Machine Learning Topological Invariants with Neural Networks

The MIT Faculty has made this article openly available. Please share how this access benefits you. Your story matters.

<table>
<thead>
<tr>
<th>Citation</th>
<th>Zhang, Pengfei et al. &quot;Machine Learning Topological Invariants with Neural Networks.&quot; Physical Review Letters 120, 6 (February 2018): 066401 © 2018 American Physical Society</th>
</tr>
</thead>
<tbody>
<tr>
<td>As Published</td>
<td><a href="http://dx.doi.org/10.1103/PhysRevLett.120.066401">http://dx.doi.org/10.1103/PhysRevLett.120.066401</a></td>
</tr>
<tr>
<td>Publisher</td>
<td>American Physical Society</td>
</tr>
<tr>
<td>Version</td>
<td>Final published version</td>
</tr>
<tr>
<td>Accessed</td>
<td>Thu Apr 04 18:10:07 EDT 2019</td>
</tr>
<tr>
<td>Citable Link</td>
<td><a href="http://hdl.handle.net/1721.1/114406">http://hdl.handle.net/1721.1/114406</a></td>
</tr>
<tr>
<td>Terms of Use</td>
<td>Article is made available in accordance with the publisher's policy and may be subject to US copyright law. Please refer to the publisher's site for terms of use.</td>
</tr>
<tr>
<td>Detailed Terms</td>
<td></td>
</tr>
</tbody>
</table>
Machine Learning Topological Invariants with Neural Networks

Pengfei Zhang,1 Huitao Shen,2,* and Hui Zhai1,3,1
1Institute for Advanced Study, Tsinghua University, Beijing 100084, China
2Department of Physics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA
3Collaborative Innovation Center of Quantum Matter, Beijing 100084, China

(Received 6 September 2017; revised manuscript received 4 December 2017; published 6 February 2018)

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.

DOI: 10.1103/PhysRevLett.120.066401

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 paradigm, topological phases are the general form of such two-band Hamiltonians is \( H(k) = \mathbf{h}(k) \cdot \mathbf{\sigma} \), where \( \mathbf{\sigma} \equiv (\sigma_x, \sigma_y, \sigma_z) \) is the vector of Pauli matrices. The chiral symmetry in AIII class requires \( S H(k)S^{-1} = -H(k) \). 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

\[
\begin{pmatrix}
\tilde{h}_x(0) & \tilde{h}_x(2\pi/L) & \tilde{h}_x(4\pi/L) & \cdots & \tilde{h}_x(2\pi) \\
\tilde{h}_y(0) & \tilde{h}_y(2\pi/L) & \tilde{h}_y(4\pi/L) & \cdots & \tilde{h}_y(2\pi)
\end{pmatrix}^T
\]  

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 $S^1 \rightarrow S^1 : k \mapsto U(k)$, $k \in [0, 2\pi]$. $|U(k)| = 1$ and $U(k + 2\pi) = U(k)$. For the Hamiltonians given above, we identify $U(k) = \tilde{h}_x(k) + \imath \tilde{h}_y(k)$. 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 $2\pi$. Its sign denotes the clockwise ($w < 0$) or the anticlockwise ($w > 0$) winding. The winding number could be formally computed as

$$w = -\frac{i}{2\pi} \int_0^{2\pi} U^*(k) \partial_k U(k) dk.$$  

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

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

where $\Delta \theta(n) = [\theta(n) - \theta(n-1)] \mod 2\pi$ 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 $w$, and the predicted winding number is interpreted as the integer that is closest to $w$. 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 with the winding numbers computed by Eq. (3), 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 $w = 0$, 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 $H_{\text{SSH}}(k) = \sigma_y H_{\text{SSH}}(-k) \sigma_y$, one can read out the winding number directly from the Hamiltonian at the high symmetry point $k = \pi$:

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

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

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

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

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

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

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

$c$ 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 $w = 0$, $\pm 1$, and $\pm 2$, 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 $2 \times 2$ and 1 kernel of size $1 \times 1$, 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 $w \in \{\pm 2, \pm 1, 0\}$ that are not included in the training set. (ii) $10^4$ Hamiltonians with the following functional form:

$$\begin{align*}
h_x(k) &= \theta(\pi - k) \cos f_1(k) \\
&\quad + \theta(k - \pi) \cos[-f_2(k - \pi) + f_1(\pi)], \\
h_y(k) &= \theta(\pi - k) \sin f_1(k) \\
&\quad + \theta(k - \pi) \sin[-f_2(k - \pi) + f_1(\pi)],
\end{align*}$$

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 clockwise winds back during $k \in [\pi, 2\pi]$. The resulting winding numbers should always be zero. (iii) $10^4$ Hamiltonians with winding numbers $w = \pm 3$. (iv) $10^4$ Hamiltonians with winding numbers $w = \pm 4$.

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 $2 \times 2$ kernel $A_i$, $i = 1, \ldots, N$:

$$\begin{align*}
\tilde{B}^i(n) &= A_{i1}^x \tilde{h}_x(2\pi(n-1)/L) + A_{i1}^y \tilde{h}_y(2\pi(n-1)/L) \\
&\quad + A_{i2}^x \tilde{h}_x(2\pi n/L) + A_{i2}^y \tilde{h}_y(2\pi n/L) + A_{i0},
\end{align*}$$

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.,

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

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.

<table>
<thead>
<tr>
<th>Network</th>
<th>Test (i)</th>
<th>Test (ii)</th>
<th>Test (iii)</th>
<th>Test (iv)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully connected</td>
<td>82.2%</td>
<td>99.1%</td>
<td>22.8%</td>
<td>1.4%</td>
</tr>
<tr>
<td>Convolutional</td>
<td>99.6%</td>
<td>100.0%</td>
<td>98.2%</td>
<td>99.3%</td>
</tr>
</tbody>
</table>
(iii) Finally, the $L$-dimensional vector $D(n)$ is mapped to the winding number $\tilde{w}$ through

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

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

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

All above, $A^i$, $c^i$, $M_{\eta n}$, $N_{\eta}$, $P_\eta$, 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

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

and expect

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

In Fig. 3 we show $D(n = 1)$ as a function of $\phi$ and $\Delta \phi$. It is very clear that, except for little fluctuations, $D(n = 1)$ is independent of $\phi$ 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 target 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 $2 \times 2$ kernels. As a result, the convolutional network performs better than the fully connected network.

This work is supported by MOST under Grant No. 2016YFA0301600, and NSFC Grants No. 11325418 and No. 11734010. H. S. thanks IASTU for hosting his visit to Beijing, where the key parts of this work were done. H. S. is supported by the MIT Alumni Fellowship Fund For Physics.

\[\text{huitao@mit.edu}\]
\[\text{hzhai@mail.tsinghua.edu.cn}\]


[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).