Neural-network quantum states at finite temperature

Naoki Irikura and Hiroki Saito

Department of Engineering Science, University of Electro-Communications, Tokyo 182-8585, Japan

(Received 7 November 2019; accepted 21 February 2020; published 9 March 2020)

We propose a method to obtain the thermal-equilibrium density matrix of a many-body quantum system using artificial neural networks. The variational function of the many-body density matrix is represented by a convolutional neural network with two input channels. We first prepare an infinite-temperature state, and the temperature is lowered by imaginary-time evolution. We apply this method to the one-dimensional Bose-Hubbard model and compare the results with those obtained by exact diagonalization.

DOI: 10.1103/PhysRevResearch.2.013284

I. INTRODUCTION

One of the challenging problems in physics is the determination of the properties of quantum many-body systems. Quantum many-body problems are difficult to solve, since the size of the Hilbert space exponentially increases with the size of the system. An approximate method to overcome this difficulty is the variational method, in which the huge Hilbert space is represented by a variational wave function with a tractable number of variational parameters. However, the variational method relies greatly on the physical insight of researchers to find sophisticated variational wave functions [1,2].

Carleo and Troyer [3] proposed the use of artificial neural networks to represent variational wave functions for quantum many-body states. It is known that artificial neural networks are very flexible and can approximate any function if the number of hidden units in the neural networks is sufficient. Using artificial neural networks as variational functions, therefore, we expect that quantum many-body wave functions can be approximated efficiently, in which the essential features of quantum many-body states are automatically captured as variational network parameters are optimized. This method has been applied to a variety of quantum many-body problems, and various properties of quantum many-body states represented by neural networks have been investigated [4–27].

Recently, artificial neural networks were also used to represent the density matrices of open quantum many-body systems [28–32]. A density operator $\hat{\rho}$ contains more information than a pure state $|\psi\rangle$, and open quantum systems need more representation ability of neural networks than closed quantum systems. In Refs. [29–32], the master equations in the Lindblad form are solved using the variational Monte Carlo method, and the steady states of dissipative spin systems are obtained. The successful use of neural networks to represent density matrices opens up the application of machine learning not only to dissipative quantum systems, but also to finite-temperature states of quantum many-body systems.

Although the Boltzmann machine was used in previous studies [28–32], in this paper, we use a convolutional neural network (CNN) [33] to represent the density matrix of a finite-temperature state. The CNN has been used to represent the ground states, i.e., pure states, of quantum many-body systems [10,16,18,22]. In the case of the pure state $|\psi\rangle$, for the base $|\boldsymbol{n}\rangle$, a configuration of particles or spins $\boldsymbol{n}$ is input into the CNN, and the output of the CNN gives the amplitude $\langle\boldsymbol{n}|\psi\rangle$. For the density matrix, in the present study, we input $\boldsymbol{n}$ and $\boldsymbol{n}'$ into the CNN with two input channels, and the output of the CNN gives the matrix element $\langle\boldsymbol{n}|\hat{\rho}|\boldsymbol{n}'\rangle$ of the density operator $\hat{\rho}$. We first prepare the density matrix at infinite temperature with $\beta = (k_B T)^{-1} = 0$, and the imaginary-time propagator $e^{-\Delta\beta\hat{H}}$ is applied successively to the density matrix to obtain $\hat{\rho} = e^{-\beta\hat{H}}$ at each β. A similar imaginary-time method was used to obtain the thermal equilibrium in matrix product states [34–36]. We apply our method to the Bose-Hubbard model, which describes cold bosonic atoms in optical lattices [37]. We calculate the finite-temperature density matrix of the Bose-Hubbard model in one-dimensional space and compare the results with those obtained by exact diagonalization. We also investigate the dependence of the accuracy of our method on various conditions, such as CNN structures.

This paper is organized as follows. Section II explains the method, Sec. III shows the numerical results, and Sec. IV provides the conclusions of the study.

II. METHOD

To demonstrate the neural-network method for obtaining the finite-temperature density matrix, we apply it to the Bose-Hubbard model in one-dimensional space. The Hamiltonian is given by

$$\hat{H} = -\sum_{\langle i,j\rangle} \hat{a}_i \hat{a}_j^\dagger + \frac{U}{2}\sum_i \hat{n}_i(\hat{n}_i - 1), \tag{1}$$

where U is the on-site interaction energy, $\hat{a}_i$ is the annihilation operator of a boson at the ith site, $\hat{n}_i = \hat{a}_i^\dagger \hat{a}_i$ is the number

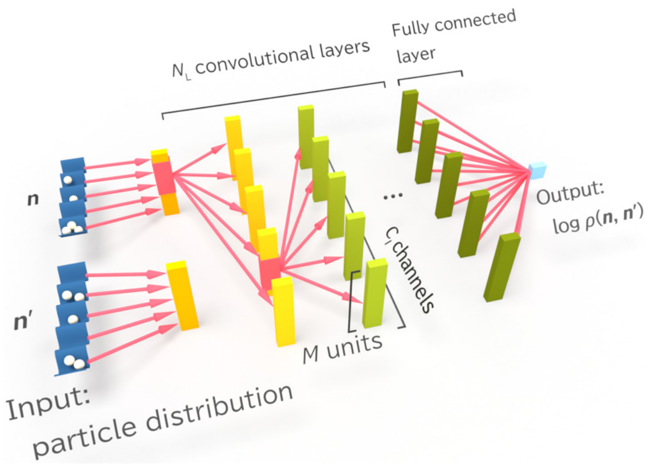
N_L convolutional layers
Fully connected layer
n
n'
...
C channels
M units
Output: $\log \rho(n, n')$
Input: particle distribution

FIG. 1. Schematic illustration of the CNN to represent the density matrix $\rho(\boldsymbol{n}, \boldsymbol{n}') = \langle \boldsymbol{n}|\hat{\rho}|\boldsymbol{n}'\rangle$. The one-dimensional configurations of bosons $\boldsymbol{n}$ and $\boldsymbol{n}'$ on M sites are input into the two input channels. Successive N_L convolutional layers are followed by a fully connected layer, which gives output $u^{(\text{out})}$. The matrix element of the density matrix $\rho(\boldsymbol{n}, \boldsymbol{n}')$ is given by $e^{u^{(\text{out})}}$.

operator, and $\langle i, j\rangle$ represents adjacent sites. The energy is normalized in such a way that the hopping coefficient becomes unity. Such a system can be realized by ultracold bosonic atoms loaded in an optical lattice [37]. We assume the periodic boundary condition $\hat{a}_{M+1} = \hat{a}_1$, where M is the number of sites. A pure quantum state can be expanded by the Fock-state bases $|\boldsymbol{n}\rangle$, where $\boldsymbol{n} = (n_1, n_2, \dots, n_M)$ represents the number of bosons in each site. We consider a canonical ensemble at temperature $T = (k_B\beta)^{-1}$ with a total number of bosons N. The number of Fock-state bases $|\boldsymbol{n}\rangle$ satisfying $\sum_i n_i = N$ is $N_{\text{base}} = (N + M - 1)!/[N!(M - 1)!]$, which increases exponentially with N and M. We restrict ourselves to the case of $N = M$ in the following analysis. In this case, at zero temperature, the system becomes a Mott insulator for large U, and the system exhibits superfluidity for small U. At finite temperature, the normal phase emerges [38] around the Mott insulator and superfluid regions in the phase diagram. Since all the matrix elements $\langle \boldsymbol{n}|e^{-\epsilon\hat{H}}|\boldsymbol{n}'\rangle$ for infinitesimal $\epsilon > 0$ can be taken to be real and non-negative without loss of generality, all the matrix elements of the thermal density matrix $\langle \boldsymbol{n}|e^{-\beta\hat{H}}|\boldsymbol{n}'\rangle$, which are decomposed to matrix products of $\langle \boldsymbol{n}|e^{-\epsilon\hat{H}}|\boldsymbol{n}'\rangle$, can be taken to be real and non-negative [39].

We employ the CNN [33] to represent the density matrix $\langle \boldsymbol{n}|\hat{\rho}|\boldsymbol{n}'\rangle \equiv \rho(\boldsymbol{n}, \boldsymbol{n}')$ of the system (see Fig. 1). The inputs into the CNN are vectors of integers $\boldsymbol{n}$ and $\boldsymbol{n}'$, and the CNN outputs a single real value $u^{(\text{out})}$. The matrix element of the density matrix is represented as

$$\rho(\boldsymbol{n}, \boldsymbol{n}') = e^{u^{(\text{out})}}. \tag{2}$$

We generate the thermal density matrix as follows. We first prepare the initial CNN that represents the density matrix at infinite temperature $\beta = 0$, which is used as the initial density matrix of the imaginary-time evolution. In general, we can construct a CNN that approximates a desired density matrix $\rho_{\text{target}}(\boldsymbol{n}, \boldsymbol{n}')$ by the method described below. The initial CNN can thus be constructed with

$$\rho_{\text{target}}(\boldsymbol{n}, \boldsymbol{n}') = \lim_{\beta\to 0}\langle \boldsymbol{n}|e^{-\beta\hat{H}}|\boldsymbol{n}'\rangle = \delta_{\boldsymbol{n},\boldsymbol{n}'}. \tag{3}$$

Next, we set the target as $\rho_{\text{target}}(\boldsymbol{n}, \boldsymbol{n}') = \rho_{\Delta\beta}(\boldsymbol{n}, \boldsymbol{n}') = \langle \boldsymbol{n}|e^{-\Delta\beta\hat{H}}|\boldsymbol{n}'\rangle$, and construct another CNN that represents this target. Repeating this procedure, we obtain CNNs that represent $\rho_{2\Delta\beta}, \rho_{3\Delta\beta}, \dots$, successively.

We can calculate matrix elements of the density matrix $\rho_{\beta+\Delta\beta}(\boldsymbol{n}, \boldsymbol{n}')$ at $\beta + \Delta\beta$ using $\rho_\beta(\boldsymbol{n}, \boldsymbol{n}')$ at β as

$$\begin{aligned}\rho_{\beta+\Delta\beta}(\boldsymbol{n}, \boldsymbol{n}') &= \langle \boldsymbol{n}|e^{-(\beta+\Delta\beta)\hat{H}}|\boldsymbol{n}'\rangle \\ &= \sum_{\boldsymbol{n}_1,\boldsymbol{n}_2}\langle \boldsymbol{n}|e^{-\frac{\Delta\beta\hat{H}}{2}}|\boldsymbol{n}_1\rangle\rho_\beta(\boldsymbol{n}_1, \boldsymbol{n}_2)\langle \boldsymbol{n}_2|e^{-\frac{\Delta\beta\hat{H}}{2}}|\boldsymbol{n}'\rangle.\end{aligned} \tag{4}$$

We expand $e^{-\Delta\beta\hat{H}/2}$ with respect to $\Delta\beta$ as

$$\begin{aligned}\rho_{\beta+\Delta\beta}(\boldsymbol{n}, \boldsymbol{n}') \simeq{}& \rho_\beta(\boldsymbol{n}, \boldsymbol{n}') - \frac{\Delta\beta}{2}\sum_{\boldsymbol{n}_1}[\langle \boldsymbol{n}|\hat{H}|\boldsymbol{n}_1\rangle\rho_\beta(\boldsymbol{n}_1, \boldsymbol{n}') \\ &+ \rho_\beta(\boldsymbol{n}, \boldsymbol{n}_1)\langle \boldsymbol{n}_1|\hat{H}|\boldsymbol{n}'\rangle] \\ &+ \frac{\Delta\beta^2}{8}\sum_{\boldsymbol{n}_1}[\langle \boldsymbol{n}|\hat{H}^2|\boldsymbol{n}_1\rangle\rho_\beta(\boldsymbol{n}_1, \boldsymbol{n}') \\ &+ \rho_\beta(\boldsymbol{n}, \boldsymbol{n}_1)\langle \boldsymbol{n}_1|\hat{H}^2|\boldsymbol{n}'\rangle] \\ &+ \frac{\Delta\beta^2}{4}\sum_{\boldsymbol{n}_1,\boldsymbol{n}_2}\langle \boldsymbol{n}|\hat{H}|\boldsymbol{n}_1\rangle\rho_\beta(\boldsymbol{n}_1, \boldsymbol{n}_2)\langle \boldsymbol{n}_2|\hat{H}|\boldsymbol{n}'\rangle \\ &+ \cdots + (\text{terms proportional to } \Delta\beta^K),\end{aligned} \tag{5}$$

where we cut off $O(\Delta\beta^{K+1})$ terms in Eq. (5). By this approximation, the number of terms in Eq. (5) is reduced to $O(M^K)$, since the number of nonzero matrix elements $\langle \boldsymbol{n}|\hat{H}^K|\boldsymbol{n}'\rangle$ is $O(M^K)$. We can thus calculate any matrix elements of $\rho_{\text{target}}(\boldsymbol{n}, \boldsymbol{n}') = \rho_{\beta+\Delta\beta}(\boldsymbol{n}, \boldsymbol{n}')$, when we have the CNN that represents the density matrix $\rho_\beta(\boldsymbol{n}, \boldsymbol{n}')$.

The above method to obtain the imaginary-time evolution is based on the technique to construct a CNN that approximates $\rho_{\text{target}}(\boldsymbol{n}, \boldsymbol{n}')$. This is done by minimizing

$$L = \frac{1}{2}\sum_{\boldsymbol{n},\boldsymbol{n}'}[\rho(\boldsymbol{n}, \boldsymbol{n}') - \rho_{\text{target}}(\boldsymbol{n}, \boldsymbol{n}')]^2, \tag{6}$$

where $\rho(\boldsymbol{n}, \boldsymbol{n}')$ is the density matrix represented by the CNN to be optimized. Usually the gradient of L with respect to the network parameters is used to update the network parameters for minimizing L. Here, instead of such a gradient [Eq. (A2)], we introduce a modified gradient in Eq. (A7) to stabilize the network updates and facilitate the reduction of L (see the Appendix). Using the modified gradient with the Adam optimizer [33,40], we can optimize a CNN to approximate $\rho_{\text{target}}(\boldsymbol{n}, \boldsymbol{n}')$. Using this technique with Eq. (3), we can construct a CNN that represents ρ_0, and using this technique with Eq. (5), we can construct the CNNs that represent $\rho_{\Delta\beta}, \rho_{2\Delta\beta}, \dots$, successively. To prepare the CNN that represents ρ_0, we use random network parameters to start the above optimization. After that, to facilitate convergence, the initial values of the network parameters are set to those of $\rho_{(n-1)\Delta\beta}$ to construct the CNN for $\rho_{n\Delta\beta}$.

The expectation value of an observable $\hat{A}$ is written as

$$\langle\hat{A}\rangle = \frac{\mathrm{Tr}(\hat{\rho}\hat{A})}{\mathrm{Tr}\,\hat{\rho}} = \frac{\sum_{\boldsymbol{n},\boldsymbol{n}'}\rho(\boldsymbol{n},\boldsymbol{n}')\langle\boldsymbol{n}'|\hat{A}|\boldsymbol{n}\rangle}{\sum_{\boldsymbol{n}}\rho(\boldsymbol{n},\boldsymbol{n})} = \sum_{\boldsymbol{n}} P(\boldsymbol{n})A(\boldsymbol{n}), \tag{7}$$

where Tr indicates trace, $P(\boldsymbol{n}) = \rho(\boldsymbol{n},\boldsymbol{n})/\sum_{\boldsymbol{n}}\rho(\boldsymbol{n},\boldsymbol{n})$, and $A(\boldsymbol{n}) = \sum_{\boldsymbol{n}'}\rho(\boldsymbol{n},\boldsymbol{n}')\langle\boldsymbol{n}'|\hat{A}|\boldsymbol{n}\rangle/\rho(\boldsymbol{n},\boldsymbol{n})$. The summation in the second line of Eq. (7) is calculated by the Monte Carlo method with a Metropolis sampling of $\boldsymbol{n}$ with probability distribution $P(\boldsymbol{n})$. To evaluate $\sum_{\boldsymbol{n}'}$ in $A(\boldsymbol{n})$, the number of nonzero matrix elements $\langle\boldsymbol{n}'|\hat{A}|\boldsymbol{n}\rangle$ should not be exponentially large with respect to the system size.

Here, we briefly describe the structure of the CNN [33] to make the present paper self-contained. The input into the CNN is $\boldsymbol{n}$ and $\boldsymbol{n}'$, which we denote as $\boldsymbol{u}_1^{(0)}$ and $\boldsymbol{u}_2^{(0)}$, respectively, i.e., the CNN has two input channels, with each of size M. The first hidden layer is calculated as

$$u_{m,j}^{(1)} = \sum_{k=1}^{2}\sum_{p=0}^{F_1-1} W_{k,m,p}^{(1)} u_{k,j+p}^{(0)} + b_m^{(1)}, \tag{8}$$

and these are propagated to the deeper layers as

$$u_{m,j}^{(l)} = \sum_{k=1}^{C_{l-1}}\sum_{p=0}^{F_l-1} W_{k,m,p}^{(l)} f\big(u_{k,j+p}^{(l-1)}\big) + b_m^{(l)}, \tag{9}$$

where $\boldsymbol{W}_{k,m}^{(l)}$ is the one-dimensional filter with size F_l, $\boldsymbol{b}^{(l)}$ is the bias, and C_l is the number of channels in the lth hidden layer. In Eqs. (8) and (9), the subscripts m and k identify the channels. The number of units in each channel in the input and hidden layers is M, i.e., $\boldsymbol{u}_m^{(l)} = (u_{m,1}^{(l)}, u_{m,2}^{(l)}, \dots, u_{m,M}^{(l)})$, which satisfies the periodic boundary condition $u_{m,M+1}^{(l)} = u_{m,1}^{(l)}$. We use the rectified linear unit (ReLU) [33] as the activation function f,

$$f(x) = \begin{cases} x & (x \geqslant 0), \\ 0 & (x < 0), \end{cases} \tag{10}$$

which is frequently used in the application of the CNN. After N_L convolutional layers, the CNN finally gives a single output value $u^{(\mathrm{out})}$ through the fully connected layer as

$$u^{(\mathrm{out})} = \sum_{m=1}^{C_L}\sum_{j=1}^{M} W_{m,j}^{(\mathrm{fc})} u_{m,j}^{(L)}. \tag{11}$$

The network parameters are thus the filters $\boldsymbol{W}_{k,m}^{(l)}$ and biases $\boldsymbol{b}^{(l)}$ in the convolutional layers, and weights $\boldsymbol{W}_m^{(\mathrm{fc})}$ in the fully connected layer, which are all taken to be real, and therefore the output $u^{(\mathrm{out})}$ is real. Using the output $u^{(\mathrm{out})}$ in Eq. (11), the matrix element of the density matrix is represented by Eq. (2). In the present model, all the matrix elements of the density matrix are positive, and therefore the positive definiteness is assured. However, the Hermiticity of the density matrix is not assured, since $\rho(\boldsymbol{n},\boldsymbol{n}')$ is not always equal to $\rho(\boldsymbol{n}',\boldsymbol{n})$ in the CNN representation of the density matrix. Our CNN representation is thus redundant compared with the Boltzmann-machine representation proposed in Ref. [28]. Nevertheless, the correct density matrix is obtained by the imaginary-time evolution, as demonstrated in the next section.

III. RESULTS

We consider a system of $M = 5$ sites with $N = 5$ particles. The CNN consists of $N_L = 4$ convolutional layers with filter size $F_1 = F_2 = F_3 = F_4 = 5$ and $C_1 = C_2 = C_3 = C_4 = 16$ channels. The imaginary-time evolution is generated with $\Delta\beta = 0.01$, where we take the terms up to the second order of $\Delta\beta$ in the expansion in Eq. (5) (i.e., $K = 2$). We take 2000 samples in the Metropolis sampling to calculate the gradient in Eq. (A7) in each Adam optimization step. The optimization steps are performed 10^4 times to obtain the next density matrix $\rho_{\beta+\Delta\beta}$ from ρ_β in the imaginary-time evolution. In the Adam optimization, the learning rate is decreased from 10^{-3} to 10^{-8} quadratically as $(10^{-3} - 10^{-8})(1 - \text{steps}/10^4)^2 + 10^{-8}$. In order to suppress exponential growth or decay of the matrix elements $\langle\boldsymbol{n}|e^{-\beta\hat{H}}|\boldsymbol{n}'\rangle$ in the imaginary-time evolution, we subtract an average energy from the Hamiltonian $\hat{H} - \langle\hat{H}\rangle_\beta$ to obtain $\rho_{\beta+\Delta\beta}$. By this subtraction, numerical errors are reduced, while the results with and without the subtraction are mathematically equivalent to each other since $e^{-\Delta\beta(\hat{H}-\langle\hat{H}\rangle_\beta)/2}\hat{\rho}e^{-\Delta\beta(\hat{H}-\langle\hat{H}\rangle_\beta)/2} \propto e^{-\Delta\beta\hat{H}/2}\hat{\rho}e^{-\Delta\beta\hat{H}/2}$.

Figure 2(a) shows the imaginary-time evolution of the expectation value of the energy $\langle\hat{H}\rangle$ obtained by our method for $U = 1$, 4, and 10. In Fig. 2(a), we also plot the exact energy E_β^{exact} obtained by the exact diagonalization of the Hamiltonian. The lines of $\langle\hat{H}\rangle$ almost overlap with those of E_β^{exact}. In Fig. 2(b), we plot the mean-squared error in the matrix elements of the density matrix, defined as [31]

$$\delta\rho = \frac{1}{N_{\mathrm{base}}^2}\sum_{\boldsymbol{n},\boldsymbol{n}'}\big[\rho_\beta(\boldsymbol{n},\boldsymbol{n}') - \rho_\beta^{\mathrm{exact}}(\boldsymbol{n},\boldsymbol{n}')\big]^2, \tag{12}$$

where $\rho_\beta^{\mathrm{exact}}(\boldsymbol{n},\boldsymbol{n}')$ is the density matrix obtained by exact diagonalization of the Hamiltonian, and the summation is taken over all $\boldsymbol{n}$ and $\boldsymbol{n}'$. In calculating $\delta\rho$, the density matrix is normalized as $\sum_{\boldsymbol{n}}\rho_\beta(\boldsymbol{n},\boldsymbol{n}) = 1$. The error $\delta\rho$ in Fig. 2(b) is less than 10^{-8}. Thus, our method works well for the whole temperature region and both for superfluid and Mott insulator regimes.

In Fig. 2(b), $\delta\rho$ increases for the early stage of the imaginary-time evolution, and then $\delta\rho$ decreases with β. This is because the imaginary-time evolution $e^{-\Delta\beta\hat{H}/2}\hat{\rho}e^{-\Delta\beta\hat{H}/2}$ eliminates excited states in $\hat{\rho}$, and then also eliminates errors arising during the imaginary-time evolution. For $\beta \to \infty$, the density matrix converges to the ground state, even if errors arise during the imaginary-time evolution.

The errors arise from various sources: the cutoff in the expansion in Eq. (5), the representation ability of the CNN, the statistical errors due to the Monte Carlo sampling, and insufficient convergence in the Adam optimization. Figure 3 shows the dependence of the results on various conditions. We see that the errors are increased by reducing the cutoff order in Eq. (5) from $K = 2$ to $K = 1$ [line (ii) in Fig. 3]. The errors are also increased by reducing the number of convolutional

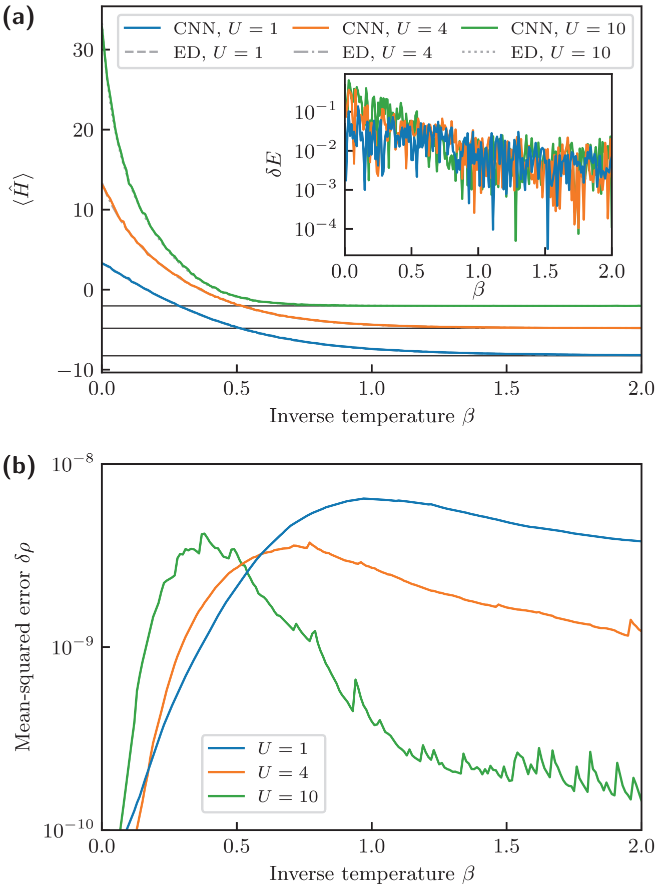

FIG. 2. Imaginary-time evolution of the density matrix represented by a CNN for $U = 1$, 4, and 10. (a) Expectation value of the Hamiltonian $\langle \hat{H} \rangle$. The lines of the energies E_β^{exact} obtained by exact diagonalization (ED) are also drawn, which however almost overlap with the lines obtained by our method and cannot be seen. The horizontal lines represent the exact energies of the ground states. The inset shows the error in the energy $\delta E = \langle \hat{H} \rangle - E_\beta^{\text{exact}}$. (b) Mean-squared error $\delta\rho$ defined in Eq. (12).

layers from $N_L = 4$ to $N_L = 2$ [line (iii)]. We have confirmed that the errors for $K = 3$ and those for $N_L = 6$ are similar to the errors in line (i), and therefore $K = 2$ and $N_L = 4$ are sufficient for the present system. We also confirmed that the accuracy is lowered by reducing the number of samples in the Metropolis sampling or the number of iterations in the Adam optimization.

To identify the errors arising from the cutoff in Eq. (5), we examined the imaginary-time evolution of the full density matrix in Eq. (5) (data not shown). The errors for the full density matrix with $K = 1$ are intermediate between lines (i) and (ii) in Fig. 3. The errors for the full density matrix with $K = 2$ are $\delta\rho \lesssim 10^{-11}$, which are much smaller than line (i). This indicates that the cutoff errors of $O(\Delta\beta^3)$ are almost negligible in line (i), and the errors in line (i) are mainly due to the reasons other than the cutoff. This indicates again that $K = 2$ is sufficient in the present condition. We also confirmed that reduction of $\Delta\beta$ does not improve the accuracy.

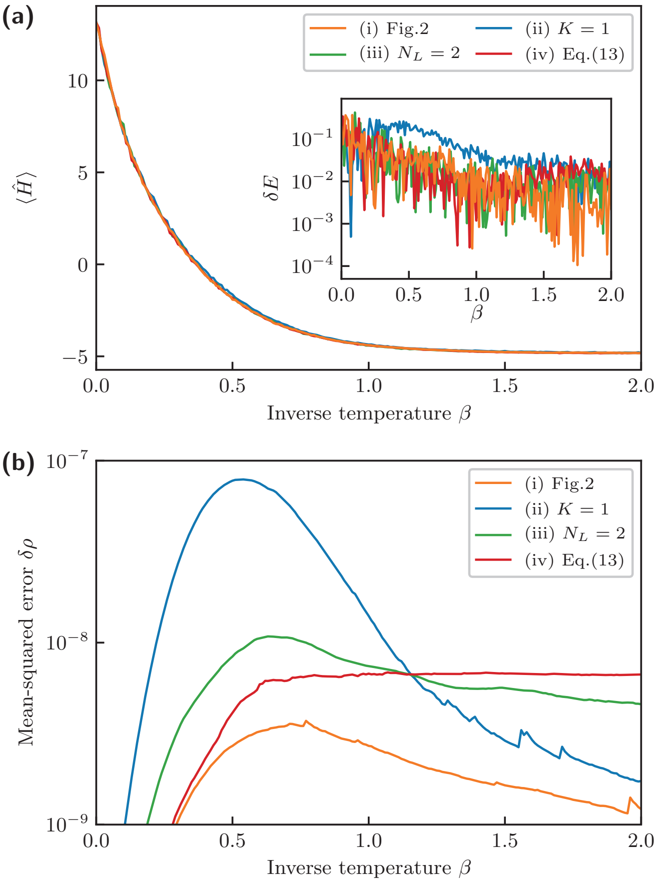

FIG. 3. Dependence of the accuracy on various conditions for $U = 4$, where (i) the same as in Fig. 2, (ii) the cutoff order in Eq. (5) is reduced to $K = 1$, (iii) the number of convolutional layers is reduced to $N_L = 2$, and (iv) the asymmetric expansion in Eq. (13) is used with $K = 2$. (a) Expectation value of the Hamiltonian $\langle \hat{H} \rangle$. The inset shows the error in the energy $\delta E = \langle \hat{H} \rangle - E_\beta^{\text{exact}}$. (b) Mean-squared error $\delta\rho$ defined in Eq. (12).

In Fig. 3, we also examine a different form of expansion of $e^{-\Delta\beta\hat{H}}$ instead of the symmetric expansion in Eq. (5),

$$\begin{aligned}\rho_{\beta+\Delta\beta}(\boldsymbol{n}, \boldsymbol{n}') &= \langle \boldsymbol{n} | e^{-(\beta+\Delta\beta)\hat{H}} | \boldsymbol{n}' \rangle \\ &= \sum_{\boldsymbol{n}''} \langle \boldsymbol{n} | e^{-\Delta\beta\hat{H}} | \boldsymbol{n}'' \rangle \langle \boldsymbol{n}'' | e^{-\beta\hat{H}} | \boldsymbol{n}' \rangle \\ &\simeq \sum_{k=0}^{K} \frac{(-\Delta\beta)^k}{k!} \sum_{\boldsymbol{n}''} \langle \boldsymbol{n} | \hat{H}^k | \boldsymbol{n}'' \rangle \rho_\beta(\boldsymbol{n}'', \boldsymbol{n}'), \end{aligned} \quad (13)$$

where the propagator $e^{-\Delta\beta\hat{H}}$ always operates from the left-hand side of ρ_β. Line (iv) in Fig. 3 shows the result using this asymmetric expansion with $K = 2$. The mean-squared error $\delta\rho$ monotonically increases for Eq. (13) and never decreases. This is because the operator $e^{-\Delta\beta\hat{H}}$ in the asymmetric form in Eq. (13) only eliminates excited states in the ket vectors in the density operator, and therefore, once errors arise in the bra vectors during the imaginary-time evolution, the errors remain for $\beta \to \infty$.

Finally, we consider how the method scales with the system size M. In the numerical procedure, the computationally most expensive part is the evaluation of $\rho_{\text{target}}(\boldsymbol{n}, \boldsymbol{n}')$ in Eqs. (A5) and (A8), which is calculated by the expansion in Eq. (5). Since the number of terms in $\hat{H}^K$ is $O(M^K)$, and the evaluation of ρ_β using the CNN costs $O(M)$, the total efficiency scales as $O(M^{K+1})$. The size of the CNN should also be increased with M, which is expected to be at most polynomial in M [27].

IV. CONCLUSIONS

We proposed a method to represent a many-body density matrix using a convolutional neural network (CNN), where the particle configurations $\boldsymbol{n}$ and $\boldsymbol{n}'$ are input into the CNN to produce the value of $\langle \boldsymbol{n}|\hat{\rho}|\boldsymbol{n}'\rangle$. We also proposed a method to obtain the density matrix at finite temperature through the imaginary-time evolution of the density matrix represented by the CNN. We applied our method to the one-dimensional Bose-Hubbard model, and demonstrated the imaginary-time evolution, which showed that the finite-temperature density matrix obtained by our method agrees well with that obtained by exact diagonalization of the Hamiltonian. We have also investigated the dependence of the accuracy on different conditions.

Neural-network quantum states are also efficient for representing the many-body states of fermions [9], and therefore we expect that our method can also be used to investigate the finite-temperature properties of fermions, which may be complementary to the path-integral quantum Monte Carlo method. The present method can also be extended to higher spatial dimensions in a straightforward manner. For these purposes, we need to suppress computational cost to study larger systems efficiently [41].

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grants No. JP17K05595 and No. JP17K05596.

APPENDIX: A METHOD TO OPTIMIZE NETWORK PARAMETERS

We can optimize a CNN so as to represent a desired density matrix $\rho_{\text{target}}(\boldsymbol{n}, \boldsymbol{n}')$ by minimizing

$$L = \frac{1}{2}\sum_{\boldsymbol{n},\boldsymbol{n}'}[\rho(\boldsymbol{n}, \boldsymbol{n}') - \rho_{\text{target}}(\boldsymbol{n}, \boldsymbol{n}')]^2, \tag{A1}$$

where $\rho(\boldsymbol{n}, \boldsymbol{n}')$ is the density matrix represented by the CNN to be optimized. We can reduce the value of L using its gradient with respect to the network parameters as

$$\frac{\partial L}{\partial w} = \sum_{\boldsymbol{n},\boldsymbol{n}'}\frac{\partial \rho(\boldsymbol{n}, \boldsymbol{n}')}{\partial w}[\rho(\boldsymbol{n}, \boldsymbol{n}') - \rho_{\text{target}}(\boldsymbol{n}, \boldsymbol{n}')], \tag{A2}$$

where w is one of the network parameters. Since the summation $\sum_{\boldsymbol{n},\boldsymbol{n}'}$ cannot be taken exactly for a large system, we may evaluate the summation in Eq. (A2) by the Monte Carlo method with a random sampling of $\boldsymbol{n}$ and $\boldsymbol{n}'$, i.e., taking random samples of $\mathcal{N}$ sets of $(\boldsymbol{n}, \boldsymbol{n}')$, we can evaluate Eq. (A2) as

$$\frac{1}{\mathcal{N}}\sum_{\text{sampling}}\frac{\partial \rho(\boldsymbol{n}, \boldsymbol{n}')}{\partial w}[\rho(\boldsymbol{n}, \boldsymbol{n}') - \rho_{\text{target}}(\boldsymbol{n}, \boldsymbol{n}')], \tag{A3}$$

where $\sum_{\text{sampling}}$ represents the sum over the samples. However, for the huge Hilbert space, the random sampling is inefficient and we need importance sampling. To do this, we take the gradient of $\ln L$ as

$$\begin{aligned}\frac{\partial \ln L}{\partial w} &= \frac{2\sum_{\boldsymbol{n},\boldsymbol{n}'}\frac{\partial \rho(\boldsymbol{n},\boldsymbol{n}')}{\partial w}[\rho(\boldsymbol{n}, \boldsymbol{n}') - \rho_{\text{target}}(\boldsymbol{n}, \boldsymbol{n}')]}{\sum_{\boldsymbol{n},\boldsymbol{n}'}[\rho(\boldsymbol{n}, \boldsymbol{n}') - \rho_{\text{target}}(\boldsymbol{n}, \boldsymbol{n}')]^2},\\ &= 2\sum_{\boldsymbol{n},\boldsymbol{n}'}P(\boldsymbol{n}, \boldsymbol{n}')\frac{\partial \rho(\boldsymbol{n}, \boldsymbol{n}')}{\partial w}\frac{1}{\rho(\boldsymbol{n}, \boldsymbol{n}') - \rho_{\text{target}}(\boldsymbol{n}, \boldsymbol{n}')},\end{aligned} \tag{A4}$$

where

$$P(\boldsymbol{n}, \boldsymbol{n}') = \frac{[\rho(\boldsymbol{n}, \boldsymbol{n}') - \rho_{\text{target}}(\boldsymbol{n}, \boldsymbol{n}')]^2}{\sum_{\boldsymbol{n},\boldsymbol{n}'}[\rho(\boldsymbol{n}, \boldsymbol{n}') - \rho_{\text{target}}(\boldsymbol{n}, \boldsymbol{n}')]^2} \tag{A5}$$

can be regarded as a probability distribution satisfying $\sum_{\boldsymbol{n},\boldsymbol{n}'}P(\boldsymbol{n}, \boldsymbol{n}') = 1$. Since the probability distribution $P(\boldsymbol{n}, \boldsymbol{n}')$ emphasizes the samples with larger deviation $|\rho(\boldsymbol{n}, \boldsymbol{n}') - \rho_{\text{target}}(\boldsymbol{n}, \boldsymbol{n}')|$, we expect that this is an appropriate probability distribution for the importance sampling. Taking $\mathcal{N}$ sets of samples $(\boldsymbol{n}, \boldsymbol{n}')$ using Metropolis sampling with probability $P(\boldsymbol{n}, \boldsymbol{n}')$, Eq. (A4) is evaluated as

$$\frac{2}{\mathcal{N}}\sum_{\text{sampling}}\frac{\partial \rho(\boldsymbol{n}, \boldsymbol{n}')}{\partial w}\frac{1}{\rho(\boldsymbol{n}, \boldsymbol{n}') - \rho_{\text{target}}(\boldsymbol{n}, \boldsymbol{n}')}. \tag{A6}$$

However, using this form of gradient to update the network parameters, we found that numerical instability arises, which may be due to accidental small denominators in the summation in Eq. (A6). To avoid the instability, we introduce a modified gradient instead of Eqs. (A2) and (A4),

$$\sum_{\boldsymbol{n},\boldsymbol{n}'}P(\boldsymbol{n}, \boldsymbol{n}')\frac{\partial \rho(\boldsymbol{n}, \boldsymbol{n}')}{\partial w}[\rho(\boldsymbol{n}, \boldsymbol{n}') - \rho_{\text{target}}(\boldsymbol{n}, \boldsymbol{n}')]. \tag{A7}$$

Taking the Metropolis sampling with probability $P(\boldsymbol{n}, \boldsymbol{n}')$, this is evaluated by

$$\frac{1}{\mathcal{N}}\sum_{\text{sampling}}\frac{\partial \rho(\boldsymbol{n}, \boldsymbol{n}')}{\partial w}[\rho(\boldsymbol{n}, \boldsymbol{n}') - \rho_{\text{target}}(\boldsymbol{n}, \boldsymbol{n}')]. \tag{A8}$$

Using this modified gradient, the network parameters are updated using the Adam scheme [33,40], until $\rho(\boldsymbol{n}, \boldsymbol{n}')$ converges sufficiently. We found that this scheme numerically stabilizes the network updates, and reduces L efficiently.

[1] R. P. Feynman, Application of quantum mechanics to liquid helium, Prog. Low Temp. Phys. **1**, 17 (1955).

[2] J. Bardeen, L. N. Cooper, and J. R. Schrieffer, Theory of superconductivity, Phys. Rev. **108**, 1175 (1957).

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

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

[5] 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).

[6] Z. Cai and J. Liu, Approximating quantum many-body wave-functions using artificial neural networks, Phys. Rev. B **97**, 035116 (2018).

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

[8] H. Saito, Solving the Bose-Hubbard Model with machine learning, J. Phys. Soc. Jpn. **86**, 093001 (2017).

[9] Y. Nomura, A. S. Darmawan, Y. Yamaji, and M. Imada, Restricted Boltzmann machine learning for solving strongly correlated quantum systems, Phys. Rev. B **96**, 205152 (2017).

[10] H. Saito and M. Kato, Machine learning technique to find quantum many-body ground states of bosons on a lattice, J. Phys. Soc. Jpn. **87**, 014001 (2018).

[11] 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).

[12] M. Ruggeri, S. Moroni, and M. Holzmann, Nonlinear Network Description for Many-Body Quantum Systems in Continuous Space, Phys. Rev. Lett. **120**, 205302 (2018).

[13] G. Carleo, Y. Nomura, and M. Imada, Constructing exact representations of quantum many-body systems with deep neural networks, Nat. Commun. **9**, 5322 (2018).

[14] S. Czischek, M. Gärttner, and T. Gasenzer, Quenches near Ising quantum criticality as a challenge for artificial neural networks, Phys. Rev. B **98**, 024311 (2018).

[15] H. Saito, Method to solve quantum few-body problems with artificial neural networks, J. Phys. Soc. Jpn. **87**, 074002 (2018).

[16] Y. Levine, O. Sharir, N. Cohen, and A. Shashua, Quantum Entanglement in Deep Learning Architectures, Phys. Rev. Lett. **122**, 065301 (2019).

[17] K. Choo, G. Carleo, N. Regnault, and T. Neupert, Symmetries and Many-Body Excitations with Neural-Network Quantum States, Phys. Rev. Lett. **121**, 167204 (2018).

[18] X. Liang, W.-Y. Liu, P.-Z. Lin, G.-C. Guo, Y.-S. Zhang, and L. He, Solving frustrated quantum many-particle models with convolutional neural networks, Phys. Rev. B **98**, 104426 (2018).

[19] D. Luo and B. K. Clark, Backflow Transformations via Neural Networks for Quantum Many-Body Wave Functions, Phys. Rev. Lett. **122**, 226401 (2019).

[20] S. Lu, X. Gao, and L.-M. Duan, Efficient representation of topologically ordered states with restricted Boltzmann machines, Phys. Rev. B **99**, 155136 (2019).

[21] K. McBrian, G. Carleo, and E. Khatami, Ground state phase diagram of the one-dimensional Bose-Hubbard model from restricted Boltzmann machines, J. Phys.: Conf. Ser. **1290**, 012005 (2019).

[22] K. Choo, T. Neupert, and G. Carleo, Two-dimensional frustrated J_1-J_2 model studied with neural network quantum states, Phys. Rev. B **100**, 125124 (2019).

[23] H.-Q. Shi, X.-Y. Sun, and D.-F. Zeng, Neural-network quantum state of transverse-field Ising model, Commun. Theor. Phys. **71**, 1379 (2019).

[24] G. Torlai, B. Timar, E. P. L. van Nieuwenburg, H. Levine, A. Omran, A. Keesling, H. Bernien, M. Greiner, V. Vuletić, M. D. Lukin, R. G. Melko, and M. Endres, Integrating Neural Networks with a Quantum Simulator for State Reconstruction, Phys. Rev. Lett. **123**, 230504 (2019).

[25] O. Sharir, Y. Levine, N. Wies, G. Carleo, and A. Shashua, Deep Autoregressive Models for the Efficient Variational Simulation of Many-Body Quantum Systems, Phys. Rev. Lett. **124**, 020503 (2020).

[26] F. Ferrari, F. Becca, and J. Carrasquilla, Neural Gutzwiller-projected variational wave functions, Phys. Rev. B **100**, 125131 (2019).

[27] D. Sehayek, A. Golubeva, M. S. Albergo, B. Kulchytskyy, G. Torlai, and R. G. Melko, Learnability scaling of quantum states: Restricted Boltzmann machines, Phys. Rev. B **100**, 195125 (2019).

[28] G. Torlai and R. G. Melko, Latent Space Purification via Neural Density Operators, Phys. Rev. Lett. **120**, 240503 (2018).

[29] N. Yoshioka and R. Hamazaki, Constructing neural stationary states for open quantum many-body systems, Phys. Rev. B **99**, 214306 (2019).

[30] A. Nagy and V. Savona, Variational Quantum Monte Carlo Method with a Neural-Network Ansatz for Open Quantum Systems, Phys. Rev. Lett. **122**, 250501 (2019).

[31] M. J. Hartmann and G. Carleo, Neural-Network Approach to Dissipative Quantum Many-Body Dynamics, Phys. Rev. Lett. **122**, 250502 (2019).

[32] F. Vicentini, A. Biella, N. Regnault, and C. Ciuti, Variational Neural-Network Ansatz for Steady States in Open Quantum Systems, Phys. Rev. Lett. **122**, 250503 (2019).

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

[34] F. Verstraete, J. J. García-Ripoll, and J. I. Cirac, Matrix Product Density Operators: Simulation of Finite-Temperature and Dissipative Systems, Phys. Rev. Lett. **93**, 207204 (2004).

[35] M. Zwolak and G. Vidal, Mixed-State Dynamics in one-Dimensional Quantum Lattice Systems: A Time-Dependent Superoperator Renormalization Algorithm, Phys. Rev. Lett. **93**, 207205 (2004).

[36] A. E. Feiguin and S. R. White, Finite-temperature density matrix renormalization using an enlarged Hilbert space, Phys. Rev. B **72**, 220401(R) (2005).

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

[38] D. B. M. Dickerscheid, D. van Oosten, P. J. H. Denteneer, and H. T. C. Stoof, Ultracold atoms in optical lattices, Phys. Rev. B **68**, 043623 (2003).

[39] Although we restrict ourselves to the real-valued CNN, we may extend it to the complex-valued CNN to represent complex-valued $\langle \boldsymbol{n}|e^{-\beta\hat{H}}|\boldsymbol{n}'\rangle$. In this case, we need to use a complex-valued activation function, such as a generalized ReLU function used in Ref. [22], and the squared error $[\rho(\boldsymbol{n},\boldsymbol{n}')-\rho_{\text{target}}(\boldsymbol{n},\boldsymbol{n}')]^2$ in Eqs. (A1) and (A5) should be replaced with $|\rho(\boldsymbol{n},\boldsymbol{n}')-\rho_{\text{target}}(\boldsymbol{n},\boldsymbol{n}')|^2$.

[40] D. P. Kingma and J. L. Ba, Adam: A method for stochastic optimization, *Proceedings of the 3rd International Conference on Learning Representations, San Diego*, 2015, arXiv:1412.6980.

[41] In real time, it takes several days to obtain each line in Fig. 2 using our computer with four Tesla V100 GPUs.