Chin. Phys. Lett. **39**, 050701 (2022)

Network-Initialized Monte Carlo Based on Generative Neural Networks

Hongyu Lu(卢虹宇)[1], Chuhao Li(李楚豪)[2,3], Bin-Bin Chen(陈斌斌)[1], Wei Li(李伟)[4,5*], Yang Qi(戚扬)[6,7*], and Zi Yang Meng(孟子杨)[1*]

[1]*Department of Physics and HKU-UCAS Joint Institute of Theoretical and Computational Physics, The University of Hong Kong, Hong Kong, China*
[2]*Beijing National Laboratory for Condensed Matter Physics, and Institute of Physics, Chinese Academy of Sciences, Beijing 100190, China*
[3]*School of Physical Sciences, University of Chinese Academy of Sciences, Beijing 100190, China*
[4]*Institute of Theoretical Physics, Chinese Academy of Sciences, Beijing 100190, China*
[5]*School of Physics, Beihang University, Beijing 100191, China*
[6]*State Key Laboratory of Surface Physics, Fudan University, Shanghai 200438, China*
[7]*Center for Field Theory and Particle Physics, Department of Physics, Fudan University, Shanghai 200433, China*

(Received 16 February 2022; accepted 11 April 2022; published online 29 April 2022)

We design generative neural networks that generate Monte Carlo configurations with complete absence of autocorrelation from which only short Markov chains are needed before making measurements for physical observables, irrespective of the system locating at the classical critical point, fermionic Mott insulator, Dirac semimetal, or quantum critical point. We further propose a network-initialized Monte Carlo scheme based on such neural networks, which provides independent samplings and can accelerate the Monte Carlo simulations by significantly reducing the thermalization process. We demonstrate the performance of our approach on the two-dimensional Ising and fermion Hubbard models, expect that it can systematically speed up the Monte Carlo simulations especially for the very challenging many-electron problems.

DOI: 10.1088/0256-307X/39/5/050701

Monte Carlo (MC) method is a widely used numerical method to investigate problems in statistical and quantum many-body physics. Though generally polynomial when the sign problem is absent, the complexity of MC simulation can still be prohibitively high for certain challenging problems. In particular, quantum MC simulations of generic interacting fermion systems,[1–5] such as the Hubbard model for correlated electrons,[2,3] the Holstein model for electron-phonon interaction,[6–9] the spin-fermion model of non-Fermi liquid,[5,10–13] and the momentum space QMC for quantum moiré systems,[14–17] to name a few, have very high complexity that scales at least to $\sim\beta N^3$ with $\beta = 1/T$ being the inverse temperature and the $N = L^d$ the total site number for a lattice model in d dimensions. Such type of MC approach for the many-electron problems is termed as determinant quantum Monte Carlo (DQMC) where the complexity comes from the matrix operation on the fermion determinant. The situation worsens when dealing with critical points like the novel quantum critical points ubiquitously present in the metal-insulator transition, pseudogap and non-Fermi liquids, and dynamics and transport properties in high-temperature superconductivity, etc. This can largely be ascribed to that the widely used Markov-chain Monte Carlo (MCMC) method with local updates (or the Metropolis algorithm) suffers from very long Markov-chain autocorrelation that scales with the system size to a high power due to the critical slowing down.

Although there exist powerful and highly efficient cluster/loop update schemes for classical problems such as the Ising model[18,19] and quantum spin/boson systems such as Heisenberg and Bose–Hubbard models,[20–23] these ingeniously designed algorithms do not exist for generic quantum many-body problems, especially for the interacting fermion systems, for example, in DQMC. Therefore, efforts for more efficient algorithms that could overcome these difficulties and boost the (quantum) MC simulations for the aforementioned systems are called for.

In recent years, the application of machine learning technique has achieved notable success in the studies of systems ranging from statistical models to condensed matter and quantum materials research.[24–38] The previous attempts of learning effective low-energy Hamiltonian that could help with accelerating the (quantum) MC sampling process, dubbed self-learning Monte Carlo scheme have shown the power of ideas from machine learning in improving the MC simula-

*Corresponding author. Email: w.li@itp.ac.cn; qiyang@fudan.edu.cn; zymeng@hku.hk
© 2022 Chinese Physical Society and IOP Publishing Ltd

tions of interacting fermion systems and inspired many extensions,[8−12,39−47] which are, however, still technically limited and have not enjoyed the developments of neural networks. On the other hand, studies of (generative) neural network models have shown interesting results such as normalizing flows[48−50] and autoregressive models,[51−53] and some networks have been successfully applied to assist MC simulations.[54−58]

Inspired by these recent developments, we design generative neural networks to approximate the distribution of MC configurations, and find that the network can provide configurations with complete absence of autocorrelation, thus serving as a good starting point for MCMC with significantly shortened Markov chain. We demonstrate the powerful fitting ability of the neural networks by successfully generating configurations from which the correct physical observables can be directly measured after a short Markov chain, as showcased in both the classical $d = 2$ Ising model and quantum critical point of the $d = 2$ fermion Hubbard model. In particular, the latter cannot be mapped into a classical model with short-range interactions due to the existence of gapless Dirac fermions, but can still be successfully dealt with by the network. This suggests that our approach can be applied to quantum many-body systems, in particular the interacting fermion models.

Moreover, we design a network-initialized Monte Carlo (NIMC) scheme with the assistance of such neural networks, which "heals" the initial bias due to the unthermalized training set of the neural networks and thus provides very accurate results. We find that the NIMC scheme with independent samplings can accelerate simulations by reducing the long thermalization time and measurement processes, which is especially important for simulating systems with long autocorrelation time in traditional MCMC.

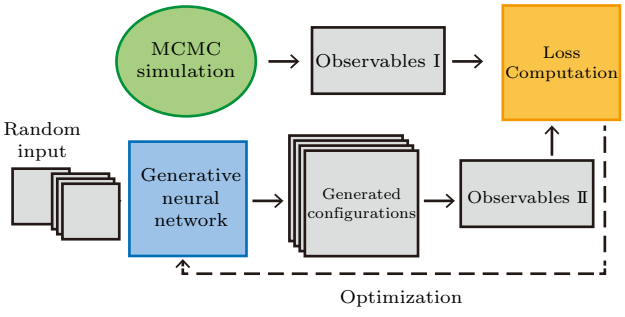

Fig. 1. Flow diagram for training the generative neural networks in this study. Observables I and observables II refer to observables measured from training samples and from generated configurations, respectively. The other terms are as explained in the main text and the loss functions are defined in Eqs. (2) and (4).

Models. In this work, we aim to train neural networks to generate MC configurations that approach the distribution of the original Hamiltonian using convolutional and transposed convolutional architectures. Different from the reference work such as autoregressive models and referring to the idea of the Generator in Generative Adversarial Nets (GAN),[59] we use random configurations (each element of the configurations is randomly set to be ± 1 with equal probability) as input, which, after operations inside the networks, turn into the output of MC configurations distributed according to the physical Hamiltonian. The difference of our approach from GAN is that we do not need to build any discriminator, but directly compare the physical observables, like the internal energy and magnetization (or magnetic structure factor) measured from a batch of configurations generated by the neural network with those obtained from MC simulations to optimize the parameters inside neural networks, as demonstrated in the flow diagram for training in Fig. 1.

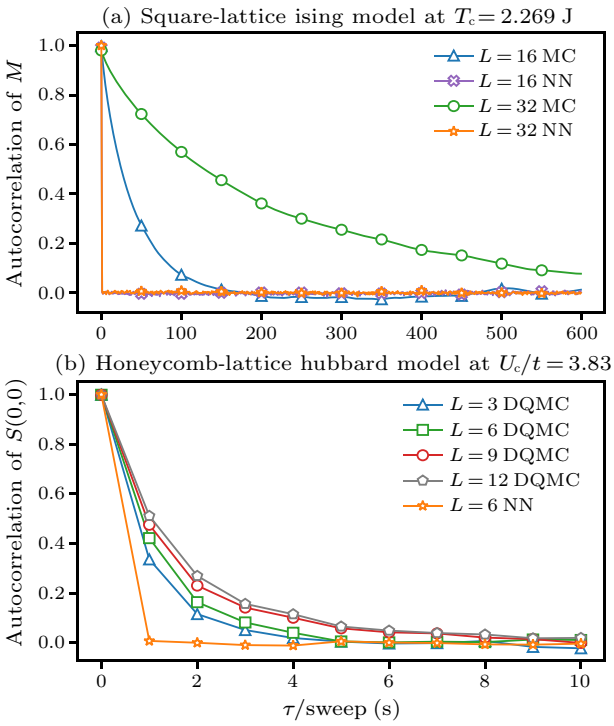

Fig. 2. Comparison of autocorrelation functions (here NN represents configurations generated from neural networks and (DQ)MC refers to samplings from Markov chains). (a) The autocorrelation of magnetization M for the 2D Ising model from both MCMC and NN generated results at critical point $T_c = 2.269J$ for $L = 16, 32$. The MC results exhibit the typical critical slowing down whereas the NN results have zero autocorrelation, i.e., direct *i.i.d.* sampling. (b) The autocorrelation for $S(0, 0)$ of the Hubbard model on honeycomb lattice at the Gross–Neveu QCP ($U_c/t = 3.83$) from DQMC with $L = 3, 6, 9, 12$, the critical slowing down manifested. The NN results again show no sign of autocorrelation.

For the 2D square lattice Ising model with

$$H = -J \sum_{\langle i,j \rangle} \sigma_i \sigma_j, \tag{1}$$

we study the system at the critical point ($T_c \approx 2.269J$) with system sizes of 16×16 and 32×32. We design the loss function directly related to the physical observables, i.e.,

$$\begin{aligned}\text{Loss}(\boldsymbol{G}_l; M_l, E_l) &= w_1 \sum_l [|M(\boldsymbol{G}_l)| - |M_l|]^2 \\ &+ w_2 \sum_l [M^2(\boldsymbol{G}_l) - M_l^2]^2 + w_3 \sum_l [E(\boldsymbol{G}_l) - E_l]^2, \quad (2)\end{aligned}$$

where $\boldsymbol{G}_l$ is the lth generated configuration, M_l and E_l refer to the magnetization $\frac{1}{N}|\sum_j \sigma_j|$ and the energy density $\frac{1}{N}\langle H\rangle$ measured from the corresponding lth MCMC configuration, and w_1, w_2, and w_3 are constants to balance each part of the loss to be in the same magnitude.

For the 2D Hubbard model with

$$\begin{aligned}H = &-t \sum_{\langle i,j\rangle,\sigma} (c_{i,\sigma}^\dagger c_{j,\sigma} + \text{h.c.}) \\ &+ U \sum_i \left(n_{i,\uparrow} - \frac{1}{2}\right)\left(n_{i,\downarrow} - \frac{1}{2}\right), \quad (3)\end{aligned}$$

we consider the half-filled honeycomb lattices where it experiences a chiral Heisenberg Gross–Neveu quantum critical point at $U_c/t \approx 3.83$ between the Dirac semimetal and the antiferromagnetic Mott insulator.[60–64] Here the configuration is made of the auxiliary fields used in the discrete Hubbard–Stratonovich transformation to decouple the fermion interaction terms in DQMC, so the configurational space is of the size $L \times L \times \beta$ compared with the $L \times L$ for the Ising case (see detailed explanation in Section I of Supplemental Materials (SM), which include detailed description of lattice models, determinantal quantum Monte Carlo for fermion Hubbard model and architecture, and optimization of neural networks). We then design the loss function as

$$\begin{aligned}\text{Loss}(\boldsymbol{G}_l; S(\boldsymbol{Q})_l, E_{k_l}) = &\, w_1 \sum_l [S(\boldsymbol{Q})(\boldsymbol{G}_l) - S(\boldsymbol{Q})_l]^2 \\ &+ w_2 \sum_l [E_k(\boldsymbol{G}_l) - E_{k_l}]^2. \quad (4)\end{aligned}$$

Here, $\boldsymbol{G}_l$ is the lth generated configuration, $S(\boldsymbol{Q}) = \frac{1}{N}\sum_{ij} e^{-i\boldsymbol{Q}\cdot(\boldsymbol{r}_i - \boldsymbol{r}_j)}\langle s_i^z s_j^z\rangle$, where $s_i^z = \frac{1}{2}(n_{i,\uparrow} - n_{i,\downarrow})$ refer to the magnetic structure factor [so $S(\boldsymbol{Q})_l$ and $S(\boldsymbol{Q})(\boldsymbol{G}_l)$ refer to the structure factor measured from the lth training sample and from the lth generated configuration] and $S[\Gamma = (0,0)]$ is chosen for honeycomb lattice as the $\Gamma = (0,0)$ is the ordered wave vector for the antiferromagnetic long-range order. E_k is the kinetic energy $\frac{1}{N}\sum_{\langle i,j\rangle,\sigma}\langle(c_{i,\sigma}^\dagger c_{j,\sigma} + \text{h.c.})\rangle$, with constants w_1 and w_2.

The design of the loss functions comes from physical intuition, and we choose the key physical observables such as the energy and magnetic structure factor, which reflect the phase transition of our interests in the loss function. Similar considerations can be generalized to other problems. In practice, our networks are all based on TensorFlow,[65] and we complement with more details of the Ising and Hubbard models in Section I and those of neural networks in Section II of SM.

Autocorrelation Analysis. In MCMC, the autocorrelation time, associated with the particular update scheme, can be extremely long in the physically interesting parameter regime like, e.g., near the classical and quantum critical points. And it is even worse when one performs the finite size analysis as the autocorrelation time usually increases with system size with a high power.[8,18,41] It therefore benefits a lot that the independent and direct samplings from the neural networks are completely free of autocorrelation. We demonstrate these behaviors quantitatively by measuring the autocorrelation function (defined in Section I.3 of SM) of magnetization for the 2D Ising model at $T_c = 2.269J$ in Fig. 2(a), and of $S(0,0)$ for the honeycomb-lattice Hubbard model at the Gross–Neveu QCP ($U_c/t = 3.83$) in Fig. 2(b), between the results generated by neural networks and MCMC results. It can be observed that, for the MCMC the autocorrelation functions decay slowly in Monte Carlo steps and the autocorrelation time for classical (quantum) critical points increases with the system size, but in results from neural networks, such autocorrelation has been completely eliminated.

NIMC Algorithm Based on Neural Networks. We now introduce the NIMC algorithm. In order to demonstrate its ability of correcting the error in the training sample, we no longer train the networks to approximate well-thermalized MC results, but take unthermalized MC configurations generated from short Markov chains. After training the neural networks and generate independent configurations, we add MCMC updates starting from these generated configurations (one Markov chain for each configuration) (For direct measurement on the generated configurations, we map each generated number from the range of $(0,1)$ to $(-1,1)$ and the relationship is: numberformeasurement = generatednumber $\times 2 - 1$. For MCMC update, we use Bernoulli distribution to discretize each element with the initial generated number as the possibility to take the value 1, and map every 0 to -1). We find that the NIMC results quickly converge to the well-thermalized results. The whole process is summarized in Algorithm 1.

Algorithm 1: NIMC scheme based on neural networks.

(1) Run a short Markov-chain (unthermalized) as training samples.

(2) Train a generative neural network using observables from training samples.

(3) Generate configurations from the trained neu-

ral network.

(4) Start short MCMC steps from the generated *i.i.d.* configurations and make measurements.

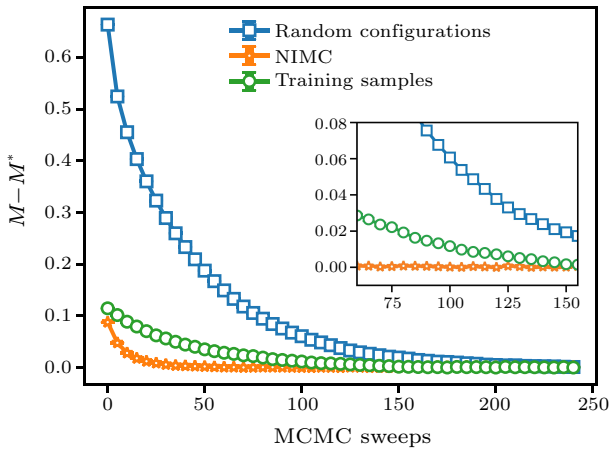

Fig. 3. Comparing the convergence time of magnetization (MCMC sweeps needed to converge) between NIMC and the two MCMC simulations of random configurations and configurations used for training the neural network, respectively. We set $M^* = 0.713$ for reference. For all three sets (each consisted of 1000 configurations), we run 50 iterations from the beginning to derive the expectations with standard errors. The inset further highlights the superiority of NIMC and the scale on y-axis is not changed.

Table 1. Numerical results of NIMC on the square lattice Ising model ($L = 16$, $T_c = 2.269J$). The four sets of data respectively refer to training samples (unthermalized), neural network generated results, NIMC results (30 more MCMC sweeps), and well-thermalized MC results. Results shown here are expectations with standard errors from 1000 values (and good MC takes 1000 bins).

Metric	MC train	NN	NIMC	Good MC
$\|M\|$	0.600(7)	0.606(3)	0.713(6)	0.713(3)
E	−1.373(6)	−1.329(4)	−1.454(5)	−1.453(2)

Numerical Experiments on the Ising Model. In the standard MCMC, the Markov chains must first be well thermalized such that it converges to the correct distribution, and then generate enough bins of statistically uncorrelated configurations to yield expectation values of observables with satisfactory statistical error, which requires the thermalization and each bin to be longer than the system-size-dependent autocorrelation time. To demonstrate that NIMC largely reduces the thermalization process, we compare the performance of NIMC on the 16×16 Ising model at critical point, with those starting MCMC simulations from random configurations and configurations used as training samples in NIMC, by computing the MCMC sweeps (local update) needed for converged magnetization results ($M^* = 0.713$ for $L = 16$ and $T = T_c$ as reference). We run 50 iterations for all three sets of data with each set consisting of 1000 configurations (for each iteration, we rerun Markov chains from all the configurations), and the results are shown in Fig. 3. For training samples, almost about 150 sweeps are needed to reach a convergence, and for random configurations the number is more than 200. Remarkably, the NIMC scheme greatly lowers the thermalization time and arrive at the convergence after about only 30 sweeps.

To further quantify the comparison, we take the training part for neural networks into consideration, and compute on the same CPU (Intel Xeon E5-2680) that for the 150 epochs of network training that are more than sufficient to converge the parameters in the neural network. It turns out that the training process takes around 125 s (could be made even faster on GPU), and each MCMC sweep for 1000 configurations takes around 2.5 s. That is, NIMC takes $125 + 2.5 \times 30 = 200$ s for the 1000 thermalized and independent configurations here from training samples, while the 1000 samples themselves need up to $2.5 \times 150 = 375$ s for thermalization. Therefore, we claim that NIMC can overall speed up the MC simulations. In addition, the independent NIMC samples also save time in the process of measurements as there is completely no autocorrelation and we can directly measure observables once the thermalization is completed.

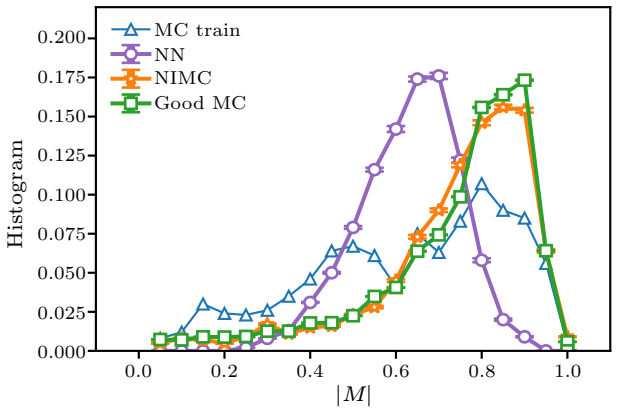

Fig. 4. Histograms of the distribution of magnetization and the abbreviations are the same as those in Table 1. We obtain standard errors from 50 iterations of NIMC (30 more MCMC sweeps and 1000 configurations each) in total with MC train fixed, and the Good MC are from 50 sets (each set consists of 200000 data).

To better compare the accuracy, we then show the observables by taking 1000 configurations from the beginning of a rather short Markov-chain as training samples to run NIMC and the numerical results are shown in Table 1. In addition, we also plot the corresponding histograms of distribution of magnetization in Fig. 4 after 50 iterations of NIMC (1000 configurations for each iteration) from the same training samples. The observables obtained from unthermalized training samples are far worse than those thermalized. After training, the neural network generates configurations with approximately the same ob-

servables but of Gaussian-like distributions. Here, we think that this change of distribution is the reason for the faster convergence for MCMC simulations. After we add 30 MCMC updates on the generated configurations, they quickly converge to the results of the so-called Good MC, which consists of totally 200000 well thermalized MC configurations for reference (then we take 1000 bins with 200 configurations per bin). Moreover, the distribution of NIMC has a large overlap with Good MC. To quantitatively analyze the overlap of distribution, we compute the percentage overlap (%OL) between two histograms,[57] i.e., $(\%\mathrm{OL}(P_r, P_\theta) = \sum_i \min[P_r(i), P_\theta(i)]$, where P_r and P_θ corresponds to the distributions to be compared and we divide them into 20 bins. Again we count the NIMC samples from 50 iterations for statistical expectation with standard error. The result is $\%\mathrm{OL} = 94.78(18)$, which reconfirms that the distribution of NIMC with just 1000 configurations has a large overlap with that of Good MC.

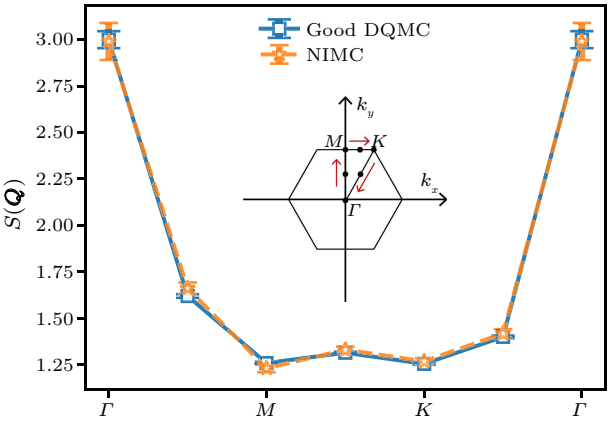

Fig. 5. Comparison of magnetic structure factor $S(\boldsymbol{Q})$ between Good DQMC (10000 samples) and NIMC (15 sweeps), which are not used for network training except $S(\Gamma)$. The black dots in the inset are the momenta measured along the high-symmetry path (with red arrows).

Table 2. Numerical results of NIMC on the honeycomb lattice Hubbard model at $(U_c/t = 3.83, L = 6)$. The three sets of data respectively refers to training samples (unthermalized), NIMC results, and well-thermalized DQMC results. Results shown here are expectations with standard errors from 1000 values (Good DQMC takes 1000 bins).

Metric	DQMC train	NIMC	Good DQMC
E_k	−1.360(4)	−1.357(3)	−1.358(1)
$S(\Gamma)$	3.11(10)	2.99(10)	3.00(4)

Hubbard Model on the Honeycomb Lattice. Beyond the classical Ising model, here we take the 2D Hubbard model on a honeycomb lattice at its unique Gross–Neveu quantum criticality between the Dirac semimetal and antiferromagnetic Mott insulator $(U_c/t = 3.83, L = 6)$ as an example of interacting fermion systems. We run NIMC with 15 DQMC sweeps from 1000 generated configurations and the mean values with error bars of physical observables are listed in Table 2. The NIMC results again present almost identical expectation values of physical observables as the Good DQMC (containing 10000 configurations in total that are divided into 1000 bins). Furthermore, in Fig. 5, we show the magnetic structure $S(\boldsymbol{Q})$ along the high-symmetry path of the Brillouin zone, which are observables not used in training the networks. We find again excellent agreements between Good MC and NIMC only after 15 sweeps. The peak at the Γ point highlights the quantum critical fluctuations towards the magnetic order that will gap out the Dirac cones. These results confirm that the neural networks with optimizers like ADAM[66] are capable of fixing the large-scale determinant computations. We must clarify that, in NIMC, we do not necessarily need up to 1000 configurations (as we use for example here), which can further reduce the computational cost.

Discussion. By developing an NIMC method based on the generative neural networks, our numerical results support firmly that the latter as independent and direct sampling approach can approximately capture the weight distributions of MC configurations and completely get rid of autocorrelation in the study of classical and quantum many-body systems. NIMC therefore provides a scheme for speeding up the MC simulations in reducing the thermalization time and saving time for direct measurement of observables from those independent NIMC samples. Admittedly, our NIMC still needs a few steps of the MCMC simulation, therefore it does not change the overall scaling of the computational cost but can nevertheless lead to a significant factor reduction.

From the perspective of neural networks, we verify that large-scale simulations, such as DQMC calculations, can be implemented through the evaluation of the neural network loss functions. The next step from here is to use generative neural networks to provide efficient and accurate system-size extrapolations of MC configurations and in this way many more complicated yet fundamental problems, such as the quantum Moiré materials that the DQMC has just been shown to be able to solve[14–17,67–69] but with heavy computational costs, could be improved in NIMC.

Acknowledgments. We thank Zheng Yan, Shangqiang Ning, Bin-bin Mao, Jiarui Zhao, Chengkang Zhou, and Xu Zhang for the enjoyable discussions and happy conversations in No. 16 Pavilion of Lung Fu Shan Country Park. We acknowledge support from the RGC of Hong Kong SAR of China (Grant Nos. 17303019, 17301420, 17301721, and AoE/P-701/20), the National Natural Science Foundation of China (Grant Nos. 11974036, 11874115, and 11834014), the Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No. XDB33000000), and the K. C. Wong Education

Foundation (Grant No. GJTD-2020-01). We thank the Center for Quantum Simulation Sciences in Institute of Physics, Chinese Academy of Sciences, the Computational Initiative at the Faculty of Science and the Information Technology Services at the University of Hong Kong for their technical support and generous allocation of GPU/CPU time. This work was also supported by the Seed Funding "Quantum-Inspired explainable-AI" at the HKU-TCL Joint Research Centre for Artificial Intelligence, Hong Kong.

References

[1] Blankenbecler R, Scalapino D J and Sugar R L 1981 *Phys. Rev. D* **24** 2278
[2] Hirsch J E 1983 *Phys. Rev. B* **28** 4059
[3] Hirsch J E 1985 *Phys. Rev. B* **31** 4403
[4] Assaad F F and Evertz H G 2008 *World-Line and Determinantal Quantum Monte Carlo Methods for Spins, Phonons and Electrons* (Berlin: Springer) p 277
[5] Xu X Y, Liu Z H, Pan G, Qi Y, Sun K and Meng Z Y 2019 *J. Phys.: Condens. Matter* **31** 463001
[6] Scalettar R T, Bickers N E and Scalapino D J 1989 *Phys. Rev. B* **40** 197
[7] Noack R M, Scalapino D J and Scalettar R T 1991 *Phys. Rev. Lett.* **66** 778
[8] Chen C, Xu X Y, Liu J, Batrouni G, Scalettar R and Meng Z Y 2018 *Phys. Rev. B* **98** 041102(R)
[9] Chen C, Xu X Y, Meng Z Y and Hohenadler M 2019 *Phys. Rev. Lett.* **122** 077601
[10] Xu X Y, Sun K, Schattner Y, Berg E and Meng Z Y 2017 *Phys. Rev. X* **7** 031058
[11] Liu Z H, Pan G, Xu X Y, Sun K and Meng Z Y 2019 *Proc. Natl. Acad. Sci. USA* **116** 16760
[12] Jiang W, Liu Y, Klein A, Wang Y, Sun K, Chubukov A V and Meng Z Y 2021 arXiv:2105.03639 [cond-mat.str-el]
[13] Liu Y, Jiang W, Klein A, Wang Y, Sun K, Chubukov A V and Meng Z Y 2022 *Phys. Rev. B* **105** L041111
[14] Zhang X, Pan G, Zhang Y, Kang J and Meng Z Y 2021 *Chin. Phys. Lett.* **38** 077305
[15] Hofmann J S, Khalaf E, Vishwanath A, Berg E and Lee J Y 2021 arXiv:2105.12112 [cond-mat.str-el]
[16] Pan G, Zhang X, Li H, Sun K and Meng Z Y 2022 *Phys. Rev. B* **105** L121110
[17] Zhang X, Sun K, Li H, Pan G and Meng Z Y 2021 arXiv:2111.10018 [cond-mat.str-el]
[18] Swendsen R H and Wang J S 1987 *Phys. Rev. Lett.* **58** 86
[19] Wolff U 1989 *Phys. Rev. Lett.* **62** 361
[20] Sandvik A W 1999 *Phys. Rev. B* **59** R14157
[21] Prokof'ev N V, Svistunov B V and Tupitsyn I S 1998 *J. Exp. Theor. Phys.* **87** 310
[22] Prokof'ev N V, Svistunov B V and Tupitsyn I S 1998 *Phys. Lett. A* **238** 253
[23] Sandvik A W 2010 *AIP Conf. Proc.* **1297** 135
[24] Carleo G, Cirac I, Cranmer K, Daudet L, Schuld M, Tishby N, Vogt-Maranto L and Zdeborová L 2019 *Rev. Mod. Phys.* **91** 045002
[25] Carrasquilla J 2020 *Adv. Phys. X* **5** 1797528
[26] Bedolla E, Padierna L C and Casta P R 2020 *J. Phys.: Condens. Matter* **33** 053001
[27] Ch'ng K, Carrasquilla J, Melko R G and Khatami E 2017 *Phys. Rev. X* **7** 031038
[28] Broecker P, Carrasquilla J, Melko R G and Trebst S 2017 *Sci. Rep.* **7** 8823
[29] Carrasquilla J and Melko R G 2017 *Nat. Phys.* **13** 431
[30] Carleo G, Nomura Y and Imada M 2018 *Nat. Commun.* **9** 5322
[31] Carleo G and Troyer M 2017 *Science* **355** 602
[32] Cai Z and Liu J 2018 *Phys. Rev. B* **97** 035116
[33] Choo K, Carleo G, Regnault N and Neupert T 2018 *Phys. Rev. Lett.* **121** 167204
[34] Cheng S, Wang L, Xiang T and Zhang P 2019 *Phys. Rev. B* **99** 155131
[35] Guo C, Jie Z, Lu W and Poletti D 2018 *Phys. Rev. E* **98** 042114
[36] Han Z Y, Wang J, Fan H, Wang L and Zhang P 2018 *Phys. Rev. X* **8** 031012
[37] Xie H, Zhang L and Wang L 2022 *Journal of Machine Learning* **1** 38
[38] Efthymiou S, Beach M J S and Melko R G 2019 *Phys. Rev. B* **99** 075113
[39] Liu J, Qi Y, Meng Z Y and Fu L 2017 *Phys. Rev. B* **95** 041101(R)
[40] Liu J, Shen H, Qi Y, Meng Z Y and Fu L 2017 *Phys. Rev. B* **95** 241104(R)
[41] Xu X Y, Qi Y, Liu J, Fu L and Meng Z Y 2017 *Phys. Rev. B* **96** 041119(R)
[42] Nagai Y, Shen H, Qi Y, Liu J and Fu L 2017 *Phys. Rev. B* **96** 161102(R)
[43] Huang L and Wang L 2017 *Phys. Rev. B* **95** 035105
[44] Huang L, Yang Y F and Wang L 2017 *Phys. Rev. E* **95** 031301(R)
[45] Endo K, Nakamura T, Fujii K and Yamamoto N 2020 *Phys. Rev. Res.* **2** 043442
[46] Liu Z H, Xu X Y, Qi Y, Sun K and Meng Z Y 2018 *Phys. Rev. B* **98** 045116
[47] Liu Z H, Xu X Y, Qi Y, Sun K and Meng Z Y 2019 *Phys. Rev. B* **99** 085114
[48] Zhang L, E W and Wang L 2018 arXiv:1809.10188 [cs.LG]
[49] Hartnett G S and Mohseni M 2020 arXiv:2001.00585 [cs.LG]
[50] Li S H and Wang L 2018 *Phys. Rev. Lett.* **121** 260601
[51] Wu D, Wang L and Zhang P 2019 *Phys. Rev. Lett.* **122** 080602
[52] Sharir O, Levine Y, Wies N, Carleo G and Shashua A 2020 *Phys. Rev. Lett.* **124** 020503
[53] Liu J G, Mao L, Zhang P and Wang L 2021 *Mach. Learn.: Sci. Technol.* **2** 025011
[54] Alcalde P D and Eremin I M 2020 *Phys. Rev. B* **102** 195148
[55] Albergo M S, Kanwar G and Shanahan P E 2019 *Phys. Rev. D* **100** 034515
[56] McNaughton B, Milošević M V, Perali A and Pilati S 2020 *Phys. Rev. E* **101** 053312
[57] Singh J, Arora V, Gupta V and Scheurer M S 2021 *SciPost Phys.* **11** 043
[58] Wu D, Rossi R and Carleo G 2021 *Phys. Rev. Res.* **3** L042024
[59] Goodfellow I J *et al.* 2014 *NIPS'14: Proceedings of the 27th International Conference on Neural Information Processing Systems* vol 2 pp 2672–2680
[60] Meng Z Y, Lang T C, Wessel S, Assaad F F and Muramatsu A 2010 *Nature* **464** 847
[61] Sorella S, Otsuka Y and Yunoki S 2012 *Sci. Rep.* **2** 992
[62] Assaad F F and Herbut I F 2013 *Phys. Rev. X* **3** 031010
[63] Lang T C and Läuchli A M 2019 *Phys. Rev. Lett.* **123** 137602
[64] Liu Y, Wang W, Sun K and Meng Z Y 2020 *Phys. Rev. B* **101** 064308
[65] Abadi M *et al.* 2016 arXiv:1603.04467 [cs.DC]
[66] Kingma D P and Ba J 2015 arXiv:1412.6980 [cs.LG]
[67] Liao Y D, Xu X Y, Meng Z Y and Kang J 2021 *Chin. Phys. B* **30** 017305
[68] Liao Y D, Kang J, Breiø C N, Xu X Y, Wu H Q, Andersen B M, Fernandes R M and Meng Z Y 2021 *Phys. Rev. X* **11** 011014
[69] Liao Y D, Meng Z Y and Xu X Y 2019 *Phys. Rev. Lett.* **123** 157601

Supplemental Material: Network-Initialized Monte Carlo Based on Generative Neural Networks

Hongyu Lu,[1] Chuhao Li,[2,3] Bin-Bin Chen,[1] Wei Li,[4,5,*] Yang Qi,[6,7,†] and Zi Yang Meng[1,‡]

[1]*Department of Physics and HKU-UCAS Joint Institute of Theoretical and Computational Physics, The University of Hong Kong, Pokfulam Road, Hong Kong SAR, China*
[2]*Beijing National Laboratory for Condensed Matter Physics and Institute of Physics, Chinese Academy of Sciences, Beijing 100190, China*
[3]*School of Physical Sciences, University of Chinese Academy of Sciences, Being 100190, China*
[4]*Institute of Theoretical Physics, Chinese Academy of Sciences, Beijing 100190, China*
[5]*School of Physics, Beihang University, Beijing 100191, China*
[6]*State Key Laboratory of Surface Physics, Fudan University, Shanghai 200438, China*
[7]*Center for Field Theory and Particle Physics, Department of Physics, Fudan University, Shanghai 200433, China*

PHYSICAL MODELS

1. Model Details

First we consider two-dimensional Ising model on square lattice at the critical point $T_c \approx 2.269J$ and the Hamiltonian is shown in Eq.(1) of the main text, which can be solved with classica Monte Carlo simulations. σ_i and σ_j refers to nearest neighbors of classical spins and take values: ± 1. We set the interaction strength J of all pairs of neighbors to be 1.

Next, we consider Hubbard model on the honeycomb lattice, which can be solved with determinant quantum Monte Carlo (DQMC) at half-filling. In Eq.(3) of main text, t is the hopping parameter for the kinetic energy, U is the repulsive Coulomb interaction between electrons on the same lattice site, $\langle i, j\rangle$ represents a pair of nearest-neighbor sites in the lattice, the operators $c^\dagger_{i,\sigma}$ and $c_{i,\sigma}$ are the fermion creation and annihilation operators for fermions with z component of spin-up ($\sigma = \uparrow$) or spin-down ($\sigma = \downarrow$), and the operators $n_{i,\sigma} = c^\dagger_{i,\sigma} c_{i,\sigma}$ are the number operators which count numbers of fermions of spin σ on the site i. In loss function Eq.(4) in the main text, S refers to the structure factor: $S(\boldsymbol{Q}) = \frac{1}{L^2}\sum_{ij} e^{-i\mathbf{Q}\cdot(\mathbf{r}_i - \mathbf{r}_j)}\langle s^z_i s^z_j\rangle$, and the spin operator $\mathbf{s}_i = \frac{1}{2}\sum_{\alpha\alpha'} c^\dagger_{i,\alpha}\overrightarrow{\sigma}_{\alpha\alpha'} c_{i,\alpha'}$. Note here $\overrightarrow{\sigma}$ denotes the Pauli matrices and so the z-component $s^z_i = \frac{1}{2}(n_{i\uparrow} - n_{i\downarrow})$.

2. Determinant Quantum Monte Carlo (DQMC)

The partition function $Z = \mathrm{Tr}\{e^{-\beta H}\}$ is expressed as a path integral by discretizing the inverse temperature β into L_τ slices of length $\Delta\tau$. Then, after Trotter-Suzuki decomposition, the Hamiltonian are separated in each time slice and Z can be written as

$$Z = \mathrm{Tr}\left[\prod_{l=1}^{L_\tau} e^{-\Delta\tau H_k} e^{-\Delta\tau H_U}\right] + O(\Delta\tau^2), \tag{S1}$$

where $H_k = -t\sum_{\langle i,j\rangle,\sigma}(c^\dagger_{i,\sigma} c_{j,\sigma} + h.c.)$ is the kinetic term and $H_U = U\sum_i (n_{i,\uparrow} - \frac{1}{2})(n_{i,\downarrow} - \frac{1}{2})$ is interaction term of the Hubbard model.

To treat the quartic interaction term, the discrete Hubbard-Stratonovich transformation[1-3] can be applied and then $e^{-\Delta\tau H_U} = e^{-U\Delta\tau(n_{i,\uparrow}-\frac{1}{2})(n_{i,\downarrow}-\frac{1}{2})} = \frac{1}{2}e^{-\frac{U\Delta\tau}{4}}\sum_{s_i=\pm1} e^{\nu s_i(n_{i,\uparrow}-n_{i,\downarrow})}$, where the scalar ν is defined by $\cosh\nu = e^{\frac{U\Delta\tau}{2}}$. Putting kinetic part and interaction part together, the term in partition function becomes

$$e^{-\Delta\tau H_k} e^{-\Delta\tau H_U} = \prod_\sigma e^{-\Delta\tau T_\sigma} e^{\sigma\nu s_i n_{i,\sigma}}, \tag{S2}$$

where H_k is rewritten with operator $T_\sigma = -t\sum_{i,j} c^\dagger_{i,\sigma} c_{j,\sigma} + h.c.$. According to the feature of fermion operator[1], a fermion operator ($\hat{M}_l$ for example) with a quadratic form like $\hat{M}_l = \sum_{i,j} c^\dagger_i (M_l)_{ij} c_j$ satisfies

$$\mathrm{Tr}\left[e^{-\hat{M}_1} e^{-\hat{M}_2}\cdots e^{-\hat{M}_L}\right] = \mathrm{Det}[I + e^{-M_1} e^{-M_2}\cdots e^{-M_L}]. \tag{S3}$$

Then the partition function could be finally written in determinant form as

$$Z = \left(\frac{1}{2}e^{-\frac{U\Delta\tau}{4}}\right)^{NL_\tau}\sum_{s_{i,l}}\prod_\sigma \mathrm{Det}\left[I + B^\sigma(L_\tau, L_\tau - 1)\cdots B^\sigma(1,0)\right], \tag{S4}$$

in which

$$B^{\sigma}(l_2, l_1) = \prod_{l=l_1+1}^{l_2} e^{\sigma\nu \mathrm{Diag}\{s_{i,l}\}} e^{-\Delta\tau T}, \tag{S5}$$

where $\sigma = \{1, -1\}$ in calculation corresponding mark $\{\uparrow, \downarrow\}$, and T is the matrix corresponding to the operator T_σ. Thus, we have introduced a sum over the field of auxiliary variables $s_{i,l}$ in a $(d+1)$-dimension space (d for spatial denoted by i and 1 for imaginary time denoted by l) as shown in Fig. S1 (b), and the fermionic degrees of freedom in the quadratic form have been integrated out analytically. Note B^{σ} is an $N \times N$ matrix that depends on the auxiliary configurations.

3. Autocorrelation Function

The autocorrelation function for an observable O is defined as:

$$A_O(t) = \frac{\langle O(i+t)O(i)\rangle - \langle O\rangle^2}{\langle O^2\rangle - \langle O\rangle^2}. \tag{S6}$$

For MCMC, i and t denotes the simulation time, normally in units of the MC sweeps (one sweep means doing flipping attempts over all the spins of the configuration), and the averages are over the reference time i. For Neural Network results, when computing the formal autocorrelation function, i and t simply refers to the serial numbers of the configurations.

NEURAL NETWORK DETAILS

1. Architecture

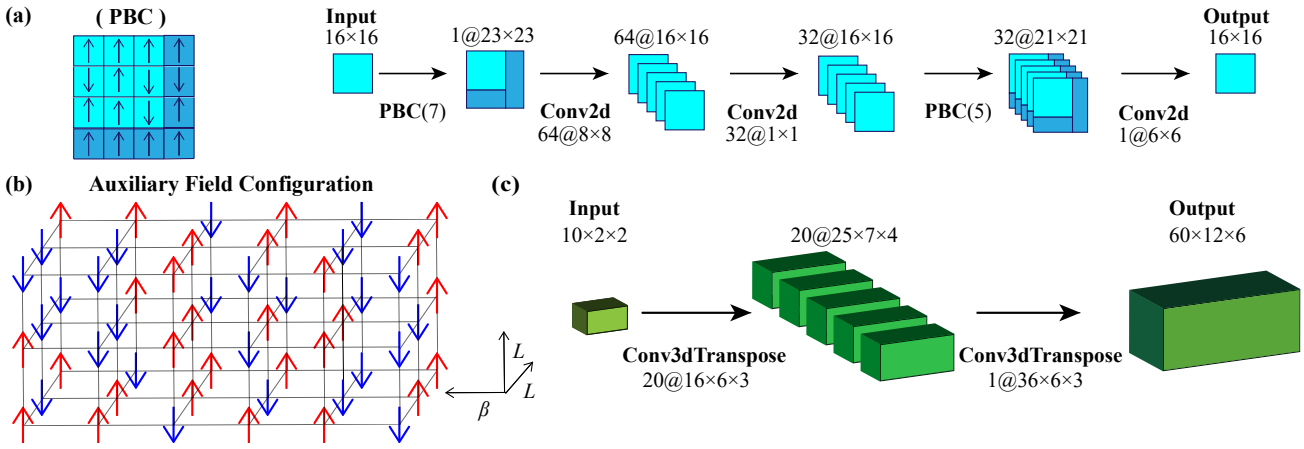

FIG. S1. Schematic figures for the generative neural networks used in this study. (a) shows an example of periodic boundary condition on the left, and we use the neural network structure on the right for the training of the 2d Ising model, where Conv2d stands for 2d convolutional layer. (b) displays the auxiliary field configuration in DQMC for the Hubbard model, with the $(2+1)d$ space-time of $\beta \times L \times L$. The auxiliary fields of ± 1 live on each space-time lattice site. (c) demonstrates the schematic structure of 3d transposed convolutional network we employed for the training of the 2d Hubbard model on $6 \times 6 \times 2$ honeycomb lattice.

For 2d Ising model on square lattice (16×16 for example), the network structure is shown in Fig. S1 (a). For the random input, we fix its shape to be the same as that of the Monte Carlo configurations. Inside the network, inspired by the network structure in Ref.[30], we use three 2d convolutional layers, with 64 filters of size 8×8 for the first layer, 32 filters of size 1 for the second, and 1 filter of size 6×6 for the third layer. We choose rectified linear function ReLU(x) = max(0, x) in the first and second convolutional layers and sigmoid function $\sigma(x) = 1/(1+e^{-x})$ in the third. We also apply periodic boundary condition (PBC) layers each time before the convolutional layers if the kernel dimension is larger than one, which provides the configuration tensors with paddings of boundary elements instead of 0s (the padding size is decided by the size of filters in the corresponding convolutional layer, so as to ensure the shape invariance of the configuration tensors after convolutional operations).

In the case of 2d Hubbard model on the honeycomb lattice ($6\times6\times2$ for example), we switch from classical Ising configurations to the auxiliary field configurations of the DQMC[3,33], with the configuration space of $\beta \times L \times L \times 2$ where $\beta = 1/T$ is the inverse temperature. We set $\beta = 6$ and the Trotter discretization $\Delta\tau = 0.1$, so the DQMC auxiliary field configurations are of shape $60 \times 12 \times 6$ (note the 2 site per unit cell for the honeycomb lattice), as schematically shown in Fig. S1 (b). As shown in Eq.(4), we have to carry out large-scale determinant computation to measure the observables from generated configurations in order to compute the loss, and thus the complexity for the optimization is much larger than the case of classical model. Therefore, instead of the previous PBC-based network structure, we try another architecture of transposed convolutional layers and feed smaller random configurations as input. We find it is still capable of fitting the results we want and optimize faster. As shown in Fig. S1 (c) for the example model, we build two 3d transposed convolutional layers, with all filters using valid padding and sigmoid function as activation. There are 20 filters of size $16 \times 6 \times 3$ in the first layer and 1 filter of size $36 \times 6 \times 3$ in the second and the random input configurations are of shape $10 \times 2 \times 2$ while the predicted output are of shape $60 \times 12 \times 6$.

2. Optimization

In order to optimize such neural networks, for both classical and quantum models, we prepare 1000 sets of observables measured from MC simulations and take 1000 input random configurations which will be processed into generated configurations. The comparison in loss function is randomly distributed without any grouping. For the choice of observables in the defined loss functions, we simply pick some from the ones we usually focus on. In the Ising case, we take the batch size to be 5 and epoch number to be up to 150 as the computation is quite easy. While in the Hubbard case, we run at most 15 epochs with a batch size of 3. We optimize the network parameters using Adam[63] with conventional learning rate 10^{-3}, $\beta_1 = 0.9$, and $\beta_2 = 0.999$.

* w.li@itp.ac.cn
† qiyang@fudan.edu.cn
‡ zymeng@hku.hk