Rapid Communications

Deep learning-enhanced variational Monte Carlo method for quantum many-body physics

Li Yang,[1,*] Zhaoqi Leng,[2] Guangyuan Yu,[3] Ankit Patel,[4,5] Wen-Jun Hu,[1,†] and Han Pu[1,‡]

[1]*Department of Physics and Astronomy & Rice Center for Quantum Materials, Rice University, Houston, Texas 77005, USA*
[2]*Department of Physics, Princeton University, Princeton, New Jersey 08544, USA*
[3]*Department of Physics and Astronomy & The Center for Theoretical Biological Physics, Rice University, Houston, Texas 77005, USA*
[4]*Department of Electrical and Computer Engineering, Rice University, Houston, Texas 77005, USA*
[5]*Department of Neuroscience, Baylor College of Medicine, Houston, Texas 77030, USA*

(Received 26 May 2019; accepted 12 December 2019; published 14 February 2020)

Artificial neural networks have been successfully incorporated into the variational Monte Carlo method (VMC) to study quantum many-body systems. However, there have been few systematic studies exploring quantum many-body physics using deep neural networks (DNNs), despite the tremendous success enjoyed by DNNs in many other areas in recent years. One main challenge of implementing DNNs in VMC is the inefficiency of optimizing such networks with a large number of parameters. We introduce an importance sampling gradient optimization (ISGO) algorithm, which significantly improves the computational speed of training DNNs by VMC. We design an efficient convolutional DNN architecture to compute the ground state of a one-dimensional SU(N) spin chain. Our numerical results of the ground-state energies with up to 16 layers of DNNs show excellent agreement with the Bethe ansatz exact solution. Furthermore, we also calculate the loop correlation function using the wave function obtained. Our work demonstrates the feasibility and advantages of applying DNNs to numerical quantum many-body calculations.

DOI: 10.1103/PhysRevResearch.2.012039

Introduction. Over the past few years, artificial neural networks have been introduced to study quantum many-body systems [1–8]. In their seminal paper [1], Carleo and Troyer proposed to represent the quantum many-body states by a restricted Boltzmann machine (RBM), which contains one visible and one hidden layer. The many-body wave function is represented by a visible layer after integrating out the hidden layer. The parameters in the RBM are trained by the variational Monte Carlo (VMC) method. Following this work, the RBM and a few other networks have been applied to study several quantum many-body systems with good accuracy [1–11]. So far, the networks that have been implemented in quantum physics studies are not deep and hence are not powerful enough to represent more complicated many-body states. As a result, there has been no clear evidence that their performance far exceeds the more traditional state-of-the-art numerical algorithms such as quantum Monte Carlo, density matrix renormalization group, or tensor networks, to name a few. To overcome this problem, deep neural networks (DNNs) have been suggested. Theoretical studies have shown that the DNNs can efficiently represent any tensor network states and most quantum many-body states, and possess distinct advantages over shallow networks [12–14]. In fact, DNN-based deep learning has become the most successful model of many machine learning tasks and has dominated the field since 2012. DNNs have been demonstrated to have a comparable or superior performance in various tasks when compared to human experts, such as playing Atari games [15], Go [16,17], manipulating robots [18,19], etc., and have led to rapid advances in artificial intelligence.

Despite great interest, there have been relatively few works in applying DNNs to quantum many-body computations [20,21]. This perhaps is due to the fact that applying DNNs to represent quantum many-body states faces two main challenges: inefficient optimization and insufficient information for the proper choice of DNN architectures. The former arises because a DNN typically contains a large number of parameters to train, while a proper choice of the architecture often requires physical insights about the nature of the quantum systems.

In this Rapid Communication, we propose an efficient convolutional DNN architecture to represent the ground state of quantum many-body systems. Most of the quantum systems consist of particles interacting with each other through a finite range. Such a local interacting character can be ideally captured by a convolutional neural network (CNN). We have developed an importance sampling gradient optimization (ISGO) algorithm within the VMC method, which significantly improves the optimization speed and hence enables us to utilize DNN architectures. Our method can take advantage of the automatic differentiation, which automatically computes the gradient update via a backward-propagation algorithm [22]. We show that our method can be parallelized and take full advantage of the acceleration provided by graphic

*lyliyang@google.com
†nuaahwj@gmail.com
‡hpu@rice.edu

processing units (GPUs). The ISGO method achieves at least one order of magnitude speed-up when trained on GPUs [23].

For benchmark purposes, we construct DNNs to represent the ground-state wave function of the one-dimensional (1D) SU(N) spin chain, which has an exact solution under the Bethe ansatz. We systematically test different DNN architectures with ISGO for systems with different complexities. Our numerical results for the ground-state energies of a 1D SU(N) spin chain show excellent agreement with the exact solutions. Furthermore, we are able to compute correlation functions which are extremely difficult to obtain by the Bethe ansatz. The convolutional DNN architecture we constructed for this work can be readily generalized to represent the ground states of other quantum many-body systems. The ISGO method can also be used to accelerate the computation based on general VMC methods.

Network architectures. We consider a homogeneous 1D SU(N) spin chain with N_{site} spins, which is the simplest prototypical model with SU(N) symmetry, governed by the Hamiltonian

$$H = \sum_{i=1}^{N_{\text{site}}} P_{i,i+1}, \tag{1}$$

where $P_{i,i+1}$ is the spin exchange operator which exchanges two neighboring spins: $P_{i,i+1}|a_i, b_{i+1}\rangle = |b_i, a_{i+1}\rangle$. This model can describe the behavior of 1D strongly interacting quantum spinor gases [24–27], and has attracted significant attention both experimentally and theoretically [28–35]. Here, we will use the DNN to represent the ground-state wave function of this model.

A general state takes the form

$$|\Psi\rangle = \sum_{\{s_i\}} \Psi\big(s_1, s_2, \ldots, s_{N_{\text{site}}}\big)\big|s_1, s_2, \ldots, s_{N_{\text{site}}}\big\rangle,$$

where each s_i represents one of the N spin states for the SU(N) model. The goal is to build a network that takes the basis state $|\{s_i\}\rangle$ as the input and compute the ground-state wave function $\Psi(\{s_i\})$ such that the energy functional $\langle\Psi|H|\Psi\rangle/\langle\Psi|\Psi\rangle$ is minimized. The first step is to encode the input basis state into a 2D tensor $S_{j,\beta}$, where the first and the second indices j and β represent the spatial site and the local spin state, respectively. In this work, we consider two kinds of state encodings: value encoding, which encodes each spin state into a number, and one-hot encoding, which encodes the spin state into a one-hot Boolean vector. The tensor S is fed into the DNN as an input. The output of the first hidden layer, which follows immediately after the input layer, is given by

$$A^{[1]}_{i,f'} = \sigma\left(\sum_{k=1}^{K}\sum_{f=1}^{N_{\text{in}}} W^{[1]}_{k,f,f'} S_{i+k,f} + b^{[1]}_{f'}\right), \tag{2}$$

where $A^{[1]}$ is the activation (or feature map) of the first hidden layer, $\sigma(x) = \max(x, 0)$ is the rectified linear unit (ReLU) activation function, which has been demonstrated to outperform traditional sigmoid activation function for DNNs [36], and $W^{[1]}$ is a 3D tensor of shape (K, N_{in}, F), where K is the convolution kernel size, F the number of channels of the hidden layer, and N_{in} the number of the channels of the input layer which is 1 for value encoding and N for one-hot encoding. In this work, we use the same K and F, which determines the width of the network, for every hidden layer. $b^{[1]}$ is a bias vector of size F. The output from the remaining hidden layers is given by

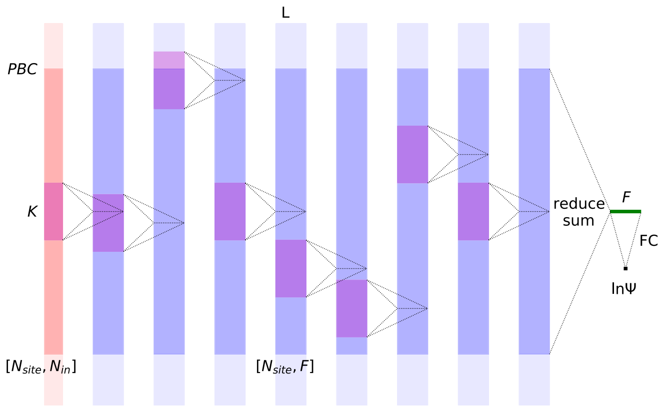

FIG. 1. The architecture of a convolutional DNN with $L = 8$ hidden layers. The input state is encoded into a 2D tensor of shape $[N_{\text{site}}, N_{\text{in}}]$, and fed into the input layer (represented by the leftmost pink rectangle). For value encoding $N_{\text{in}} = 1$ and for one-hot encoding $N_{\text{in}} = N$. The blue rectangles stand for the activation (feature maps) of the hidden layers. Convolution filters (the small pink rectangles) transform one hidden layer to the next one. The last hidden layer (on the right) is reduce summed and followed by a fully connected layer to give the ln Ψ output.

$$A^{[l]}_{i,f'} = \sigma\left(\sum_{k=1}^{K}\sum_{f=1}^{F} W^{[l]}_{k,f,f'} A^{[l-1]}_{i+k,f} + b^{[l]}_{f'}\right), \quad l = 2, 3, \ldots, L, \tag{3}$$

where L is the total number of hidden layers that determine the depth of the network, $W^{[l]}$ is a 3D tensor of shape $[K, F, F]$, and $b^{[l]}$ is a bias vector of size F. After the last hidden layer, its output is summed along the spatial dimension, and followed by a single fully connected layer to give the final output of the network,

$$\ln\Psi(S) = \sum_{f=1}^{F} a_f \sum_{i=1}^{N_{\text{site}}} A^{[L]}_{i,f}, \tag{4}$$

where a is a weight vector of size F. The full structure of the network is illustrated in Fig. 1. Each magenta rectangular object corresponds to a convolutional filter. We use periodic padding for each convolutional layer to enforce the periodic boundary condition. This network is fully convolutional [37], which means the network architecture is compatible with different system sizes, and we can easily do transfer learning. Here, $W^{[l]}$, $b^{[l]}$, and a are the network parameters that need to be optimized. The total number of parameters is roughly KF^2L.

Importance sampling gradient optimization. Before introducing the ISGO method, we first revisit the conventional gradient optimization method in VMC. The wave function $\Psi(\{w\})$ is encoded by the set of network parameters $w \in \{W^{[l]}, b^{[l]}, a\}$. In every iteration step, N_{sample} quantum states following the distribution $P^0_x \propto |\Psi^0_x|^2$ are sampled using a Markov chain. Here, Ψ^0 is the input wave function from

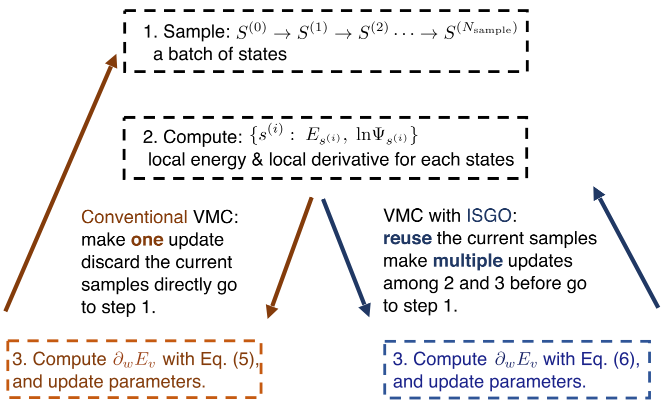
1. Sample: $S^{(0)} \to S^{(1)} \to S^{(2)} \dots \to S^{(N_{\text{sample}})}$
a batch of states
2. Compute: $\{s^{(i)}: E_{s^{(i)}}, \ln\Psi_{s^{(i)}}\}$
local energy & local derivative for each states
Conventional VMC:
make one update
discard the current
samples directly go
to step 1.
VMC with ISGO:
reuse the current samples
make multiple updates
among 2 and 3 before go
to step 1.
3. Compute $\partial_w E_v$ with Eq. (5),
and update parameters.
3. Compute $\partial_w E_v$ with Eq. (6),
and update parameters.

FIG. 2. The flowchart comparing the conventional VMC algorithm and the VMC with ISGO algorithm.

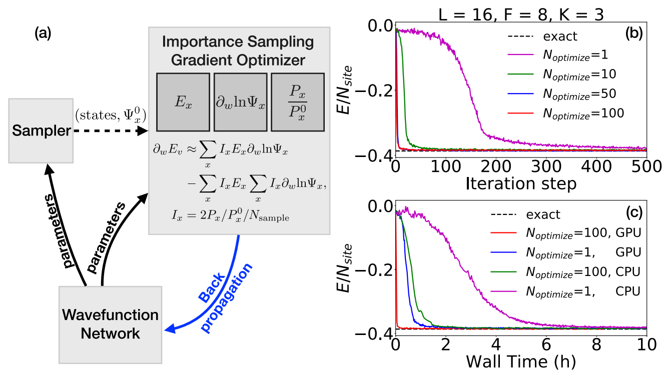

FIG. 3. (a) The flowchart of the ISGO algorithm within the VMC method. A network is used to represent the ground-state wave function. In every iteration step, the sampler generates N_{sample} samples following distribution $P_x^0 \propto \Psi_x^{*0}\Psi_x^0$. Then the importance sampling optimizer updates the network parameters through back propagation in a loop for N_{optimize} times. $N_{\text{optimize}} = 1$ corresponds to the conventional gradient optimization method. The whole process is iterated until convergence. The sampler and the optimizer share the same wave function. We compare the training curves for a $N_{\text{site}} = 60$ SU(2) spin chain using a $(L, F, K) = (16, 8, 3)$ CNN with one-hot encoding. (b) Variational energy vs iteration steps. (c) Variational energy vs wall time on GPU/CPU. The initial learning rate for Adam is 10^{-4}.

the previous step, and x indexes the sampled states. The variational energy functional $E_v(\{w\}) = \frac{\langle\Psi^0(\{w\})|H|\Psi^0(\{w\})\rangle}{\langle\Psi^0(\{w\})|\Psi^0(\{w\})\rangle}$ is then computed. To minimize $E_v(\{w\})$, the network parameters are updated as $w \leftarrow w - \alpha\partial_w E_v$, where the "learning rate" α is a small parameter [38]. In our work, we use Adam [39], a variant of the stochastic gradient descent algorithm. Here, $\partial_w E_v$ is approximated by the variational wave function Ψ^0 with the N_{sample} samples,

$$\partial_w E_v \approx \sum_x I^0 E_x^0 \partial_w \ln\Psi_x^0 - \sum_x I^0 E_x^0 \sum_x I^0 \partial_w \ln\Psi_x^0, \quad (5)$$

where $E_x^0 = \sum_{x'} H_{x,x'}\Psi_{x'}^0/\Psi_x^0$ is the local energy under Ψ^0 and $I^0 = 2/N_{\text{sample}}$. After the parameters w are updated, the wave function changes from Ψ^0 to Ψ which serves as the input for the next iteration step, where a new set of states is sampled based on Ψ and the previously sampled states based on Ψ^0 are discarded.

Inspired by the off-policy policy gradient method in reinforcement learning (RL) [40,41], we develop an efficient importance sampling gradient optimization (ISGO) method that utilizes the mismatched samples, as shown in Fig. 3(a). The key is to renormalize the distribution of those mismatched samples to $|\Psi_x|^2$ by multiplying the local energies and derivatives in Eq. (5) with importance sampling factors,

$$\partial_w E_v \approx \sum_x I_x E_x \partial_w \ln\Psi_x - \sum_x I_x E_x \sum_x I_x \partial_w \ln\Psi_x, \quad (6)$$

where E_x is the local energy under the last updated wave function Ψ, and $\frac{I_x}{I^0} = \frac{P_x}{P_x^0} = \mathcal{C}\frac{|\Psi_x|^2}{|\Psi_x^0|^2}$ with $\mathcal{C}$ the normalization factor which can also be approximated using $\sum_x I_x/I^0 = 1$. The key difference as summarized in Fig. 2, in comparison to the conventional method, is that, within each iteration step, the network parameters w (and hence the wave functions) are updated multiple times. This enables us to use the N_{sample} sampled states much more efficiently. Furthermore, the update procedure can be efficiently parallelized and run on GPUs.

We plot the variational energies versus iteration step and wall time for a 60-site SU(2) chain with a 16-layer CNN in Figs. 3(b) and 3(c) [23]. As can be seen, the ISGO method converges with much fewer samples and much faster GPU than the conventional method. We also implement the ISGO method using TENSORFLOW with autodifferentiation [42], which allows us to try different network architectures much more easily. Our code for both RBM and CNN can be found in Ref. [43]. We emphasize that, although we choose a particular gradient optimization algorithm Adam in our work, the concept of ISGO is general and can be implemented with any other optimization methods.

Numerical results for 1D SU(N) spin chain. We test our DNN on the Sutherland model, the 1D homogeneous SU(N) spin chain governed by Hamiltonian (1). We pick this model for two main reasons. First, the ground-state energy of this model can be exactly solved by the Bethe snsatz [44], which allows us to benchmark our results [23]. Second, the number of spin states N controls the complexity of the system, which allows us to systematically study the efficiency and accuracy of the DNN as the complexity of model grows. Numerical details can be found in Ref. [23].

Figure 4 shows our main results of the ground-state energies for various N on an $N_{\text{site}} = 60$ chain with a DNN with varying depth (i.e., number of layers L) and width (i.e., kernel size K). We tested two encoding methods for the input state. Figures 4(a) and 4(b) correspond to the value encoding, while Figs. 4(c) and 4(d) correspond to the one-hot encoding. The value encoding imposes ordinality, i.e., different spin states are encoded into a number with the average value to be zero. For one-hot encoding, different spin states are encoded into a vector orthogonal to each other, and thus are not ordinal. For example, for an $N = 3$ system, the three spin states are encoded into values of $-1/2$, 0, $1/2$ in value encoding, and into vectors $(1, 0, 0)$, $(0, 1, 0)$, $(0, 0, 1)$ in one-hot encoding. The one-hot encoding requires more computational resources (in terms of both memory and computational time) than the value encoding, and scales linearly with respect to N. However, in general it yields better accuracy than the value

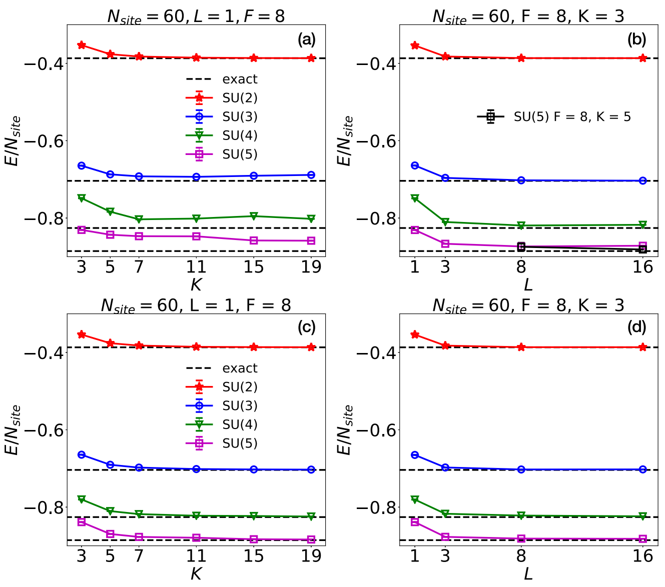

FIG. 4. The ground-state energy for $N_{\text{site}} = 60$ SU(N) spin chains ($N = 2, 3, 4, 5$) using CNN with (a), (b) value encoding and (c), (d) one-hot encoding. (a) and (c) are for one layer $L = 1$ with fixed channel number $F = 8$ and different kernel size K. (b) and (d) are for fixed channel number and kernel size ($F = 8$, $K = 3$) but different number of layers L. In (b), the black squares are for $F = 8$ and $K = 5$. The horizontal black dashed lines are exact results from the Bethe ansatz.

encoding method. This could be due to the fact that one-hot encoding encodes each spin state into a vector which effectively enlarges the dimension of the parameter space. Optimization in such an artificially enlarged space helps to prevent the system from being stuck in the metastable states [45].

Figures 4(a) and 4(c) display the results from a single-layer network with varying width. As one can see, increasing the kernel size K helps bring the ground-state energy closer to the exact solution represented by horizontal dashed lines, which indicates that, for such a shallow network, a large kernel size is necessary for capturing long-range effects mediated by nearest-neighbor interactions. Here, one-hot encoding performs significantly better than value encoding, especially for SU($N > 2$), where, no matter how large the kernel size is, the energies computed via value encoding do not converge to exact solutions. In Figs. 4(b) and 4(d), we fix the width of the network, but vary its depth by adjusting the number of layers L. Even for a relatively small kernel size $K = 3$, increasing L helps to bring the computed ground-state energy closer to the exact result. Therefore, a DNN can capture a long-range effect even with a small kernel size. For the SU(5) model (the largest N we used in the calculation), the energy does not converge to an exact solution using value encoding with a kernel size 3. Simply by increasing the kernel size to 5, we can reduce the computed ground-state energy and match it with the exact solution as the black squares shown in Fig. 4(b). We vary the number of channels F and find that the energy results are not sensitive to F. More details can be found in the Supplemental Material [23].

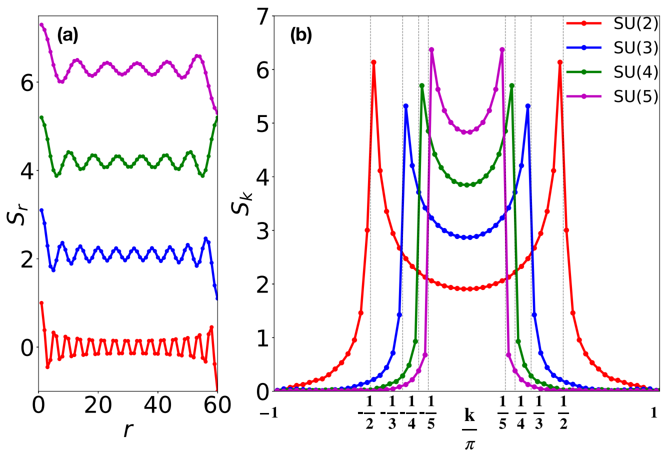

FIG. 5. (a) Real-space loop correlation functions S_r for the SU(N) spin chain with $N = 5, 4, 3, 2$ from top to bottom. (b) The Fourier transform of S_r: $S_k = |\sum_r S_r e^{ikr}|$ with peaks at $k = \pm\pi/N$.

The Bethe ansatz method can yield an energy spectrum and, in principle, the many-body wave function for exactly solvable models such as the one considered here. However, due to the complexity of a general many-body wave function, it remains a tremendous challenge to compute other useful quantities such as the correlation functions. Often, advanced numerical techniques are needed for such tasks [46,47]. To further demonstrate the power of DNN, here we show our results for the loop correlation function,

$$S_{m,n} = (-1)^{m-n}\langle(m\cdots n)\rangle, \tag{7}$$

where the expectation value is taken with respect to the ground state, and $(m\cdots n)$ is the loop permutation operator that permutes the spatial indices in the wave function by $m \to m+1$, $m+1 \to m+2, \ldots, n-1 \to n$, $n \to m$. Physically, this operator puts the spin in the original nth position to the mth position and correspondingly moves the spins at the original ith (with $i = m, \ldots, n-1$) positions to their neighboring position on the right. The loop correlation function appears in the definition of the one-body density matrix of 1D strongly interacting quantum spinor gases whose ground state can be represented by a strong-coupling ansatz wave function [24,26,27,48,49] due to the fact that such wave functions must obey the permutation symmetry rule originating from quantum indistinguishability.

For the homogeneous system we considered here, $S_{m,n}$=S_r with $r \equiv n - m$. We plot S_r for an SU(N) spin chain with $N_{\text{site}} = 60$ spins in Fig. 5(a), and its discrete Fourier transform $S_k = |\sum_r S_r e^{ikr}|$ in Fig. 5(b). S_r and S_k characterize the correlation in the real and the momentum space, respectively. By taking the Jordan-Wigner transformation, an SU(N) spin chain with each spin component having N_{site}/N spins can be mapped to a nearest-neighbor interacting N-component fermionic system with each component having N_{site}/N fermions [49,50]. The peaks of S_k at $k = \pm\pi/N$, that can be clearly seen in Fig. 5(b), correspond to the Fermi points of those fermions. These peaks lead to the singularities of momentum distribution of strongly interacting spinor Fermi gases at the same momentum point [49,50].

Conclusion and outlook. We have constructed a DNN, combined with VMC, to study the ground state of the 1D SU(N) spin chain. The key in our work is the development of the ISGO algorithm, which can be straightforwardly applied to any type of variational wave function, for the optimization procedure. This algorithm allows us to efficiently train the network, and is particularly suitable for training DNNs which typically contain a large number of parameters. Note that the VMC with the ISGO algorithm may be interpreted as an RL process if we identify the Markov-chain state trajectories in the former as the state transitions/policies in the latter. We tested the network to solve the 1D SU(N) spin chain model and systematically investigated the performance of the network by varying its depth and width. We have found that, when using value state encoding, as the complexity of the model increases by increasing N, it is not sufficient just to increase the width (i.e., kernel size) of the network, one needs to add more depth to capture the long-range correlation of the quantum state. We only show numerical results computed by the DNN up to 16 layers. We do not observe any significant benefit by using much deeper networks up to 100 layers on this model. This could be due to a potential problem of vanishing gradients in very deep networks (see Refs. [51,52] and references therein), which may be alleviated via using other network architectures such as ResNet [52], which we leave for future studies. Finally, we note that another key finding from our work is the importance of input state encoding. We find that one-hot encoding, although requiring more computational resources, in general leads to much more accurate results than value encoding.

In conclusion, our study clearly demonstrates that it is feasible to use DNNs to represent quantum many-body wave functions and to significantly enhance the efficiency of numerical quantum many-body computations. Applying machine learning techniques to quantum many-body physics is still a young and emerging field with many open questions. We believe that such investigations will not only benefit quantum physics, but may also help us to gain deeper insights into neural networks.

Acknowledgments. H.P. acknowledges support from NSF and the Robert A. Welch Foundation (Grant No. C-1669), W.-J.H. is supported by the DOE Award No. DE-SC0018197 and the Robert A. Welch Foundation (Grant No. C-1411). The majority of the computational calculations have been performed on the K80 GPUs with 11GB memory of Google Colaboratory.

[1] G. Carleo and M. Troyer, Science **355**, 602 (2017).
[2] Y. Nomura, A. S. Darmawan, Y. Yamaji, and M. Imada, Phys. Rev. B **96**, 205152 (2017).
[3] H. Saito and M. Kato, J. Phys. Soc. Jpn. **87**, 014001 (2018).
[4] Z. Cai and J. Liu, Phys. Rev. B **97**, 035116 (2018).
[5] G. Carleo, Y. Nomura, and M. Imada, Nat. Commun. **9**, 5322 (2018).
[6] X. Liang, W.-Y. Liu, P.-Z. Lin, G.-C. Guo, Y.-S. Zhang, and L. He, Phys. Rev. B **98**, 104426 (2018).
[7] K. Choo, G. Carleo, N. Regnault, and T. Neupert, Phys. Rev. Lett. **121**, 167204 (2018).
[8] K. McBrian, G. Carleo, and E. Khatami, J. Phys.: Conf. Ser. **1290**, 012005 (2019).
[9] G. Torlai, G. Mazzola, J. Carrasquilla, M. Troyer, R. Melko, and G. Carleo, Nat. Phys. **14**, 447 (2018).
[10] I. Glasser, N. Pancotti, M. August, I. D. Rodriguez, and J. I. Cirac, Phys. Revi. X **8**, 011006 (2018).
[11] R. Kaubruegger, L. Pastori, and J. C. Budich, Phys. Rev. B **97**, 195136 (2018).
[12] X. Gao and L.-M. Duan, Nat. Commun. **8**, 662 (2017).
[13] S. R. Clark, J. Phys. A: Math. Theor. **51**, 135301 (2018).
[14] Y. Levine, O. Sharir, N. Cohen, and A. Shashua, Phys. Rev. Lett. **122**, 065301 (2019).
[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, Nature (London) **518**, 529 (2015).
[16] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, Nature (London) **529**, 484 (2016).
[17] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, Nature (London) **550**, 354 (2017).
[18] S. Gu, E. Holly, T. Lillicrap, and S. Levine, arXiv:1610.00633.
[19] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, arXiv:1603.02199.
[20] O. Sharir, Y. Levine, N. Wies, G. Carleo, and A. Shashua, Phys. Rev. Lett. **124**, 020503 (2020).
[21] K. Choo, T. Neupert, and G. Carleo, Phys. Rev. B **100**, 125124 (2019).
[22] R. Hecht-Nielsen, in *Neural Networks for Perception* (Elsevier, Amsterdam, 1992), pp. 65–93.
[23] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevResearch.2.012039 for details of the Bethe ansatz solutions, the unitary transformation, more numerical details and results, and information about comparison of training on GPUs and CPUs.
[24] F. Deuretzbacher, D. Becker, J. Bjerlin, S. M. Reimann, and L. Santos, Phys. Rev. A **90**, 013611 (2014).
[25] A. Volosniev, D. V. Fedorov, A. S. Jensen, M. Valiente, and N. T. Zinner, Nat. Commun. **5**, 5300 (2014).
[26] L. Yang, L. Guan, and H. Pu, Phys. Rev. A **91**, 043634 (2015).
[27] L. Yang and H. Pu, Phys. Rev. A **94**, 033614 (2016).
[28] C. Wu, J.-P. Hu, and S.-C. Zhang, Phys. Rev. Lett. **91**, 186402 (2003).
[29] C. Honerkamp and W. Hofstetter, Phys. Rev. Lett. **92**, 170403 (2004).
[30] A. V. Gorshkov, M. Hermele, V. Gurarie, C. Xu, P. S. Julienne, J. Ye, P. Zoller, E. Demler, M. D. Lukin, and A. Rey, Nat. Phys. **6**, 289 (2010).
[31] S. Taie, R. Yamazaki, S. Sugawa, and Y. Takahashi, Nat. Phys. **8**, 825 (2012).
[32] G. Pagano, M. Mancini, G. Cappellini, P. Lombardi, F. Schäfer, H. Hu, X.-J. Liu, J. Catani, C. Sias, M. Inguscio *et al.*, Nat. Phys. **10**, 198 (2014).
[33] F. Scazza, C. Hofrichter, M. Höfer, P. De Groot, I. Bloch, and S. Fölling, Nat. Phys. **10**, 779 (2014).

[34] X. Zhang, M. Bishof, S. L. Bromley, C. V. Kraus, M. S. Safronova, P. Zoller, A. M. Rey, and J. Ye, Science **345**, 1467 (2014).
[35] C. Hofrichter, L. Riegger, F. Scazza, M. Höfer, D. R. Fernandes, I. Bloch, and S. Fölling, Phys. Rev. X **6**, 021030 (2016).
[36] X. Glorot, A. Bordes, and Y. Bengio, in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, edited by G. Gordon, D. Dunson, and M. Dudík, Proceedings of Machine Learning Research Vol. 15 (PMLR, Fort Lauderdale, FL, 2011), pp. 3–15.
[37] J. Long, E. Shelhamer, and T. Darrell, arXiv:1411.4038.
[38] For stochastic gradient descent (SGD), α is a constant learning rate. For Adam [39], it is an adaptive learning rate depending on the first and second moment of the previous gradients. For stochastic reconfiguration (SR) [53], it is an inverse of the covariance matrix of local derivatives.
[39] D. P. Kingma and J. Ba, arXiv:1412.6980.
[40] N. Meuleau, L. Peshkin, and K.-E. Kim, *Exploration in Gradient-Based Reinforcement Learning* (MIT Press, Cambridge, MA, 2001).
[41] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction* (MIT Press, Cambridge, MA, 2018).
[42] A loss function $\mathcal{L} = \sum_x I_x E_x \ln \Psi_x - \sum_x I_x E_x \sum_x I_x \ln \Psi_x$ can be defined, and tf.stop_gradient() can be applied on $I_x = P_x/P_x^0/N_{\text{sample}}$ and E_x to keep the gradient only flow back through $\ln \Psi_x$.
[43] https://github.com/liyang2019/VMC-ISGO.
[44] B. Sutherland, Phys. Rev. B **12**, 3795 (1975).
[45] C. J. Pickard, Phys. Rev. B **99**, 054102 (2019).
[46] J. Dufour, P. Nataf, and F. Mila, Phys. Rev. B **91**, 174427 (2015).
[47] P. Nataf and F. Mila, Phys. Rev. B **97**, 134420 (2018).
[48] J. Levinsen, P. Massignan, G. M. Bruun, and M. M. Parish, Sci. Adv. **1**, e1500197 (2015).
[49] L. Yang and H. Pu, Phys. Rev. A **95**, 051602(R) (2017).
[50] M. Ogata and H. Shiba, Phys. Rev. B **41**, 2326 (1990).
[51] R. K. Srivastava, K. Greff, and J. Schmidhuber, in *Advances in Neural Information Processing Systems* (MIT Press, Cambrdige, MA, 2015), pp. 2377–2385.
[52] K. He, X. Zhang, S. Ren, and J. Sun, in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE Computer Society, Los Alamitos, CA, 2016), p. 770.
[53] S. Sorella, M. Casula, and D. Rocca, J. Chem. Phys. **127**, 014105 (2007).