Self-learning Monte Carlo with deep neural networks

Huitao Shen,[1,*] Junwei Liu,[1,2,†] and Liang Fu[1]
[1]*Department of Physics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA*
[2]*Department of Physics, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong, China*

(Received 16 January 2018; revised manuscript received 16 May 2018; published 29 May 2018)

The self-learning Monte Carlo (SLMC) method is a general algorithm to speedup MC simulations. Its efficiency has been demonstrated in various systems by introducing an effective model to propose global moves in the configuration space. In this paper, we show that deep neural networks can be naturally incorporated into SLMC, and without any prior knowledge can learn the original model accurately and efficiently. Demonstrated in quantum impurity models, we reduce the complexity for a local update from $O(\beta^2)$ in Hirsch-Fye algorithm to $O(\beta \ln \beta)$, which is a significant speedup especially for systems at low temperatures.

DOI: 10.1103/PhysRevB.97.205140

I. INTRODUCTION

As an unbiased method, Monte Carlo (MC) simulation plays an important role in understanding condensed matter systems. Although great successes have been made in the past several decades, there are still many interesting systems that are practically beyond the capability of conventional MC methods, due to the strong autocorrelation of local updates or due to the heavy computational cost of a single local update. In the midst of recent developments of machine learning techniques in physics [1–17], a general method called self-learning Monte Carlo (SLMC) was introduced to reduce or solve these problems, first in classical statistical mechanics models [18,19], later extended to classical spin-fermion models [20], determinantal quantum Monte Carlo (DQMC) [21], continuous-time quantum Monte Carlo [22,23], and hybrid Monte Carlo [24]. Recently, it helped understand itinerant quantum critical point by setting up a new record of system size in DQMC simulations [25].

Designed under the philosophy of "first learn, then earn," the central ingredient of SLMC is an effective model that is trained to resemble the dynamics of the original model. The advantage of SLMC is twofold. First, simulating the effective model is much faster, which enables the machine to propose global moves to accelerate MC simulations on the original model. Second, the effective model can directly reveal the underlying physics, such as the RKKY interaction in the double-exchange model [20] and the localized spin-spin imaginary-time correlation [23]. We note that there have been many previous works incorporating effective potentials or proposing various kinds of global moves to improve Monte Carlo simulation efficiency [26–30].

The efficiency of SLMC depends on the accuracy of the effective model, which is usually invented based on the human understanding of the original system [18,20–23,25]. To further extend SLMC to complex systems where an accurate effective model is difficult to write down, in this work we employ deep neural networks (DNNs) as effective models in SLMC. Instead of treating neural networks as black boxes with a huge number of parameters and training them blindly, we show how to design highly efficient neural networks that respect the symmetry of the system, with very few parameters yet capturing the dynamics of the original model quantitatively. The generality of this approach is guaranteed by the mathematical fact that DNNs are able to accurately approximate any continuous functions given enough fitting parameters [31,32]. Practically, our DNNs can be trained with ease using back-propagation-based algorithms [33], and can be directly evaluated in dedicated hardware [34]. Compared with other machine learning models in SLMC such as the restricted Boltzmann machine [19,24], which can be regarded as a fully connected neural network with one hidden layer, our DNNs have greater expressibility (more hidden layers) and flexibility (respecting the symmetry of the model).

As a concrete example, we demonstrate SLMC with DNNs on quantum impurity models. In the following, we first review SLMC for fermion systems. We then implement the simplest neural networks and test their performances. Next, we show how the visualization of these networks helps design a more sophisticated convolutional neural network that is more accurate and efficient. Finally, we discuss the complexity of our algorithm.

II. SELF-LEARNING MONTE CARLO FOR FERMIONS

For an interacting fermion system, the partition function is given by $Z = \mathrm{Tr}[e^{-\beta \hat{H}_f}]$, where $\beta = 1/T$ is the inverse temperature, and the trace is over the grand-canonical ensemble. One often applies the Trotter decomposition $e^{-\beta \hat{H}_f} = \prod_{i=1}^{L} e^{-\Delta\tau \hat{H}_f}$, $\Delta\tau = \beta/L$, the Hubbard-Stratonovich transformation $\mathrm{Tr}[e^{-\Delta\tau \hat{H}_f}] = \sum_{s^j=\pm 1}^{N} \mathrm{Tr}[e^{-\Delta\tau \hat{H}[s^j]}]$, and then integrates out fermions. We denote s_i^j as the jth auxiliary Ising spin on the ith imaginary time slice. At this stage, the partition function is written purely on the auxiliary Ising spin degrees

*huitao@mit.edu
†liuj@ust.hk

of freedom $\mathcal{S} \equiv \{s_i^j\}$ [35–37]:

$$Z = \sum_{\mathcal{S}} \det\left[I + \prod_{i=1}^{L} e^{-\Delta\tau \hat{H}[s_i]}\right] \equiv \sum_{\mathcal{S}} W[\mathcal{S}]. \quad (1)$$

The Monte Carlo sampling is in the configuration space of $\mathcal{S}$. The probability p of accepting a move, for example in the Metropolis-Hastings algorithm, is the weight ratio between two configurations: $p(\mathcal{S}_1 \to \mathcal{S}_2) = \min(1, W[\mathcal{S}_2]/W[\mathcal{S}_1])$. Generally, one must evaluate the determinant in Eq. (1), which is time consuming.

The idea of SLMC is to introduce an effective model $H_{\text{eff}}[\mathcal{S};\alpha]$ that depends on some trainable parameters α. We would like to optimize α so that $W_{\text{eff}}[\mathcal{S};\alpha] \equiv e^{-\beta H_{\text{eff}}[\mathcal{S};\alpha]}$ and $W[\mathcal{S}]$ are as close as possible. More formally, we would like to minimize the mean-squared error (MSE) of the logarithmic weight difference:

$$\min_{\alpha} \mathbb{E}_{\mathcal{S}\sim W[\mathcal{S}]/Z}(\ln W_{\text{eff}}[\mathcal{S};\alpha] - \ln W[\mathcal{S}])^2. \quad (2)$$

The rationale of minimizing this error will be discussed shortly later. Following the maximum likelihood principle, in practice one could minimize the MSE on a given data set called the training set $\{\mathcal{S}, W[\mathcal{S}]\}$, which is obtained from the Monte Carlo simulation of the original model. This training set is of small size compared with that of the whole configuration space, but is considered typical mimicking the distribution of the original model because it is generated by the importance sampling. Importantly, the training data taken from the Markov chain should be independently distributed in order for the maximum likelihood estimation to work well.

One then uses the trained effective model to propose global moves. Starting from a given configuration $\mathcal{S}_1$, one first performs standard Monte Carlo simulations on the effective model $\mathcal{S}_1 \to \mathcal{S}_2 \to \cdots \to \mathcal{S}_n$. Configuration $\mathcal{S}_n$ is accepted by the original Markov chain with the probability [18,20]

$$p(\mathcal{S}_1 \to \mathcal{S}_n) = \min\left(1, \frac{W[\mathcal{S}_n]}{W[\mathcal{S}_1]} \frac{W_{\text{eff}}[\mathcal{S}_1;\alpha]}{W_{\text{eff}}[\mathcal{S}_n;\alpha]}\right). \quad (3)$$

As proven in the Supplemental Material [38], the acceptance rate $\langle p\rangle$, defined as the expectation of configuration acceptation probability p defined in Eq. (3), is directly related to the MSE in Eq. (2) as $\langle(\ln p)^2\rangle = \text{MSE}$. This means MSE serves as a very good estimation of the acceptance rate. Indeed, we will see in the following that these two quantities correspond with each other very well.

The acceleration of SLMC can be analyzed as follows. Denote the computational cost of computing $W[\mathcal{S}]$ and $W_{\text{eff}}[\mathcal{S};\alpha]$ given $\mathcal{S}$ as T and T_{eff}, and the autocorrelation time of a measurement without SLMC as τ. Suppose $T \geqslant T_{\text{eff}}$, one can always to make enough ($\gtrsim\tau$) updates in the effective model so that the proposed configuration $\mathcal{S}_n$ is uncorrelated with $\mathcal{S}_1$. In this way, to obtain two independent configurations without or with SLMC, it takes time $O(\tau T)$ and $O[(\tau T_{\text{eff}} + T)/\langle p\rangle]$. If simulating the effective model is efficient $\tau T_{\text{eff}} \ll T$, as demonstrated by cases studied in Refs. [20,21], the acceleration is of order $\langle p\rangle\tau$.

In principle, there is no limitation on the functional form of $H_{\text{eff}}[\mathcal{S};\alpha]$. In the following, we choose to construct $H_{\text{eff}}[\mathcal{S};\alpha]$ using neural networks of different architectures. To be concrete, we study the asymmetric Anderson model with a single impurity [39]

$$\hat{H} = \hat{H}_0 + \hat{H}_1, \quad (4)$$

$$\hat{H}_0 = \sum_{\mathbf{k}} \varepsilon_{\mathbf{k}} \hat{c}_{\mathbf{k}}^\dagger \hat{c}_{\mathbf{k}} + V \sum_{\mathbf{k}\sigma} (\hat{c}_{\mathbf{k}}^\dagger \hat{d}_\sigma + \text{H.c.}) + \mu \hat{n}_d, \quad (5)$$

$$\hat{H}_1 = U\left(\hat{n}_{d,\uparrow} - \frac{1}{2}\right)\left(\hat{n}_{d,\downarrow} - \frac{1}{2}\right), \quad (6)$$

where $\hat{d}_\sigma$ and $\hat{c}_k$ are the fermion annihilation operator for the impurity and for the conduction electrons respectively. $\hat{n}_{d,\sigma} \equiv \hat{d}_\sigma^\dagger \hat{d}_\sigma$, $\hat{n}_d = \sum_{\sigma=\uparrow/\downarrow} \hat{n}_{d,\sigma}$. With different fillings, this model hosts very different low-temperature behaviors identified by the three regimes: local moment, mixed valence, and empty orbital [40]. The Hubbard-Stratonovich transformation on the impurity site is (up to a constant factor)

$$e^{-\Delta\tau \hat{H}_1} = \frac{1}{2} \sum_{s=\pm1} e^{\lambda s(\hat{n}_{d,\uparrow} - \hat{n}_{d,\downarrow})}, \quad (7)$$

with $\cosh\lambda = e^{\Delta\tau U/2}$. In total there are L auxiliary spins, one at each imaginary time slice denoted as a vector $\mathbf{s} \equiv \mathcal{S}$. For impurity problems, one may integrate out the continuous band of conducting electrons explicitly and update with Hirsch-Fye algorithm [41,42]. In the following, the conduction band is assumed to have semicircular density of states $\rho(\varepsilon) = 2\sqrt{1-(\varepsilon/D)^2}/(\pi D)$. The half-bandwidth $D = 1$ is set to be the energy unit.

III. FULLY CONNECTED NEURAL NETWORKS

To gain some insight on how to design the neural network as the effective model, we first implement the simplest neural network, which consists of several fully connected layers. Its structure is shown schematically in the inset of Fig. 1. The effect of the ith fully connected layer can be summarized as $\mathbf{a}_{i+1} = f_i(W_i\mathbf{a}_i + \mathbf{b}_i)$, where $\mathbf{a}_i$ is the input/output vector of the ith/$(i-1)$th layer. W_i, $\mathbf{b}_i$, f_i are the weight matrix, bias vector, and the nonlinear activation function of such layer. The layer is said to have N_i neurons when W_i is of size $N_i \times N_{i-1}$. This structure as the variational wave function in quantum many-body systems has recently been studied extensively [43–52].

We take the auxiliary Ising spin as the input vector $\mathbf{s} \equiv \mathbf{a}_1$. It is propagated through two nonlinear hidden layers with N_1 and N_2 neurons, and then a linear output layer. The output is a number represents the corresponding weight $\ln W_{\text{eff}}[\mathbf{s}]$. We trained the fully connected neural networks of different architectures and in different physical regimes. The details on the networks and training can be found in the Supplemental Material [38].

As shown in Fig. 1, the trained neural network resembles the original model very well. It retains high acceptance rates (>70%) steadily throughout all the parameter regimes. In addition, $e^{-\sqrt{\text{MSE}}}$ indeed shares the same trend with the acceptance rate $\langle p\rangle$. This suggests that to compare different effective models, one can directly compare the MSE instead of computing the acceptance rate every time.

To extract more features from neural networks, we visualize the weight matrix of the first layers in Fig. 2. The most

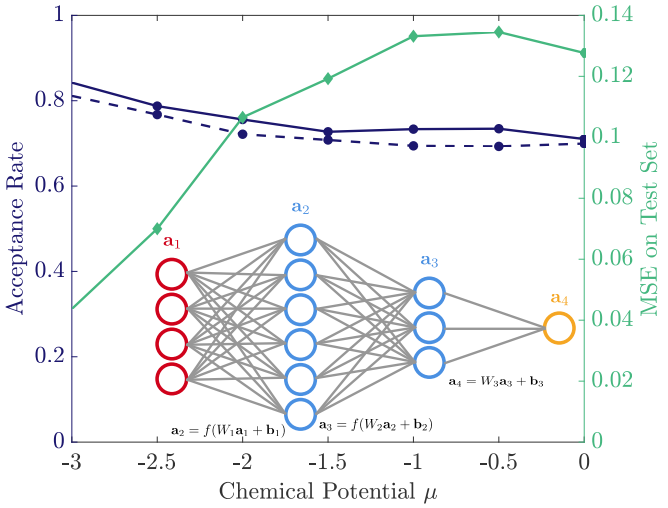

FIG. 1. The performance of the effective model at different chemical potentials. The dashed blue line is the estimation of acceptance rates through MSE computed in a test data set [38]: $e^{-\sqrt{\mathrm{MSE}}}$. The discrepancy between the solid and the dashed line is due to the fact that in general $\langle p\rangle \neq e^{-\sqrt{\langle(\ln p)^2\rangle}} = e^{-\sqrt{\mathrm{MSE}}}$ and the effective model is biased. Here $\beta = 20, U = 3.0, V = 1.0$, and $L = 120$. The number of neurons in the first and second hidden layers are set as $N_1 = 100$ and $N_2 = 50$. The activation function is the rectified linear unit $f(x) = \max\{x, 0\}$. Inset: A schematic show of the fully connected neural network. The red circles represent neurons in the input layer with $L = 4$, and the blue circles represent neurons in two hidden layers with $N_1 = 6$ and $N_2 = 3$. The last layer is a linear output layer.

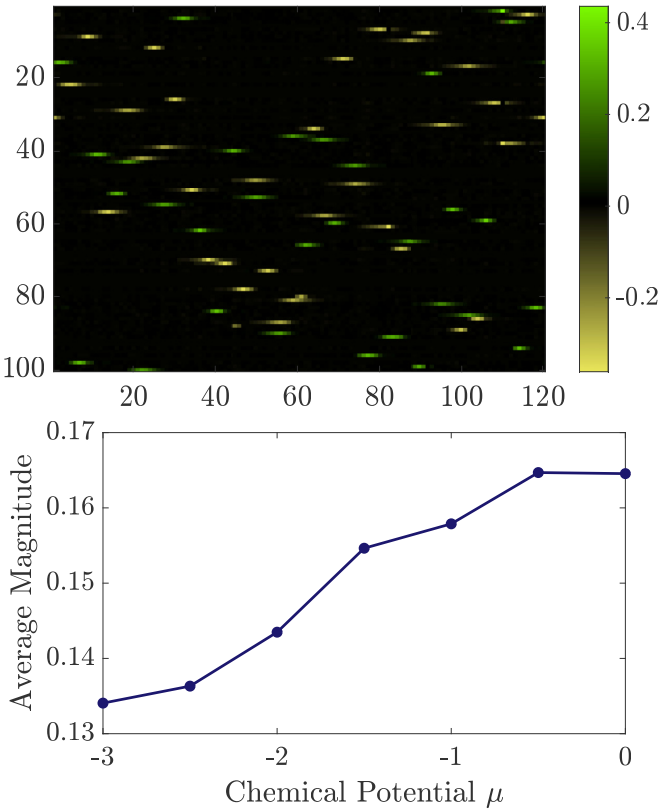

FIG. 2. Top: Weight matrix W_1 taken from the fully connected neural network of $\mu = 0$ in Fig. 1. Bottom: Average magnitude of nonzero matrix elements in W_1 of fully connected neural networks in Fig. 1. The nonzero matrix element is defined as the element that is greater than 10% of the maximum element in all seven weight matrices W_1 from networks trained at seven different chemical potentials. Only around 3% elements are nonzero in these weight matrices.

striking feature is its sparsity. Although the network is fully connected by construction, most of the weight matrix elements vanish after the training, and thus the network is essentially sparsely connected. Clearly, even without any prior knowledge of the system, and using only field configurations and their corresponding energies, the neural network can actually learn that the correlation between auxiliary spins is short ranged in imaginary time.

Weight matrices in neural networks of different chemical potentials look similar. The main difference lies in the magnitude of the matrix elements. Shown in Fig. 2, the neural network could capture the relative effective interaction strength. When the chemical potential moves away from half-filling, less occupation on the impurity site $\langle n_d\rangle$ leads to a weaker coupling between the auxiliary spins and the impurity electrons, according to the Hubbard-Stratonovich transformation Eq. (7). This further causes the decrease of interaction between the auxiliary spins induced by conducting electrons.

We end this section by briefly discussing the complexity of fully connected neural networks. The forward propagation of the spin configuration in the fully connected network involves three matrix-vector multiplications. Each multiplication takes computational cost $O(L^2)$, with L the number of imaginary-time slices that is usually proportional to βU. Thus the running time for a local update in the effective model is $T_{\mathrm{eff}} = O(L^2)$. In the Hirsch-Fye algorithm, each local update also takes time $T = O(L^2)$ [41,42]. Since $T_{\mathrm{eff}} = T$, SLMC based on fully connected networks has no advantage in speed over the original Hirsch-Fye algorithm. In the next section, we will show that by taking advantage of the sparse connection found in the neural network, we can reduce the complexity and make the acceleration possible.

IV. CONVOLUTIONAL NEURAL NETWORKS

The sparsity found in the previous section inspires us to design a neural network that is sparsely connected by construction. Moreover, it is known physically that the interaction between the auxiliary spins in imaginary time is translationally invariant, i.e., only depends on $|\tau_i - \tau_j|$. A neural network that has both properties is known as the "one-dimensional convolutional neural network" [33]. Instead of doing matrix multiplications as in fully connected networks, convolutional networks produce their output by sliding inner product denoted by $*$: $a_{i+1} = f_i(a_i * h_i + b_i)$ (Fig. 3). h_i and b_i are called the kernel and the bias of such layer. A detailed mathematical description of such networks can be found in the Supplemental Material [38].

The key parameters in convolutional neural networks are the number and size of kernels and the stride of the sliding inner product. The setup of these parameters can be guided from the fully connected neural networks: The number and the size of kernels is determined according to the pattern of weight

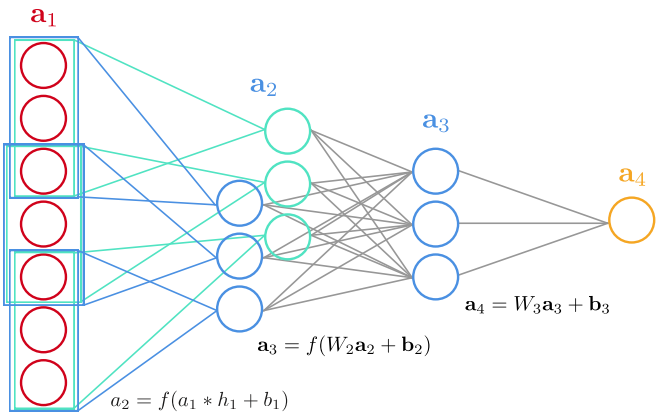

FIG. 3. The structure of the convolutional neural network. The first layer is a convolutional layer with two kernels (dark and light blue) of size 3. The stride of the sliding inner product is 2. The second layer is a fully connected layer and the last layer is a linear layer.

matrices in the fully connected neural networks, and the stride could be chosen to be half of the kernel size to avoid missing local correlations. For example, for the model whose weight matrix is shown in Fig. 2, we could choose two kernels of size 9 for the first convolutional layer with a stride 3. Then several convolutional layers designed in the same spirit are stacked until the size of the output is small enough. Finally, one adds a fully connected layer to produce the final output.

Compared with the fully connected network, the convolutional network has much fewer trainable parameters, which explicitly reduces the redundancy in the parametrization. The fully connected networks in Fig. 2 typically have 10^5 trainable parameters, while the convolutional networks have only 10^2—smaller by three orders of magnitude.

The performance of convolutional networks are shown in Fig. 4. The measured fermion imaginary-time Green's function is shown in the Supplemental Material [38]. Interestingly, the acceptance rate of global moves proposed by convolutional networks are sometimes even higher than those proposed by fully connected networks. In principle, fully connected networks with more parameters have greater expressibility. However, for this specific Anderson model, the parametrization of the effective model has a lot of redundancy, as the auxiliary spin interactions are local and are translationally invariant. The convolutional networks reduce this redundancy by construction, and are easier to train potentially due to the smaller parameter space.

The fewer parameters not only make the network easier to train, but also faster to evaluate. Each slide inner product has the computational cost $O(L)$. It is important to notice that the strides of the sliding inner product are greater than one so that the dimensions of intermediate outputs keep decreasing. In this way, the number of the convolutional layers is of order $O(\ln L)$ because of the large stride. The final fully connected layer is small. Propagating through such layer only costs a constant computational time that is insensitive to L. To summarize, each local update on the effective model has complexity $T_{\text{eff}} = O(L \ln L)$, while that in Hirsch-Fye algorithm is $T = O(L^2)$. Since the autocorrelation time for the desired observable is at least of order $\tau = \Omega(L)$ in order for every auxiliary spin to

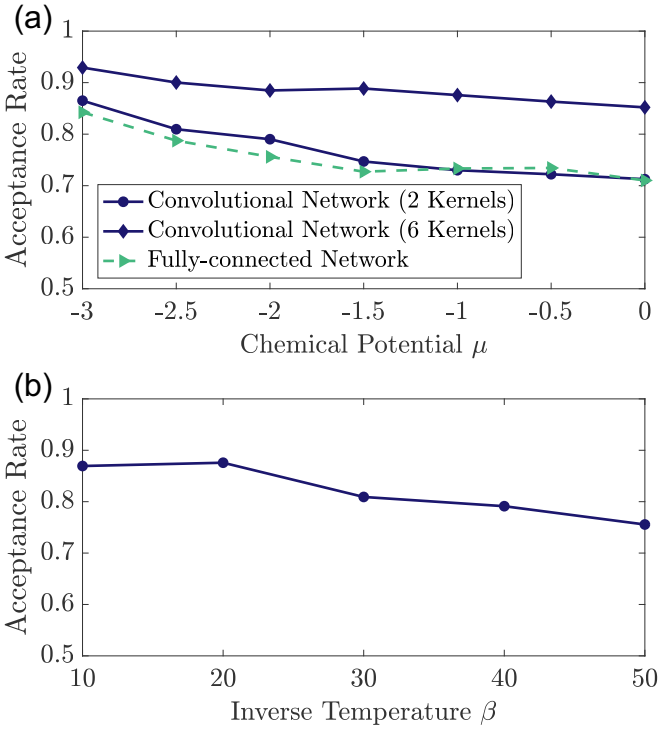

FIG. 4. (a) The performance of the convolutional network compared with the fully connected network in Fig. 1. The number of trainable parameters is 211 (two kernels in the first convolutional layer) or 291 (six kernels in the first convolutional layer). Here $\beta = 20, U = 3.0, V = 1.0$, and $L = 120$. (b) The performance of the convolutional network at different temperatures. Here $U = 3.0, \mu = -1.0, V = 1.0$, and $L = 2\beta U$. The conventional neural network details are described in Supplemental Material [38].

be updated once, i.e. $\tau T_{\text{eff}} \geqslant T$, the acceleration with respect to the original Hirsch-Fye algorithm is then $\tau T/(\tau T_{\text{eff}} + T) \approx T/T_{\text{eff}}$, of order $\langle p \rangle L/\ln L$. It is especially significant for large L. This efficiency allows us to train effective models at very low temperatures very effectively [Fig. 4(b)], whereas training a fully connected network is very costly, if possible at all.

V. CONCLUSION

In this paper, we showed how to integrate neural networks into the framework of SLMC. Both the architecture of the networks and the way we design these networks are general and not restricted to impurity models. This work can help design neural networks as effective models in more complicated systems, thereby introducing the state-of-the-art deep learning hardware into the field of computational physics.

Particularly for impurity models, we demonstrated that the complexity of the convolutional network for a local update is improved to $O(L \ln L)$. We note that there exist continuous-time Monte Carlo algorithms that generally outperform the discrete-time Hirsch-Fye algorithm [53,54]. Although similar self-learning approaches have already been implemented in these systems [22,23], designing an accurate effective model in these continuous-time algorithms is not straightforward as the size of the field configuration keeps changing during the simulation. Looking forward, there have already been

attempts introducing machine learning into dynamical mean-field theory (DMFT) [55,56]. It will be interesting to accelerate DMFT simulation by integrating SLMC into their impurity solvers [57]. Moreover, it is worthwhile to develop more advanced network architectures beyond convolutional networks, e.g., networks that are invariant under permutations of the input [58]. We leave all these attempts for future work.

ACKNOWLEDGMENTS

H.S. thanks Bo Zeng for helpful discussions. This work is supported by DOE Office of Basic Energy Sciences, Division of Materials Sciences and Engineering under Award DE-SC0010526. J.L. is supported by the start-up funds from HKUST. L.F. is partly supported by the David and Lucile Packard Foundation.

[1] J. Carrasquilla and R. G. Melko, Nature Phys. **13**, 431 (2017).
[2] L. Wang, Phys. Rev. B **94**, 195105 (2016).
[3] A. Tanaka and A. Tomiya, J. Phys. Soc. Jpn. **86**, 063001 (2017).
[4] T. Ohtsuki and T. Ohtsuki, J. Phys. Soc. Jpn. **85**, 123706 (2016).
[5] E. P. L. van Nieuwenburg, Y.-H. Liu, and S. D. Huber, Nature Phys. **13**, 435 (2017).
[6] Y. Zhang and E.-A. Kim, Phys. Rev. Lett. **118**, 216401 (2017).
[7] K. Mills, M. Spanner, and I. Tamblyn, Phys. Rev. A **96**, 042113 (2017).
[8] S. J. Wetzel, Phys. Rev. E **96**, 022140 (2017).
[9] W. Hu, R. R. P. Singh, and R. T. Scalettar, Phys. Rev. E **95**, 062122 (2017).
[10] F. Schindler, N. Regnault, and T. Neupert, Phys. Rev. B **95**, 245134 (2017).
[11] S. Lu, S. Huang, K. Li, J. Li, J. Chen, D. Lu, Z. Ji, Y. Shen, D. Zhou, and B. Zeng, arXiv:1705.01523.
[12] M. Cristoforetti, G. Jurman, A. I. Nardelli, and C. Furlanello, arXiv:1705.09524.
[13] C. Wang and H. Zhai, Phys. Rev. B **96**, 144432 (2017).
[14] P. Zhang, H. Shen, and H. Zhai, Phys. Rev. Lett. **120**, 066401 (2018).
[15] W.-J. Rao, Z. Li, Q. Zhu, M. Luo, and X. Wan, Phys. Rev. B **97**, 094207 (2018).
[16] N. Yoshioka, Y. Akagi, and H. Katsura, Phys. Rev. B **97**, 205110 (2018).
[17] P. Huembeli, A. Dauphin, and P. Wittek, Phys. Rev. B **97**, 134109 (2018).
[18] J. Liu, Y. Qi, Z. Y. Meng, and L. Fu, Phys. Rev. B **95**, 041101 (2017).
[19] L. Huang and L. Wang, Phys. Rev. B **95**, 035105 (2017).
[20] J. Liu, H. Shen, Y. Qi, Z. Y. Meng, and L. Fu, Phys. Rev. B **95**, 241104 (2017).
[21] X. Y. Xu, Y. Qi, J. Liu, L. Fu, and Z. Y. Meng, Phys. Rev. B **96**, 041119 (2017).
[22] L. Huang, Y.-F. Yang, and L. Wang, Phys. Rev. E **95**, 031301(R) (2017).
[23] Y. Nagai, H. Shen, Y. Qi, J. Liu, and L. Fu, Phys. Rev. B **96**, 161102 (2017).
[24] A. Tanaka and A. Tomiya, arXiv:1712.03893.
[25] Z. H. Liu, X. Y. Xu, Y. Qi, K. Sun, and Z. Y. Meng, arXiv:1706.10004.
[26] S. Baroni and S. Moroni, Phys. Rev. Lett. **82**, 4745 (1999).
[27] C. Pierleoni and D. M. Ceperley, Chem. Phys. Chem. **6**, 1872 (2005).
[28] O. F. Syljuåsen and A. W. Sandvik, Phys. Rev. E **66**, 046701 (2002).
[29] V. G. Rousseau, Phys. Rev. E **78**, 056707 (2008).
[30] G. Carleo, F. Becca, S. Moroni, and S. Baroni, Phys. Rev. E **82**, 046710 (2010).
[31] G. Cybenko, Math. Control. Signals, Syst. **2**, 303 (1989).
[32] K. Hornik, Neural Networks **4**, 251 (1991).
[33] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, Cambridge, 2016).
[34] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P.-l. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami *et al.*, in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ISCA '17 (ACM, New York, 2017), pp. 1–12.
[35] R. Blankenbecler, D. J. Scalapino, and R. L. Sugar, Phys. Rev. D **24**, 2278 (1981).
[36] J. E. Hirsch, Phys. Rev. B **31**, 4403 (1985).
[37] S. R. White, D. J. Scalapino, R. L. Sugar, E. Y. Loh, J. E. Gubernatis, and R. T. Scalettar, Phys. Rev. B **40**, 506 (1989).
[38] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevB.97.205140 for (i) proof of the relation between the acceptance rate and the training MSE; (ii) basics of neural networks; (iii) details of the neural network structure and the training hyperparameters; and (iv) measured fermion imaginary-time Green's function.
[39] P. W. Anderson, Phys. Rev. **124**, 41 (1961).
[40] F. D. M. Haldane, Phys. Rev. Lett. **40**, 416 (1978).
[41] J. E. Hirsch and R. M. Fye, Phys. Rev. Lett. **56**, 2521 (1986).
[42] R. M. Fye and J. E. Hirsch, Phys. Rev. B **38**, 433 (1988).
[43] G. Carleo and M. Troyer, Science **355**, 602 (2017).
[44] J. Chen, S. Cheng, H. Xie, L. Wang, and T. Xiang, Phys. Rev. B **97**, 085104 (2018).
[45] D.-L. Deng, X. Li, and S. Das Sarma, Phys. Rev. X **7**, 021021 (2017).
[46] X. Gao and L.-M. Duan, Nature Commun. **8**, 662 (2017).
[47] Y. Huang and J. E. Moore, arXiv:1701.06246.
[48] G. Torlai, G. Mazzola, J. Carrasquilla, M. Troyer, R. Melko, and G. Carleo, Nature Phys. **14**, 447 (2018).
[49] H. Saito and M. Kato, J. Phys. Soc. Jpn. **87**, 014001 (2018).
[50] Y. Nomura, A. S. Darmawan, Y. Yamaji, and M. Imada, Phys. Rev. B **96**, 205152 (2017).
[51] S. R. Clark, J. Phys. A Math. Theor. **51**, 135301 (2018).
[52] I. Glasser, N. Pancotti, M. August, I. D. Rodriguez, and J. I. Cirac, Phys. Rev. X **8**, 011006 (2018).
[53] E. Gull, P. Werner, A. Millis, and M. Troyer, Phys. Rev. B **76**, 235123 (2007).
[54] E. Gull, A. J. Millis, A. I. Lichtenstein, A. N. Rubtsov, M. Troyer, and P. Werner, Rev. Mod. Phys. **83**, 349 (2011).

[55] L.-F. Arsenault, A. Lopez-Bezanilla, O. A. von Lilienfeld, and A. J. Millis, Phys. Rev. B **90**, 155136 (2014).

[56] L.-F. Arsenault, O. A. von Lilienfeld, and A. J. Millis, arXiv:1506.08858.

[57] A. Georges, G. Kotliar, W. Krauth, and M. J. Rozenberg, Rev. Mod. Phys. **68**, 13 (1996).

[58] R. Kondor, H. T. Son, H. Pan, B. Anderson, and S. Trivedi, arXiv:1801.02144.