MIT Open Access Articles

Machine Learning Topological Invariants with Neural Networks

Massachusetts Institute of Technology

Machine Learning Topological Invariants with Neural Networks

Pengfei Zhang,[1] Huitao Shen,[2,*] and Hui Zhai[1,3,†]

[1]*Institute for Advanced Study, Tsinghua University, Beijing 100084, China*
[2]*Department of Physics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA*
[3]*Collaborative Innovation Center of Quantum Matter, Beijing 100084, China*

In this Letter we supervisedly train neural networks to distinguish different topological phases in the context of topological band insulators. After training with Hamiltonians of one-dimensional insulators with chiral symmetry, the neural network can predict their topological winding numbers with nearly 100% accuracy, even for Hamiltonians with larger winding numbers that are not included in the training data. These results show a remarkable success that the neural network can capture the global and nonlinear topological features of quantum phases from local inputs. By opening up the neural network, we confirm that the network does learn the discrete version of the winding number formula. We also make a couple of remarks regarding the role of the symmetry and the opposite effect of regularization techniques when applying machine learning to physical systems.

Recently, machine learning has emerged as a novel tool for studying physical systems and has demonstrated its ability in problems such as inferring numerical solutions [1–4], classifying phases [5–24], accelerating Monte Carlo algorithms [25–30], detecting entanglement [31], and controlling quantum dynamics [32–35]. Among all these applications, learning phases is a particularly intriguing one, as it paves a new route toward discovering new phases or even new physics without prior human knowledge [36]. Indeed, there have already been quite a few works on this direction of identifying phase transitions or even extracting order parameters unsupervisedly, i.e., without the awareness of any concept of phases [14–24].

Aside from the current success of machine learning phases within Landau's paradigm, topological phases are especially challenging to learn for several reasons. First, these phases are characterized by topological properties, e.g., the topological invariants, which are intrinsically nonlocal. Second, these topological invariants are nonlinear with respect to the field configuration. Third, topological invariants are intensive instead of extensive compared to the conventional order parameters. As a result, many commonly used techniques in machine learning turn out to be ineffective. For example, the intensiveness makes it futile to distinguish topological phases with the method of (kernel) principal component analysis [22].

The neural network is nonetheless a promising tool for learning topological phases due to its great expressibility and versatility. Mathematically, these networks are able to approximate any continuous functions if the number of fitting parameters can grow indefinitely [37,38]. This great expressibility, together with the development of many effective training algorithms [39–43], makes the neural network an indispensable ingredient in the boom of modern machine learning [44]. In this Letter, we report that properly designed neural networks can successfully learn topological invariants for topological band insulators [45–49]. Our formalism and results possess the following key features that make them significantly beyond those in the existing works on this topic: (i) The input data are completely local; (ii) our study is not restricted to any specific model in the symmetry class; (iii) our neural network has generalization power after training. We will elaborate these points in the following.

To be concrete, we consider one-dimensional topological band insulators of the AIII symmetry class [47–49]. The general form of such two-band Hamiltonians is $H(k) = \mathbf{h}(k) \cdot \boldsymbol{\sigma}$, where $\boldsymbol{\sigma} \equiv (\sigma_x, \sigma_y, \sigma_z)$ is the vector of Pauli matrices. The chiral symmetry in AIII class requires $SH(k)S^{-1} = -H(k)$. Without loss of generality, we can always choose $S = \sigma_z$ so that only h_x and h_y are nonzero. In our study, we feed neural networks directly with normalized Hamiltonians $\tilde{H}(k) \equiv \tilde{h}_x(k)\sigma_x + \tilde{h}_y(k)\sigma_y$ at L points discretized uniformly along the Brillouin zone. Here $\tilde{h}_i(k) \equiv h_i(k)/|\mathbf{h}(k)|$, $i = x, y$. In other words, the input data are $(L + 1) \times 2$ matrices of the form

$$\begin{pmatrix} \tilde{h}_x(0) & \tilde{h}_x(2\pi/L) & \tilde{h}_x(4\pi/L) & \dots & \tilde{h}_x(2\pi) \\ \tilde{h}_y(0) & \tilde{h}_y(2\pi/L) & \tilde{h}_y(4\pi/L) & \dots & \tilde{h}_y(2\pi) \end{pmatrix}^T. \quad (1)$$

In reality, these input data are available in quantum simulators [50]. In the following, we choose $L = 32$ and confirm all our results are insensitive to L as long as $L \geq 32$.

The topological invariant for the AIII class is the winding number, as the first homotopy group of a circle $\pi_1(S^1)$. It is

defined for a continuous mapping $S^1 \to S^1 : k \mapsto U(k)$, $k \in [0, 2\pi]$. $|U(k)| = 1$ and $U(k + 2\pi) = U(k)$. For the Hamiltonians given above, we identify $U(k) = \tilde{h}_x(k) + i\tilde{h}_y(k)$. Intuitively, the winding number $w \in \mathbb{Z}$ is an integer that counts how many times $U(k)$ winds around the origin when k changes from 0 to 2π. Its sign denotes the clockwise ($w < 0$) or the anticlockwise ($w > 0$) winding. The winding number could be formally computed as

$$w = -\frac{i}{2\pi} \oint_0^{2\pi} U^*(k) \partial_k U(k) dk. \quad (2)$$

For discretized $U(k)$, this reduces to

$$w = -\frac{1}{2\pi} \sum_{n=1}^{L} \Delta\theta(n), \quad (3)$$

where $\Delta\theta(n) \equiv [\theta(n) - \theta(n-1)] \mod 2\pi$ so that $\Delta\theta(n) \in [-\pi, \pi)$ and $\theta(n) \equiv \arg[U(2\pi n/L)]$.

Our machine learning workflow is shown schematically in Fig. 1. The output of the neural network is a real number $\tilde{w}$, and the predicted winding number is interpreted as the integer that is closest to $\tilde{w}$. Notice that the input data of form Eq. (1) is completely local and highly nonlinear with respect to the formula Eq. (3). We first train neural networks with both Hamiltonians and their corresponding winding numbers. At the testing stage, we feed only the Hamiltonians to the neural networks and compare their predictions with the winding numbers computed by Eq. (3), from which we determine the percentage of the correct predictions as accuracy. The details of the networks and the training can be found in the Supplemental Material [51].
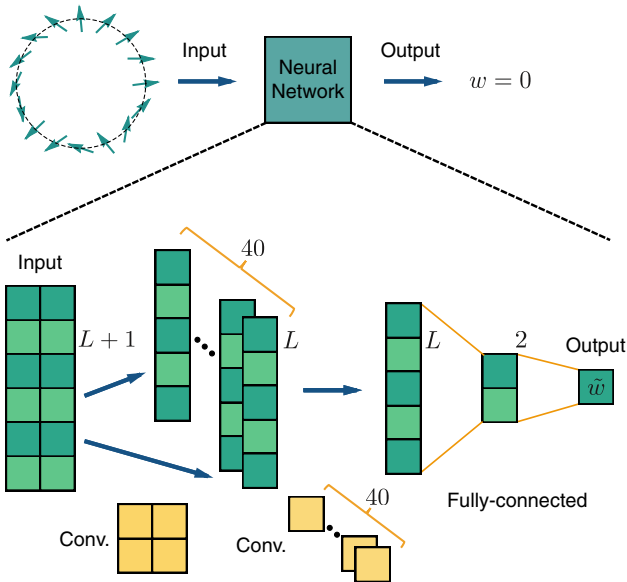

FIG. 1. Schematic of the machine learning workflow and the structure of the convolutional neural network. The input Hamiltonians are represented by vectors $\tilde{\mathbf{h}}(k) \equiv (\tilde{h}_x(k), \tilde{h}_y(k))$, where $k \in [0, 2\pi]$ is in the Brillouin zone.

The Su-Schrieffer-Heeger (SSH) model [52] is one of the most simple and widely studied models within the AIII symmetry class, whose Hamiltonian is

$$H_{\text{SSH}}(k) = (t + t' \cos k)\sigma_x + (t' \sin k)\sigma_y. \quad (4)$$

This model hosts two topologically distinct gapped phases with winding number $w = 0$ for $t > t'$ and $w = 1$ for $t < t'$, respectively. We first report the results when the training data are restricted within this model.

The training set consists of 10^5 SSH Hamiltonians whose $(t - t')/t$ are uniformly distributed within $[-10, 10]$, and the test set consists of 10^4 similar Hamiltonians that are not included in the training set. Surprisingly, even the most simple neural network with no hidden layer nor nonlinear activation function—essentially a linear model used for linear regression—can correctly compute the winding number with nearly 100% accuracy in the test set after only one training epoch. Further increasing the network complexity by introducing a hidden layer will push the accuracy to exactly 100%. However, if we test these networks with more general Hamiltonians of winding number $w = 0, 1$, the accuracy sharply drops to around 50%, which is just the accuracy of blind guesses. This situation could not be improved by increasing model complexity or using more sophisticated neural networks.

Obviously, these networks compute the winding number with a shortcut that is dedicated to SSH Hamiltonians and is only linear with respect to the input data. In fact, due to the additional inversion symmetry in the SSH model $H_{\text{SSH}}(k) = \sigma_x H_{\text{SSH}}(-k)\sigma_x$, one can read out the winding number directly from the Hamiltonian at the high symmetry point $k = \pi$:

$$w = 0 \leftrightarrow \tilde{\mathbf{h}}(\pi) = (1, 0),$$
$$w = 1 \leftrightarrow \tilde{\mathbf{h}}(\pi) = (-1, 0). \quad (5)$$

This local feature is exactly what the networks exploited, for they can predict the winding number perfectly even for $L = 2$, where only $\mathbf{h}'(0)$ and $\mathbf{h}'(\pi)$ are present.

The lesson is that, if the training data are restricted to some certain model, the neural network would only exploit less universal features of this specific model instead of the universal ones. In the above example, the neural networks do not learn the general formula Eq. (3), but "cleverly" reduce Eq. (3) to Eq. (5). Therefore, they fail to make any correct prediction for Hamiltonians not respecting the inversion symmetry.

To examine whether the neural networks have the ability to learn the winding number in its most general form, we generate training data with the most general one-dimensional Hamiltonians with chiral symmetry,

$$H(k) = h_x(k)\sigma_x + h_y(k)\sigma_y, \quad (6)$$

where $h_i(k)$, $i = x, y$ are periodic functions in k expanded by the Fourier series

$$h_i(k) = \sum_{n=0}^{c} [a_{i,n} \cos(nk) + b_{i,n} \sin(nk)]. \qquad (7)$$

c is a cutoff that determines the highest possible winding number of the Hamiltonian, and is set to $c = 4$ in the following. $a_{i,n}$, $b_{i,n}$ are randomly sampled from a uniform distribution within $[-1, 1]$. Among 10^5 training Hamiltonians, 37%, 50%, and 13% of them having winding numbers $w = 0, \pm1$, and ±2, respectively. Different cutoff c and the distribution of w of the training data will not affect the network performance (See Supplemental Material [51]).

We consider two classes of neural networks: the fully connected network and the convolutional network [53]. The fully connected network has three hidden layers with 40, 32, and 2 neurons, respectively. The total number of trainable parameters is 4061. The convolutional network has two convolutional layers with 40 kernels of size 2×2 and 1 kernel of size 1×1, followed by a fully connected layer of 2 neurons before the output layer. The total number of trainable parameters is 310. The structure of the convolutional network is shown in Fig. 1. All the hidden layers have rectified linear units $f(x) = \max\{0, x\}$ as activation functions and the output layer has linear activation $f(x) = x$.

We test these networks with four different test sets, schematically shown in Fig. 2. (i) 10^4 Hamiltonians with winding numbers $w \in \{\pm2, \pm1, 0\}$ that are not included in the training set. (ii) 10^4 Hamiltonians with the following functional form:

$$h_x(k) = \theta(\pi - k) \cos f_1(k)$$
$$\qquad + \theta(k - \pi) \cos[-f_2(k - \pi) + f_1(\pi)],$$
$$h_y(k) = \theta(\pi - k) \sin f_1(k)$$
$$\qquad + \theta(k - \pi) \sin[-f_2(k - \pi) + f_1(\pi)], \qquad (8)$$
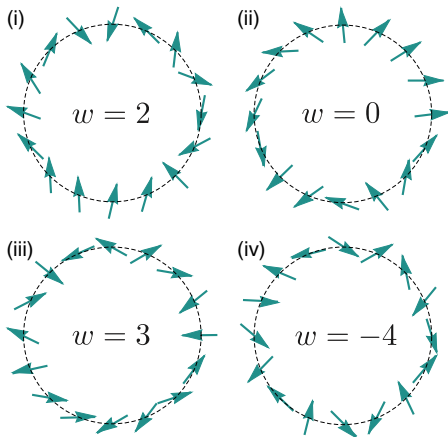

FIG. 2. Typical Hamiltonians arranged in the Brillouin zone in the test set (i)–(iv). Here $L = 16$.

where $\theta(x)$ is the Heaviside step function, $f_1(k)$ and $f_2(k)$ are monotonic increasing functions bounded by $f_1(0) = f_2(0) = 0$, and $f_1(\pi) = f_2(\pi) \leq c\pi$. Intuitively, the Hamiltonian first winds the circle anticlockwisely during $k \in [0, \pi]$, then clockwisely winds back during $k \in [\pi, 2\pi]$. The resulting winding numbers should always be zero. (iii) 10^4 Hamiltonians with winding numbers $w = \pm3$. (iv) 10^4 Hamiltonians with winding numbers $w = \pm4$.

The test results are presented in Table I. The convolutional network works generally better than the fully connected network. The Hamiltonian configurations in test (ii) have a strong local twist at $k = \pi$, but the global topological numbers are always zero. That both neural networks perform well in this test is an indication that they have learned the global structures in the data instead of the local features. Surprisingly, the convolutional network can perform extremely well even in tests (iii) and (iv), which consist of Hamiltonians with larger winding numbers not seen by neural networks during the training. The fact that the convolutional network can pass test (iii) and (iv) shows that it has generalization power, and is also a strong indication that it really learns the general formula for the winding number.

Open the black box.—Inspired by its performance, we open up the black box of the convolutional network and explore what it learns. Mathematically, our convolutional network can be described by the composition of the following functions:

(i) The first layer performs $N = 40$ different convolutions with respect to the input Hamiltonians using the 2×2 kernel A^i, $i = 1, \ldots, N$:

$$\tilde{B}^i(n) = A_{11}^i \tilde{h}_x(2\pi(n-1)/L) + A_{12}^i \tilde{h}_y(2\pi(n-1)/L)$$
$$\qquad + A_{21}^i \tilde{h}_x(2\pi n/L) + A_{22}^i \tilde{h}_y(2\pi n/L) + A_0^i, \qquad (9)$$

for $n = 1, \ldots, L$, followed by $B^i(n) = f(\tilde{B}^i(n))$, where $f(x)$ is the activation function.

(ii) The second layer performs another linear mapping across different kernels and is diagonal in n, i.e.,

$$\tilde{D}(n) = \sum_{i=1}^{N} c^i B^i(n) + c^0, \qquad (10)$$

followed by $D(n) = f(\tilde{D}(n))$.

TABLE I. Performance (accuracy with respect to different test sets) of neural networks for learning topological phases in general models.

Network	Test (i)	Test (ii)	Test (iii)	Test (iv)
Fully connected	82.2%	99.1%	22.8%	1.4%
Convolutional	99.6%	100.0%	98.2%	99.3%
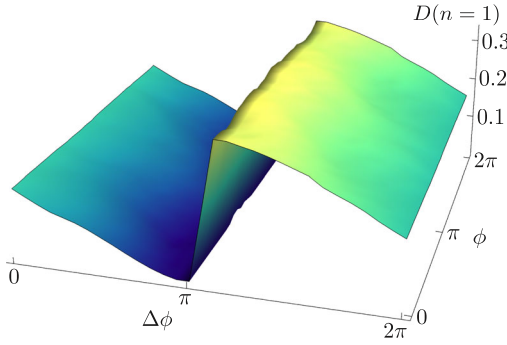
FIG. 3. $D(n = 1)$ as a function of ϕ and $\Delta\phi$ when the input is Eq. (14).

FIG. 4. The performance of the convolutional network on various test sets with respect to the L_2 regularization strength. Solid lines, without introducing noise; inset dashed lines, randomly adding ± 1 to the label of the winding number ω for 4% of the training data to mimic noise.

(iii) Finally, the L-dimensional vector $D(n)$ is mapped to the winding number $\tilde{w}$ through

$$\tilde{F}_\eta = \sum_{n=1}^{L} M_{\eta n} D(n) + N_\eta, \qquad \eta = 1, 2, \qquad (11)$$

$$F_\eta = f(\tilde{F}_\eta), \qquad (12)$$

$$\tilde{w} = \sum_{\eta=1}^{2} P_\eta F_\eta + Q. \qquad (13)$$

All above, A^i, c^i, $M_{\eta n}$, N_η, P_η, and Q are fitting parameters that are determined during the training. If the neural network successfully learns the discrete version of the winding number formula Eq. (3), we should expect $D(n)$ reproduces $\Delta\theta(n)$ and then the rest of the layers are basically summing over all $\Delta\theta(n)$. To verify this, we consider the input Hamiltonian

$$\begin{pmatrix} \cos\phi & \cos(\phi + \Delta\phi) & \dots \\ \sin\phi & \sin(\phi + \Delta\phi) & \dots \end{pmatrix}^T, \qquad (14)$$

and expect

$$D(n = 1) \propto -\frac{1}{2\pi} \begin{cases} \Delta\phi, & 0 \le \phi < \pi, \\ \Delta\phi - 2\pi, & \pi < \phi \le 2\pi. \end{cases} \qquad (15)$$

In Fig. 3 we show $D(n = 1)$ as a function of ϕ and $\Delta\phi$. It is very clear that, except for little fluctuations, $D(n = 1)$ is independent of ϕ and depends on $\Delta\phi$ with the same function form as Eq. (15).

With the above analysis, we can gain some understanding why our neural network has great generalization power. The convolutional layers that extract local windings are universal, and are unaffected by the global winding numbers of the data. As long as the training Hamiltonians are enough to cover the full surface of Fig. 3, the convolutional layers are always interpolating Eq. (15) instead of extrapolating it, however large the global winding numbers are. Extrapolation only happens in the last two layers when $\Delta\theta$ is summed. This is only a linear extrapolation, and is
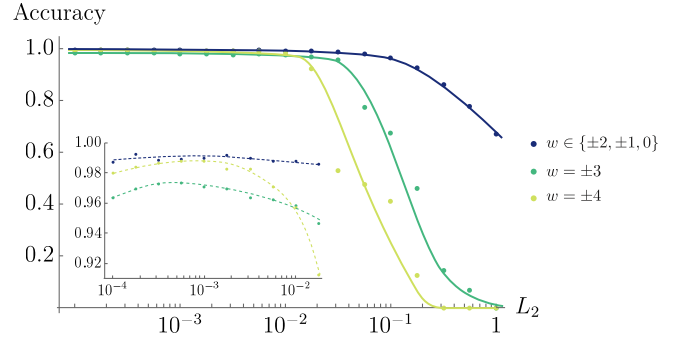
relatively easy for neural networks. In this way, the trained convolutional network computes winding numbers through the discrete version of the winding number formula Eq. (3).

Regularization techniques.—Finally, we remark on the regularization technique, which is usually considered necessary in training neural networks in order to avoid overfitting and to enhance networks' generalization power [40,42,53]. However, in our case we find the result to be contrary. In Fig. 4, the ability of the network to compute larger winding numbers decays rapidly with the L_2 regularization strength, although the network could still very accurately compute winding numbers that are within the same range as the training set [54]. We attribute this phenomenon to the lack of noise. The data used here are generated by randomly sampling Hamiltonians [55], where there is much less noise, if noise exists at all. However, imagine the training data are taken directly from experiments. In this scenario the noise should exist and regularization should be useful. Indeed, this is demonstrated to be true in Fig. 4 if we artificially introduce noise into the training data. The situation is similar when data are generated by Monte Carlo sampling [5–7,13–15,17–24], where thermal noise may exist and regularization will be useful [56].

Concluding remarks.—In summary, we successfully train a neural network that learns global and nonlinear topological features from a large data set of Hamiltonians in the momentum space. We illustrate that our neural network has great generalization power to correctly compute larger winding numbers not seen in the training data. By analyzing the neural network, we confirm that our network does learn the discrete version of the winding number formula. Our network can directly be used to analyze the data from quantum simulators [50]. It is also possible to generalize our results to the topological model in higher dimension and other classes [57]. We hope this work opens up a lot of possibilities of using machine learning to study rich topological physics.

Before concluding, we would like to make a couple of remarks on the role of symmetry when applying machine learning to physical problems. First, the symmetry of the training data matters. In order for neural networks to learn general rules, the training data have to be as general as possible to avoid unnecessary symmetry constraints. As demonstrated by the counterexample of learning the SSH model, the neural network exploits the inversion symmetry and learns a shortcut to the winding number. Second, the symmetry of the neural network matters. The structure of the neural network should be designed to be compatible with the symmetry of the targeting physics law. It is tempting to ask why the convolutional network performs better than the fully connected network, as shown in Table I, even though the later has more trainable parameters and hence greater expressibility in principle. This is because the translation of Hamiltonian configurations in the momentum space does not change the winding number. In practice, the translational symmetry is hard to be rediscovered for the fully connected network during training. The convolutional network, on the other hand, respects this symmetry explicitly, reducing the redundancy in the parametrization. Thus, it is easier for the training algorithm to find the optimal fitting parameters. Furthermore, the winding number formula is the summation of many local phase winding $\Delta\theta$. The convolutional layer takes this notion of locality directly through the 2×2 kernels. As a result, the convolutional network performs better than the fully connected network.

*huitao@mit.edu
†hzhai@mail.tsinghua.edu.cn

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

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

[3] K. Mills, M. Spanner, and I. Tamblyn, Phys. Rev. A 96, 042113 (2017).

[4] K. Mills and I. Tamblyn, arXiv:1706.09779.

[5] J. Carrasquilla and R. G. Melko, Nat. Phys. 13, 431 (2017).

[6] P. Broecker, J. Carrasquilla, R. G. Melko, and S. Trebst, Sci. Rep. 7, 8823 (2017).

[7] K. Ch'ng, J. Carrasquilla, R. G. Melko, and E. Khatami, Phys. Rev. X 7, 031038 (2017).

[8] Y. Zhang and E.-A. Kim, Phys. Rev. Lett. 118, 216401 (2017).

[9] Y. Zhang, R. G. Melko, and E.-A. Kim, Phys. Rev. B 96, 245119 (2017).

[10] T. Ohtsuki and T. Ohtsuki, J. Phys. Soc. Jpn. 85, 123706 (2016).

[11] T. Ohtsuki and T. Ohtsuki, J. Phys. Soc. Jpn. 86, 044708 (2017).

[12] F. Schindler, N. Regnault, and T. Neupert, Phys. Rev. B 95, 245134 (2017).

[13] P. Ponte and R. G. Melko, Phys. Rev. B 96, 205146 (2017).

[14] L. Wang, Phys. Rev. B 94, 195105 (2016).

[15] A. Tanaka and A. Tomiya, J. Phys. Soc. Jpn. 86, 063001 (2017).

[16] E. P. L. van Nieuwenburg, Y.-H. Liu, and S. D. Huber, Nat. Phys. 13, 435 (2017).

[17] Y.-H. Liu and E. P. L. van Nieuwenburg, arXiv:1706.08111.

[18] S. J. Wetzel, Phys. Rev. E 96, 022140 (2017).

[19] S. J. Wetzel and M. Scherzer, Phys. Rev. B 96, 184410 (2017).

[20] W. Hu, R. R. P. Singh, and R. T. Scalettar, Phys. Rev. E 95, 062122 (2017).

[21] N. C. Costa, W. Hu, Z. J. Bai, R. T. Scalettar, and R. R. P. Singh, Phys. Rev. B 96, 195138 (2017).

[22] C. Wang and H. Zhai, Phys. Rev. B 96, 144432 (2017).

[23] P. Broecker, F. F. Assaad, and S. Trebst, arXiv:1707.00663.

[24] K. Ch'ng, N. Vazquez, and E. Khatami, arXiv:1708.03350 [Phys. Rev. E (to be published)].

[25] J. Liu, Y. Qi, Z. Y. Meng, and L. Fu, Phys. Rev. B 95, 041101 (2017).

[26] J. Liu, H. Shen, Y. Qi, Z. Y. Meng, and L. Fu, Phys. Rev. B 95, 241104 (2017).

[27] X. Y. Xu, Y. Qi, J. Liu, L. Fu, and Z. Y. Meng, Phys. Rev. B 96, 041119 (2017).

[28] Y. Nagai, H. Shen, Y. Qi, J. Liu, and L. Fu, Phys. Rev. B 96, 161102 (2017).

[29] L. Huang, Y.-F. Yang, and L. Wang, Phys. Rev. E 95, 031301 (2017).

[30] L. Huang and L. Wang, Phys. Rev. B 95, 035105 (2017).

[31] S. Lu, S. Huang, K. Li, J. Li, J. Chen, D. Lu, Z. Ji, Y. Shen, D. Zhou, and B. Zeng, arXiv:1705.01523.

[32] C. Chen, D. Dong, H. X. Li, J. Chu, and T. J. Tarn, IEEE Trans. Neural Networks Learning Syst. 25, 920 (2014).

[33] P. Palittapongarnpim, P. Wittek, E. Zahedinejad, S. Vedaie, and B. C. Sanders, Neurocomputing 268, 116 (2017).

[34] M. Bukov, A. G. R. Day, D. Sels, P. Weinberg, A. Polkovnikov, and P. Mehta, arXiv:1705.00565.

[35] V. Dunjko and H. J. Briegel, arXiv:1709.02779.

[36] M. Schmidt and H. Lipson, Science 324, 81 (2009).

[37] G. Cybenko, Math. Control Signals Syst. 2, 303 (1989).

[38] K. Hornik, Neural Netw. 4, 251 (1991).

[39] X. Glorot and Y. Bengio, in Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, edited by Y. W. T. Titterington and Mike (PMLR, Chia Laguna Resort, Sardinia, Italy, 2010), Vol. 9, p. 249.

[40] Y. Bengio, in Practical Recommendations for Gradient-Based Training of Deep Architectures, edited by G. Montavon, G. B. Orr, and K. R. Müller. Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science Vol. 7700 (Springer, Berlin, Heidelberg, 2012), p. 437.

[41] D. P. Kingma and J. Ba, arXiv:1412.6980.

[42] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, J. Mach. Learn. Res. 15, 1929 (2014).

[43] S. Ioffe and C. Szegedy, in Proceedings of 32nd International Conference on Machine Learning, edited by

D. Blei and F. Bach (PMLR, Lille, France, 2015), Vol. 37, p. 448.

[44] Y. LeCun, Y. Bengio, and G. Hinton, Nature (London) **521,** 436 (2015).

[45] C. L. Kane and E. J. Mele, Phys. Rev. Lett. **95,** 146802 (2005).

[46] J. E. Moore and L. Balents, Phys. Rev. B **75,** 121306 (2007).

[47] A. P. Schnyder, S. Ryu, A. Furusaki, and A. W. W. Ludwig, Phys. Rev. B **78,** 195125 (2008).

[48] A. Kitaev, AIP Conf. Proc. **1134,** 22 (2009).

[49] C.-K. Chiu, J. C. Y. Teo, A. P. Schnyder, and S. Ryu, Rev. Mod. Phys. **88,** 035005 (2016).

[50] N. Flaschner, B. S. Rem, M. Tarnowski, D. Vogel, D.-S. Luhmann, K. Sengstock, and C. Weitenberg, Science **352,** 1091 (2016).

[51] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevLett.120.066401 for Details of neural networks and their training process.

[52] W. P. Su, J. R. Schrieffer, and A. J. Heeger, Phys. Rev. Lett. **42,** 1698 (1979).

[53] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, Cambridge, MA, 2016), http://www.deeplearningbook.org.

[54] All the results in this section are similar for other regularization techniques such as weight clipping.

[55] See also Ref. [16].

[56] Still, this depends on what to learn. Imagine the goal is to extract temperature from the Markov chain dynamics, the "thermal noise" is not noise, but instead important features.

[57] N. Sun *et al.* (to be published).