties of our model can be characterized analytically. Then, we train MPS as an associative memory to learn random binary patterns to study properties such as capacity and length dependences. Finally, we test our model on the Modified National Institute of Standards and Technology (MNIST) database to illustrate its generalization ability for generating and reconstructing images of handwritten digits.

A. Bars and Stripes

Bars and Stripes (BS) [48] is a data set containing 4×4 binary images. Each image has either four-pixel-length vertical bars or horizontal stripes, but not both. In total there are 30 different images in the data set out of all 2^{16} possible ones, as shown in Fig. 1(a). These images appear with equal probability in the data set. This toy problem allows a detailed analysis and reveals key characteristics of the MPS probabilistic model.

To use MPS for modeling, we unfold the 4×4 images into one-dimensional vectors as shown in Fig. 1(b). After being trained over four loops of batch gradient descent training, the cost function converges to its minimum value, which is equal to the Shannon entropy of the BS data set



(a) (b)

FIG. 1. (a) The Bars and Stripes data set. (b) Ordering of the pixels when transforming the image into a one-dimensional vector. The numbers between pixels indicate the bond dimensions of the well-trained MPS.

$\mathcal{S} = \ln(30)$, within an accuracy of 1×10^{-10}. Here, what the MPS has accomplished is memorizing the 30 images rigidly, by increasing the probability of the instances that appear in the data set, and suppressing the probability of not-shown instances towards zero. We have checked that the result is insensitive to the choice of hyperparameters.

The bond dimensions of the learned MPS have been annotated in Fig. 1(b). It is clear that part of the symmetry of the data set has been preserved. For instance, the 180° rotation around the center or the transposition of the second and the third rows would change neither the data set nor the bond dimension distribution. The open boundary condition results in the decrease of bond dimensions at both ends. In fact, when conducting SVD at bond k, there are at most $2^{\min(k,N-k)}$ nonzero singular values because the two parts linked by bond k have their Hilbert spaces of dimension 2^k, 2^{N-k}. In addition, the turnings bonds have slightly smaller bond dimension ($\mathcal{D}_4 = \mathcal{D}_8 = \mathcal{D}_{12} = 15$) than others inside the second row and the third row, which can be explained qualitatively as these bonds carrying less entanglement than the bonds in the bulk.

One can directly write down the exact "quantum wave function" of the BS data set, which has finite and uniform amplitudes for the training images and zero amplitude for other images. For division on each bond, one can construct the reduced density matrix whose eigenvalues are the square of the singular values. Analyzed in this way, it is confirmed that the trained MPS achieves the minimal number of required bond dimension to exactly describe the BS data set.

We have generated $N_s = 10^6$ independent samples from the learned MPS. All these samples are training images shown in Fig. 1(a). Carrying out the likelihood ratio test [49], we got the log-likelihood ratio statistic $G^2 = 2N_s D_{\mathrm{KL}}(\{n_j/N_s\}||\{p_j\}) = 22.0$, equivalently $D_{\mathrm{KL}}(\{n_j/N_s\}||\{p_j\}) = 1.10 \times 10^{-5}$. The reason for adopting this statistic is that it is asymptotically χ^2-distributed [49]. The p-value of this test is 0.820, which indicates a high probability that the uniform distribution holds true for the sampling outcomes.

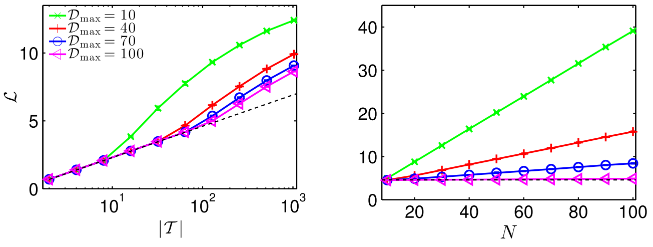
Note that $D_{\mathrm{KL}}(\{n_j/N_s\}||\{p_j\})$ quantifies the deviation from the expected distribution to the sampling outcomes, so it reflects the performance of the sampling method rather than merely the training performance. In contrast to our model, for energy-based models, one typically has to resort to the MCMC method for sampling new patterns. It suffers from the slow mixing problem, since various patterns in the BS data set differ substantially, and it requires many MCMC steps to obtain one independent pattern.

B. Random patterns

Capacity represents how much about data could be learned by the model. Usually, it is evaluated using randomly generated patterns as data. For the classic Hopfield model [12] with pairwise interactions given by Hebb's rule among $N \to \infty$ variables, it has been shown [50] that, in the low-temperature region at the thermodynamic limit, there is the retrieval phase, where, at most, $|\mathcal{T}|_c = 0.14N$ random binary patterns could be remembered. In this sense, each sample generated by the model has a large overlap with one of the training patterns. If the number of patterns in the Hopfield model is larger than $|\mathcal{T}|_c$, the model would enter the spin glass state, where samples generated by the model are not correlated with any training pattern.

Thanks to the tractable evaluation of the partition function Z in MPS, we are able to evaluate exactly the likelihood of every training pattern. Thus, the capability of the model can be easily characterized by the mean negative log-likelihood $\mathcal{L}$. In this section, we focus on the behavior of $\mathcal{L}$ with varying numbers of training samples and varying system sizes.

In Fig. 2(a), we plot $\mathcal{L}$ as a function of the number of patterns used for training for several maximal bond dimensions $\mathcal{D}_{\max}$. The figure shows that we obtain $\mathcal{L} = \ln|\mathcal{T}|$ for a



(a) (b)

FIG. 2. NLL averaged as a function of (a) number of random patterns used for training, with system size $N = 20$. (b) System size N, trained using $|\mathcal{T}| = 100$ random patterns. In both (a) and (b), different symbols correspond to different values of maximal bond dimension $\mathcal{D}_{\max}$. Each data point is averaged over 10 random instances (i.e., sets of random patterns); error bars are also plotted, although they are much smaller than symbol size. The black dashed lines in figures denote $\mathcal{L} = \ln|\mathcal{T}|$.

training set no larger than $\mathcal{D}_{\max}$. As shown in the previous section, this means that all training patterns are remembered exactly. As the number of training patterns increases, MPS with a fixed $\mathcal{D}_{\max}$ will eventually fail in remembering exactly all the training patterns, resulting in $\mathcal{L} > \ln|\mathcal{T}|$. In this regime, generations of the model usually deviate from training patterns (as illustrated in Fig. 3 on the MNIST data set). We notice that, with $|\mathcal{T}|$ increasing, the curves in the figure deviate from $\ln|\mathcal{T}|$ continuously. We note that this is very different from the Hopfield model, where the overlap between the generation and training samples changes abruptly due to the first order transition from the retrieval phase to the spin glass phase.

Figure 2(a) also shows that a larger $\mathcal{D}_{\max}$ enables MPS to remember exactly more patterns and produce smaller $\mathcal{L}$ with the number of patterns $|\mathcal{T}|$ fixed. This is quite natural because enlarging $\mathcal{D}_{\max}$ amounts to the increase of the parameter number of the model and, hence, enhances the capacity of the model. In principle, if $\mathcal{D}_{\max} = \infty$, our model has infinite capacity, since arbitrary quantum states can be decomposed into MPS [17]. Clearly, this is an advantage of our model over the Hopfield model and inverse Ising model [14], whose maximal model capacity is proportional to system size.

Careful readers may complain that the inverse Ising model is not the correct model to compare with, because its variation with hidden variables, i.e., Boltzmann machines, do have infinite representation power. Indeed, increasing the bond dimensions in MPS has similar effects to increasing the number of hidden variables in other generative models.

In Fig. 2(b), we plot $\mathcal{L}$ as a function of system size N, trained on $|\mathcal{T}| = 100$ random patterns. As shown in the figure, with $\mathcal{D}_{\max}$ fixed, $\mathcal{L}$ increases linearly with system size N, which indicates that our model gives a worse memory capability with a larger system size. This is due to the fact that keeping the joint distribution of variables becomes more and more difficult for MPS when the number of variables increases, especially for long-range correlated data. This is a drawback of our model when compared with fully pairwise-connected models such as the inverse Ising model, which is able to capture long-distance correlations of the training data easily. Fortunately, Fig. 2(b) also shows that the decay of memory capability with system size can be compensated by increasing $\mathcal{D}_{\max}$.

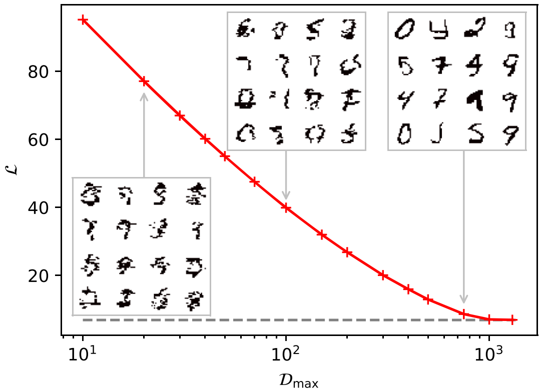


FIG. 3. NLL average of a MPS trained using $|\mathcal{T}| = 1000$ MNIST images of size 28×28, with varying maximum bond dimensions $\mathcal{D}_{\max}$. The horizontal dashed line indicates the Shannon entropy of the training set $\ln|\mathcal{T}|$, which is also the minimal value of $\mathcal{L}$. The inset images are generated by the MPS trained with different $\mathcal{D}_{\max}$ (denoted by the arrows).

C. MNIST data set of handwritten digits

In this subsection, we perform experiments on the MNIST data set [51]. In preparation, we turn the grayscale images into binary numbers by threshold binarization and flattened the images row by row into a vector. For the purpose of unsupervised generative modeling, we do not need the labels of the digits. Here, we further test the capacity of the MPS for this larger-scale and more meaningful data set. Then, we investigate its generalization ability via examining its performance on a separated test set, which is crucial for generative modeling.

1. Model capacity

Having chosen $|\mathcal{T}| = 1000$ MNIST images, we train the MPS with different maximal bond dimensions $\mathcal{D}_{\max}$, as shown in Fig. 3. As $\mathcal{D}_{\max}$ increases, the final $\mathcal{L}$ decreases to its minimum $\ln|\mathcal{T}|$, and the images generated become more and more clear. It is interesting that, with a relatively small maximum bond dimension, e.g., $\mathcal{D}_{\max} = 100$, some crucial features show up, though some of the images were not as clear as the original ones. For instance, the hooks and loops that partly resemble the numerals “2,” “3,” and “9” emerge. These clear characters of handwritten digits illustrate that the MPS has learned many “prototypes.” Similar feature-to-prototype transitions in pattern recognitions could also be observed by using a many-body interaction in the Hopfield model, or equivalently, using a higher-order rectified polynomial activation function in the deep neural networks [52]. It is remarkable that, in our model, this can be achieved by simply adjusting the maximum bond dimension of the MPS.

Next, we train another model with the restriction of $\mathcal{D}_{\max} = 800$. The NLL on the training data set reaches 16.8, and many bonds have reached maximal dimension $\mathcal{D}_{\max}$. Figure 4 shows the distribution of bond dimensions. Large bond dimensions are concentrated in the center of the image, where the variation of the pixels is complex. The bond dimensions around the top and bottom edge of the image remain small, because those pixels are always inactivated in the images. They carry no information and have no correlations with the remaining part of the image. Remarkably, although the pixels on the left and right edges are also white, they also have large bond dimensions

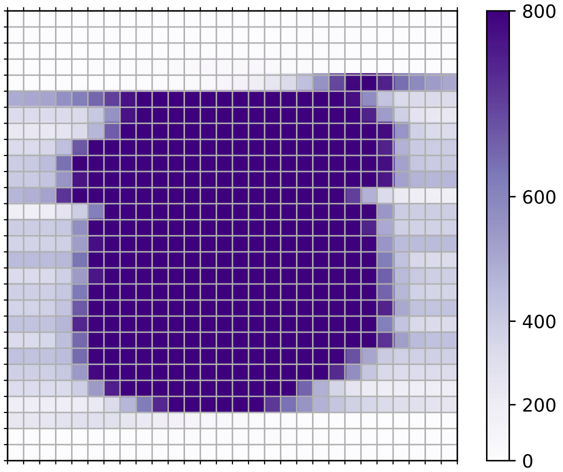
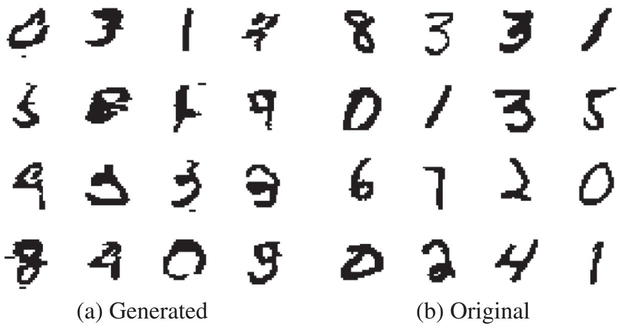


FIG. 4. Bond dimensions of the MPS trained with $|\mathcal{T}| = 1000$ MNIST samples, constrained to $\mathcal{D}_{\max} = 800$. Final average NLL reaches 16.8. Each pixel in this figure corresponds to the bond dimension of the right leg of the tensor associated to the identical coordinate in the original image.

because these bonds learn to mediate the correlations between the rows of the images.

The samples directly generated after training are shown in Fig. 5(a). We also show a few original samples from the training set in Fig. 5(b) for comparison. Although many of the generated images cannot be recognized as digits, some aspects of the result are worth mentioning. Firstly, the MPS learned to leave margins blank, which is the most obvious common feature in the MNIST database. Secondly, the activated pixels compose pen strokes that can be extracted from the digits. Finally, a few of the samples could already be recognized as digits. Unlike the discriminative learning task carried out in Ref. [32], it seems we need to use much larger bond dimensions to achieve a good performance in the unsupervised task. We postulate the reason to be that, in the classification task, local features of an image are sufficient for predicting the label. Thus, MPS is not required to remember longer-range correlation between pixels. For generative modeling, however, it is necessary because learning the joint distribution from the data consists of (but not limited to) learning two-point correlations between pairs of variables that could be far from each other.

With the MPS restricted to $\mathcal{D}_{\max} = 800$ and trained with 1000, we carry out image restoration experiments. As shown in Fig. 6, we remove part of the images in Fig. 5(b) and then reconstruct the removed pixels (in yellow) using conditional direct sampling. For column reconstruction, its performance is remarkable. The reconstructed images in Fig. 6(a) are almost identical to the original ones in Fig. 5(b). On the other hand, for row reconstruction in Fig. 6(b), it makes interesting but reasonable deviations. For instance, for the rightmost image in the first row, the “1” shape has been bent to a “7.”

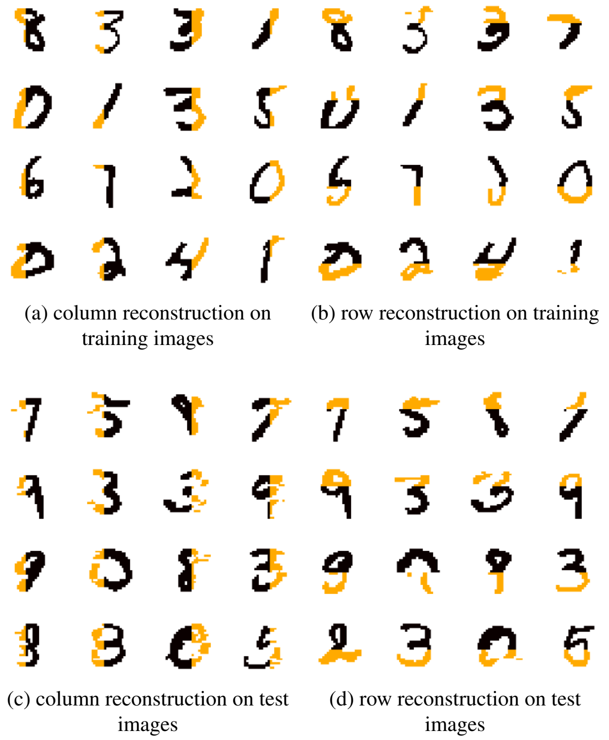


(a) Generated (b) Original

FIG. 5. (a) Images generated from the same MPS as in Fig. 4. (b) Original images randomly selected from the training set.

2. Generalization ability

In a glimpse of its generalization ability, we also tried reconstructing MNIST images other than the training images, as shown in Figs. 6(c) and 6(d). These results indicate that the MPS has learned crucial features of the data set, rather than merely memorizing the training instances. In fact, even as early as only 11 loops trained, the MPS could perform column reconstruction with similar



(a) column reconstruction on training images
(b) row reconstruction on training images
(c) column reconstruction on test images
(d) row reconstruction on test images

FIG. 6. Image reconstruction from partial images by direct sampling with the same MPS as in Fig. 4. (a,b) Restoration of images in Fig. 5(b), which are selected from the training set. (c,d) Reconstruction of 16 images chosen from the test set. The test set contains images from the MNIST database that were not used for training. The given parts are in black (dark) and the reconstructed parts are in yellow (light). The reconstructed parts are 12 columns from either (a,c) the left or the right and (b,d) the top or the bottom.

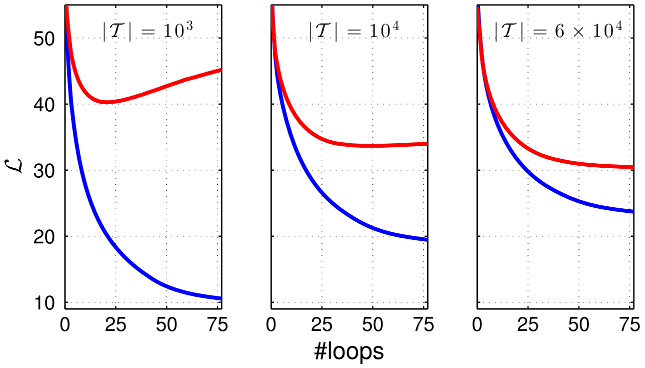


FIG. 7. Evolution of the average negative log-likelihood $\mathcal{L}$ for both training images (blue, bottom lines) and 10^4 test images (red, top lines) during training. From left to right, the numbers of images in the training set $|\mathcal{T}|$ are 10^3, 10^4, and 6×10^4, respectively.

image quality, but its row reconstruction performance was much worse than that trained over 251 loops. It is reflected that the MPS has learned about short-range patterns within each row earlier than those with long-range correlations between different rows, since the images have been flattened into a one-dimensional vector row by row.

To further illustrate our model's generalization ability, in Fig. 7, we plotted $\mathcal{L}$ for the same 10^4 test images after training on different numbers of images. To save computing time, we worked on rescaled images of size 14×14. The rescaling has also been adopted by past works, and it is shown that the classification on the rescaled images is still comparable with those obtained using other popular methods [32].

For different $|\mathcal{T}|$, $\mathcal{L}$ for training images always decreases monotonically to different minima, and with a fixed $\mathcal{D}_{\max}$, it is easier for the MPS to fit fewer training images. The $\mathcal{L}$ for test images, however, behaves quite differently: For $|\mathcal{T}| = 10^3$, test $\mathcal{L}$ decreases to about 40.26 and then starts climbing quickly, while for $|\mathcal{T}| = 10^4$, the test $\mathcal{L}$ decreases to 33.65 and then increases slowly to 34.18. For $|\mathcal{T}| = 6 \times 10^4$, test $\mathcal{L}$ kept decreasing in 75 loops. The behavior shown in Fig. 7 is quite typical in machine learning problems. When training data are not enough, the model quickly overfits the training data, giving worse and worse generalization to the unseen test data. An extreme example is when our model is able to decrease training $\mathcal{L}$ to $\ln |\mathcal{T}|$, i.e., completely overfits the training data, then all other images, even the images with only one pixel difference from one of the training images, have zero probability in the model, and hence $\mathcal{L} = \infty$. We also observe that the best test NLL decreases as the training set volume enlarges, which means the tendency of memorizing is constrained and that of generalization is enhanced.

The histograms of log-likelihoods for all training and test images are shown in Fig. 8. Notice that, if the model just memorized some of the images and ignored the others, the histograms would be bimodal. It is not the case, as shown in the figure, where all distributions are centered around. This indicates that the model learns all images well rather than concentrates on some images while completely ignoring the others. In the bottom panel, we show the detailed $\mathcal{L}$ histogram by categories. For some digits, such as "1" and "9," the difference between training and test log-likelihood distribution is insignificant, which suggests that the model has particularly great generalization ability to these images.

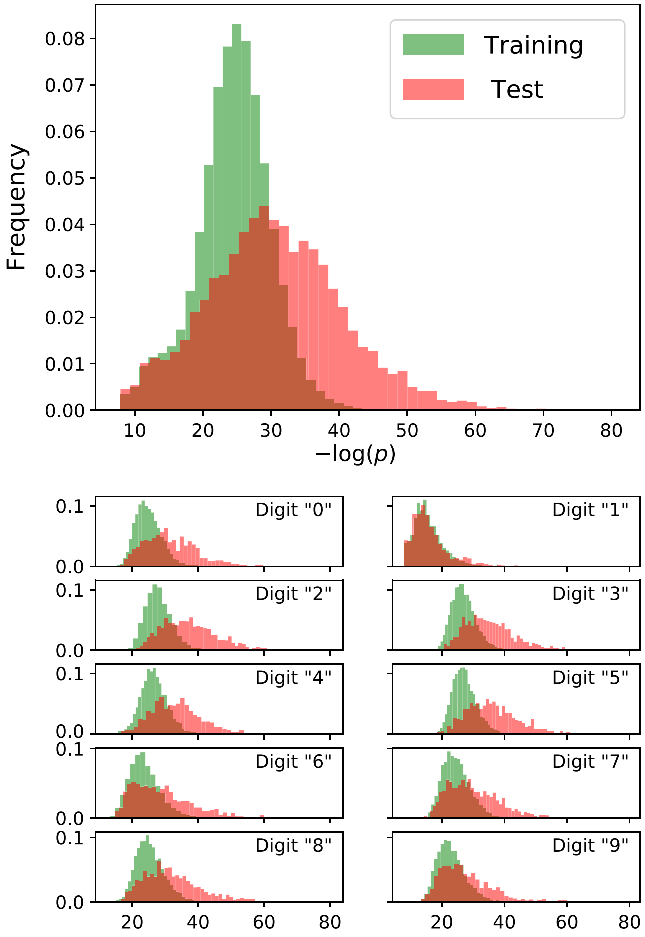


FIG. 8. Top: Distribution of $-\ln p$ of 60 000 training images and 10 000 test images given by a trained MPS with $\mathcal{D}_{\max} = 500$. The training negative log likelihood $\mathcal{L}_{\text{train}} = 24.2$, and the test $\mathcal{L}_{\text{test}} = 30.3$. Bottom: Distributions for each digit.

IV. SUMMARY AND OUTLOOK

We have presented a tensor-network-based unsupervised model, which aims at modeling the probability distribution of samples in given unlabeled data. The probabilistic model is structured as a matrix product state, which brings several advantages, as discussed in Sec. II D, such as adaptive and efficient learning and direct sampling.

Since we use the square of the TN states to represent probability, the sign is redundant for probabilistic modeling besides the gauge freedom of MPS. It is likely that, during the optimization, MPS develops different signs for different configurations. The sign variation may unnecessarily increase the entanglement in MPS and, therefore, the bond dimensions [53]. However, restricting the sign of MPS may also impair the expressibility of the model. One probable approach to obtain a low entanglement representation is adding a penalty term in the target function, for instance, a term proportional to Rényi entanglement entropy as in our further work on quantum tomography [54]. In light of these discussions, we would like to point to future research on the differences and connections of MPS with non-negative matrix entries [55] and the probabilistic graphical models such as the hidden Markov model.

Binary data modeling links closely to quantum many-body systems with spin-$1/2$ constituents and could be straightforwardly generalized for higher-dimensional data. One can also follow Refs. [32,56] to use a local feature map to lift continuous variables to a spinor space for continuous data modeling. The ability and efficiency of this approach may also depend on the specific way of performing the mapping, so in terms of continuous input, there is still a lot to be explored in this algorithm. Moreover, for colored images, one can encode the RGB values to three physical legs of each MPS tensor.

Similar to using MPS for studying two-dimensional quantum lattice problems [31], modeling images with MPS faces the problem of introducing long-range correlations for some neighboring pixels in two dimensions. An obvious generalization of the present approach is to use more expressive TN with more complex structures. In particular, the projected entangled pair states (PEPS) [57] is particularly suitable for images, because it takes care of correlation between pixels in two dimensions. Similar to the studies of quantum systems in 2D, however, this advantage of PEPS is partially compensated by the difficulty of contracting the network and the loss of convenient canonical forms. Exact contraction of a PEPS is #P hard [58]. Nevertheless, one can employ tensor renormalization group methods for approximated contraction of PEPS [59–62]. Thus, it remains to be seen whether judicious combination of these techniques really brings a better performance to generative modeling.

In the end, we would like to remark that perhaps the most exciting feature of quantum-inspired generative models is the possibility of being implemented by quantum devices [63], rather than merely being simulated in classical computers. In that way, neither the large bond dimension nor the high computational complexity of tensor contraction would be a problem. The tensor network representation of probability may facilitate quantum generative modeling because some of the tensor network states can be prepared efficiently on a quantum computer [64,65].

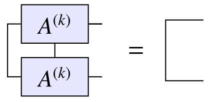
ACKNOWLEDGMENTS

We thank Liang-Zhu Mu, Hong-Ye Hu, Song Cheng, Jing Chen, Wei Li, Zhengzhi Sun, and E. Miles Stoudenmire for inspiring discussions. We also acknowledge suggestions from anonymous reviewers. P. Z. acknowledges the Swarm Club workshop on “Geometry, Complex Network and Machine Learning” sponsored by the Kai Feng Foundation in 2016. L. W. is supported by the Ministry of Science and Technology of China under Grant No. 2016YFA0300603 and the National Natural Science Foundation of China under Grant No. 11774398. J. W. is supported by the National Training Program of Innovation for Undergraduates of China. P. Z. is supported by Key Research Program of Frontier Sciences CAS Grant No. QYZDB-SSW-SYS032 and Project No. 11747601 of National Natural Science Foundation of China. Part of the computation was carried out at the High Performance Computational Cluster of ITP, CAS.

Z.-Y. H. and J. W. contribute equally to this work.

APPENDIX A: CANONICAL CONDITIONS FOR MPS AND COMPUTATION OF THE PARTITION FUNCTION

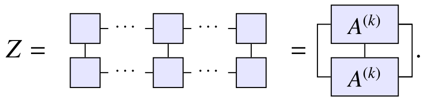
The MPS representation has gauge degrees of freedom, which means that the state is invariant after inserting identity $I = MM^{-1}$ on each bond (M can be different on each bond). Exploiting the gauge degrees of freedom, one can bring the MPS into its canonical form: for example, the tensor $A^{(k)}$ is called left canonical if it satisfies $\sum_{v_k \in \{0,1\}} (A^{(k)v_k})^{\dagger} A^{(k)v_k} = I$. In diagrammatic notation, the left-canonical condition reads



(A1)

The right-canonical condition is defined analogously. Canonicalization of each tensor can be done locally and only involves the single tensor at consideration [17,18].

Each tensor in the MPS can be in a different canonical form. For example, given a specific site k, one can conduct a gauge transformation to make all the tensors on the left, $\{A^{(i)} | i = 1, 2, \ldots, k-1\}$, left canonical and tensors on the right, $\{A^{(i)} | i = k+1, k+2, \ldots, N\}$, right canonical, while leaving $A^{(k)}$ neither left canonical nor right canonical. This is called the mixed-canonical form of the MPS [17]. The normalization of the MPS is particularly easy to compute in the canonical from. In the graphical notation, it reads



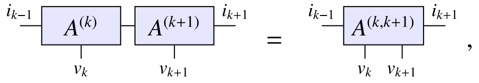
(A2)

We note that, even if the MPS is not in the canonical form, its normalization factor Z can still be computed efficiently if one pays attention to the order of contraction [17,18].

APPENDIX B: DMRG-LIKE GRADIENT DESCENT ALGORITHM FOR LEARNING

A standard way of minimization of the cost function in Eq. (4) is done by performing the gradient descent algorithm on the MPS tensor elements. Crucially, our method allows dynamical adjustment of the bond dimension during the optimization, thus being able to allocate resources to the spatial regions where correlations among the physical variables are stronger.

Initially, we set the MPS with random tensors with small bond dimensions. For example, all the bond dimensions are set to $\mathcal{D}_k = 2$ except those on the boundaries [66]. We then carry out the canonicalization procedure so that all the tensors except the rightmost one $A^{(N)}$ are left canonical. Then, we sweep through the matrices back and forth to tune the elements of the tensors, i.e., the parameters of the MPS. The procedure is similar to the DMRG algorithm with the two-site update, where one optimizes two adjacent tensors at a time [30]. At each step, we firstly merge two adjacent tensors into an order-4 tensor,

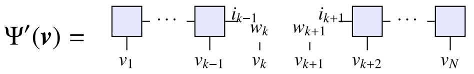


(B1)

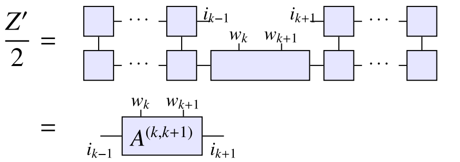
followed by adjusting its elements in order to decrease the cost function $\mathcal{L} = \ln Z - (1/|\mathcal{T}|)\sum_{\mathbf{v}\in\mathcal{T}} \ln |\Psi(\mathbf{v})|^2$. It is straightforward to check that its gradient with respect to an element of the tensor in Eq. (B1) reads

$$\frac{\partial \mathcal{L}}{\partial A^{(k,k+1)w_k w_{k+1}}_{i_{k-1} i_{k+1}}} = \frac{Z'}{Z} - \frac{2}{|\mathcal{T}|}\sum_{\mathbf{v}\in\mathcal{T}} \frac{\Psi'(\mathbf{v})}{\Psi(\mathbf{v})}, \tag{B2}$$

where $\Psi'(\mathbf{v})$ denotes the derivative of the MPS with respect to the tensor in Eq. (B1), and $Z' = 2\sum_{\mathbf{v}\in\mathcal{V}} \Psi'(\mathbf{v})\Psi(\mathbf{v})$. In diagram language, they read



(B3)



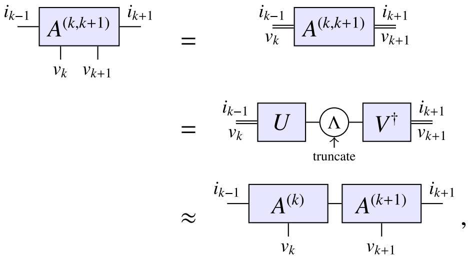
(B4)

The direct vertical connections of w_k, v_k and w_{k+1}, v_{k+1} in Eq. (B3) stand for Kronecker delta functions $\delta_{w_k v_k}$ and $\delta_{w_{k+1} v_{k+1}}$, respectively, meaning that only those input data with pattern $v_k v_{k+1}$ contribute to the gradient with respect to the tensor elements $A^{(k,k+1)v_k v_{k+1}}$. Note that, although Z and Z' involve summations over an exponentially large number of terms, they are tractable in MPS via efficient contraction schemes [17]. In particular, if the MPS is in the mixed canonical form, the computation only involves local manipulations illustrated in Eq. (B4).

Next, we carry out gradient descent to update the components of the merged tensor. The update is flexible and is open to various gradient descent techniques. Firstly, the stochastic gradient descent is considerable. Instead of averaging the gradient over the whole data set, the second term of the gradient in Eq. (B2) can be estimated by randomly chosen minibatches of samples, where the size of the minibatch m_{batch} plays the role of a hyperparameter in the training. Secondly, on a specific contracted tensor, one can conduct several steps of gradient descent. Note that, although the local update of $A^{(k,k+1)}$ does not change its environment, the shifting of $A^{(k,k+1)}$ makes a difference between n_{des} steps of the update with learning rate η and one update step with $\eta' = n_{\text{des}} \times \eta$. Thirdly, especially when several steps are conducted on each contracted tensor, the learning rate (the ratio of the update to the gradient) can be adaptively tuned by meta-algorithms such as RMSProp and Adam [43].

In practice, it is observed that sometimes the gradients become very small, while it is not in the vicinity of any local minimum of the landscape. In that case, a plateau or a saddle point may have been encountered, and we simply increase the learning rate so that the norm of the update is a function of the dimensions of the contracted tensor.

After updating the order-4 tensor in Eq. (B1), it is decomposed by unfolding the tensor to a matrix, subsequently applying singular value decomposition (SVD), and finally unfolding the obtained two matrices back to two order-3 tensors:



(B5)

where U, V are unitary matrices and Λ is a diagonal matrix containing singular values on the diagonal. The number of nonvanishing singular values will generally increase compared to the original value in Eq. (B1) because the MPS observes correlations in the data and tries to capture them. We truncate those singular values whose ratios to the largest one are smaller than a prescribed hyperparameter cutoff ϵ_{cut}, along with their corresponding row vectors and column vectors deleted in U and $V^\dagger$.

If the next bond to train on is the $(k+1)$th bond on the right, take $A^{(k)} = U$ so that it is left canonical, and consequently $A^{(k+1)} = \Lambda V^\dagger$. Meanwhile, if the MPS is about to be trained on the $(k-1)$th bond, analogously,

$A^{(k+1)} = V^{\dagger}$ will be right canonical and $A^{(k)} = U\Lambda$. This keeps the MPS in a mixed-canonical form.

The whole training process consists of many loops. In each loop, the training starts from the rightmost bond [between $A^{(N-1)}$ and $A^{(N)}$] and sweeps to the leftmost $A^{(1)}$, then back to the rightmost.

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

[2] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, *A Learning Algorithm for Boltzmann Machines*, Cogn. Sci. **9**, 147 (1985).

[3] P. Smolensky, *Information Processing in Dynamical Systems: Foundations of Harmony Theory*, in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* (MIT Press, Cambridge, MA, 1986), Vol. 1, pp. 194–281.

[4] R. Salakhutdinov, *Learning Deep Generative Models*, Annu. Rev. Stat. Appl. **2**, 361 (2015).

[5] D. P. Kingma and M. Welling, *Auto-encoding Variational Bayes*, arXiv:1312.6114.

[6] B. Uria, M.-A. Côté, K. Gregor, I. Murray, and H. Larochelle, *Neural Autoregressive Distribution Estimation*, J. Mach. Learn. Res. **17**, 1 (2016).

[7] A. Van Oord, N. Kalchbrenner, and K. Kavukcuoglu, *Pixel Recurrent Neural Networks*, in *Proceedings of The 33rd International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 48 (PMLR, New York, New York, USA, 2016), pp. 1747–1756.

[8] L. Dinh, D. Krueger, and Y. Bengio, *NICE: Non-linear Independent Components Estimation*, arXiv:1410.8516.

[9] L. Dinh, J. Sohl-Dickstein, and S. Bengio, *Density Estimation Using Real NVP*, arXiv:1605.08803.

[10] D. Rezende and S. Mohamed, *Variational Inference with Normalizing Flows*, in *Proceedings of the 32nd International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 37 (PMLR, Lille, France, 2015), pp. 1530–1538.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative Adversarial Nets*, in *Advances in Neural Information Processing Systems 27* (Curran Associates, Inc., 2014), pp. 2672–2680.

[12] J. J. Hopfield, *Neural Networks and Physical Systems with Emergent Collective Computational Abilities*, Proc. Natl. Acad. Sci. U.S.A. **79**, 2554 (1982).

[13] H. J. Kappen and F. de Borja Rodríguez Ortiz, *Boltzmann Machine Learning Using Mean Field Theory and Linear Response Correction*, in *Advances in Neural Information Processing Systems 10* (MIT Press, Cambridge, MA, 1998), pp. 280–286.

[14] Y. Roudi, J. Tyrcha, and J. Hertz, *Ising Model for Neural Data: Model Quality and Approximate Methods for Extracting Functional Connectivity*, Phys. Rev. E **79**, 051915 (2009).

[15] W. M. C. Foulkes, L. Mitas, R. J. Needs, and G. Rajagopal, *Quantum Monte Carlo Simulations of Solids*, Rev. Mod. Phys. **73**, 33 (2001).

[16] G. Carleo and M. Troyer, *Solving the Quantum Many-Body Problem with Artificial Neural Networks*, Science **355**, 602 (2017).

[17] U. Schollwöck, *The Density-Matrix Renormalization Group in the Age of Matrix Product States*, Ann. Phys. **326**, 96 (2011).

[18] R. Orús, *A Practical Introduction to Tensor Networks: Matrix Product States and Projected Entangled Pair States*, Ann. Phys. **349**, 117 (2014).

[19] M. B. Hastings, *An Area Law for One-Dimensional Quantum Systems*, J. Stat. Mech. 2007, P08024 (2007).

[20] J. Eisert, M. Cramer, and M. B. Plenio, *Colloquium: Area Laws for the Entanglement Entropy*, Rev. Mod. Phys. **82**, 277 (2010).

[21] J. A. Bengua, H. N. Phien, and H. D. Tuan, *Optimal Feature Extraction and Classification of Tensors via Matrix Product State Decomposition*, arXiv:1503.00516.

[22] A. Novikov, A. Rodomanov, A. Osokin, and D. Vetrov, *Putting MRFs on a Tensor Train*, in *International Conference on Machine Learning* (2014), pp. 811–819.

[23] A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov, *Tensorizing Neural Networks*, in *Advances in Neural Information Processing Systems* (2015), pp. 442–450.

[24] N. Cohen, O. Sharir, and A. Shashua, *On the Expressive Power of Deep Learning: A Tensor Analysis*, arXiv:1509.05009.

[25] A. Cichocki, N. Lee, I. Oseledets, A.-H. Phan, Q. Zhao, D. P. Mandic *et al.*, *Tensor Networks for Dimensionality Reduction and Large-Scale Optimization: Part 1 Low-Rank Tensor Decompositions*, Foundations and Trends in Machine Learning **9**, 249 (2016).

[26] Y. Levine, D. Yakira, N. Cohen, and A. Shashua, *Deep Learning and Quantum Entanglement: Fundamental Connections with Implications to Network Design*, arXiv:1704.01552.

[27] D. Perez-Garcia, F. Verstraete, M. M. Wolf, and J. I. Cirac, *Matrix Product State Representations*, Quantum Inf. Comput. **7**, 401 (2007).

[28] I. V. Oseledets, *Tensor-Train Decomposition*, SIAM J. Sci. Comput. **33**, 2295 (2011).

[29] Z. Landau, U. Vazirani, and T. Vidick, *A Polynomial Time Algorithm for the Ground State of 1D Gapped Local Hamiltonians*, Nat. Phys. **11**, 566 (2015).

[30] S. R. White, *Density Matrix Formulation for Quantum Renormalization Groups*, Phys. Rev. Lett. **69**, 2863 (1992).

[31] E. M. Stoudenmire and S. R. White, *Studying Two-Dimensional Systems with the Density Matrix Renormalization Group*, Annu. Rev. Condens. Matter Phys. **3**, 111 (2012).

[32] E. M. Stoudenmire and D. J. Schwab, *Supervised Learning with Quantum-Inspired Tensor Networks*, *Advances in Neural Information Processing Systems*, edited by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Curran Associates, Inc., 2016), pp. 4799–4807.

[33] A. Novikov, M. Trofimov, and I. Oseledets, *Exponential Machines*, arXiv:1605.03795.

[34] A. J. Gallego and R. Orus, *The Physical Structure of Grammatical Correlations: Equivalences, Formalizations and Consequences*, arXiv:1708.01525.

[35] 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).

[36] A. J. Ferris and G. Vidal, *Perfect Sampling with Unitary Tensor Networks*, Phys. Rev. B **85**, 165146 (2012).

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

[38] M. Born, *Zur Quantenmechanik der Stoßvorgänge*, Z. Phys. **37**, 863 (1926).

[39] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang, *A Tutorial on Energy-Based Learning*, http://yann.lecun.com/exdb/publis/orig/lecun-06.pdf.

[40] M.-J. Zhao and H. Jaeger, *Norm-Observable Operator Models*, Neural Comput. **22**, 1927 (2010).

[41] R. Bailly, *Quadratic Weighted Automata: Spectral Algorithm and Likelihood Maximization*, J. Mach. Learn. Res. **20**, 147 (2011).

[42] S. Kullback and R. A. Leibler, *On Information and Sufficiency*, Ann. Math. Stat. **22**, 79 (1951).

[43] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, arXiv:1412.6980.

[44] G. E. Hinton, *A Practical Guide to Training Restricted Boltzmann Machines*, in *Neural Networks: Tricks of the Trade* (Springer, New York, 2012), pp. 599–619.

[45] M. Gabrie, E. W. Tramel, and F. Krzakala, *Training Restricted Boltzmann Machine via the Thouless-Anderson-Palmer Free Energy*, in *Advances in Neural Information Processing Systems 28*, edited by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Curran Associates, Inc., 2015), pp. 640–648.

[46] M. J. Wainwright and M. I. Jordan, *Graphical Models, Exponential Families, and Variational Inference*, Foundations and Trends in Machine Learning **1**, 1 (2008).

[47] D. Koller and N. Friedman, *Probabilistic Graphical Models, Principles and Techniques* (The MIT Press, Cambridge, MA, 2009).

[48] D. J. C. MacKay, *Information Theory, Inference and Learning Algorithms* (Cambridge University Press, Cambridge, England, 2003).

[49] S. S. Wilks, *The Large-Sample Distribution of the Likelihood Ratio for Testing Composite Hypotheses*, Ann. Math. Stat. **9**, 60 (1938).

[50] D. J. Amit, H. Gutfreund, and H. Sompolinsky, *Spin-Glass Models of Neural Networks*, Phys. Rev. A **32**, 1007 (1985).

[51] Y. LeCun, C. Cortes, and C. J. C. Burges, *The MNIST Database of Handwritten Digits*, 1998, http://yann.lecun.com/exdb/mnist.

[52] D. Krotov and J. J. Hopfield, *Dense Associative Memory for Pattern Recognition*, in *Advances in Neural Information Processing Systems* (2016), pp. 1172–1180.

[53] Y. Zhang, T. Grover, and A. Vishwanath, *Entanglement Entropy of Critical Spin Liquids*, Phys. Rev. Lett. **107**, 067202 (2011).

[54] H. Zhao-Yu, W. Jun, W. Song-Bo, L. Ze-Yang, M. Liang-Zhu, F. Heng, and L. Wang, *Efficient Quantum Tomography with Fidelity Estimation*, arXiv:1712.03213.

[55] K. Temme and F. Verstraete, *Stochastic Matrix Product States*, Phys. Rev. Lett. **104**, 210502 (2010).

[56] E. M. Stoudenmire, *Learning Relevant Features of Data with Multi-scale Tensor Networks*, Quantum Sci. Technol. **3**, 034003 (2018).

[57] F. Verstraete and J. I. Cirac, *Renormalization Algorithms for Quantum-Many Body Systems in Two and Higher Dimensions*, arXiv:cond-mat/0407066.

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

[59] M. Levin and C. P. Nave, *Tensor Renormalization Group Approach to Two-Dimensional Classical Lattice Models*, Phys. Rev. Lett. **99**, 120601 (2007).

[60] Z. Y. Xie, H. C. Jiang, Q. N. Chen, Z. Y. Weng, and T. Xiang, *Second Renormalization of Tensor-Network States*, Phys. Rev. Lett. **103**, 160601 (2009).

[61] C. Wang, S.-M. Qin, and H.-J. Zhou, *Topologically Invariant Tensor Renormalization Group Method for the Edwards-Anderson Spin Glasses Model*, Phys. Rev. B **90**, 174201 (2014).

[62] G. Evenbly and G. Vidal, *Tensor Network Renormalization*, Phys. Rev. Lett. **115**, 180405 (2015).

[63] A. Perdomo-Ortiz, M. Benedetti, J. Realpe-Gómez, and R. Biswas, *Opportunities and Challenges for Quantum-Assisted Machine Learning in Near-Term Quantum Computers*, arXiv:1708.09757.

[64] M. Schwarz, K. Temme, and F. Verstraete, *Preparing Projected Entangled Pair States on a Quantum Computer*, Phys. Rev. Lett. **108**, 110502 (2012).

[65] W. Huggins, P. Patel, K. B. Whaley, and E. M. Stoudenmire, *Towards Quantum Machine Learning with Tensor Networks*, arXiv:1803.11537.

[66] Setting $\mathcal{D}_k = 1$ for all bonds makes the bond dimension difficult to grow in the initial training phase, since the rank of the two-site tensor is $1 \times 2 \times 2 \times 1$ and the number of the nonzero singular value is at most 2, which is likely to be truncated back to $\mathcal{D}_k = 1$ with small cutoff.

[67] A. Hyvärinen, *Consistency of Pseudolikelihood Estimation of Fully Visible Boltzmann Machines*, Neural Comput. **18**, 2283 (2006).