Renormalization-group-inspired neural networks for computing topological invariants

Gilad Margalit,[1,*] Omri Lesser,[1,*] T. Pereg-Barnea,[2] and Yuval Oreg[1]
[1]*Department of Condensed Matter Physics, Weizmann Institute of Science, Rehovot 7610001, Israel*
[2]*Department of Physics, McGill University, Montréal, Québec H3A 2T8, Canada*

(Received 1 March 2022; revised 18 May 2022; accepted 18 May 2022; published 27 May 2022)

We show that artificial neural networks (ANNs) can, to high accuracy, determine the topological invariant of a disordered system given its two-dimensional real-space Hamiltonian. Furthermore, we describe a "renormalization-group" (RG) network, an ANN which converts a Hamiltonian on a large lattice to another on a small lattice while preserving the invariant. By iteratively applying the RG network to a "base" network that computes the Chern number of a small lattice of set size, we are able to process larger lattices without retraining the system. We therefore show that it is possible to compute real-space topological invariants for systems larger than those on which the network was trained. This opens the door for computation times significantly faster and more scalable than previous methods.

DOI: 10.1103/PhysRevB.105.205139

I. INTRODUCTION

Machine learning (ML) has been redefining the way computer-related problems are solved [1,2]. Until about a decade ago, the methodology of algorithmic problem solving had been quite clear: A human programmer looks for rules and regularities, which are then transformed into conditional statements run by the computer. Machine learning takes a different approach, based on the concept of *training*, i.e., repeatedly changing the values of internal parameters in the algorithm in order to reach a desired goal. After the training, the ML algorithm finds a solution by itself, earning it the name *artificial intelligence*. Owing to massive improvements in both software and hardware, machine learning has emerged as a game-changing approach to formidably difficult problems, well beyond the reach of traditional algorithms. Popular examples of such problems are image and pattern recognition [3] and natural language processing [4,5].

The great promise held by machine learning did not evade the notice of physicists [6–9]. ML approaches are now widespread in many areas of experimental physics involving the analysis of large amounts of data [10–14]. Additionally, ML techniques—and in particular, artificial neural networks (ANNs)—keep finding applications in theoretical condensed matter physics [15]. For example, ANNs have been used as a means to detect phase transitions [16–20] and solve quantum many-body problems [21–26]. Typically, a representation of the system is chosen (like Hamiltonian parameters or wave function), and the ML algorithm learns to calculate a desired property (like phase classification, ground-state energy, or average magnetization). An important advantage of ML techniques in solving such problems is the sizable speedups they offer compared to traditional methods.

With the advent of topological phases of matter in the past several decades [27–31], calculating topological invariants has become a highly relevant problem in contemporary condensed matter physics [32–37]. These integer-valued invariants are important markers of the bulk topology and the edge properties. A famous example is the quantum Hall effect [38,39], where the relevant topological invariant—the Chern number—counts the number of chiral edge modes [32,40]. In translation-invariant and noninteracting systems, calculating topological invariants is usually a straightforward task [31,41]. However, the presence of disorder alters the picture and makes the calculation challenging. In two spatial dimensions, there are several methods for solving this problem [42,43], but their scaling with the system size makes them unfeasible for large systems. It is therefore tempting to utilize ML techniques to tackle this problem. In particular, since topological invariants are integers, the task at hand is actually a classification problem, for which ML algorithms are particularly well suited [44–52].

Despite the apparent appeal and compatibility of ML techniques to the task of calculating topological invariants of large disordered systems, there is one clear drawback. The overwhelming majority of ML algorithms assume a constant input size, whereas we would like our method to be general—we do not wish to train a separate ANN for each system size. Furthermore, generating labeled samples and training an ANN both become computationally intractable, for large lattice sizes.

To overcome these hurdles, we employ a two-stage solution strategy, as illustrated in Fig. 1(a). First, we train an ANN to solve the simple problem of calculating the Chern number of a small system with a fixed size. We shall refer to this as the "base" network. Then, we train a second ANN whose goal is to map a large system to a smaller one, while preserving the Chern number. This resizing network applies a non-linear transformation to the original system and reduces its linear size by a factor of 2. Crucially the network is capable of reducing any $N \times N$ system to an effective $\frac{N}{2} \times \frac{N}{2}$

[*]These authors contributed equally to the work.

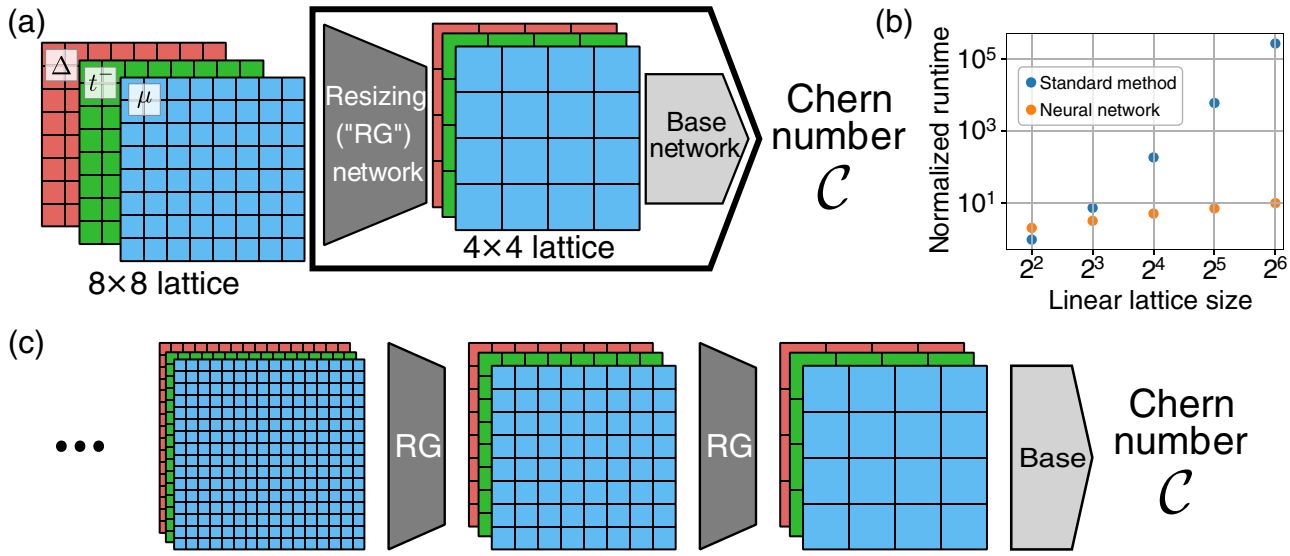

FIG. 1. (a) RG and base network block structure. In the first stage, the base network is trained to find the Chern numbers of 4×4 lattices. Then a compound network (thick outline), which includes the pretrained base network and a resizing "RG" network, is trained on 8×8 lattices. The RG network reduces the lattice dimensions by 2 while preserving the Chern number, allowing it to be solved by the base network. (b) Runtime scaling comparison between the Bott index method and successive application of an RG network. The times for both methods are normalized by the runtime for the 4×4 case calculated using the Bott index method. The Bott index method, used as a baseline in this work, is $\mathcal{O}(n^4)$ to $\mathcal{O}(n^6)$, depending on the efficiency of diagonalization methods, where n is the linear lattice size. Extreme parallelization allows the RG method to run in sublinear time. Notice that the accuracy of the result is not shown; it is likely that the accuracy of our method declines for particularly large lattices, though additional research may change this. (c) Extension of the diagram in panel (a) to arbitrary lattice sizes by iteratively applying the RG network, then computing the Chern number via the base network once the lattice has been reduced to size 4×4.

one. It then becomes possible to start with a large system and iteratively apply the resizing network, without further training, until it reaches the fixed small size accepted by the base network. At this point, the reduced system is fed into the base network, which outputs its Chern number. The iterative process is demonstrated in Fig. 1(c). This architecture allows for a very substantial speedup in computation time compared to the standard methods, as shown in Fig. 1(b). Though the base network is not the source of the computational speedup, expressing the base Chern solver as a neural network allows easier integration with the RG network during training.

Mapping a large system onto a smaller effective one is a well-established paradigm in condensed matter physics, and the overarching theoretical framework for doing so is the renormalization group (RG) [53–55]. In particular, our approach is similar in spirit to real-space RG or block-spin decimation [56,57]. There, the idea is to cluster neighboring degrees of freedom, and treat them as the effective degrees of freedom in a new, coarse-grained system. Such methods have had some success in elementary problems, but they are typically insufficient for capturing more complex situations. They may fail by generating new types of interactions that were absent in the original system, or by ignoring the entanglement between coarse-grained blocks—the ground state of a large system is not necessarily composed of the ground states of each block. Here, we aim to address the difficulties common in real-space RG methods by assuming very little about the resizing transformation (which we refer to as the "RG" network): The high versatility and nonlinearity of the ANN allow it to realize extremely complicated transformations. The network, by construction, *learns* an appropriate RG-like transformation that preserves the topological invariant. Furthermore, the built-in separation into network blocks allows us to gain some insight into the nature of the learned transformation. Such insight is not typically possible in ANNs (see Appendix C).

II. SYSTEM AND BASE NETWORK

Our example system is a disordered $p + ip$ superconductor in real space [31,58,59], described by the following tight-binding Hamiltonian:

$$H = \sum_{i,j} \left[\mu_{i,j} c^\dagger_{i,j} c_{i,j} + t^x_{i,j} c^\dagger_{i+1,j} c_{i,j} + t^y_{i,j} c^\dagger_{i,j+1} c_{i,j} + \Delta_{i,j} c^\dagger_{i+1,j} c^\dagger_{i,j} + i\Delta_{i,j} c^\dagger_{i,j+1} c^\dagger_{i,j} + \text{H.c.} \right], \tag{1}$$

where $c^\dagger_{i,j}$, $c_{i,j}$ are creation and annihilation operators for an electron on the site (i, j) of the square lattice with periodic boundary conditions. Each site has an on-site energy μ, hopping energies t^x and t^y to adjacent sites, and a superconducting pair potential Δ with $p + ip$ symmetry. It is helpful to define $t^\pm_{i,j} \equiv (t^x_{i,j} \pm t^y_{i,j})/2$ and normalize the parameters such that $t^+ = 1$ for all sites. This leaves us with three parameters per site: $\mu_{i,j}$, $t^-_{i,j}$, and $\Delta_{i,j}$ (the latter two are bond parameters, but they can equivalently be regarded as site parameters).

In the clean case the Chern number is independent of Δ, and it is determined by the uniform values of μ and t^- [59], as shown in the phase diagram in Fig. 2(a). In the disordered case, we can construct a similar phase diagram. A disorder

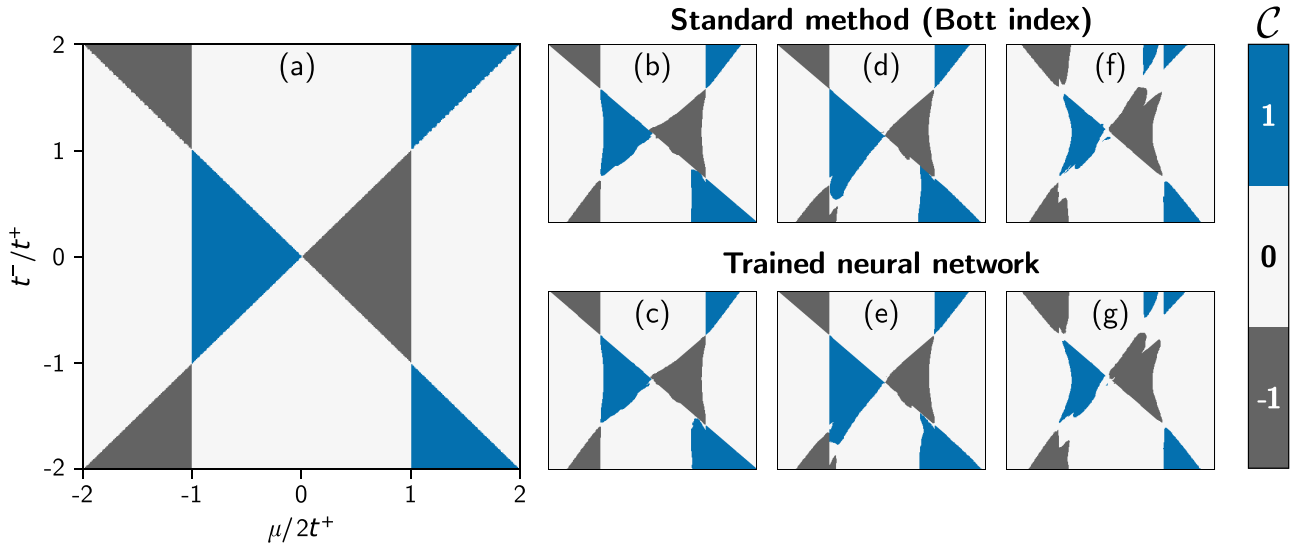

FIG. 2. (a) Phase diagram of a clean system (every site identical) in terms of the chemical potential μ and hopping anisotropy t^-. Color indicates the Chern number $\mathcal{C}$ of the system. [(b), (d), (f)] Examples of disordered phase diagrams of 4×4 lattices, calculated using the Bott index method [42]. [(c), (e), (g)] Corresponding phase diagrams calculated by the base neural network.

realization is defined by a list of random values of μ, t^-, and Δ at each site with zero mean. On top of that, we uniformly vary the average values of μ and t^-, and calculate the real-space Chern number for the resulting lattice via the Bott index method [42]. Figure 2(b), 2(d), and 2(f) each correspond to a single disorder realization for a 4×4 lattice.

We shall now reproduce this calculation using an ANN. Our base neural network acts on a 4×4 lattice, and thus takes as input a set of 48 numbers: 16 sites times 3 "channels" per site: μ, t^-, Δ. We used a convolutional ResNet architecture with 9 layers, which proved fast to train (see Appendix A for more details on the network structure, loss function, and training process). The network was trained on 10^8 disorder realizations, whose Chern number was calculated using the Bott index method and treated as a label. For each sample, the network was trained to minimize the difference between its estimate of the Chern number and the sample's label. This process continued until the network reached a maximum accuracy of 99.02%.

As a visualization of the network's performance, we computed the phase diagrams corresponding to several disorder realizations. This was done by feeding the disorder configuration with the uniform μ, t^- offsets into the trained network. The resulting phase diagrams, shown in Figs. 2(c), 2(e), and 2(g), are in very good agreement with the directly computed ones shown in Figs. 2(b), 2(d), and 2(f).

III. RG NETWORK

Now that we have a base network capable of accurately determining the Chern number of disordered 4×4 lattices, the next step is generalizing to larger lattices. We do this by training an "RG" network that transforms an 8×8 lattice to a 4×4 one. For this network, we use another nine-layer convolutional ResNet, whereby each layer consists of several "kernels"—small blocks which are convolved with the lattice, incorporating periodic boundary conditions into the network structure (see Appendix B for more details on the network's architecture). This type of neural network is ideal for an RG-like operation because it operates on neighboring sites and transforms them into an effective site, regardless of the system size. Each layer is also reminiscent of a renormalization group in that it is spatially local.

For the training of this RG network, we generated 10^7 disorder realizations of 8×8 lattices, and calculated the Chern number for each of them using the Bott index method. The RG network takes these samples as input, keeping the same input structure of $8 \times 8 \times 3$ parameters per sample, and outputs an effective 4×4 lattice: a list of $4 \times 4 \times 3$ parameters. This effective reduced lattice is then combined with the previously trained 4×4 base network to form a compound network, illustrated in a thick outline in Fig. 1(a), which, as a whole, takes a disordered 8×8 lattice and returns an estimate of its Chern number. The resulting Chern number is compared to the correct one, and the network is trained to minimize the difference between the two, under the constraint that the base network's weights are not allowed to change.

This process alone can be used to train an RG network to high accuracy, 98.72%, when evaluating 8×8 lattices. The crucial question, though, is what happens when trying to apply this network to a larger lattice. The simplest test is taking a 16×16 lattice, applying the RG network to obtain a 8×8 lattice, and then applying it again to obtain a 4×4 lattice. We find that the Chern number accuracy of this repeated network drops significantly (to 72.49%). This behavior is not surprising: the RG network has over 9×10^6 free parameters, and its training has no generalization requirement, so it has a very low probability of converging to a generalizable solution.

To fix this issue, we initialize the RG network, prior to its training, to perform a naïve decimation mapping: constructing the smaller output lattice by simply averaging adjacent sites in each channel of the larger input. Initializing the network in

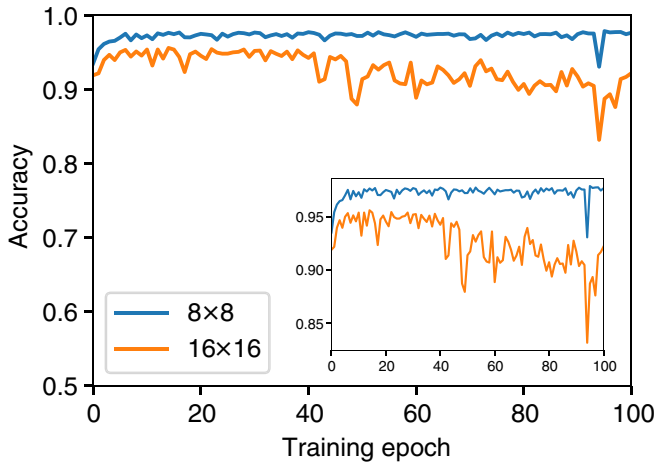

FIG. 3. Accuracy of the RG network on 8×8 and (by applying the network twice in succession) 16×16 lattices, as a function of how long the network has been training. The accuracy for 16×16 initially increases, reaches a maximum at epoch 14, then declines.

this way can be thought of as using the decimation mapping as a zeroth-order approximation of our desired function to be performed by the RG network (details on the initialization procedure are given in Appendix B 1). The main step in our training procedure, which is training on the actual Chern numbers of 8×8 lattices, is then a refinement of this approximation. As long as the network is only allowed to make small changes beyond naïve decimation, we expect it to stay generalizable.

The training process for the RG network is plotted in Fig. 3. After each epoch—exposure to all of the samples in the 8×8 training set—the weights were saved and the current network was tested against a testing set of 8×8 lattices (blue curve). In addition, the double-RG layer was tested against a testing set of 16×16 lattices (orange curve). Thus, the two curves in the figure represent the performance of the same RG network on multiple scales. The initial accuracies—93.52% and 91.91% for 8×8 and 16×16, respectively—are the accuracies of the initial decimation function. The network's performance on 8×8 samples increases roughly monotonically, as expected since this is the target function. The 16×16 performance, however, reaches a maximum and then declines. Our interpretation of this result is that the RG network begins in a generalizable subspace, but as the training process seeks an optimal 8×8 accuracy, it eventually drifts out of this subspace.

Since our goal is an RG network that generalizes well to large scales, we halt the training process when the 16×16 accuracy is at its peak. The resulting RG network has an accuracy of 97.64% on 8×8 lattices—slightly less than the best accuracy we attained, due to the fact that the training process was halted. The accuracy on 16×16 lattices is 95.60%, a significant improvement over the uninitialized case. We emphasize that this performance is despite the system never being exposed to a 16×16 lattice during training.

IV. CONCLUSION

We have demonstrated that neural networks can accurately compute the Chern numbers of disordered Hamiltonians on lattices of a fixed size, and that RG-like networks can resize lattices while preserving the Chern number. In combination, these two types of networks can, in principle, evaluate the topological invariants of lattices of arbitrary size, including sizes that are not computationally tractable with previous methods. Indeed, our proof-of-concept RG network generalizes well to lattices a factor of 2 larger than those on which it was trained.

Given this promising result, the natural question is how large the input lattice can become before the compound network's accuracy decays beyond the point of usefulness. Unfortunately, this question is logistically difficult to answer. As discussed in Appendix B 2, an appropriate disorder scaling convention must be found to counteract disorder averaging at larger scales. However, finding a viable convention is computationally expensive, as it requires slowly generating many 32×32 or larger lattices and computing their Chern numbers via the Bott index. Further study is needed to determine a more efficient means of solving this step; otherwise, our method's use remains restricted to systems for which a valid disorder scaling is known. For this reason, we constrained our scope to the 16×16 demonstration, as it is the simplest result that conveys the viability of our approach.

However, the accuracy does drop between the 16×16 and 8×8 scales, and we have no guarantee that it does not drop further for larger lattices. A possible way to address this, which is left to future study, is an additional training stage on 16×16 samples. We envision using two copies of the RG network in sequence, constrained to be identical throughout the training process, which reduce the input size by an overall factor of 4. This training step should further refine the RG block's ability to generalize to larger system sizes.

Beyond 2D systems and physics in general, iterated resizing networks represent an under-explored tool in computer science with three significant benefits. One is their ability, as we have demonstrated, to be applicable over many input sizes. The second is the ability to reach high accuracies on inputs outside of the training distribution, which may be helpful in cases where training samples are more computationally difficult to produce (as is the case for our 16×16 lattices) or for which training data cannot be easily collected. Finally, our modular design offers a degree of interpretability. Most neural networks are "black boxes" with very limited means of determining why they act the way they do. Our network with 8×8 inputs, for instance, can be examined at the intermediate 4×4 scale to study the network's interpretation of "RG" (see Appendix C for details). These benefits and extensions to our concept suggest that the use of base networks and iterated resizing blocks could prove useful in and beyond computational physics.

Models and related functions used in this research were written in PYTHON using the machine learning packages Keras [60] and TensorFlow [61]. Samples were generated using MATLAB. Our full code is available in Ref. [62] .

ACKNOWLEDGMENTS

We are grateful to Oren Pereg, Moshe Wasserblat, Roman Beliy, and Itsik Adiv for helpful discussions. The work at Weizmann was supported by the European Union's Horizon 2020 research and innovation programme (Grant Agreement No. LEGOTOP No. 788715), the DFG (CRC/Transregio 183, Grant No. EI 519/7-1), ISF Quantum Science and Technology (Grant No. 2074/19), and the BSF and NSF (Grant No. 2018643). T.P.B. acknowledges financial support from NSERC and FRQNT.

APPENDIX A: STRUCTURE AND TRAINING: BASE NETWORK

The base network is the final part of our network which receives a small (4×4 lattice points) input and returns the Chern number as an output. Here we describe the details of this network.

The base network is a nine-layer convolutional ResNet [63] with skip connections between layers 2 and 3, 4 and 5, and so on. Each two-dimensional (2D) convolutional layer has 256 kernels of size 3×3, with ReLU activation [64] and zero padding (periodic padding is only used for the RG network). The output from these layers (of size $4 \times 4 \times 256$) is then combined via a max-pooling layer, flattened, and then passed through a dense layer with a single node and linear activation. This outputs the predicted Chern number.

The network was trained on an NVIDIA GeForce RTX 2080Ti GPU using 10^8 samples of $4 \times 4 \times 3$ disordered lattices labeled with their corresponding Chern numbers (calculated via the Bott method [42,65]) in minibatches of size 640. We used stochastic gradient descent (SGD) with a momentum of 0.9 and an initial learning rate of 0.01. No learning rate decay was used, but a callback function reduced the learning rate by a factor of 0.1 each time the validation loss did not reach a new minimum for 50 consecutive epochs. When the learning rate would have been reduced to 10^{-7}, the training instead terminated.

1. Sample distribution

Each $4 \times 4 \times 3$ sample lattice is produced in two steps. First, in each of the three channels (chemical potential μ, hopping anisotropy t^-, and pairing energy Δ), a random disorder value is generated for each site, chosen uniformly from a disorder range specific to each parameter type and subtracting the average so that each channel's disorder averages to 0. The disorder ranges used in this work are $[-0.15, 0.15]$ for μ, $[-0.1, 0.1]$ for t^-, and $[-0.025, 0.025]$ for Δ. These disorder ranges were chosen to allow a noticeable impact on Chern number while not resulting in phase diagrams that are vastly different from the clean diagram.

Next, we choose an average value for each of the three channels and add it to every site in the lattice. Unlike the disorder distribution, these average values are not selected uniformly in each range. In order to use training time more efficiently, our training set is boundary biased, meaning that the distribution of average site values is biased toward points close to the phase boundaries, where the Chern number is most strongly affected by disorder (only the training set, not the testing set, is biased in this way; the latter is generated uniformly within the average and disorder ranges). Specifically, the training set distribution of samples is

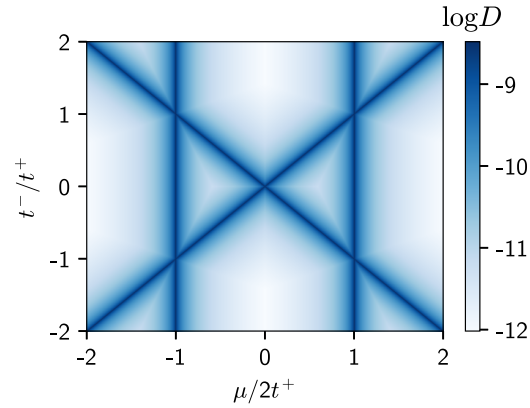

FIG. 4. Biased distribution of training samples according to their distance from the phase boundaries [see Eq. (A1)], with $\alpha = 0.03$.

$$D(\mu, t^-, \Delta) = \frac{N}{\alpha + x(\mu, t^-)}, \tag{A1}$$

where N is a normalization factor, $\alpha = 0.03$ is a constant that indicates the level of bias, and $x(\mu, t^-)$ is the distance between the point $(\frac{\mu}{2}, t^-)$ and the nearest clean phase boundary [the vertical and diagonal lines where the Chern number changes in Fig. 2(a) in the main text; we set t^+ equal to 1]. The sample distribution is visualized in Fig. 4. Note that since the phase boundaries of clean systems do not depend on Δ, the distribution of average Δ values is uniform. μ, t^-, and Δ are constrained to the ranges $[-4, 4]$, $[-2, 2]$, and [0.1,0.2], respectively, and are subjected to the boundary-biasing distribution within those ranges.

2. Custom loss

We make use of a custom loss function in all of our training throughout this research. The function is

$$L(y_{\text{pred}} - y_{\text{true}}) = |y_{\text{pred}} - y_{\text{true}}| + (y_{\text{pred}} - y_{\text{true}})^2, \tag{A2}$$

where y_{pred} and y_{true} are the predicted value and true value of the label, respectively. For the base network, we apply this loss on each channel independently and then add the results to form an overall loss function.

This loss has the advantage of harshly punishing large deviations from the labels, as a mean square loss does, while also being unforgiving of small errors in the latter stages of training, since the absolute value function is larger for values close to zero. We found substantially improved accuracy on early tests when using this loss function compared with mean-square or linear loss functions, as well as with discrete losses such as categorical cross-entropy. In later tests with our final training set and disorder distribution, the difference between this loss and conventional losses was less pronounced, but the loss was not modified further.

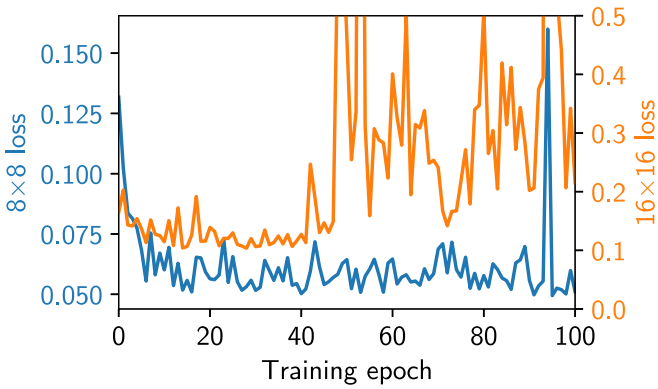

FIG. 5. Training loss for 8×8 (blue) and 16×16 (orange) evaluations of the RG network.

APPENDIX B: STRUCTURE AND TRAINING: RG NETWORK

The structure of the RG network is nearly identical to the base network: a nine-layer ResNet. All parameters are the same as the base network except as indicated below.

The input layer is of size $8 \times 8 \times 3$, and the output is of size $4 \times 4 \times 3$. In order to reduce the dimensions of the lattice, the first convolutional layer has its stride value set to (2,2), meaning that the kernels only evaluate on sites with odd numbers as both of their indices. The sites with even index still contribute to the output, however, since each is within several 3×3 kernels centered at nearby odd-indexed sites. All layers after the first use the standard (1,1) stride, and in addition, the number of kernels in each layer after the first is 512 so as to keep the total number of parameters consistent between rows.

Rather than use zero padding, we employ periodic padding (described in more detail in Sec. B 3) on the nine main convolutional layers. After the nine main layers, there is a single convolutional layer with only three kernels of size 1, which results in an output of the desired shape; the padding on this layer is irrelevant since the kernel cannot extend beyond the lattice boundary, so it is set to zero padding for simplicity.

For further illustration of the training process, the loss is plotted in Fig. 5 for the same training run whose accuracy is depicted in Fig. 3.

1. Initialization

As described in the main text, before the RG network can be trained to classify Chern numbers, it must first be initialized such that it approximates a decimation mapping. This preliminary step is motivated by the fact that decimation—equivalent to an average pooling layer of pool size (2,2)—performs somewhat well on both 8×8 and 16×16 lattices (with accuracy 93.52% and 91.91%, respectively). We hypothesized that an RG network initialized in this way would retain some of the generalizability (functionality at multiple lattice scales) of the decimation function in the early stages of its training to preserve the Chern number when it changes the system size.

The goal of the initialization process is to begin with a network that performs a simple decimation, meaning that it merely averages nearby parameters, creating a coarse-grained lattice. Practically, it is easier to train the network to perform this decimation rather than setting its weights by hand. We therefore begin with a normal distribution of kernel weights and biases and the RG ResNet is trained on 5×10^7 samples of 8×8 lattices. Rather than being labeled by their Chern number, these samples' labels are the 4×4 lattices that would result from a decimation mapping being applied to them. The loss during training is the custom loss described inAppendix A 2 applied to each channel and site of the output, comparing them to the corresponding channel and site of the label, and then summing the result with all sites and channels weighted equally. The training procedure is identical to that of the base network, except that the learning rate is set to 0.0005 and not adjusted during training.

The samples used in the initialization stage form a much wider distribution than those used in the Chern number training. Specifically, each site and channel of a sample contains a uniformly chosen random number in the range $[-5, 5]$. It is critical that this domain includes, but is much larger than, the parameter range of the base and RG network's Chern number training. If the initialization step is trained only on lattices from the same distribution as the Chern-learning step, the initialized network does not approximate a general decimation, but rather a decimation that has been overfitted to function only on 8×8 lattices in our specific input domain. Naturally, this prevents it from generalizing successfully to 16×16 lattices.

2. Chern training and results

In the main training step, a compound network, composed of the initialized RG network feeding into the base network, is trained on 5×10^7 samples of 8×8 lattices and their Chern numbers. The base network's weights are frozen, so that they do not update and only the RG network is optimized. The training process is identical to the base network case except for a learning rate of 0.0005, which is not changed during training.

The samples are distributed identically to the 4×4 samples described in Sec. A 1, except that the range of disorder in μ in both training and testing samples is increased from $[-0.15, 0.15]$ to $[-0.3, 0.3]$. This counteracts disorder averaging, which causes the effect of disorder on the Chern number to become less significant at larger lattice scales. Without scaling the disorder in this way, a trivial RG mapping that ignores disorder altogether can still achieve high accuracy. Since we are free to define how lattice parameters scale in our renormalization scheme, we choose to increase this disorder value in order to prevent the RG network from converging to this trivial map. This ensures that our compound network will be able to solve for the Chern number for large lattices in the strong-disorder regime.

The compound network is saved after each epoch. The RG network corresponding to each epoch is then loaded and tested on an 8×8 testing set of size 10^6. Additionally, each network is placed in a compound network with another copy of itself and a base network, allowing it to evaluate 16×16 lattices. This requires only that the input layer of each RG network be removed, that the two be placed in sequence, and that a new input layer of size $16 \times 16 \times 3$ be added. The latter compound network is tested on a 16×16 testing set of size 10^6. The

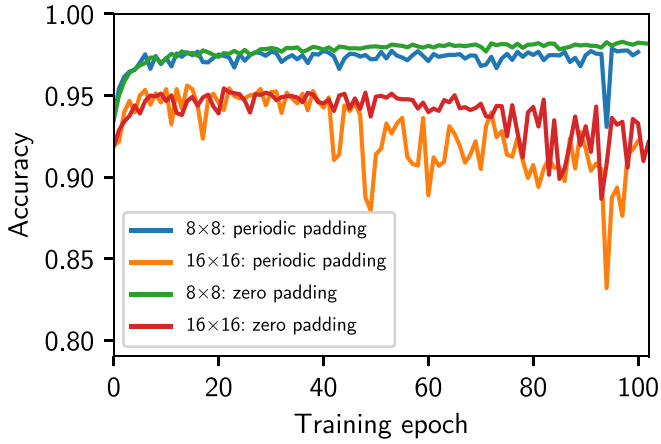

FIG. 6. Accuracy as a function of epochs trained for 8×8 and 16×16 evaluations of the RG network.

accuracy of these tests as a function of epoch number are given in Fig. 3 in the main text and in Fig. 6 below.

3. Periodic padding

Unlike in the image processing applications that are the usual use case for convolutional neural networks, the lattices we use as inputs represent periodic systems. As such, it is physically reasonable to have the network interpret the lattice in such a way that periodicity is preserved.

To do this, we define a periodic padding. When the 3×3 kernel is centered at a site on the edge or corner of the input lattice, some of the positions in the kernel will be outside the lattice. Instead of interpreting the values of these nonexistent so-called pad sites as uniformly zero in each channel, as most convolutional networks do, we treat their value as being what it would be if the lattice was periodic; that is, equal to the corresponding site on the other side of our input lattice. Practically, this is accomplished by inserting a Lambda layer—a layer that performs a predefined function—before each convolution, which adds a periodic pad. The input to each convolutional layer is thus two sites larger in each dimension, corresponding to the one-site-thick pad. The padding procedure of the convolutional layers is set to "valid," meaning that the border sites are not allowed to be the center sites of kernels, which reduces the output of the convolutional layer by two, back to its correct size.

The effect of periodic padding on the computed Chern number accuracy of 8×8 and 16×16 lattices is shown in Fig. 6. For unknown reasons, periodic padding causes the accuracy to grow more slowly in the 8×8 case and to fall more rapidly in the 16×16 case for later epochs of training. However, in our testing, it provided the best single saved network at evaluating 16×16 lattices (95.60% accuracy, compared to 95.43% accuracy for the best nonperiodic network). Periodic padding also aids in interpretability, since it removes a notable and unphysical distortion along the system boundary when using the RG network as a resizing mapping (see Sec. C). The frequent layer resizing causes RG training time to increase by about 40%, however.

APPENDIX C: PROPERTIES OF THE RG NETWORK

Our network has a two-stage structure: a down-sizing RG block is followed by a Chern number calculation block. This structure allows us to examine the properties of the RG network alone, without looking at the Chern number itself. In this section, we describe some basic observations on the nature of this network.

1. Points mapping

As a first step, we ask how the RG transformation acts in parameter space. We know that the Chern number is mostly preserved, but that alone does not mean the lattice does not drift far away in parameter space—many points have the same Chern number despite being distant from each other in parameter space. However, our training starts from simple decimation, which leaves parameter space invariant, so we expect the output lattice's parameters not to stray too far from those of the input lattice.

To check this notion of locality in parameter space, we generated clean 128×128 lattices with values of μ and t^- evenly spaced in the interval $[-2, 2]$ (as usual, we set $t^+ = 1$). Simple decimation leaves these points exactly where they started in the 2D parameter space of μ, t^-. Our network, however, acts slightly differently. The 64×64 plot in Fig. 7 shows the distribution of points after applying the RG network. Each point in this plot corresponds to a clean lattice which resulted from the RG mapping being applied to a clean lattice from the 128×128 plot. Rather than being evenly spaced, we observe a high density of points deep inside the phases, and generally a much lower density close to the phase transition lines. Our interpretation is that the network is trying to avoid regions with large uncertainty, where the error in the Chern number would be much higher. Physically, the topological gap in these regions is small, and therefore the Chern number becomes harder to determine reliably.

The network has the freedom to choose an asymmetric transformation, as we do not force any symmetry; hence, the point plots do not obey the mirror symmetries of the phase boundaries in general. Moreover, different runs of the training procedure may randomly lead to different asymmetries.

As the RG mapping is applied iteratively, the points cluster into fixed points. Since this mapping has been trained to preserve Chern number, we expect at least one fixed point per phase region, and this is indeed what we find. If this holds for most disorder realizations as well, it implies that in large-scale implementations of this method, the inputs will be clustered and effectively classified by the time they are reduced to the 4×4 scale. Thus, in these large-scale implementations, it is likely that the base network could be substituted with a much smaller, simpler network, further improving runtime.

2. Mapping of a single impurity

Going one step further, we would like to understand how simple configurations are mapped under the RG network. As an example, we consider a single impurity in an otherwise clean 8×8 lattice. The impurity is in the form of a different chemical potential, $\mu_0 + \delta\mu$, at a specific site, where all the

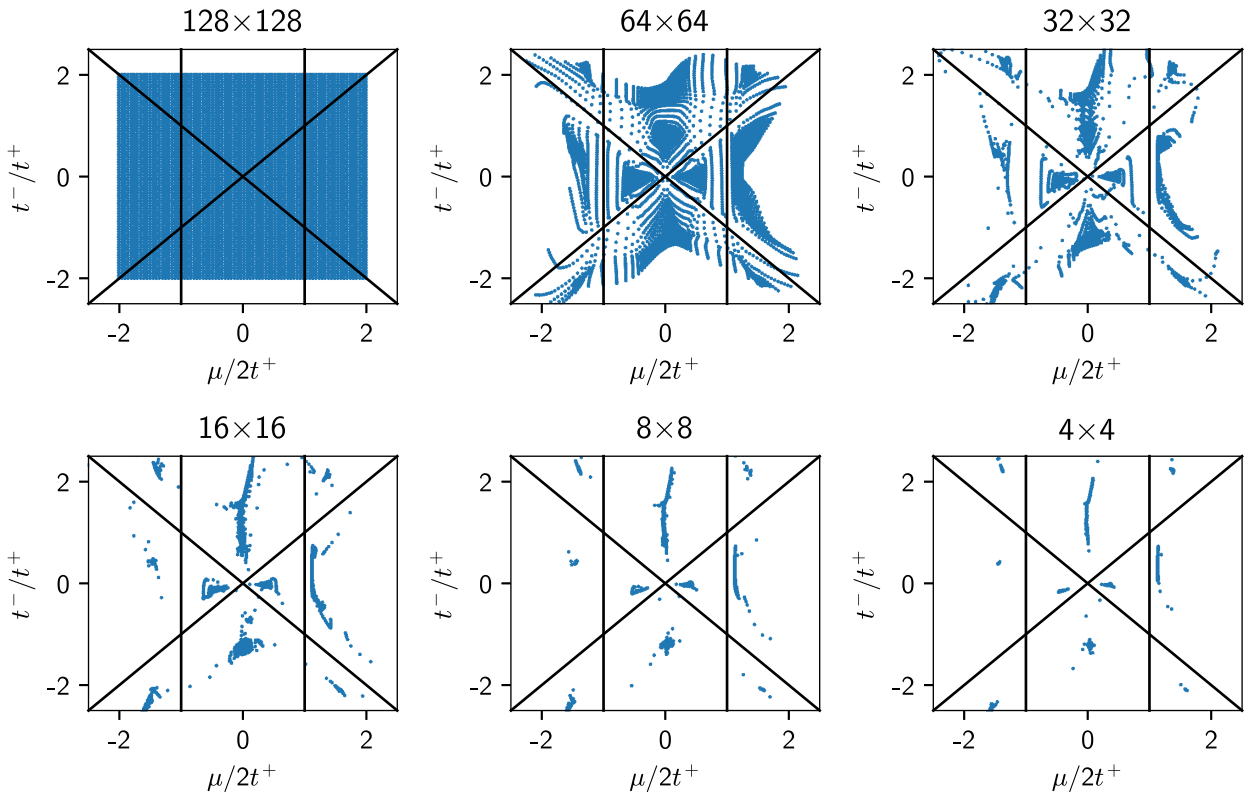

FIG. 7. Visualization of the RG mapping. Each point in the 128 × 128 plot corresponds to a single clean lattice, characterized by its chemical potential μ and the hopping anisotropy t^-. The solid black lines represent the phase boundaries in the clean case. As the RG network is applied iteratively, the points cluster in each phase, demonstrating that the RG mapping actively preserves the Chern number.

other sites have chemical potential μ_0. Specifically, here we take $\mu_0 = 0.5t^+$ and place the impurity at the site (4, 6).

In Fig. 8, we show the 4 × 4 lattices resulting from applying the RG network to this impurity configuration, for four values of the impurity $\delta\mu$. For each $\delta\mu$, we plot the values of μ, t^-, Δ at each point in the 4 × 4 lattice, subtracting their initial values to get a difference map.

Without an impurity ($\delta\mu = 0$), the 4 × 4 lattice is clean, as its 8 × 8 ancestor, and the parameters closely match the original ones—see Fig. 8(a). As $\delta\mu$ is increased, we observe the impurity propagating to the 4 × 4 lattice in a *local* way. First, the impurity mostly affects the sites close to the location of the original impurity (if one were to coarse-grain the 8 × 8 lattice into a 4 × 4 one). Second,

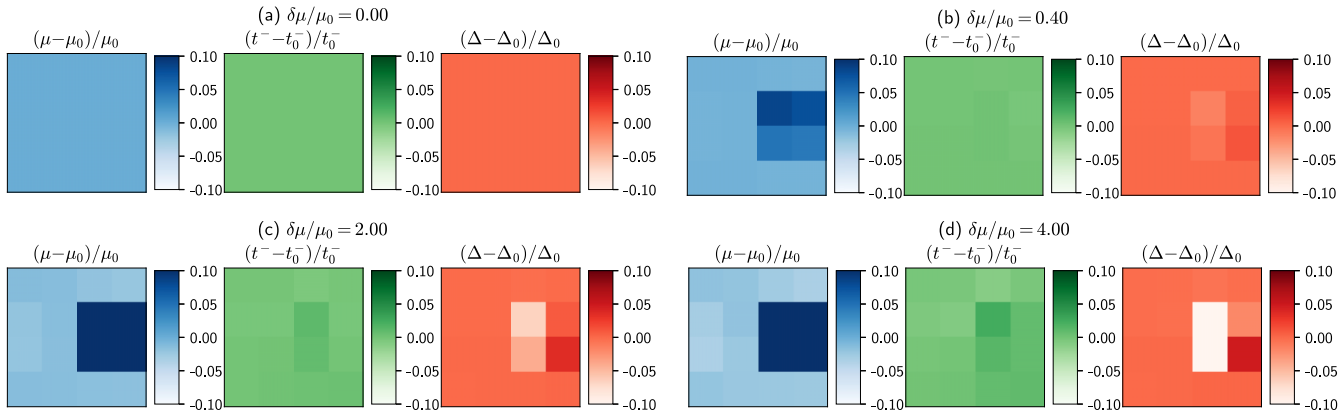

FIG. 8. Propagation of a single impurity in the chemical potential from an 8 × 8 lattice down to the renormalized 4 × 4 lattice calculated by the RG network. The values used are $\mu_0 = 0.5$, $t_0^- = 1$, $\Delta_0 = 0.15$. The impurity $\delta\mu$ is placed at the site (4, 6), and its values are indicated on the different panels.

the propagation is mostly local also in channel space: The variation in μ affects mostly the renormalized μ, and its effect on the renormalized t^-, Δ is weaker. This demonstrates that although the Chern number is a global property, the RG-like transformation that preserves it is mostly local in nature.

[1] I. Goodfellow, Y. Bengio, and A. Courville, in *Deep Learning*, edited by F. Bach, Adaptive Computation and Machine Learning Series (MIT Press, Cambridge, MA, 2016).

[2] M. A. Nielsen, *Neural Networks and Deep Learning* (Determination Press, 2015).

[3] M. Pak and S. Kim, in *2017 4th International Conference on Computer Applications and Information Processing Technology (CAIPT)* (IEEE, Indonesia, 2017), pp. 1–3.

[4] P. M. Nadkarni, L. Ohno-Machado, and W. W. Chapman, J. Am. Med. Inform. Assoc. **18**, 544 (2011).

[5] E. Mankolli and V. Guliashki, in *ICT Innovations 2020. Machine Learning and Applications*, Communications in Computer and Information Science, edited by V. Dimitrova and I. Dimitrovski (Springer International, Cham, 2020), pp. 71–86.

[6] 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).

[7] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, Rev. Mod. Phys. **91**, 045002 (2019).

[8] J. Schmidt, M. R. G. Marques, S. Botti, and M. A. L. Marques, npj Comput. Mater. **5**, 83 (2019).

[9] S.-M. Udrescu and M. Tegmark, Sci. Adv. **6**, eaay2631 (2020).

[10] A. Radovic, M. Williams, D. Rousseau, M. Kagan, D. Bonacorsi, A. Himmel, A. Aurisano, K. Terao, and T. Wongjirad, Nature (London) **560**, 41 (2018).

[11] D. Guest, K. Cranmer, and D. Whiteson, Annu. Rev. Nucl. Part. Sci. **68**, 161 (2018).

[12] D. Bourilkov, Int. J. Mod. Phys. A **34**, 1930019 (2019).

[13] E. Lustig, O. Yair, R. Talmon, and M. Segev, Phys. Rev. Lett. **125**, 127401 (2020).

[14] N. Käming, A. Dawid, K. Kottmann, M. Lewenstein, K. Sengstock, A. Dauphin, and C. Weitenberg, Mach. Learn.: Sci. Technol. **2**, 035037 (2021).

[15] J. Carrasquilla, Adv. Phys. X **5**, 1797528 (2020).

[16] J. Venderley, V. Khemani, and E.-A. Kim, Phys. Rev. Lett. **120**, 257204 (2018).

[17] N. Sun, J. Yi, P. Zhang, H. Shen, and H. Zhai, Phys. Rev. B **98**, 085402 (2018).

[18] M. J. S. Beach, A. Golubeva, and R. G. Melko, Phys. Rev. B **97**, 045207 (2018).

[19] E. van Nieuwenburg, E. Bairey, and G. Refael, Phys. Rev. B **98**, 060301(R) (2018).

[20] E.-J. Kuo and H. Dehghani, arXiv:2111.08747 [cond-mat, physics:quant-ph].

[21] G. Carleo and M. Troyer, Science **355**, 602 (2017).

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

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

[24] M. Koch-Janusz and Z. Ringel, Nat. Phys. **14**, 578 (2018).

[25] S.-H. Li and L. Wang, Phys. Rev. Lett. **121**, 260601 (2018).

[26] D. Luo and B. K. Clark, Phys. Rev. Lett. **122**, 226401 (2019).

[27] G. E. Volovik, *The Universe in a Helium Droplet*, International Series of Monographs on Physics (Oxford University Press, Oxford, UK, 2009).

[28] M. Z. Hasan and C. L. Kane, Rev. Mod. Phys. **82**, 3045 (2010).

[29] J. E. Moore, Nature (London) **464**, 194 (2010).

[30] X.-L. Qi and S.-C. Zhang, Rev. Mod. Phys. **83**, 1057 (2011).

[31] B. A. Bernevig and T. L. Hughes, *Topological Insulators and Topological Superconductors* (Princeton University Press, Princeton, NJ, 2013).

[32] D. J. Thouless, M. Kohmoto, M. P. Nightingale, and M. den Nijs, Phys. Rev. Lett. **49**, 405 (1982).

[33] Q. Niu, D. J. Thouless, and Y.-S. Wu, Phys. Rev. B **31**, 3372 (1985).

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

[35] Y. Hatsugai, Phys. Rev. Lett. **71**, 3697 (1993).

[36] T. Fukui, Y. Hatsugai, and H. Suzuki, J. Phys. Soc. Jpn. **74**, 1674 (2005).

[37] I. C. Fulga, F. Hassler, and A. R. Akhmerov, Phys. Rev. B **85**, 165409 (2012).

[38] K. v. Klitzing, G. Dorda, and M. Pepper, Phys. Rev. Lett. **45**, 494 (1980).

[39] D. C. Tsui, H. L. Stormer, and A. C. Gossard, Phys. Rev. Lett. **48**, 1559 (1982).

[40] S. M. Girvin, in *Aspects Topologiques de La Physique En Basse Dimension [Topological Aspects of Low Dimensional Systems]*, Les Houches - Ecole d'Ete de Physique Theorique, edited by A. Comtet, T. Jolicœur, S. Ouvry, and F. David (Springer, Berlin, 1999), pp. 53–175.

[41] M. V. Berry, Proc. R. Soc. London, Ser. A **392**, 45 (1984).

[42] T. A. Loring and M. B. Hastings, EPL **92**, 67004 (2011).

[43] J. Borchmann, A. Farrell, and T. Pereg-Barnea, Phys. Rev. B **93**, 125133 (2016).

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

[45] D.-L. Deng, X. Li, and S. Das Sarma, Phys. Rev. B **96**, 195145 (2017).

[46] P. Zhang, H. Shen, and H. Zhai, Phys. Rev. Lett. **120**, 066401 (2018).

[47] D. Carvalho, N. A. García-Martínez, J. L. Lado, and J. Fernández-Rossier, Phys. Rev. B **97**, 115453 (2018).

[48] N. Yoshioka, Y. Akagi, and H. Katsura, Phys. Rev. B **97**, 205110 (2018).

[49] J. F. Rodriguez-Nieva and M. S. Scheurer, Nat. Phys. **15**, 790 (2019).

[50] N. L. Holanda and M. A. R. Griffith, Phys. Rev. B **102**, 054107 (2020).

[51] M. S. Scheurer and R.-J. Slager, Phys. Rev. Lett. **124**, 226401 (2020).

[52] T. Mertz and R. Valentí, Phys. Rev. Res. **3**, 013132 (2021).

[53] B. Hu, Phys. Rep. **91**, 233 (1982).

[54] S. R. White and R. M. Noack, Phys. Rev. Lett. **68**, 3487 (1992).

[55] G. Refael and E. Altman, C. R. Phys.Disordered Syst.**14**, 725 (2013).

[56] L. P. Kadanoff, Phys. Phys. Fiz. **2**, 263 (1966).

[57] K. G. Wilson, Phys. Rev. B **4**, 3174 (1971).

[58] J. Alicea, Rep. Prog. Phys. **75**, 076501 (2012).
[59] D. F. Mross, Y. Oreg, A. Stern, G. Margalit, and M. Heiblum, Phys. Rev. Lett. **121**, 026801 (2018).
[60] F. Chollet *et al.*, Keras, https://github.com/fchollet/keras.
[61] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg *et al.*, TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, software available from tensorflow.org.
[62] https://github.com/giladmargalit/ML-topology.
[63] K. He, X. Zhang, S. Ren, and J. Sun, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, Las Vegan, NV, USA, 2016), pp. 770–778.
[64] V. Nair and G. E. Hinton, in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, edited by J. Fürnkranz and T. Joachims (Omnipress, Madison, WI, USA,2010), pp. 807–814.
[65] Z. Yi-Fu, Y. Yun-You, J. Yan, S. Li, S. Rui, S. Dong-Ning, and X. Ding-Yu, Chin. Phys. B **22**, 117312 (2013).