CHIN. PHYS. LETT. Vol. 37, No. 1 (2020) 018401

Predicting Quantum Many-Body Dynamics with Transferable Neural Networks *

Ze-Wang Zhang(张泽旺)[1], Shuo Yang(杨硕)[2], Yi-Hang Wu(吴亦航)[1], Chen-Xi Liu(刘晨曦)[1], Yi-Min Han(韩翊民)[1], Ching-Hua Lee[3,4], Zheng Sun(孙政)[1], Guang-Jie Li(李光杰)[1], Xiao Zhang(张笑)[1]**

[1]*School of Physics, Sun Yat-sen University, Guangzhou 510275*
[2]*State Key Laboratory of Low-Dimensional Quantum Physics and Department of Physics, Tsinghua University, Beijing 100084*
[3]*Department of Physics, National University of Singapore, 117542, Singapore*
[4]*Institute of High Performance Computing, 138632, Singapore*

(Received 24 October 2019)

Advanced machine learning (ML) approaches such as transfer learning have seldom been applied to approximate quantum many-body systems. Here we demonstrate that a simple recurrent unit (SRU) based efficient and transferable sequence learning framework is capable of learning and accurately predicting the time evolution of the one-dimensional (1D) Ising model with simultaneous transverse and parallel magnetic fields, as quantitatively corroborated by relative entropy measurements between the predicted and exact state distributions. At a cost of constant computational complexity, a larger many-body state evolution is predicted in an autoregressive way from just one initial state, without any guidance or knowledge of any Hamiltonian. Our work paves the way for future applications of advanced ML methods in quantum many-body dynamics with knowledge only from a smaller system.

PACS: 84.35.+i, 05.50.+q, 02.70.−c DOI: 10.1088/0256-307X/37/1/018401

Machine learning (ML) approaches, particularly neural networks (NNs), have achieved great success in solving real-world industrial and social problems.[1−8] Inspired by its widespread applicability, ML was soon adopted by condensed matter physicists in the modeling of quantum many-body behavior and phase transition discovery.[9−19] Compared to many advanced applications, it is natural to ask if recent progress in these more sophisticated ML architectures can benefit or even revolutionize the modeling of quantum systems. For instance, can quantum many-body dynamics be "learned" through transferable learning?[20,21]

Thus, the main objective of this work is to demonstrate the novel application of NNs in the transferable learning and prediction of the evolution of a many-body wavefunction, an otherwise computationally intensive task that has not been solved by generative models. Focusing on static problems, it is proven that deep NNs like the restricted Boltzmann machine (RBM) can represent most physical states,[22] and a recent work based on deep CNNs shows the ability to circumvent the need for Markov Chain sampling on the two-dimensional interacting spin model of larger systems.[23] Lately, physical properties of spin Hamiltonians are reproduced by the deep Boltzmann machine (DBM), as an alternative to the standard path integral.[24] Our approach is fundamentally in contrast with conventional approaches in computing many-body dynamics: instead of evolving the wavefunction explicitly with the Hamiltonian, we directly predict the dynamical wavefunction from the initial state by propagating it with an efficient and transferable framework based on unified spin encoding, chain encoding and SRU[25] module. With the same level of parallelism as feed-forward CNNs and scalable context-dependent capacity of recurrent connections, our proposed framework is naturally suited for learning many-body systems with unified parameters, although they have never been harnessed for an exact quantum state evolution; in our scenario, a 1D Ising model with both parallel and transverse magnetic field.

Inspired by end-to-end training[26] and domain adaptation,[27,28] we specialize in the many-body dynamics of a 1D Ising chain with transverse and parallel magnetic fields. Comparison with exact conventionally computed results with up to seven spins reveals high predictive accuracy, as quantified by the relative entropy as well as magnetization. Indeed, our SRU-propagated wavefunction shows a strong grasp of the periodicity in the time evolution, despite being unaware of the Hamiltonian that sets the energy (inverse periodicity) scale. Encouraged by circumventing the problem of exponential computational complexity through unified encoding mechanisms and parallel recurrent connections, we hope that such encouraging results from our pioneering transferable learning approach will inspire further applications of transferable learning methods to build a shared model suited for quantum systems with vast spin variables.

We consider a 1D Ising spin chain composed of N spin variables with local transverse (g) and parallel (h) magnetic fields, described by the Hamiltonian

$$H = -\sum_{i}^{N-1} \sigma_i^z \otimes \sigma_{i+1}^z - h\sum_{i}^{N} \sigma_i^z - g\sum_{i}^{N} \sigma_i^x, \quad (1)$$

*Supported by the National Natural Science Foundation of China under Grant Nos 11874431 and 11804181, the National Key R&D Program of China under Grant No 2018YFA0306800, and the Guangdong Science and Technology Innovation Youth Talent Program under Grant Nos 2016TQ03X688 and 2018YFA0306504, and the Research Fund Program of the State Key Laboratory of Low-Dimensional Quantum Physics under Grant No ZZ201803.

**Corresponding author. Email: zhangxiao@mail.sysu.edu.cn

© 2020 Chinese Physical Society and IOP Publishing Ltd

where σ^x and σ^z denote the Pauli matrices, and i denotes the spin variable. When the magnetic field is parallel ($g = 0$) or transverse ($h = 0$), the Hamiltonian is exactly solvable. However, when $g \neq 0$ and $h \neq 0$, the dynamics of N spins must be numerically computed in the 2^N-dimensional many-body Hilbert space spanned by direct product states Ψ of single-spin wavefunctions ψ_i:

$$\Psi = \prod_i^N \otimes \psi_i = \prod_i^N \otimes \begin{pmatrix} \phi_i^{\uparrow} \\ \phi_i^{\downarrow} \end{pmatrix}, \quad \dim \Psi = 2^N. \quad (2)$$

Wavefunction dynamics can be exactly computed through unitary time evolution of the Hamiltonian

$$\begin{aligned} |\Psi(t)\rangle &= \exp\left(-i\frac{H}{\hbar}t\right)|\Psi(0)\rangle \\ &= V\exp\left(-i\frac{E}{\hbar}t\right)V^{-1}|\Psi(0)\rangle, \end{aligned} \quad (3)$$

where $E = V^{-1}HV$ is the diagonal eigenenergy matrix.

The 2^N-dimensional N-body wave function $|\Psi(t)\rangle$ quickly becomes expensive to compute as N increases. Inspired by the sequence generation model,[29] our framework (as shown in Fig. 1(b)) is composed of a spin encoding layer, chain encoding layer, SRU layers, and spin decoding layer, which instead attempts to predict its time evolution based on prior knowledge of the time evolution behavior of known training states. This training (learning) only has to be performed once for the relatively inexpensive prediction of any number of initial states. Importantly, the training and prediction process captures solely the intrinsic evolution patterns of the wavefunctions, and does not involve any explicit knowledge about the Hamiltonian. Moreover, as we shall explain, our SRU-based framework is transferable. We next outline the broad principles behind our NN approach of predicting quantum state evolution, for details see the Supplementary Material. The vanilla SRU NN with peephole connections (Fig. 1(a)) substitutes inherent matrix multiplication with parallelizable element-wise multiplication operations ($\odot$ in Fig. 1(a)) associated with c_{t-1}, hence the calculation of f_t does not have to wait until the whole c_{t-1} is updated. With the help of spin encoding and decoding layers, the amount of trained parameters is fixed, and thus the complexity has an upper bound instead of increase exponentially.

Our procedure occurs in two main stages: the training stage and the inference stage. We first "train" or optimize the weight parameters of our SRU-based framework in a teacher forcing mode[30] by feeding it with a large number of training sequences, which are the time-evolved wavefunction data of 10^4 randomly chosen initial 2-spin to 7-spin state sequences sampled over 500 time steps, obtained via conventional exact diagonalization (ED). The SRU-based framework is fully optimized by the Adam optimization algorithm[31] to minimize the mean squared error between the ED-evolved and SRU-evolved states at all time steps in a mini-batch (see the Supplementary Material).

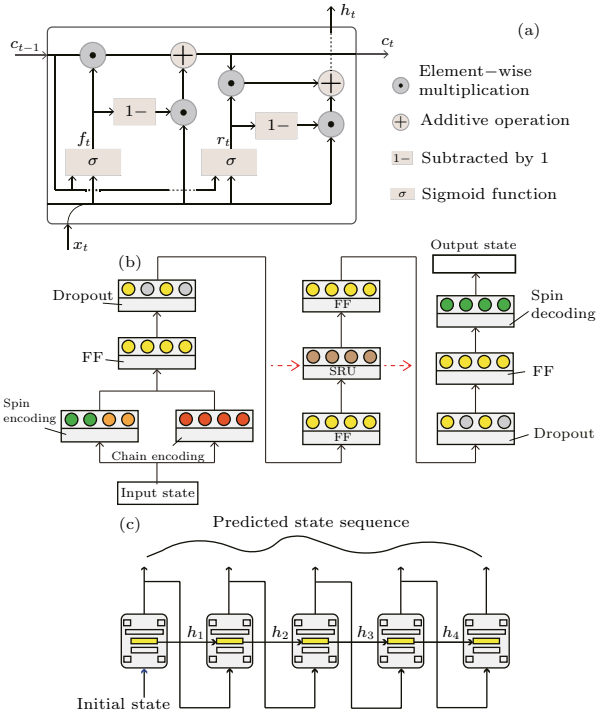


Fig. 1. (a) Details of a vanilla SRU cell with forget gate f_t, skip gate r_t, memory cell c_t and weighted connections from previous memory cell c_{t-1} to both forget gate f_t and skip gate r_t. X_t is the input and h_t is the output hidden state of SRU cell. The forget gate decides what to forget from previous memory cell c_{t-1} and what redundant information to drop when adapting to other systems. The skip gate, along with current memory cell c_t, decides what to skip from input and what to output as h_t. Here $\sigma(x) = \frac{e^x}{e^x+1}$ is the activation function. (b) Proposed architecture we take in this study as a block composed of a spin encoding layer, a chain encoding layer, SRU layers and a spin decoding layer at each time step. Spin encoding, chain encoding and spin decoding are all feed-forward layers (labeled as FF). (c) Autoregressive procedure of generating new quantum states, given an initial state at the beginning. Each time-evolved quantum state is predicted from our proposed block by SRU's cell memory information (shown as arrows in the middle) and previous predicted state.

Once trained well, the SRU-based framework is ready to predict the evolution of arbitrarily given initial states. As sketched in Fig. 1(c), the initial state $|\Psi(t=0)\rangle$ enters the leftmost block at $t = 0$, and is then processed by a spin encoding layer, a chain encoding layer, and two fully connected layers, and its output is propagated as an input state to the next block with hidden layers h_t. The output of each block denotes a new quantum state at a certain time step. The combination of memory cell c_t and hidden output h_t acts as effective context-dependent behaviors. As illustrated in Fig. 1(a) and further elaborated in the Supplementary Material, context-dependent information kept in memory cell c_t is modified by its previous value c_{t-1}, new input x_t interacted with forgotten gate and skipped gate at that time step, as well as "hidden" information on h_{t-1} from the previous SRU cell. Based on the already optimized SRU-

based framework, the final predicted quantum state as a function of time would be generated from one fully connected layer and the spin decoding layer as shown in Fig. 1(b).

We report very encouraging agreements between wavefunctions evolved by $e^{-iHt/\hbar}$ as computed by ED, and wavefunction evolutions as predicted by our SRU-based framework. As for the 1D Ising model, we set the local transverse magnetic field g to be -1.05, parallel magnetic field h to be 0.5 and Δt, the time interval to be 0.002, and keep this setting constant for all computation. We find that the maximum energy eigenvalue is about $0.1 \gg 0.002$, proving that the chosen time interval is small enough. The number of spin variables studied (2 to 7) decides the cost of exactly computing the 10^4 different time evolutions over 0.2 s (100 time steps) prior to training the network, since the time complexity of the ED method is $\mathcal{O}(2^n)$.

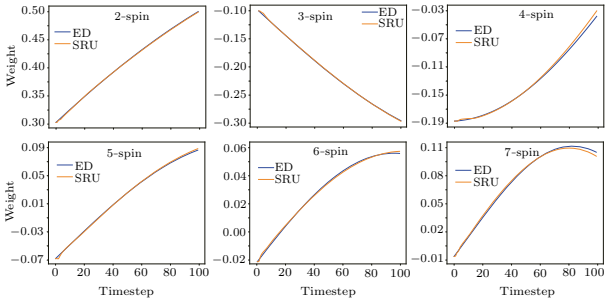


Fig. 2. Output (SRU-based) and target (ED-based) wavefunction magnitude (y-axis) for 2-spin to 7-spin systems, with the initial states given in the Supplementary Material. We plot the curves of the total 100 time steps.

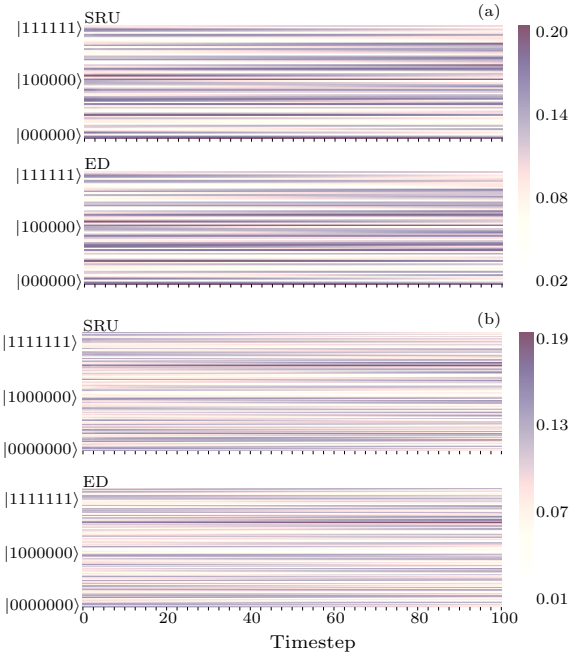


Fig. 3. Comparisons of all components of coefficients for SRU-based prediction and ED-based simulation in each transverse lattice with different colors for both (a) six-spin and (b) seven-spin systems.

Concretely, we visually illustrate the comparison for the evolution of a typical state from 2-spin to 7-spin in Fig. 2. These states are evolved from arbitrarily chosen initial states from the test set. Saliently, the evolution predicted by the SRU-based model accurately reproduces that from exact computations at the beginning 100 time steps. To confirm that this agreement is not just due to a fortuitous choice of component, we look at the evolution across all components of the same states in Fig. 3.

To further quantify the agreement of SRU and ED wavefunction evolutions, we compute the relative entropy (Kullback–Leibler divergence)[32] of their distributions over 1000 test wavefunctions sequences. For discrete probability distributions P and Q, the relative entropy is defined as

$$D_{KL}(P||Q) = \sum_x P(x) \log\left(\frac{P(x)}{Q(x)}\right). \tag{4}$$

Given ED-computed wavefunction coefficient vectors M^{ED} and SRU-predicted coefficient vectors M^{SRU}, at time t and basis vector x, the P and Q variables are taken as

$$P_{n,t,x} = \frac{|M^{\mathrm{ED}}_{n,t,x}|}{\sum_{x=1}^{2^N} |M^{\mathrm{ED}}_{n,t,x}|}, \tag{5}$$

$$Q_{n,t,x} = \frac{|M^{\mathrm{SRU}}_{n,t,x}|}{\sum_{x=1}^{2^N} |M^{\mathrm{SRU}}_{n,t,x}|}, \tag{6}$$

where n labels the test sequence. Hence the mean relative entropy (MRE) at each time step t is

$$D_{KL}(P||Q)(t) = \frac{1}{1000} \sum_{n=1}^{1000} \sum_{x=1}^{2^N} P_{n,t,x} \log \frac{P_{n,t,x}}{Q_{n,t,x}}, \tag{7}$$

and measures the amount of information lost when the distribution Q from SRU predictions is used to represent the distribution P from ED results. The smaller the value of $D_{KL}(P||Q)(t)$, the more accurate their agreement is.

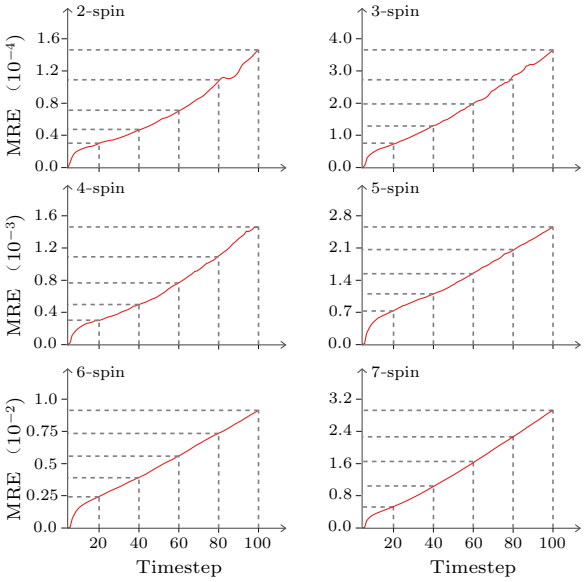


Fig. 4. MRE of generating long sequences in different systems. The MRE increases linearly with time steps.

In Fig. 4, we show the MRE varies significantly across time steps during prediction of test set. In

all six systems, the order of relative entropy is always within 0.03. Evidently, with the increase of time steps, the relative entropy generally shows an upward trend and increases linearly (see the Supplementary Material), which is caused by the accumulation of errors during the conditional generation without any external guidance, though already suppressed dropout layers.

Owing to the unified encoding and parallelism, our SRU-based NN is becoming increasingly more superior over the ED method in terms of efficiency, as the number of spins and batch size increase. Table 1 summarizes the results. When the number of spins gets larger, e.g. 6 and 7, the advantage of our SRU-based framework on inference speed becomes more and more obvious. This is attributed to its constant computational complexity. In addition, when we enlarge the batch size to 256 for 7 spins, our model demonstrates a speed of 130 times faster than the ED-based method.

Table 1. Comparison of time consumption (seconds) between the ED and SRU-based methods. Three independent runs are performed to generate sequences of different batch sizes, and the average time consumption is reported. BS denotes the batch size, and the bold black font indicates that our SRU-based method is superior to ED with increasing spin number and batch size.

	BS = 1		BS = 64		BS = 128		BS = 256	
System	ED	Ours	ED	Ours	ED	Ours	ED	Ours
2-spin	0.015	0.425	1.1	**0.74**	2.3	**0.83**	4.6	**0.69**
3-spin	0.035	0.425	2.2	**0.74**	4.4	**0.69**	8.7	**0.71**
4-spin	0.059	0.425	3.8	**0.77**	7.6	**0.66**	15.1	**0.72**
5-spin	0.271	0.425	17.5	**0.82**	34.9	**0.76**	69.7	**0.79**
6-spin	0.556	**0.425**	35.2	**0.71**	70.5	**0.66**	141.4	**0.98**
7-spin	1.15	**0.425**	73.1	**0.80**	146.3	**0.91**	292.5	**2.18**

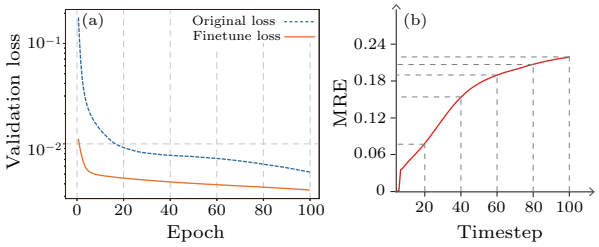


Fig. 5. (a) The validation loss for predicting the dynamics of the 8-spin system by finetuning and training from scratch, respectively. (b) The MRE of the 8-spin system.

After obtaining the base model trained with data sets of 2 to 7 spins by 300 epochs, we may continue to finetune it with the data set of the 8-spin system. We also train it from scratch for comparison. The results in Fig. 5(a) shows that the validation loss by finetuning the base model is much lower than training from scratch, demonstrating that our NN has already learned transferable features from smaller systems. The MRE of the 8-spin system is shown in Fig. 5(b).

In summary, we have successfully applied a transferable NN approach based on SRU networks to approximate the state evolution of dynamic quantum many-body systems with high accuracy and superior scalability. Our work encourages future applications of advanced ML methods in quantum many-body dynamics in a Hamiltonian-agnostic manner. One possibility is to predict the behavior of large and inhomogeneous systems lacking training data by just learning from a smaller-sized system.[33,34] Applications of these advancements in ML to quantum many-body problems are left to future work.

Xiao Zhang thanks Yingfei Gu, Meng Cheng, Yi Zhang for discussions.

References

[1] Bengio Y, Courville A and Vincent P 2013 *IEEE Trans. Pattern Anal. Mach. Intell.* **35** 1798
[2] Krizhevsky A, Sutskever I and Hinton G E 2017 *Commun. ACM* **60** 84
[3] Gatys L A, Ecker A S and Bethge M 2016 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* p 2414
[4] Van Den Oord A, Dieleman S and Zen H 2016 *CoRR abs/1609.03499*
[5] Sun Z, Liu J and Zhang Z 2018 *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)* p 1864
[6] Wu Y, Schuster M and Chen Z 2016 arXiv:1609.08144 [cs.CL]
[7] Voulodimos A, Doulamis N, Doulamis A and Protopapadakis E 2018 *Comput. Intell. Neurosci.* **2018**
[8] Deng L, Hinton G and Kingsbury B 2013 *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* p 8599
[9] Van Nieuwenburg E P, Liu Y H and Huber S D 2017 *Nat. Phys.* **13** 435
[10] Cai Z and Liu J 2018 *Phys. Rev. B* **97** 035116
[11] Torlai G and Melko R G 2016 *Phys. Rev. B* **94** 165134
[12] Torlai G and Melko R G 2017 *Phys. Rev. Lett.* **119** 030501
[13] Deng D L, Li X and Sarma S D 2017 *Phys. Rev. X* **7** 021021
[14] Gray J, Banchi L, Bayat A and Bose S 2018 *Phys. Rev. Lett.* **121** 150503
[15] Ch'ng K, Carrasquilla J, Melko R G and Khatami E 2017 *Phys. Rev. X* **7** 031038
[16] Broecker P, Carrasquilla J, Melko R G and Trebst S 2017 *Sci. Rep.* **7** 8823
[17] Amin M H, Andriyash E, Rolfe J, Kulchytskyy B and Melko R 2018 *Phys. Rev. X* **8** 021050
[18] Zhang Y and Kim E A 2017 *Phys. Rev. Lett.* **118** 216401
[19] Carleo G and Troyer M 2017 *Science* **355** 602
[20] Pan S J and Yang Q 2009 *IEEE Trans. Knowl. Data Eng.* **22** 1345
[21] Torrey L and Shavlik J 2010 *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques* (IGI Global) p 242
[22] Gao X and Duan L M 2017 *Nat. Commun.* **8** 662
[23] Sharir O, Levine Y, Wies N, Carleo G and Shashua A 2019 arXiv:1902.04057 [cond-mat.dis-nn]
[24] Carleo G, Nomura Y and Imada M 2018 *Nat. Commun.* **9** 5322
[25] Lei T, Zhang Y, Wang S I, Dai H and Artzi Y 2018 *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* p 4470
[26] Graves A and Jaitly N 2014 *International Conference on Machine Learning* p 1764
[27] Pan S J, Tsang I W, Kwok J T and Yang Q 2010 *IEEE Trans. Neural Netw.* **22** 199
[28] Glorot X, Bordes A and Bengio Y 2011 *Proceedings of the 28th International Conference on Machine Learning (ICML-11)* p 513
[29] Sutskever I, Vinyals O and Le Q V 2014 *Advances in Neural Information Processing Systems* p 3104
[30] Williams R J and Zipser D 1989 *Neural Comput.* **1** 270
[31] Kingma D P and Ba J 2014 arXiv:1412.6980 [cs.LG]
[32] Vedral V 2002 *Rev. Mod. Phys.* **74** 197
[33] Hinton G, Vinyals O and Dean J 2015 arXiv:1503.02531 [stat.ML]
[34] Kingma D P, Salimans T and Jozefowicz R 2016 *Advances in Neural Information Processing Systems* p 4743

Supplemental Material: Predicting quantum many-body dynamics with transferable neural networks

Ze-Wang Zhang,[1] Shuo Yang,[2] Yi-Hang Wu,[3] Chen-Xi Liu,[3] Yi-Min Han,[3] Ching-Hua Lee,[4,5] Zheng Sun,[3] Guang-Jie Li,[3] and Xiao Zhang[3,*]

[1]*School of Physics, Sun Yat-sen University, Guangzhou 510275, China. Telephone: 15602298593*
[2]*State Key Laboratory of Low-Dimensional Quantum Physics and Department of Physics, Tsinghua University, Beijing 100084, China*
[3]*School of Physics, Sun Yat-sen University, Guangzhou 510275, China*
[4]*Department of Physics, National University of Singapore, 117542, Singapore*
[5]*Institute of High Performance Computing, 138632, Singapore*

I. SPIN AND CHAIN ENCODING MECHANISMS

As a prelude to the unified framework, we proposed two universal encoding mechanisms for any wavefunctions of 1D Ising spin chain, they are spin encoding and chain encoding. Both encodings are actually implemented as feed-forward neural networks, the weights of them are initially sampled from the normal distribution [1]. We present the details of spin encoding and chain encoding in Fig. I1.

The input of our SRU-based framework is composed of spin encoded vector and chain encoded vector. First, we randomly initialize two spin encoding embedding matrices $E^{re}_{spin}, E^{im}_{spin} \in \mathbb{R}^{2^n \times m}$ from normal distribution, here n is the maximum size of spin variables, and m is the hidden units for representing information of spin feature. "re" denotes the embedding matrix of real part and "im" denotes embedding matrix of the imaginary part. In this paper, we set the n to be 10 (Note that $2^{10} = 1024$) and the m to be 256. Specifically, taking 7-spin system as example, a state sequence of its T timesteps can be represented as a complex matrix $S \in \mathbb{R}^{T\times 2^7}$. We split the complex matrix S into its real part S^{re} and its imaginary part S^{im}. Since $2^7 = 128$, we fetch the beginning 128 rows from E^{re}_{spin} and E^{im}_{spin}, and then multiply them by S^{re} and S^{im}, respectively. We take such process as "lookup" shown in Fig. I1. Finally, we concatenate the real part encoded vector and imaginary part encoded vector to form the spin encoded vector ($\mathbb{R}^{T\times 512}$).

Similarly, we randomly initialize a chain embedding matrix $E_{chain} \in \mathbb{R}^{n\times m}$, where n (=10) denotes the maximum size of spin variables of a chain, and m (=512) is the hidden unit for storing chain feature. Specifically, taking 7-spin system as example, we adopt one-hot vector $[0,0,0,0,1,0,0,0,0,0]$ to represent such system, and multiply it by chain embedding matrix E_{chain} to obtain the chain encoded vector of $\in \mathbb{R}^{T\times 512}$. Obtaining the spin encoded vector ($\mathbb{R}^{T\times 512}$) and chain encoded vector ($\mathbb{R}^{T\times 512}$), we take the sum of them as the input to SRU-based framework at each timestep. The workflow of spin and chain encoding is illustrated in Fig. I1.

II. SRU CELL

In most RNN architectures, including Long Short-term Memory (LSTM)[2], the new state of each timestep must be suspended until completing execution of both input and previous state. In contrast, comparing with the mainstream recurrent neural networks (RNNs), each SRU cell having a cell state and a forget gate is advantagous in terms of its efficient and parallelizable element-wise matrix product. The SRU cell is very crucial for predicting dynamics of 1D Ising model, and we assume that it can be also applied to other sequential physical tasks.

A single layer of SRU employs the following computations:

$$\begin{aligned} f_t &= \sigma(W_f x_t + v_f \odot c_{t-1} + b_f) \\ c_t &= f_t \odot c_{t-1} + (1 - f_t) \odot (W x_t) \\ r_t &= \sigma(W_r x_t + v_r \odot c_{t-1} + b_r) \\ h_t &= r_t \odot c_t + (1 - r_t) \odot x_t \end{aligned} \tag{1}$$

where f_t is a forget gate that can discard the redundant information several timesteps ago and unnecessary information of old spin chains in fusion training. It is computed from a sigmoid function (σ, $\sigma(x) = \frac{1}{1+e^{-x}}$) activated sum of input x_t weighted by matrix W_f, the previous cell state c_{t-1} weighted by vector v_f and a bias vector b_f. c_t is the

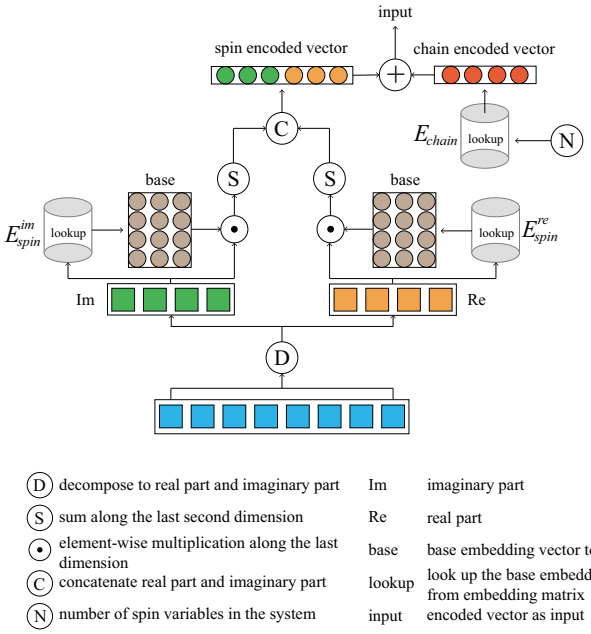


FIG. I1: Detailed calculation of spin and chain encoding mechanisms. The spin and chain encoded vector are concatenated into the final input vector for feed-forward layers.

cell state that serves to give feedbacks of latent correlated features from previous to next recursively, as the moving average of previous cell state c_t weighted by f_t plus the weighted input Wx_t. c_t is crucial for the accurate prediction of dynamics based on just one initial quantum state. r_t is the skip gate of highway network [3] that is a shortcut for information flow with learnable weights to determine the extent of the information skip, and is computed from a sigmoid function activated sum of weighted input $W_r x_t$, weighted previous cell state $v_r c_{t-1}$ and a bias vector b_r. The skip gate r_t adaptively averages the input x_t and the cell state c_t produced from the light recurrent connection. $(1-r_t)\odot x_t$ is a residual connection to settle the gradient dimishing problem, and the useage of highway network has been proved to improve the stability. Four SRU layers are stacked to learn better context-dependent latent information and transferable features.

An obvious difference between SRU and common RNN architectures is the way of c_{t-1} used in the sigmoid gate. Generally, c_{t-1} is involved in matrix multiplication with weight matrix to compute each gate. For example, the forget gate of a commonly adopted LSTM architecture is computed as $f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f)$. Obviously, the matrix multiplication between c_{t-1} and W_{cf} makes the parallelization of computing state f_t almost impossible, because each dimension of state f_t can't be derived until the whole dimensions of c_{t-1} are available. In Eq. (1), we substitute all matrix multiplications associated with c_{t-1} with element-wise multiplications (denoted as $\odot$). Based on this simplication, we make each state of SRU independent and parallelizable, and it is efficient and lossless.

To make SRU work efficiently, we follow the guidances of parallelized implementation performed in the context of CUDA programming [5]. Eq. (1) has totally three weight matrices: W, W_f and W_r, and all of them need to be multiplied by the input sequence $\{x_1 \dots x_L\}$, where L is the sequence length. To fully utilize the computational intensity of GPU, we do the batched matrix multiplications across all timesteps as:

$$U^T = \begin{pmatrix} W \\ W_f \\ W_r \end{pmatrix} [x_1, x_2, \dots, x_L], \tag{2}$$

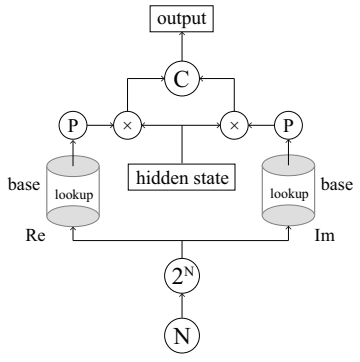
U is the computed matrix. If the hidden unit of weight is d in demension and the input is a mini-batch of BS sequences, then U would be a tensor of size $[L, BS, 3d]$.

Kernel fusion[6, 7] is an optimization method to reduce overhead of data transfer from global memory by fusing some sequential kernels into a single large one, to improve performance and memory locality. Specifically, we compile all element-wise multiplications into a single fused CUDA kernel function and parallelize the computation across each dimension of cell state. The complexity of each SRU layer is $\mathcal{O}(L \cdot B \cdot d)$ whereas the complexity of a vanilla LSTM layer is $\mathcal{O}(L \cdot B \cdot d^2)$.

III. SPIN DECODING MECHANISM

When the hidden states of sequential modeling are obtained, we need to restore the actual predicted states. Correspondingly, we propose the spin decoding mechanism (detailed in Fig. III2). To get the final predicted output, we need the explicit size of spin variable N, and then look up two embedding matrices of feature dimension 2^N. We then transpose each embedding vector and then do matrix multiplication between embedding vector and hidden state. Finally, we concatenate the real part and imaginary part to get the predicted state. The embedding matrix $E_{dec} \in \mathbb{R}^{1024\times512}$ of spin decoding layer is also randomly initialized from normal distribution.

The overall methodology for the design of our SRU-based framework follows the criteria of unity, efficiency and transferability. In addition to the spin encoding, chain encoding, SRU module and spin decoding described above, the feed-forward layers are the bottleneck layers containing full connections with a *tanh* nonlinearity and dropout[8], whose purpose is to distill the high-level latent feature and enable the framework to generalize well to unseen input. Such a gap between training and inference as context mismatch, mechanisms to alleviate such effect in sequence models are scheduled sampling[9], professor training[10] and generative adversial network (GAN)[11, 12]. We empirically observe that just adding dropout layers also yields better scalability in the context of 1D Ising spin chain. The trainable weights and biases are initialized with zero-mean and unit-variance random distributions throughout the entire framework. Each input state (the real and imaginary parts are concatenated together) is represented by the spin encoding layer and chain encoding layer, and then concatenated into two stacked feed-forward layers with *tanh* activation. The dropout layer is also applied to the inputs prior to the second feed-forward layer for regularization. To be specific, we set the dropout rate to be 0.005, that is to say, during training, about 0.5% of neurons are set to be zero randomly. The SRU module, taking these hidden features from the dropout layer as input to construct the equivalent contextual level representation, updates the cell state and output the hidden state. Finally, a feed-forward layer with the spin decoding generates a new physical state.



2^N	the dimension of feature	Im	imaginary part
P	transpose matrix along first two dimensions	Re	real part
×	matrix multiplication	base	base embedding vector to multiply
C	concatenate real part and imaginary part	lookup	look up the base embedding vector from embedding matrix
N	number of spin variables in the system	output	the final predicted quantum state

FIG. III2: Detailed calculation of spin decoding mechanism. The hidden state is multiplied by spin decoding embedding to form the final predicted quantum state.

IV. TRAINING AND INFERENCE PROCEDURE

Algorithm 1 Learning quantum many-body dynamics by SRU-based framework

Require: θ: the parameters of whole network composed of spin and chain encoding, SRU and spin decoding layers.
Input: B_S coefficient sequences in a batch with T_L timesteps each.
Output: same as input but with one timestep offset.
Preparation: concatenate the real part x_r and imaginary part x_i of each complex coefficient into x.

Training Stage:

Initialize θ
for each training epoch **do**
  for k steps **do**
    number of particles is N
    obtain the output of spin encoding for both real part and imaginary part:
      $feature_{SE} = [Emb_{SE}(x_r)x_r; Emb_{SE}(x_i)x_i]$
    obtain the output of chain encoding:
      $feature_{CE} = Emb_{CE}(N)$
    get the sum as input:
      $input = feature_{SE} + feature_{CE}$
    pass through two feed-forward layers, apply dropout and layer normalization (LN):
      $output_{f1} = LN(dropout(ReLU(W_1 input))W_2 + input)$
    pass through block of four stacked SRU layers:
      $output_s = SRU_{block}(output_{f1})$
    pass through two feed-forward layers, apply dropout and layer normalization (LN):
      $output_{f2} = LN(dropout(ReLU(W_1 output_s))W_2 + output_s)$
    obtain the real part of output coefficient from spin decoding:
      $c_r = Emb^R_{SD} output_{f2}$
    obtain the imaginary part of output coefficient from spin decoding:
      $c_i = Emb^I_{SD} output_{f2}$
    Update θ by ascending the stochastic gradient descent:
    $$\nabla(\theta)\left\{\frac{1}{T_L}\sum_{T_L}\frac{1}{B_S}\sum_{B_S}\left[(x_r - c_r)^2 + (x_i - c_i)^2\right]\right\}$$
  end for
end for

Inference Stage:

for g steps **do**
  if $g == 1$ **then**
    compute the first predicted coefficient vector c_1 by feeding with initial coefficient sequence as *input*:
      $c_1 = framework(input; \theta)$
  else
    compute new coefficients c_g by feeding with previous output c_{g-1}:
      $c_g = framework(c_{g-1}; \theta)$
  end if
end for

At the stage of training (IV3), given a state sequence of n timesteps $\{\Psi_1, ..., \Psi_n\}$, we can split it into two sub-sequences: $seq_i = \{\Psi_1, \Psi_2, ..., \Psi_{n-2}, \Psi_{n-1}\}$ and $seq_o = \{\Psi_2, \Psi_3 ..., \Psi_{n-1}, \Psi_n\}$. The state sequences are divided into training, validation and test sets, with an 80%-10%-10% split. Considering the different scales in different systems, we observed no empirical benefits from introducing data normalization pipeline. It therefore makes sense to use raw data generated from exact diagnolization (ED) method and avoid the need of data normalization and denormalization. We use a sequence learning method which passes the input seq_i into the network at each timestep, and computes output seq_o of the final layer based on a linear activation function. In contrast, at the stage of inference (Fig.1(d)), we only pass the initial wavefunction into the network as input, and the network then autoregressively generates the following states at future timesteps, by consuming the previously generated states as additional input. The full procedure of training and inference is detailed in Algorithm 1. The training loss and inference loss is shown in IV4.

To be more explicit, our dataset is gathered by computing the time evolution of wavefunctions from the two to seven-spin systems every 0.002 second, using ED method[13]. Treated as a standard ML process, we divide the whole dataset into three parts: a training set with $10,000$ sequences, a validation set with $1,000$ sequences and a test set

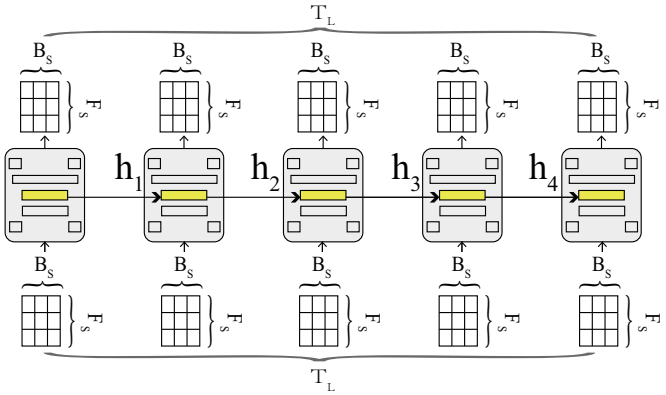


FIG. IV3: Traing the network in a teacher forcing mode[4], we take the current wave functions as input and next state as the ground label. The ground label sequence is one timestep off the input sequence.

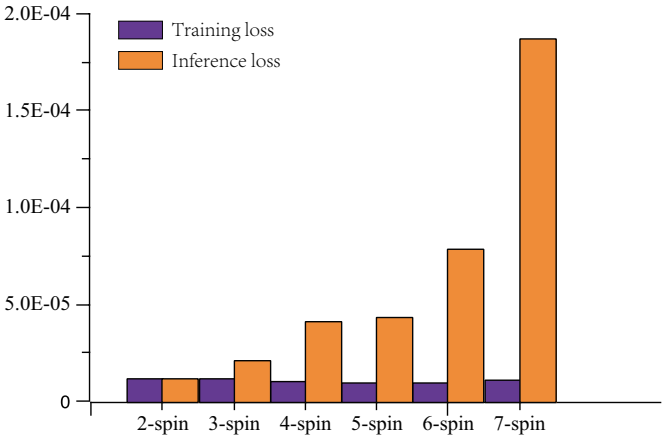


FIG. IV4: Training loss and inference loss of different 1D Ising systems. Inference loss represents the average loss of wavefunction evolution in 100 timesteps, the maximum inference loss is within 2.0E-4.

with $1,000$ sequences. During training, we use the mini-batch gradient descent framework to optimize our network and $m = 16$ is the batch size (B_S) we use, considering both NN's converging speed and GPU memory.

The target of training is to minimize the mean squared error (MSE) at all timesteps between output and target states. Before training, we initialize all weight variables based on normal distribution and set all bias variables to be zero vectors. We use the adaptive moment estimation (Adam) optimizer [14] to update the parameters for that it's very suitable for tasks with datasets of large dimension and needs less memory space, with an initial learning rate of 0.001 decayed by a factor of 0.95 every epoch until the learning rate falls below $5.0e-5$. We train our base model for 300 epoches and keep the best model with a lowest validation loss. Our model is implemented in PyTorch [15] and TensorFlow [16] framework, and all simulations are run on single NVIDIA Tesla P40 GPU and Intel Xeon CPU E5-2680. Our SRU-based framework is predominantly run on a single GPU, whereas the conventional ED-based method is run on a CPU totally. The hyper-parameters are presented in Table IV1.

In the transfer learning experiment, we select the base model and start training the new dataset using the same Adam optimizer with an initial learning rate of 0.0001 decayed by a factor of 0.97. The batch size is 16 and the total training epochs are 300.

In the main text we only showed the result of predicting state sequences of 100 timesteps. To demonstrate the ability of generating longer state sequences for different quantum many-body systems, we showed the performance of predicting state sequences as long as 250 timesteps evaluated by mean relative entropy (MRE). From Fig. IV5, the MRE increases linearly with the timesteps because of the error accumulation during the autoregressive generation. We will try to apply non-autoregressive generative model with inverse autoregressive flows[17] to better estimate the probability density of longer state sequences in our future work.

TABLE IV1: Network architecture hyper-parameters.

layer	configuration
spin encoding embedding	size=1024, unit=512
chain encoding embedding	size=1024, unit=512
feed-forward layer	unit=512
dropout layer	dropout rate=0.005
feed-forward layer	unit=512
SRU layer	layers=4, unit=512
feed-forward layer	unit=512
dropout layer	dropout rate=0.005
spin decoding embedding	size=1024, unit=512
batch size	32
learning rate (lr)	initial lr=0.001, decay rate=0.95

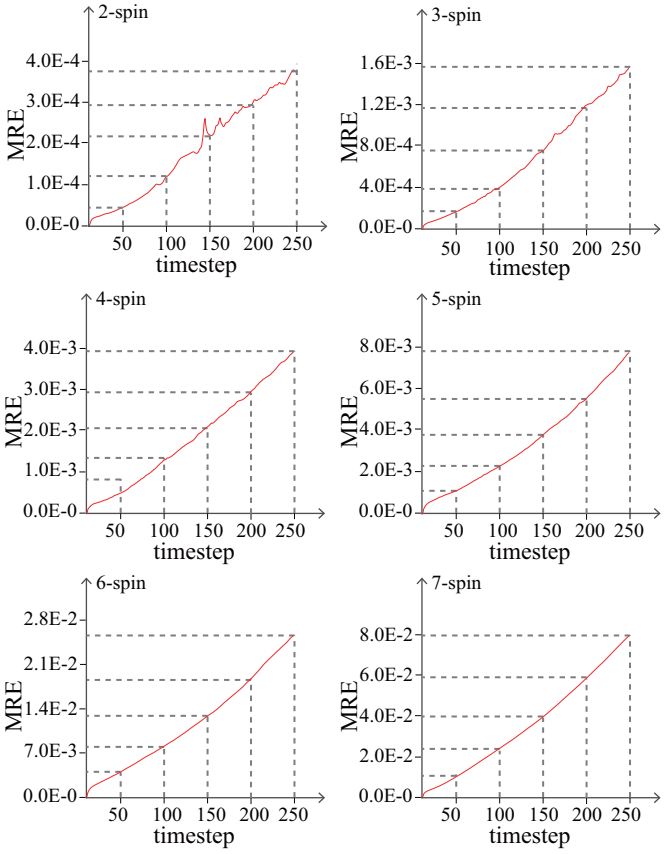


FIG. IV5: MRE of generating longer sequences of 250 timesteps in different systems. The MRE increases linearly with timesteps.

To quantify our model's performance by a physical variable, we draw the magnetization intensity calculated from both predicted (SRU) and simulated (ED) wavefunctions in Fig. IV6, which have a nice agreement. Specifically, for smaller-sized systems, such as 2-spin and 3-spin, the predicted magnetization intensity has a very nice agreement with simulated one. With the increase of spin variables, our SRU-based framework has a performance drop due to the exponentially increased computation complexity. Meanwhile, with the increase of timesteps, the difference between predicted and simulated magnetization intensity also becomes larger, which is due to the error accumulation during the autoregressive generation without any external guidance.

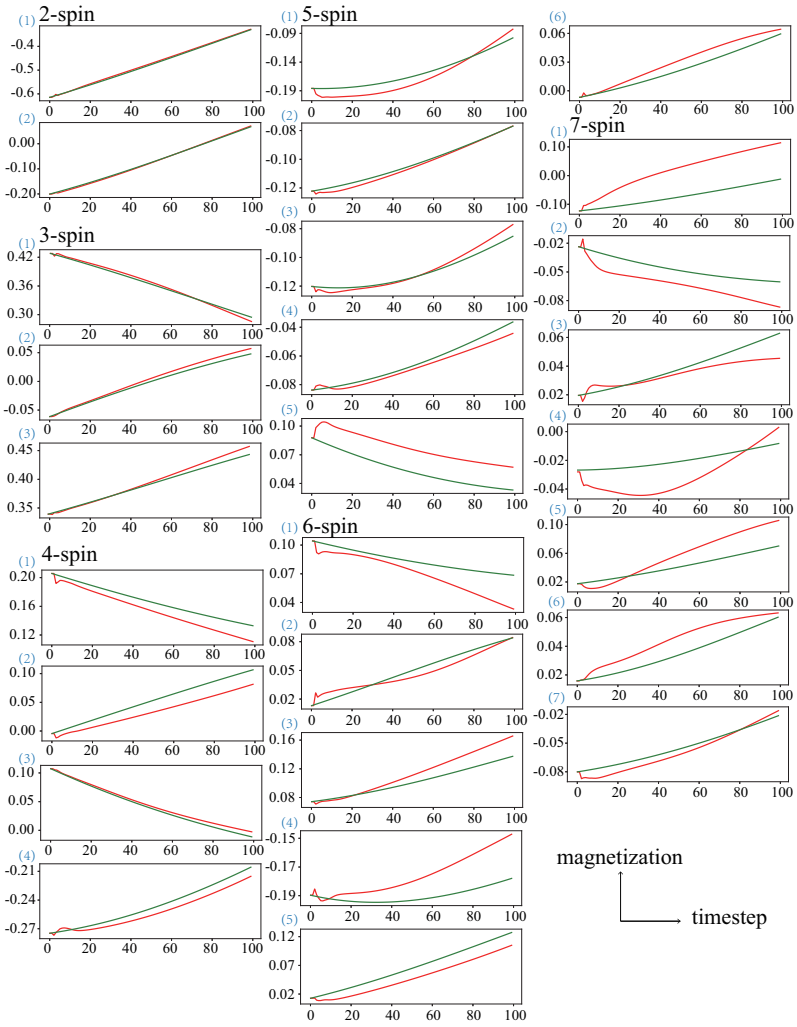


FIG. IV6: Magnetization varies with timestep calculated by both SRU (red curve) and ED (green curve) based wavefunctions for all six systems.

V. INITIAL STATES USED FOR PREDICTION

A. The initial state of two-spin system for sequence generation

$$(-0.078+0.203j)|00\rangle+(-0.458+0.359j)|01\rangle$$
$$+(\ 0.372+0.602j)|10\rangle+(-0.274+0.197j)|11\rangle$$

B. The initial state of three-spin system for sequence generation

$$(-0.127+0.134j)|000\rangle+(-0.012+0.357j)|001\rangle$$
$$+(\ 0.218+0.271j)|010\rangle+(-0.082+0.158j)|011\rangle$$
$$+(\ 0.101+0.217j)|100\rangle+(-0.175+0.499j)|101\rangle$$
$$+(-0.253+0.274j)|110\rangle+(-0.123+0.442j)|111\rangle$$

C. The initial state of four-spin system for sequence generation

(-0.091+0.008j)|0000⟩+(-0.141+0.123j)|0001⟩
+(0.002+0.382j)|0010⟩+(0.169+0.190j)|0011⟩
+(0.015+0.153j)|0100⟩+(0.042+0.028j)|0101⟩
+(-0.075+0.177j)|0110⟩+(0.137+0.255j)|0111⟩
+(-0.236+0.037j)|1000⟩+(-0.085+0.331j)|1001⟩
+(0.140+0.069j)|1010⟩+(-0.230+0.270j)|1011⟩
+(0.071+0.319j)|1100⟩+(-0.162+0.272j)|1101⟩
+(-0.149+0.132j)|1110⟩+(-0.016+0.169j)|1111⟩

D. The initial state of five-spin system for sequence generation

(0.037+0.077j)|00000⟩+(-0.110+0.026j)|00001⟩
+(0.072+0.028j)|00010⟩+(-0.026+0.036j)|00011⟩
+(-0.088+0.026j)|00100⟩+(-0.102+0.268j)|00101⟩
+(-0.099+0.253j)|00110⟩+(-0.042+0.073j)|00111⟩
+(-0.087+0.052j)|01000⟩+(0.055+0.120j)|01001⟩
+(0.003+0.128j)|01010⟩+(-0.086+0.181j)|01011⟩
+(-0.110+0.241j)|01100⟩+(0.110+0.050j)|01101⟩
+(0.107+0.249j)|01110⟩+(-0.021+0.083j)|01111⟩
+(0.036+0.099j)|10000⟩+(0.119+0.257j)|10001⟩
+(-0.072+0.160j)|10010⟩+(-0.133+0.033j)|10011⟩
+(-0.119+0.121j)|10100⟩+(-0.113+0.213j)|10101⟩
+(-0.026+0.105j)|10110⟩+(-0.081+0.038j)|10111⟩
+(-0.125+0.246j)|11000⟩+(0.027+0.185j)|11001⟩
+(-0.009+0.112j)|11010⟩+(-0.037+0.204j)|11011⟩
+(-0.005+0.121j)|11100⟩+(-0.017+0.015j)|11101⟩
+(-0.126+0.214j)|11110⟩+(-0.091+0.228j)|11111⟩

E. The initial state of six-spin system for sequence generation

(0.067+0.060j)|000000⟩+(-0.094+0.072j)|000001⟩
+(-0.034+0.174j)|000010⟩+(-0.067+0.151j)|000011⟩
+(0.005+0.069j)|000100⟩+(-0.037+0.151j)|000101⟩
+(0.034+0.131j)|000110⟩+(0.023+0.038j)|000111⟩
+(-0.064+0.013j)|001000⟩+(-0.084+0.030j)|001001⟩
+(0.082+0.061j)|001010⟩+(0.051+0.146j)|001011⟩
+(0.043+0.022j)|001100⟩+(0.001+0.141j)|001101⟩
+(0.058+0.015j)|001110⟩+(0.065+0.014j)|001111⟩
+(-0.041+0.035j)|010000⟩+(0.007+0.131j)|010001⟩
+(0.002+0.126j)|010010⟩+(0.025+0.158j)|010011⟩
+(0.042+0.013j)|010100⟩+(-0.006+0.006j)|010101⟩
+(0.090+0.086j)|010110⟩+(0.093+0.022j)|010111⟩
+(0.082+0.154j)|011000⟩+(-0.011+0.114j)|011001⟩
+(-0.040+0.081j)|011010⟩+(-0.004+0.107j)|011011⟩
+(-0.088+0.199j)|011100⟩+(-0.053+0.074j)|011101⟩
+(0.093+0.159j)|011110⟩+(-0.088+0.119j)|011111⟩
+(0.055+0.134j)|100000⟩+(0.015+0.126j)|100001⟩
+(0.031+0.006j)|100010⟩+(0.069+0.094j)|100011⟩
+(0.003+0.180j)|100100⟩+(-0.058+0.109j)|100101⟩
+(-0.034+0.075j)|100110⟩+(0.071+0.022j)|100111⟩

+(0.031+0.153j)|101000⟩+(0.033+0.191j)|101001⟩
+(-0.046+0.193j)|101010⟩+(0.025+0.013j)|101011⟩
+(0.062+0.165j)|101100⟩+(-0.045+0.068j)|101101⟩
+(0.048+0.111j)|101110⟩+(0.001+0.065j)|101111⟩
+(-0.008+0.029j)|110000⟩+(-0.071+0.106j)|110001⟩
+(0.092+0.185j)|110010⟩+(-0.066+0.075j)|110011⟩
+(-0.001+0.129j)|110100⟩+(-0.075+0.030j)|110101⟩
+(-0.031+0.079j)|110110⟩+(0.005+0.056j)|110111⟩
+(0.088+0.080j)|111000⟩+(-0.076+0.084j)|111001⟩
+(0.029+0.003j)|111010⟩+(-0.072+0.074j)|111011⟩
+(-0.076+0.175j)|111100⟩+(-0.075+0.009j)|111101⟩
+(-0.009+0.187j)|111110⟩+(0.032+0.199j)|111111⟩

F. The initial state of seven-spin system for sequence generation

(0.067+0.007j)|0000000⟩+(0.018+0.119j)|0000001⟩
+(-0.013+0.033j)|0000010⟩+(0.050+0.115j)|0000011⟩
+(-0.039+0.098j)|0000100⟩+(-0.036+0.055j)|0000101⟩
+(-0.020+0.011j)|0000110⟩+(-0.005+0.073j)|0000111⟩
+(-0.004+0.125j)|0001000⟩+(-0.050+0.098j)|0001001⟩
+(0.010+0.111j)|0001010⟩+(0.056+0.085j)|0001011⟩
+(-0.008+0.092j)|0001100⟩+(0.017+0.003j)|0001101⟩
+(0.057+0.012j)|0001110⟩+(-0.040+0.058j)|0001111⟩
+(0.021+0.053j)|0010000⟩+(-0.047+0.053j)|0010001⟩
+(0.057+0.111j)|0010010⟩+(0.052+0.006j)|0010011⟩
+(0.010+0.044j)|0010100⟩+(-0.010+0.105j)|0010101⟩
+(0.052+0.128j)|0010110⟩+(-0.005+0.072j)|0010111⟩
+(-0.034+0.119j)|0011000⟩+(-0.015+0.083j)|0011001⟩
+(0.065+0.100j)|0011010⟩+(-0.018+0.009j)|0011011⟩
+(0.019+0.072j)|0011100⟩+(-0.065+0.129j)|0011101⟩
+(0.035+0.013j)|0011110⟩+(0.040+0.117j)|0011111⟩
+(0.011+0.114j)|0100000⟩+(0.059+0.039j)|0100001⟩
+(0.057+0.035j)|0100010⟩+(0.034+0.108j)|0100011⟩
+(0.066+0.131j)|0100100⟩+(-0.002+0.078j)|0100101⟩
+(0.009+0.108j)|0100110⟩+(0.020+0.087j)|0100111⟩
+(0.010+0.069j)|0101000⟩+(0.018+0.090j)|0101001⟩
+(-0.052+0.070j)|0101010⟩+(0.031+0.022j)|0101011⟩
+(0.001+0.106j)|0101100⟩+(-0.063+0.062j)|0101101⟩
+(-0.049+0.057j)|0101110⟩+(-0.014+0.043j)|0101111⟩
+(-0.048+0.029j)|0110000⟩+(0.047+0.049j)|0110001⟩
+(0.002+0.097j)|0110010⟩+(0.065+0.124j)|0110011⟩
+(0.026+0.034j)|0110100⟩+(-0.014+0.083j)|0110101⟩
+(0.021+0.080j)|0110110⟩+(-0.051+0.001j)|0110111⟩
+(-0.052+0.044j)|0111000⟩+(0.002+0.092j)|0111001⟩
+(0.053+0.050j)|0111010⟩+(0.063+0.013j)|0111011⟩
+(0.023+0.082j)|0111100⟩+(-0.014+0.001j)|0111101⟩
+(0.011+0.123j)|0111110⟩+(-0.025+0.066j)|0111111⟩
+(-0.038+0.019j)|1000000⟩+(-0.000+0.049j)|1000001⟩
+(-0.034+0.018j)|1000010⟩+(-0.056+0.056j)|1000011⟩
+(-0.028+0.047j)|1000100⟩+(-0.016+0.057j)|1000101⟩
+(0.032+0.094j)|1000110⟩+(0.056+0.114j)|1000111⟩
+(-0.006+0.132j)|1001000⟩+(0.042+0.071j)|1001001⟩
+(0.011+0.036j)|1001010⟩+(0.034+0.093j)|1001011⟩
+(0.040+0.047j)|1001100⟩+(-0.003+0.069j)|1001101⟩

+(-0.022+0.075j)|1001110⟩+(0.030+0.078j)|1001111⟩
+(-0.014+0.049j)|1010000⟩+(-0.059+0.120j)|1010001⟩
+(0.024+0.050j)|1010010⟩+(0.008+0.118j)|1010011⟩
+(-0.045+0.047j)|1010100⟩+(0.060+0.047j)|1010101⟩
+(0.030+0.089j)|1010110⟩+(-0.031+0.007j)|1010111⟩
+(0.066+0.051j)|1011000⟩+(0.035+0.001j)|1011001⟩
+(-0.010+0.055j)|1011010⟩+(-0.062+0.043j)|1011011⟩
+(0.039+0.015j)|1011100⟩+(-0.015+0.084j)|1011101⟩
+(0.018+0.124j)|1011110⟩+(0.021+0.029j)|1011111⟩
+(0.037+0.061j)|1100000⟩+(0.053+0.027j)|1100001⟩
+(0.059+0.127j)|1100010⟩+(-0.037+0.064j)|1100011⟩
+(-0.041+0.072j)|1100100⟩+(0.007+0.121j)|1100101⟩
+(-0.054+0.045j)|1100110⟩+(0.010+0.063j)|1100111⟩
+(-0.041+0.122j)|1101000⟩+(-0.064+0.089j)|1101001⟩
+(0.037+0.053j)|1101010⟩+(-0.050+0.085j)|1101011⟩
+(0.067+0.120j)|1101100⟩+(0.030+0.055j)|1101101⟩
+(-0.007+0.063j)|1101110⟩+(-0.062+0.022j)|1101111⟩
+(0.032+0.077j)|1110000⟩+(0.044+0.102j)|1110001⟩
+(0.067+0.116j)|1110010⟩+(0.061+0.009j)|1110011⟩
+(-0.062+0.122j)|1110100⟩+(-0.065+0.037j)|1110101⟩
+(-0.015+0.024j)|1110110⟩+(0.049+0.067j)|1110111⟩
+(-0.027+0.082j)|1111000⟩+(0.041+0.003j)|1111001⟩
+(0.062+0.074j)|1111010⟩+(-0.049+0.118j)|1111011⟩
+(0.034+0.121j)|1111100⟩+(0.022+0.063j)|1111101⟩
+(-0.008+0.132j)|1111110⟩+(-0.066+0.069j)|1111111⟩

* zhangxiao@mail.sysu.edu.cn

[1] X. Glorot and Y. Bengio, in *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (2010) pp. 249–256.
[2] F. A. Gers, J. Schmidhuber, and F. Cummins, (1999).
[3] R. K. Srivastava, K. Greff, and J. Schmidhuber, arXiv preprint arXiv:1505.00387 (2015).
[5] T. Lei, Y. Zhang, S. I. Wang, H. Dai, and Y. Artzi, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (2018) pp. 4470–4481.
[6] G. Wang, Y. Lin, and W. Yi, in *2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing* (IEEE, 2010) pp. 344–350.
[7] J. Filipovič, M. Madzin, J. Fousek, and L. Matyska, The Journal of Supercomputing **71**, 3934 (2015).
[8] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, The journal of machine learning research **15**, 1929 (2014).
[9] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, in *Advances in Neural Information Processing Systems* (2015) pp. 1171–1179.
[10] A. M. Lamb, A. G. A. P. Goyal, Y. Zhang, S. Zhang, A. C. Courville, and Y. Bengio, in *Advances In Neural Information Processing Systems* (2016) pp. 4601–4609.
[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, in *Advances in neural information processing systems* (2014) pp. 2672–2680.
[12] H. Guo, F. K. Soong, L. He, and L. Xie, arXiv preprint arXiv:1904.04775 (2019).
[4] R. J. Williams and D. Zipser, Neural computation **1**, 270 (1989).
[13] R. Moessner, S. L. Sondhi, and P. Chandra, Physical Review Letters **84**, 4457 (2000).
[14] D. P. Kingma and J. Ba, arXiv preprint arXiv:1412.6980 (2014).
[15] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, (2017).
[16] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)* (2016) pp. 265–283.
[17] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, in *Advances in Neural Information Processing Systems* (2016) pp. 4743–4751.