Deep learning of topological phase transitions from entanglement aspects

Yuan-Hong Tsai,^{1,*} Meng-Zhe Yu², Yu-Hao Hsu², and Ming-Chiang Chung^{2,3,4,†}

¹Taiwan AI Academy, New Taipei, 241, Taiwan

²Physics Department, National Chung-Hsing University, Taichung, 40227, Taiwan

³Max-Planck-Institute for Physics of Complex Systems, Nöthnitzer Straße 38 01187, Dresden, Germany

⁴Physics Department, Northeastern University, 360 Huntington Avenue, Boston, Massachusetts 02115, USA

(Received 22 January 2020; revised 9 June 2020; accepted 16 July 2020; published 18 August 2020)

The one-dimensional p -wave superconductor proposed by Kitaev has long been a classic example for understanding topological phase transitions through various methods, such as examining the Berry phase, edge states of open chains, and, in particular, aspects from quantum entanglement of ground states. In order to understand the amount of information carried in the entanglement-related quantities, here we study topological phase transitions of the model with emphasis of using the deep learning approach. We feed different quantities, including Majorana correlation matrices (MCMs), entanglement spectra (ES) or entanglement eigenvectors (EE) originating from Block correlation matrices, into the deep neural networks for training, and investigate which one could be the most useful input format in this approach. We find that ES is information that is too compressed compared to MCM or EE. MCM and EE can provide us abundant information to recognize not only the topological phase transitions in the model but also phases of matter with different $U(1)$ gauges, which is not reachable by using ES only.

DOI: [10.1103/PhysRevB.102.054512](https://doi.org/10.1103/PhysRevB.102.054512)

I. INTRODUCTION

Going beyond Ginzburg-Landau theory of phase transitions [1], a topological phase transition (TPT) can occur when no symmetry is broken in a physical system. Since the discovery of the integer quantum Hall effect [2], an example of TPTs, various materials potentially exhibiting TPTs have been proposed in recent years, although only some of them have been confirmed experimentally [3,4]. Among them, the one-dimensional (1D) topological p -wave superconductor (p -SC) proposed by Kitaev [5] has become one of the most interesting proposals due to the fact that the edge modes in these superconductors can be viewed as “Majorana fermions”, whose antiparticle is the particle itself. In particular, they are essential components in forming practical fault-tolerant quantum computers [6]. To realize such a superconducting state, clever combinations of topological materials or semiconductors with an ordinary s -wave superconductor through proximity effect have been proposed [7]. In the past few years, Majorana fermions/zero modes have been claimed found at the edges of certain systems either by directly observing the scanning tunneling microscope spectroscopy [8–10] or indirectly by measuring the 4π periodicity of Josephson junction currents [11].

In fact, Majorana zero modes can be alternatively detected by measuring quantum entanglement, a concept in the subject of quantum information. For instance, the von Neumann entanglement entropy of a subsystem A embedded in the environment B can be obtained via the density matrix after tracing out the degrees of freedom in B , *i.e.*, the subentropy $S_A = -\text{Tr} \rho_A \log_2 \rho_A$ with the reduced density matrix $\rho_A = \text{Tr}_B |\Psi_{AUB}\rangle\langle\Psi_{AUB}|$ [12–16]. To reveal the topological nature of the system, one can further analyze the eigenvalues of the ρ_A -deduced entanglement Hamiltonian: The presence of Majorana zero (edge) modes can now be inferred by a corresponding degeneracy in the entanglement spectrum. In other words, the spectrum preserves topological information. Moreover, for any system with a quadratic Hamiltonian, the whole situation can be simplified to compute the eigenvalues of the correlation function matrix, also known as the one-particle entanglement spectrum [17–19]. Majorana zero modes are then shown in terms of doubly degenerate eigenvalues, $1/2$, in the spectrum [18]. This example reflects the power of the entanglement-related quantities as a good topological indicator, but sometimes computation of these quantities is time consuming and measuring entanglement is experimentally difficult. And hence, a more efficient tool for identifying TPTs might be necessary.

Machine learning (ML) is a rapidly growing field of computer science recently due to the availability of large-scaled datasets and advances in the computation hardware. Its applications have become ubiquitous in our daily life from automated machine translation, vision and speech recognition, matching news items, to email spam-filter and so on [20]. By feeding in a large amount of data (or “features”), ML algorithms can “learn” to condense them into a more accessible/meaningful form, such as distinguishable classes

*yhung.tsai@gmail.com

†mingchiangha@nchu.edu.tw

or patterns. In particular, a neural network-based learning method, called deep learning (DL), are composed of several simple but nonlinear modules and able to effectively learn suitable representations from complex raw data and distill essential information. Therefore, as a straightforward application on quantum matters, so far DL has been employed by physicists to classify different phases of matter and identify phase boundaries [21–32]. Moreover, the remarkable property of this method is also shown in considering topological [33–35] or out-of-equilibrium systems [36] where no obvious local order parameter is available.

Despite recent progress in using DL for identifying phase transitions, many of them are based on straightforward wave functions (by certain manipulation or not) as the input data for learning, while relatively few are based on the aspect of quantum information. Since quantum information is also known to be useful when there is no local order parameter available in a system, one pioneering work takes the entanglement spectrum (ES), used to compress the ground state information, as the input data and trains a neural network to distinguish the topological phase from the trivial one [21]. However, one should ask if ES is too compressed for a given quantum system and are there better ways to represent quantum information for the purpose of using DL technique. Therefore, in this work, we study topological phase transitions in 1D p -SC via the DL approach and systematically examine what kinds of quantum information related quantities could better represent the features of a quantum system, and hence a better input for DL. Concretely, we find that the block correlation matrix in Majorana representation not only tips out the phase boundaries but can even offer extra information beyond ES.

II. MODEL

The p -SC system of spinless fermions [5] in 1D is described by the Hamiltonian

$$H = \sum_i -t(c_i^\dagger c_{i+1} + c_{i+1}^\dagger c_i) + \Delta(c_i c_{i+1} + c_{i+1}^\dagger c_i^\dagger) - \mu(c_i^\dagger c_i - 1/2), \quad (1)$$

where t is the nearest-neighbor hopping amplitude, Δ is superconducting pairing potential, and μ represents on-site chemical potential. With the translational invariance, Eq. (1) can be rewritten as

$$H = - \sum_{k \in BZ} (c_k^\dagger, c_{-k}) [\mathbf{R}(k) \cdot \boldsymbol{\sigma}] (c_k, c_{-k}^\dagger)^T, \quad (2)$$

where Pauli matrices $\boldsymbol{\sigma} = (\sigma_x, \sigma_y, \sigma_z)$, and $\mathbf{R}(k) = (0, -\Delta \sin k, t \cos k + \mu/2)$ is the pseudomagnetic field. The one-particle energy spectrum is simply $\epsilon(k) = \pm 2R(k) = \pm \sqrt{(2t \cos k + \mu)^2 + 4\Delta^2 \sin^2 k}$. Note that this system preserves particle-hole symmetry, while it breaks time reversal symmetry and hence, the chiral symmetry; therefore it belongs to the class D according to the tenfold way classification for symmetry-protected topological systems [37]. It can be characterized by a Z_2 topological invariant.

FIG. 1. (a) Topological phase diagram of the 1D p -SC, where the chainlike inset pictures show schematically the Majorana representation for each phase as described in the main context. These indicate phases I and II are topological, while the others are not. (b) The infinite system is divided into a finite subsystem A with L sites and an environment B. (c) The schematic illustration of the convolutional neural network used in this work.

To make physics more transparent, we define Majorana operators, $d_{2j-1} = c_j + c_j^\dagger$ and $d_{2j} = -i(c_j - c_j^\dagger)$, and Eq. (1) becomes

$$H = \frac{i}{2} \sum_j [(-t + |\Delta|) d_{2j-1} d_{2j+2} + (t + |\Delta|) d_{2j} d_{2j+1} - \mu d_{2j-1} d_{2j}]. \quad (3)$$

As shown in Fig. 1(a), when $|\mu| > 2t$, the system Hamiltonian can be adiabatically connected to the form, $\frac{-i\mu}{2} \sum_j d_{2j-1} d_{2j}$, where $t = |\Delta| = 0$ and $\mu < 0$. It is then straightforward to see that the ground state is now composed of a paired Majorana fermions at the same site, resulting in no Majorana edge modes and hence a topologically trivial phase (phases III and IV). On the other hand, when $|\mu| < 2t$, H is adiabatically connected to the special case, $i \sum_j d_{2j} d_{2j+1}$, where $t = |\Delta| > 0$ and $\mu = 0$: Most Majorana fermions from neighboring sites are paired together while the system leaves the edge Majorana modes alone (unpaired), and thus corresponding to a nontrivial phase (phases I and II).

As we mentioned in the introduction, quantum information of the system can serve as a useful diagnostic for topological phase transitions, and in particular, the entanglement correlations should encode whole information about the focused state. In this paper, we consider two common correlators. First, by separating the system into A and B blocks [see Fig. 1(b)] and in terms of Majorana operators, the Majorana correlation matrix (MCM) of Majorana fermions at different sites within the subsystem A can be defined as $\text{MCM}_{i,j} \equiv i \text{Tr} \rho_0 d_{2i} d_{2j-1} = \text{Tr} \rho_0 (c_i - c_i^\dagger)(c_j + c_j^\dagger)$, where ρ_0 represents the density matrix of the ground state. The other correlations are either the conjugation or proportional to trivial identity when system Hamiltonian is quadratic. Second, the block correlation matrix (BCM) for subsystem A is defined as $\text{BCM}_{i,j} = \text{Tr} \rho_0 \hat{c}_i \hat{c}_j^\dagger$ with $\hat{c}_i \equiv (c_i, c_i^\dagger)^T$ and i, j being sites of the finite block A. This matrix is intimately connected

to the more familiar quantity, the reduced density matrix of the block A , $\rho_A = \bigotimes_m \begin{bmatrix} \lambda_m & 0 \\ 0 & 1 - \lambda_m \end{bmatrix}$, where λ_m are simply the eigenvalues of BCM and λ_m s are also known as one-particle entanglement spectrum. Therefore, the eigenvalues and their corresponding eigenvectors of BCM could be considered as potential representations of quantum information in the system for our deep learning purpose.

III. DL-BASED APPROACH

In order to correlate the phases outlined in the previous section with various quantum information-inspired input representations, we employ convolutional neural networks (CNNs), which are nonlinear functions particularly designed for efficiently recognizing patterns in the image-type data [38,39]. As we explain below in more detail, our choice of using CNNs is based on a natural interpretation of the input representations as “images”.

Using Keras framework [40], we build a deep CNN architecture as depicted schematically in Fig. 1(c). The architecture is mainly composed of two parts: Begins with convolutional layers followed by fully connected neural networks. The convolution part processes the input data by two consecutive convolutional layers both with filters of kernel size 3×3 and rectified linear unit (ReLU) activation functions. The number of filters (depth) is 64 for the first layer, and 128 for the second one. We use the zero-padding technique on the input data before each convolution to keep the spatial size intact [41] and do not insert pooling layers here to avoid missing subtle information due to our small image size (usually smaller than 100×100). After convolution, the processed data is then fed into a classifier made of a fully connected, ReLU activated layer with 512 neurons and a four-neuron, fully-connected softmax layer. The final outputs after softmax activation would sum up to unity and thus can be interpreted as the probabilities that the input data belongs to the four different phases as shown in Fig. 1(a) [41].

At the supervised training stage, we train the CNN on a dataset composed of $\mathcal{O}(10^4)$ representative “images” in “gray scale” (typically of 20×20 or 40×40 pixels) generated simply around six μ values (in a width of $0.05t$), corresponding to four possible phases deep inside the phase diagram [See Figs. 1(a) and 3]. Note that we collect “image” data in each *topological* phase around two different μ values for training to inform the model about two possible phase boundaries. Setting the train-validation split ratio as 0.2, the optimization for our model is then performed by ADAM algorithm [42] at learning rate 10^{-3} with cross entropy as the loss function [41]. Typically after training over 15 epochs, both train and validation losses would be less than 10^{-6} , indicating that the resultant model becomes reliable [See Fig. 2(b)]. Once training is done, at the inference stage we fix whole parameters in the trained model and feed with new data for prediction.

IV. RESULTS

Now, we take aforementioned DL approach to study topological phase transitions occurring in 1D p -SC and examine various quantum information-inspired input features in order to provide a better compressed representation of the naive

FIG. 2. (a) Each neuron output of the final softmax layer, corresponding to the probability of each phase, as a function of μ/t (unseen data) with $\Delta/t = 1, L = 20$. Although the training sets from MCMs are far beyond the μ/t region shown here, the CNN can still recognize a topological phase transition near -2.0 . The dashed line indicates the theoretical value of the transition. (b) The validation loss follows the trend of the training loss well, suggesting no overfitting happened. (c) The transition step becomes sharper as L grows, showing the finite-size effect of the subsystem A .

ground state wave function. We first prepare the training input “images” with labels by calculating a thousand of MCMs with subsystem size L (block A) under periodic boundary conditions of the full system around a given μ for each phase. Each MCM can be viewed as a $L \times L$ “image” in one (gray) channel and entries in a MCM represent pixel values. For simplicity, we fix $\Delta = t$ in the training set (except for phase II,

FIG. 3. The neuron output “phase diagram” is shown as a function of μ/t with $\Delta/t = 1, L = 20$. The training sets from the eigen-spectra of BCMs are prepared at around $\mu/t = -13, -0.3, 0.3, 13$ of a window width 0.05 (red dots). The phase boundaries are clearly recognized for unseen data points, but the probabilities corresponding to phases III and IV are equal at $|\mu/t| > 2$, suggesting they confuse the neural network.

where $\Delta = -t$). After training, we find that the model easily learns how to distinguish different phases for the given dataset and is ready to generalize for unseen data points.

As shown in Fig. 2(a), the neuron output corresponding to phase IV goes from probability 1 at $\mu/t = -2.5$ to 0 at $\mu/t = -1.5$. Such a curve gets across at $\mu^*(L=20)/t \approx -2.01$ with the curve corresponding to phase I, which behaves just oppositely, indicating that our CNN model indeed realizes a phase transition. In addition, the curves corresponding to the other phases are never activated here. Similarly, at $\mu^*(L=20)/t \approx 2.01$ two curves corresponding to phases I and III, respectively, also cross with each other and thus again it suggests another occurrence of the phase transition (not shown).

Note that for a given finite-size L , $\mu^*(L)$ is identified as the point where both crossing curves have equal probability 0.5, *i.e.*, at the moment that our trained model is not able to distinguish between the two phases. The non-abruptness of the phase transition seen from the probability curves is due to finite-size effect. As one can see in Fig. 2(c), the transition region becomes sharper when L grows longer and the finite-size trend of $\mu^*(L)/t$ is also closer to $\mu^*/t = -2$ in the thermodynamic limit where $L \rightarrow \infty$.

Alternatively, we next consider taking BCM as our training input. For a finite subsystem A of size L , there are at least three ways to represent each BCM (now of size $2L \times 2L$ due to Nambu notation) as an “image”: (i) View BCM itself as a “gray image”; (ii) arrange all eigenvalues of the BCM in ascending order into a diagonal matrix, which can be viewed as an “image”; (iii) we diagonalize a BCM and then arrange each eigenvector as one of the columns in a new matrix M . M is again of size $2L \times 2L$ and can be viewed as a “gray image”. In the DL approach, each way of representing the focused ground state would encode different levels of entanglement information, likely leading to distinguishable ability in realizing different phases, which is what we want to examine now. Since by definition of MCM, way (i) of treating BCM should be similar to the MCM case and hence we do not repeat it here and only focus on the latter two ways.

In fact, in way (ii) utilizing diagonal matrices of BCMs from all eigenvalues as input “2D images” is not efficient because the off-diagonal part contains no information (like black background). Therefore, we simply feed in all eigenvalues of a BCM, namely, entanglement spectrum, as an 1D input to a simple feed-forward neural network composed of three consecutive fully connected layers (with ReLU activation), having 32, 64, 256 neurons, respectively, followed by a four-neuron dense layer with softmax activation as the final output [43]. By taking the same training procedure (skip phase II training samples) and waiting until the training and validation losses converge, we show the final predictions of unseen data points at $\Delta/t = 1$, $L = 20$ as a function of μ in Fig. 3. It is clear that phase I can be recognized very well, while the outputs corresponding to phases III and IV are not. The predicted probabilities are 0.5 and 0.5, respectively, indicating that these two phases confuse the network due to their similar eigenspectra. If one examines phases III and IV carefully, one can see that they indeed belong to the same superconducting phase but very likely with different $U(1)$ gauges. The overall results for this case thus suggest that the representation

FIG. 4. (a) The neuron output “phase diagram” is shown as a function of μ/t with $\Delta/t = 1$, $L = 20$. The training sets from whole eigenvectors of BCMs are prepared at the same μ/t regions mentioned in Fig. 3 (red dots). The phase boundaries are still sharply defined as the MCM calculations. (b) Similar diagram as (a) but with only two middlemost eigenvectors making the input “images”. The phase boundaries are sharply shown. (c) Representative input “images” for the CNN at given μ/t values in the case (a). Boundary modes or topology-induced end modes are present at the middle-top and the middle-bottom parts of each image.

using entanglement spectrum as inputs compresses too much quantum information and may not be effective when trying to determine the global phase diagram via the DL approach.

To gain more quantum information, we finally investigate way (iii) to prepare the input dataset for training the slightly different CNN model. We keep the first convolution and then connect with two consecutive residual blocks. Then the global average 2D is used before final dense layer-based classifier [44]. Once training is completed, Figs. 4(a) and 4(b) depict again the model predictions for the unseen data points as a function of μ at $\Delta/t = 1$, $L = 20$. In the case of Fig. 4(a), we make each input “image” by including all eigenvectors of each BCM, while including only two middlemost eigenvectors in the case of Fig. 4(b). Clearly, both cases show the ability of the CNN model to recognize global phase diagrams despite different $U(1)$ phases which phases III and IV may take, and similar results are also observed when the unseen data points vary along Δ with fixed $|\mu| \leq 2$ (*i.e.*, phases I and II). But one would notice that the Fig. 4(a) case is slightly worse than 4(b) due to more eigenvectors, which may be considered as noises.

The underlying reason behind the above result can be explained as follows. Let us examine some selected input “images” made of all eigenvectors at $\mu/t = 2.5, -0.625, 0.625, 2.5$, respectively, in Fig. 4(c). The essential feature to distinguish a nontrivial topological phase from the trivial one is the presence of end modes (with entanglement eigenvalue 0.5) in the middlemost region of the figure along x . Thus, importantly, this indicates that our model can differentiate the end modes ($\mu/t = -0.625, 0.625$ cases) from the boundary eigenstates due to the finiteness of the subsystem A ($\mu/t = -2.5, 2.5$ cases), which is difficult for human eyes [45]. However, when taking all eigenvectors in making “images”, the bulk eigenvectors occupy a large portion of the image that may force our model to pay more

attention to this portion and hence weaken the ability of the model to determine the phase boundaries.

V. DISCUSSION AND CONCLUSION

There is a compelling reason for an independent check of our aforementioned results by comparing with those from seemingly unrelated 1D transverse-field Ising model. In fact, the transverse-field Ising model with a magnetic field λ , written as $H_{II} = -\sum_i \sigma_i^x \sigma_{i+1}^x - \lambda \sum_i \sigma_i^z$ with Pauli matrices σ^α , can be transformed into 1D p -SC with $t = -\Delta = 1$ and $\mu = 2\lambda$ [see Eq. (1)], by a nonlocal Jordan-Wigner transformation. Given the correlation function matrices, the entanglement spectra or entanglement eigenstates as a possible form of inputs, the transition point $\lambda = 1$ is still stood out via the proposed deep learning approach to distinguish between the ferromagnetic phase ($\lambda > 1$) and the paramagnetic one ($\lambda < 1$). It justifies the effectiveness of using the quantum information (in particular, the entanglement aspect) to encode quantum phases in the deep learning process.

Moreover, among the quantum information-related quantities used in this study, since a $2N \times 2N$ BCM has N redundant variables, we notice that using $N \times N$ MCM would be enough for recognizing topological phase transitions. And the fact that a relatively small size of MCM is already effective in our approach reflects the nature of BCMs, which indeed includes essential quantum information of the infinite chain at the ground state.

In summary, we demonstrate how to adopt the DL approach assisted via the entanglement aspect to discover topological phase transitions. Several quantum information-related quantities such as MCM, ES or entanglement eigenvectors (EEs) (from BCM) are fed into deep neural networks for training. While ES can only find the phase transition points, MCM and EE contain much abundant information to find not only the critical points but also phases of matter with different $U(1)$ gauges. Our work emphasizes utilizing quantum information, instead of naive wave functions, as inputs in the DL approach and it might be proven useful as well in higher dimensional systems.

ACKNOWLEDGMENT

M.C.C. acknowledges the MoST support under Contract No. 108-2112-M-005 -010 -MY3.

APPENDIX A: NEURAL NETWORK PRIMER

Neural networks (NNs) are neural-inspired, nonlinear models (functions), which are used in supervised or semisupervised learning tasks. Let us focus on a classification problem and denote a designed NN as a function F_θ , where θ represents the set of all trainable parameters in the NN. For a given input $\mathbf{x}_i$, the output of the NN (predicted label) can be written as $\hat{y}_i = F_\theta(\mathbf{x}_i)$. Denoting y_i as the corresponding true label, the objective of our learning task is then given by

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \sum_i \mathcal{L}(y_i, \hat{y}_i(\theta)), \quad (\text{A1})$$

where $\mathcal{L}$ is our chosen loss function and $\{\mathbf{x}_i, y_i\}$ for all i form our training dataset. We next briefly explain two NN architectures used in this work separately.

1. Simple feed-forward neural networks

The basic unit of any neural network is called a *neuron* that takes an input, say, a vector $\mathbf{x}$ with d components, and outputs a scalar z according to the following formula:

$$z = \sigma(\boldsymbol{\omega} \cdot \mathbf{x} + b), \quad (\text{A2})$$

where $\boldsymbol{\omega}$ is also a vector with d neuron-specific weights, and b is a neuron-specific bias. Note that z is an output after applying a so-called *activation function* $\sigma(\cdot)$, not simply a weighted sum of $\mathbf{x}$. In fact, the power of NNs essentially comes from σ , which is usually a *nonlinear* transformation. For instance, in this study we always choose the *rectified linear unit* (ReLU) as the activation function, i.e., $\sigma(\boldsymbol{\omega} \cdot \mathbf{x} + b) = \boldsymbol{\omega} \cdot \mathbf{x} + b$ if $\boldsymbol{\omega} \cdot \mathbf{x} + b > 0$ and 0 otherwise.

A simple feed-forward neural network (FNN) then consists of many such neurons stacked into layers. The output of a layer would be the input of the next. The layers are *fully connected* in the sense that within the same layer neurons are not connected with each other, while they are connected with each other as long as they belong to neighboring layers; each connection indicates a weight value. Often we call the first layer of a NN, the *input* layer, and the last one, the *output* layer; the layers between them are then called *hidden* layers. Note that the number of neurons in the output layer is usually equal to the number of desired classes, M , and the softmax function, $\hat{y}_i = \text{softmax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$ with $j = 1, \dots, M$, is used as the activation function of it such that the final outputs could be interpreted as the predicted probabilities for each class ($\sum_i \hat{y}_i = 1$).

2. Convolutional neural networks

Different from a FNN, a convolutional neural network (CNN) is a *translationally invariant* NN that respects the locality of the input data [46]. CNN is made of two parts: the first part consists of consecutive convolutional layers, while the second part is nothing but a simple FNN as a classifier. Thus, let us focus on this new convolutional layer.

Each convolutional layer uses its *filters* to perform 2D convolutions on its input. Similar to a fully connected layer, the output of a convolution layer would be the input of next one. Suppose the input, called the *feature map*, $\mathbf{X}$ has spatial size $(H_{\text{in}}, W_{\text{in}})$, and then the operation of the convolutional layer with C filters can be written as

$$h_k = \mathbf{W}_k * \mathbf{X} + b_k, k \in [0, C - 1], \quad (\text{A3})$$

where $\mathbf{W}_k$ and b_k are the weights and bias of the k th filter. The convolution operation is basically running this filter over all locations in the spatial plane of $\mathbf{X}$ and it finally outputs a new feature map of spatial size $(H_{\text{out}}, W_{\text{out}})$ with depth C . The spatial size is in fact determined by the following formula:

$$\begin{aligned} H_{\text{out}} &= \text{floor}((H_{\text{in}} - F + 2 \cdot P)/S + 1), \\ W_{\text{out}} &= \text{floor}((W_{\text{in}} - F + 2 \cdot P)/S + 1), \end{aligned} \quad (\text{A4})$$

FIG. 5. Gradient-based class activation maps [(d)–(f)] for selected input “images” [eigenvectors of BCM (a)–(c)], corresponding to phases IV, I, III, respectively (from left to right). The redder area indicates where the model pays more attention for predicting such class.

where the kernel size of each filter is F , the translation step for each filter when performing convolution is S , and the input is padded with P zeros around its boundary. Note that in this study, we always take an appropriate P such that $(H_{\text{in}}, W_{\text{in}}) = (H_{\text{out}}, W_{\text{out}})$. Again, the convolution operation mentioned here is just a linear mapping. After this convolution, one has to pass it through a non-linear activation function, for instance, a ReLU. Before connecting to the FNN part of the model, one can flatten the output of the last convolutional layer to a vector form with $H_{\text{out}} \times W_{\text{out}} \times C$ components.

3. Training neural networks

For classification problems, the most commonly used loss function $\mathcal{L}$ is the *categorical cross entropy*. In general, for a M -class problem one can take a “one-hot” vector to represent each class. For instance, a data point of the m -th class can be labeled as a M -component vector y_i with only m th component $y_i^m = 1$ and 0 otherwise. Also, we denote the output “probability” for m th class from the NN model as $\hat{y}_i^m(\theta)$. Then the cross entropy is defined as

$$\mathcal{L}(y_i, \hat{y}_i(\theta)) = - \sum_i \sum_{m=0}^{M-1} y_i^m \log \hat{y}_i^m(\theta) + (1 - y_i^m) \log[1 - \hat{y}_i^m(\theta)]. \quad (\text{A5})$$

Once having a model with a chosen loss function $\mathcal{L}$, one can begin training the model by feeding in training data and then update all trainable parameters θ s via a gradient descent-based optimizer that basically makes use of $\frac{\partial \mathcal{L}}{\partial \theta}$ through back-propagation for optimization.

Before closing this introduction for the NNs, there is an important remark. When preparing the dataset, it is usually not a problem to distinguish the *test* set from the *training* set, as the former one is going to test the final model performance

and certainly should not be seen by the model during training. However, one should further take 10 ~ 20% out of available training data as a *validation* set, which is used to fine tune the (nontrainable) hyperparameters of the model and to avoid overfitting to the training data. For a more comprehensive introduction to NNs and DL, in particular for physicists, we refer our readers to Ref. [47].

As a quick summary, the whole learning process can be expressed as

- (i) Collect and preprocess data,
- (ii) Build a model with a suitable architecture,
- (iii) Choose a loss function and an optimizer,
- (iv) Train the model and use the validation data to adjust the hyperparameters and avoid over-fitting,
- (v) Evaluate the model performance on the test data.

APPENDIX B: GRADIENT-BASED CLASS ACTIVATION MAP

While in this study, our trained CNN has recognized phase transitions with high accuracy, the DL model is yet still considered as a “black box” method. To gain more scientific value via this approach, one may have to solve the model interpretability or explainability for its predictions. A gradient-based class activation map (Grad-CAM), published in 2017 by Selvaraju *et al.* [48], is just such a visualization technique for CNNs to track where the network is “looking” in the input “image” and which series of neurons activated in the forward-pass during inference.

The idea behind Grad-CAM is quite simple. Suppose the outputs of the final convolutional layer in the model are $h_{k,ij}$, where subscripts denote the spatial coordinates ij on k th feature map. Since convolution operations preserve the local information of any input data, the “heat map” (which records the strongly activated places in the model) of the beginning

input image can then be estimated by the weighted sum of the all feature maps,

$$L_{\text{Grad-CAM}}^m = \text{ReLU}\left(\sum_k \alpha_k^m h_k\right), \quad (\text{B1})$$

where m indicates the class we concerned. Note that ReLU is applied to the linear combination because we are only interested in the features that have a positive influence on the class of interest. Now, the problem remaining is how to reasonably determine the weight α_k^m for each feature map. Selvaraju *et al.* propose to compute the gradients of the score for the class m , i.e., model output y^m with respect to feature maps $h_{k,ij}$, and then these gradients are global average-pooled to obtain the neuron importance weights, that is,

$$\alpha_k^m = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^m}{\partial h_{k,ij}}. \quad (\text{B2})$$

Finally, if the spatial size of the feature maps is smaller than the original input image, one can resize the heat map and put it on top of the input image to obtain the final class activation map.

Taking the study of eigenvectors of BCM as an example, in Fig. 5 we show the Grad-CAM results for three randomly selected input training "images" representing phases IV (nontopological), I (topological), and III (nontopological), respectively [similar to Fig. 4(c) in the main text]. Note that the spatial information of each eigenvector is along the y axis. The redder regions correspond to higher scores for predicting the final result. It is rather interesting to see that our trained CNN model pay much attention on the upper/lower middle regions for predicting *topological* phase, indicating its recognition of end states, which decay exponentially into the bulk, after training.

-
- [1] V. L. Ginzburg (1960); Some remarks on phase transitions of the second kind and the microscopic theory of ferroelectric materials, *Fiz. Tverd. Tela* **2**, 2031 (1960) [*Sov. Phys. Solid State* **2**, 1824 (1961)]; Landau, L. D., E. M. Lifshitz, and L. P. Pitaevskii, *Statistical Physics Part I*, Vol. 5 (Pergamon, Oxford, 1994), Chap. XIV.
- [2] K. V. Klitzing, G. Dorda, and M. Pepper, *Phys. Rev. Lett.* **45**, 494 (1980).
- [3] M. König, S. Wiedmann, C. Brüne, A. Roth, H. Buhmann, L. W. Molenkamp, X.-L. Qi, and S.-C. Zhang, *Science* **318**, 766 (2007).
- [4] For instance, see A. Bansil, H. Lin, and T. Das, *Rev. Mod. Phys.* **88**, 021004 (2016), and references therein.
- [5] A. Kitaev, *Phys. Usp.* **44**, 131 (2001).
- [6] For instance, see C. Nayak, S. H. Simon, A. Stern, M. Freedman, and S. Das Sarma, *Rev. Mod. Phys.* **80**, 1083 (2008).
- [7] L. Fu and C. L. Kane, *Phys. Rev. Lett.* **100**, 096407 (2008).
- [8] V. Mourik, K. Zuo, S. M. Frolov, S. R. Plissard, E. P. A. M. Bakkers, and L. P. Kouwenhoven, *Science* **336**, 1003 (2012).
- [9] S. Nadj-Perge, I. K. Drozdov, J. Li, H. Chen, S. Jeon, J. Seo, A. H. MacDonald, B. Andrei Bernevig, and A. Yazdani, *Science* **346**, 602 (2014).
- [10] S. M. Albrecht, A. P. Higginbotham, M. Madsen, F. Kuemmeth, T. S. Jespersen, J. Nygård, P. Krogstrup, and C. M. Marcus, *Nature (London)* **531**, 206 (2016).
- [11] J. Wiedenmann, E. Bocquillon, R. S. Deacon, S. Hartinger, O. Herrmann, T. M. Klapwijk, L. Maier, C. Ames, C. Brüne, C. Gould *et al.*, *Nat. Commun.* **7**, 10303 (2016).
- [12] For a review, see I. Peschel and V. Eisler, Reduced density matrices and entanglement entropy in free lattice models, *J. Phys. A* **42**, 504003 (2009).
- [13] M.-C. Chung and I. Peschel, *Phys. Rev. B* **64**, 064412 (2001).
- [14] I. Peschel, *J. Phys. A* **36**, L205 (2003).
- [15] S. A. Cheong and C. L. Henley, *Phys. Rev. B* **69**, 075111 (2004).
- [16] T. Barthel, M.-C. Chung and U. Schollwöck, *Phys. Rev. A* **74**, 022329 (2006).
- [17] S. Ryu and Y. Hatsugai, *Phys. Rev. B* **73**, 245115 (2006).
- [18] M.-C. Chung, Y.-H. Jhu, P. Chen, C.-Y. Mou, and X. Wan, *Sci. Rep.* **6**, 29172 (2016).
- [19] P. Fromholz, G. Magnifico, V. Vitale, T. Mendes-Santos, and M. Dalmonte, *Phys. Rev. B* **101**, 085136 (2020).
- [20] Y. LeCun, Y. Bengio, and G. Hinton, *Nature (London)* **521**, 436 (2015).
- [21] E. P. L. van Nieuwenburg, Y.-H. Liu, and S. D. Huber, *Nat. Phys.* **13**, 435 (2017).
- [22] J. Carrasquilla and R. G. Melko, *Nat. Phys.* **13**, 431 (2017).
- [23] T. Ohtsuki and T. Ohtsuki, *J. Phys. Soc. Jpn.* **85**, 123706 (2016).
- [24] L. Wang, *Phys. Rev. B* **94**, 195105 (2016).
- [25] P. Broecker, J. Carrasquilla, R. G. Melko, and S. Trebst, *Sci. Rep.* **7**, 8823 (2017).
- [26] A. Tanaka and A. Tomiya, *J. Phys. Soc. Jpn.* **86**, 063001 (2017).
- [27] S. J. Wetzel and M. Scherzer, *Phys. Rev. B* **96**, 184410 (2017).
- [28] W. Hu, R. R. P. Singh, and R. T. Scalettar, *Phys. Rev. E* **95**, 062122 (2017).
- [29] P. Broecker, F. F. Assaad, and S. Trebst, *arXiv:1707.00663* (2017).
- [30] K. Ch'ng, N. Vazquez, and E. Khatami, *Phys. Rev. E* **97**, 013306 (2018).
- [31] Y.-H. Liu and E. P. L. van Nieuwenburg, *Phys. Rev. Lett.* **120**, 176401 (2018).
- [32] J. F. Rodriguez-Nieva and M. S. Scheurer, *Nat. Phys.* **15**, 790 (2019).
- [33] Y. Zhang and E.-A. Kim, *Phys. Rev. Lett.* **118**, 216401 (2017).
- [34] Y. Zhang, R. G. Melko, and E.-A. Kim, *Phys. Rev. B* **96**, 245119 (2017).
- [35] P. Zhang, H. Shen, and H. Zhai, *Phys. Rev. Lett.* **120**, 066401 (2018).
- [36] J. Venderley, V. Khemani, and E.-A. Kim, *Phys. Rev. Lett.* **120**, 257204 (2018).
- [37] A. P. Schnyder, S. Ryu, A. Furusaki, and A. W. W. Ludwig, *Phys. Rev. B* **78**, 195125 (2008).
- [38] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, *Neural Comput.* **1**, 541 (1989).
- [39] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, *Proc. IEEE* **86**, 2278 (1998).
- [40] F. Chollet and others, <https://keras.io> (2015).
- [41] Please see Appendix A for more details.

- [42] D. P. Kingma and J. Ba, *Proceedings of the 3rd International Conference on Learning Representations (ICLR)* (Cornell University, Ithaca NY, 2015).
- [43] We have also examined some other model architectures with fewer hidden layers. Most of them can be trained to behave similarly, except for the slightly vague results near the phase boundaries in shallow networks with very few neurons. This may be understood by the capacity of each model. For instance, there are 26, 212 trainable parameters in the model mentioned in the main text for $L = 100$, comparable to our training data, totally containing $200 \times 4000 \sim \mathcal{O}(10^6)$ input pixels. Clearly, it has stronger potential to find a good fit to the data than shallower ones. However, this does not mean that the shallow ones cannot work since each input vector could have quite redundant information.
- [44] K. He, X. Zhang, S. Ren, and J. Sun, *Conference: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, Las Vegas, NV, 2016).
- [45] What the trained CNN has learned may be revealed by a visualization technique called Grad-CAM. For further explanation, please see Appendix B.
- [46] Y. LeCun and Y. Bengio, in *The Handbook of Brain Theory and Neural Networks*, edited by M. A. Arbib (MIT Press, Cambridge, 1995), p. 276.
- [47] P. Mehta, M. Bukov, C.-H. Wang, A. G. R. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, *Phys. Rep.* **810**, 1 (2019).
- [48] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, *International Conference on Computer Vision (ICCV'17)* (IEEE, Venice Italy, 2017), [arXiv:1610.02391](https://arxiv.org/abs/1610.02391).