Vulnerability of Machine Learning Phases of Matter

Si Jiang,[1,*] Sirui Lu,[1,2,*] and Dong-Ling Deng[1,†]

[1]*Center for Quantum Information, IIIS, Tsinghua University, Beijing 100084, People's Republic of China*
[2]*Max-Planck-Institut für Quantenoptik, Hans-Kopfermann-Str. 1, D-85748 Garching, Germany*

Classifying different phases and the transitions between them is a major task in condensed matter physics. Machine learning, which has achieved dramatic success recently in a broad range of artificial intelligence applications, may bring an unprecedented perspective for this challenging task. In this paper, we study the robustness of this intriguing machine-learning approach to adversarial perturbations, with a focus on supervised learning scenarios. We find that typical phase classifiers based on deep neural networks are extremely vulnerable to adversarial examples: adding a tiny amount of carefully-crafted noises, which are imperceptible to human eyes and ineffective to traditional methods, into the original legitimate data obtained from numerical simulations or real experiments will cause the classifiers to make incorrect predictions at a notably high confidence level. Our results reveal a novel vulnerability aspect in applying machine learning techniques to condensed matter physics, which provides a valuable guidance for both theoretical and experimental future studies along this direction.

Machine learning is currently revolutionizing many technological areas of modern society, ranging from image/speech recognition to content filtering on social networks and automated self-driving cars [1, 2]. Recently, its tools and techniques have been adopted to tackle intricate quantum many-body problems [3–14], where the exponential scaling of the Hilbert space dimension poses a notorious challenge. In particular, a number of supervised and unsupervised learning methods have been exploited to classify phases of matter and identify phase transitions [6, 8, 15–24]. Following these approaches, notable proof-of-principle experiments with different platforms [25–28], including electron spins in diamond nitrogen-vacancy centers [25], doped CuO_2 [28], and cold atoms in optical lattices [26, 27], have also been carried out subsequently, showing great potentials for unparalleled advantages of machine learning approaches compared to traditional means.

An important question of both theoretical and experimental relevance concerns the reliability of such machine-learning approaches to condensed matter physics: are these approaches robust to adversarial perturbations, which are deliberately crafted in a way intended to fool the classifiers? In the realm of adversarial machine learning [29–32], it has been shown that machine learning models can be surprisingly vulnerable to adversarial perturbations if the dimension of the data is high enough [33]—one can often synthesize small, imperceptible perturbations of the input data to cause the model make highly-confident but erroneous predictions. A prominent adversarial example that clearly manifests such vulnerability of classifiers based on deep neural networks was first observed by Szegedy *et al.* [34], where adding a small adversarial perturbation, although unnoticeable to human eyes, will cause the classifier to miscategorize a panda as a gibbon with confidence larger than 99%. In this paper, we investigate the vulnerability of machine learning approaches in the context of classifying different phases of matter, with a focus on supervised learning based on deep neural networks (see Fig. 1 for an illustration).

We find that typical phase classifiers based on deep neural networks are likewise extremely vulnerable to adversarial perturbations. This is demonstrated through two concrete examples, which cover different phases of matter (including both symmetry-breaking and symmetry-protected topological phases) and different strategies (such as, fast gradient sign method [35], momentum iterative method [36], and projected gradient descent [35], etc.) to obtain the adversarial perturbations. In addition, through adversarial training, we demonstrate that the robustness of such classifiers to specific types of adversarial perturbations can be significantly improved. Our results shed new light on the fledgling field of machine-learning applications in condensed matter physics, which may provides a valuable guidance for both theoretical and experimental future studies as the field matures.

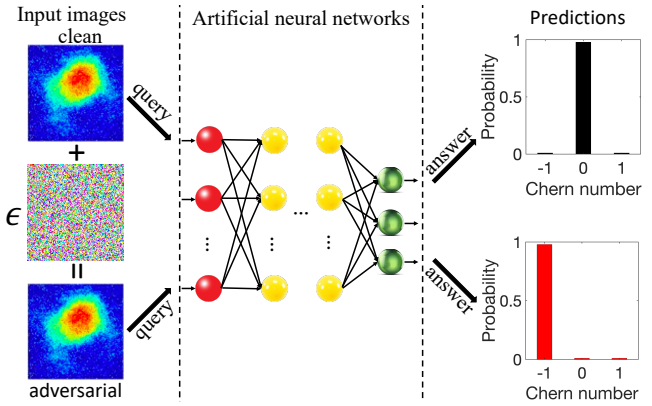

FIG. 1. A schematic illustration for the vulnerability of machine learning phases of matter. For a clean image, such as the time-of-flight image obtained in a recent cold-atom experiment [26], a trained neural network (i.e., the classifier) can successfully predict its corresponding Chern number with nearly unit accuracy. However, if we add a tiny adversarial perturbation (which is imperceptible to human eyes) to the original image, the same classifier will misclassify the resulted image into an incorrect category with nearly unit confidence probability as well.

To begin with, we first introduce the essential idea of adversarial machine learning, with a focus on the supervised learning scenarios based on deep artificial neural networks. In supervised learning, the training data is labeled: $\mathcal{D}_n = \{(\mathbf{x}^{(1)}, y^{(1)}), \cdots, (\mathbf{x}^{(n)}, y^{(n)})\}$, where $\mathbf{x}^{(i)}$ denotes a

data sample to be classified and $y^{(i)}$ is its corresponding label. The fundamental task of supervised learning is to learn a model $y = h(\mathbf{x};\theta)$ (a classifier) that provide an accurate mapping from the input $\mathbf{x}^{(i)}$ to the output $y^{(i)}$, by optimizing over some set of model parameters θ: $\min_\theta \mathcal{L}_n(\theta)$, where $\mathcal{L}_n(\theta) = \frac{1}{n}\sum_{i=1}^{n} L(h(\mathbf{x}^{(i)};\theta), y^{(i)})$ is the averaged loss function over the training data set. This minimization problem is often solved by stochastic gradient descent on the model parameters [37]. In contrast, to obtain adversarial examples, we consider our model parameters θ as fixed and instead optimize over the input space. More specifically, we search for a perturbation δ that can be added to the input sample $\mathbf{x}^{(i)}$ to *maximize* the loss function:

$$\max_{\delta\in\Delta} L(h(\mathbf{x}^{(i)}+\delta;\theta), y^{(i)}), \tag{1}$$

where we constrain δ to be from a class of appropriate small perturbations Δ, so as to ensure that the adversarial perturbation is not completely changing the input data. To solve the maximization problem in Eq. (S1), a number of methods have been proposed. Different methods have their pros and cons, and the choice of which one to use is problem-specific.

In this paper, we employ some of these methods, such as differential evolution algorithm (DEA) [38], fast gradient sign method (FGSM) [35], momentum iterative method (MIM) [36], and projected gradient descent (PGD) [35], etc., to obtain adversarial examples in learning different phases of matter. To show more precisely how this works, we give two concrete examples in the following. The first one concerns learning the conventional paramagnetic/ferromagnetic phases with a two-dimensional (2D) classical Ising model. The second example involves learning topological phases with experimental raw data generated by a solid-state quantum simulator. We use different deep neural networks to build up classifiers for different phases and show explicitly that these classifiers are highly vulnerable to adversarial perturbations.

The ferromagnetic Ising model. —The first example we consider involves the following ferromagnetic Ising model defined on a 2D square lattice:

$$H_{\text{Ising}} = -J\sum_{\langle ij\rangle}\sigma_i^z\sigma_j^z,$$

where the Ising variables $\sigma_i^z = \pm 1$ and the coupling strength $J \equiv 1$ is set to be the energy unit. This model features a well-understood phase transition at the critical temperature $T_c = 2/\ln(1+\sqrt{2}) \approx 2.366$ [39], between a high-temperature paramagnetic phase and a low-temperature ferromagnetic phase. The discrete Z_2 spin inversion symmetry is broken at temperatures below T_c and is recovered in the paramagnetic phase above T_c.

In the context of machine learning phases of matter, different pioneering approaches, including these based on supervised learning [16], unsupervised learning [8], or a confusion scheme combining both supervised and unsupervised learning [17], have been introduced to classify the ferromagnetic/-paramagnetic phases hosted by the above 2D Ising model. In particular, Carrasquilla and Melko first explored a supervised learning scheme based on a fully connected feed-forward neural network [16]. They used equilibrium spin configurations sampled from Monte Carlo simulations to train the network and demonstrated that after training it can correctly classify new samples with notably high accuracy. Moreover, through scanning the temperature the network can also locate the the transition temperature T_c and extrapolate the critical exponents that are crucial in the study of phase transitions.

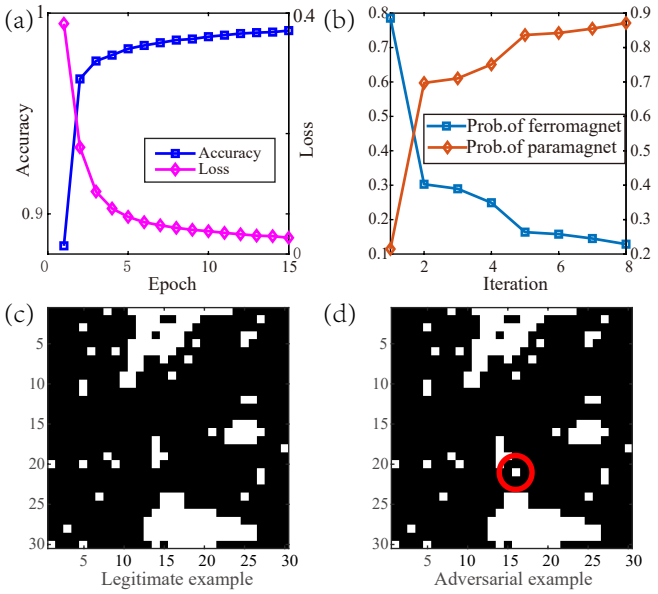

FIG. 2. (a) The average accuracy and loss versus the number of epochs. (b) We use the differential evolution algorithm to obtain the adversarial examples. This plot shows the classification probabilities as a function of the iteration number. After around two iterations, the network will begin to classify the samples incorrectly. (c) A legitimate sample of the spin configuration in the ferromagnetic phase. (d) An adversarial example obtained by the differential evolution algorithm, which only differs with the original legitimate one by a single pixel.

An important question left unexplored is: how robust are these introduced machine learning approaches to adversarial perturbations? Here, we focus on the supervised learning approach and show that it is highly vulnerable to adversarial perturbations. We first use a fully-connected feed forward neural network, implemented with TensorFlow [40], to serve as the classifier and perform supervised learning to train it with legitimate clean data generated by Monte Carlo simulations (see the Supplemental Material [41] for details). Fig. 2(a) shows both the accuracy and the loss as a function of epochs. After training, the network can successfully classify data from a test set with a high accuracy larger than 97% [41].

To obtain adversarial perturbations, we first consider a discrete attack scenario in the black-box setting, where we assume no prior information about the classifier's internal structures and modify the input data in a discrete fashion by flipping a few spins. We apply the differential evolution algo-

rithm (which is a population based optimization algorithm for solving complex multi-modal problems) [38] to the Monte Carlo sampled spin configurations and obtain the corresponding adversarial perturbations [41]. As an illustration, in Fig. 2(b) we randomly choose a sample in the ferromagnetic phase and apply the differential evolution algorithm to the sample iteratively. Fig. 2(b) shows the confidence probabilities of the classifications of slightly modified samples into ferromagnetic and paramagnetic phases, respectively. From this figure, it is clear that at the beginning, the classifier can correctly classify the sample as belonging to the ferromagnetic phase, but the corresponding confidence probability decreases rapidly as the iteration number increases. In contrast, the confidence probability for the sample to be classified into the paramagnetic phase increases rapidly. After eight iterations, the sample is incorrectly classified as belonging to the ferromagnetic phase with a confidence larger than 88%. Strikingly, as shown in Fig. 2(c-d), the final obtained adversarial sample differs from the original legitimate sample only by a single pixel—flipping a single spin will cause the classifier to misclassify the input sample at a decisively high confidence level.

If we regard H_{Ising} as a quantum Hamiltonian and allow the input data to be continuously modified, one can also consider a continuous attack scenario and obtain various adversarial examples by the FGSM, PGD and MIM methods, as discussed in details in the Supplementary Materials [41]. Similarly, the obtained adversarial examples only differs from the legitimate ones by a tiny amount of crafted perturbations. It is worthwhile to mention that, unlike FGSM, PGD, and MIM, the DEA method does not rely on the gradient information for optimizing and therefore does not require the objective function to be differentiable or known beforehand [42]. It has the main advantages of higher probability of finding global optima and requiring less information from the target systems, thus may have a wider range of applications compared to gradient based methods. Here, we only focus on using the DEA method to generate adversarial perturbations, but its applications to other machine-learning-phases-of-matter scenarios are straightforward and we leave them for future studies.

Topological phases of matter. —We now turn to the case of topological phases of matter. Unlike conventional phases (such as the paramagnetic/ferromagnetic phases discussed above), topological phases do not fit into the paradigm of symmetry breaking [43] and are described by nonlocal topological invariants [44, 45], rather than local order parameters. This makes topological phases harder to learn in general. Notably, a number of different approaches, based on either supervised [15, 23, 46] or unsupervised [22] learning paradigms, have been proposed recently and some of them been demonstrated in proof-of-principle experiments [25, 26].

The obtaining of adversarial examples might also be more challenging, since the topological invariants capture only the global properties of the systems and are insensitive to local perturbations. Here, in this section we study the vulnerability of machine-learning approaches to topological phases and show that adversarial examples do exist in this case as well.

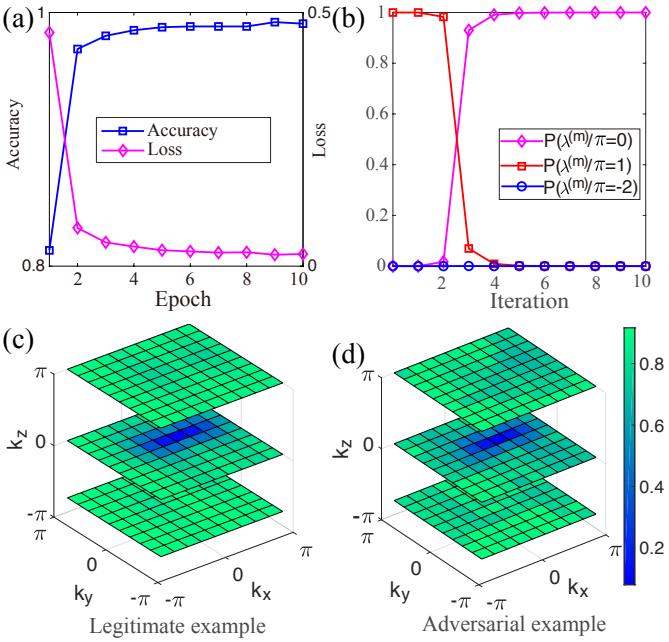

FIG. 3. (a) The average accuracy and loss of the 3D convolutional neural network to classify the topological phases. (b) We use the momentum iterative method to obtain the adversarial examples. This plot shows the classification probabilities as a function of the iteration number. After around two iterations, the network begin to misclassify the samples. (c) A legitimate sample of the first component of the input data (which is related to the density matrix in the momentum space). Here, only slices corresponding to $k_z = \pi, 0, -4\pi/5$ are displayed [41]. (d) An adversarial example obtained by the fast gradient sign method, which only differs with the original legitimate one shown in (c) by an tiny amount of noises that are imperceptible to human eyes.

We consider a simple three-band model for 3D chiral topological insulators (TIs) [47, 48]:

$$H_{\text{TI}} = \sum_{\mathbf{k}\in\text{BZ}} \Psi_{\mathbf{k}}^{\dagger} H_{\mathbf{k}} \Psi_{\mathbf{k}},$$

where $\Psi_{\mathbf{k}}^{\dagger} = (c_{\mathbf{k},1}^{\dagger}, c_{\mathbf{k},0}^{\dagger}, c_{\mathbf{k},-1}^{\dagger})$ with $c_{\mathbf{k},\mu}^{\dagger}$ the fermion creation operator at momentum $\mathbf{k} = (k_x, k_y, k_z)$ in the orbital (spin) state $\mu = -1, 0, 1$ and the summation is over the Brillouin zone (BZ); $H_{\mathbf{k}} = \lambda_1 \sin k_x + \lambda_2 \sin k_y + \lambda_6 \sin k_z - \lambda_7(\cos k_x + \cos k_y + \cos k_z + h)$ denotes the single-particle momentum-resolved Hamiltonian, with $\lambda_{1,2,6,7}$ being four traceless Gell-Mann matrices [47]. In real space, H_{TI} represents free fermions hopping on nearest-neighbor sites in a cubic lattice. It has a chiral symmetry $SH_{\mathbf{k}}S^{-1} = -H_{\mathbf{k}}$ specified by the unitary transformation $S \equiv \text{diag}(1, 1, -1)$, which protects a macroscopic zero-energy flat band that can be topologically nontrivial depending on the parameter h. The topological properties for each band can be characterized by a topological invariant

$$\chi^{(\eta)} = \frac{1}{4\pi} \int_{\text{BZ}} \epsilon^{\mu\nu\tau} A_{\mu}^{(\eta)} \partial_{k^{\nu}} A_{\tau}^{(\eta)} d^3\mathbf{k},$$

where $\epsilon^{\mu\nu\tau}$ is the Levi-Civita symbol with $\mu, \nu, \tau \in \{x, y, z\}$, and the Berry connection is $A_\mu^{(\eta)} = \langle\psi_{\mathbf{k}}^{(\eta)}|\partial_{k^\mu}|\psi_{\mathbf{k}}^{(\eta)}\rangle$ with $|\psi_{\mathbf{k}}^{(\eta)}\rangle$ denoting the Bloch state for the η band (here, $\eta = l, m, u$ denotes the lower, middle and upper band, respectively). It is straightforward to obtain that $\chi^{(m)}/\pi = 0, 1,$ and -2 for $|h| > 3, 1 < |h| < 3$, and $|h| < 1$, respectively.

Recently, an experiment has been carried out to simulate H_{TI} with the electron spins in a diamond nitrogen-vacancy (NV) center and a demonstration of the supervised learning approach to topological phases has been reported [25]. Using the measured density matrices in the momentum space (which can be obtained through quantum state tomography) as input data, a trained 3D convolutional neural network (CNN) can correctly identify distinct topological phases with exceptionally high success probability, even when a large portion of the experimentally generated raw data was dropped out or inaccessible. Here, we show that this approach is highly vulnerable to adversarial perturbations—adding a small amount noise to the measured density matrices will lead the trained CNN to make completely incorrect predictions.

To illustrate how this works, we first train a 3D CNN with numerically simulated data (for details, see [41]). Fig. 3(a) shows the training accuracy and the loss with increasing iterations. The training accuracy increases rapidly at the beginning of the training process and then saturate at a high value ($\approx 99\%$). After the training, we fix the model parameters of the CNN and utilize the FGSM, PGD and MIM methods to generate adversarial perturbations (for details, see[41]). Fig. 3(b) shows the confidence probabilities of the classification of a sample with $\chi^{(\text{m})} = 1$ as functions of the MIM iterations. From this figure, $P(\chi^{(\text{m})} = 1)$ decreases rapidly as the iteration number increases and converges to a small value ($\approx 2\%$) after about eight iterations. Meanwhile, $P(\chi^{(\text{m})} = 2)$ increases rapidly and converges to a large value ($\approx 98\%$), indicating a misclassification of the CNN—after about eight iterations, the sample originally from the category $\chi^{(\text{m})} = 1$ is misclassified to belong to the category $\chi^{(\text{m})} = 2$ with a confidence level $\approx 98\%$. In addition, as shown in Fig. 3(c-d), the obtained adversarial sample looks like the same as the original legitimate sample. They only differ by an tiny amount of noise that is imperceptible to human eyes.

The above two examples clearly demonstrate the vulnerability of machine learning approaches to classify different phases of matter. We mention that, although we have only focused on these two examples, the existence of adversarial perturbations is ubiquitous in learning various phases (independent of the learning model and input data type) and the methods used in the above examples can also be used to generate the desired adversarial perturbations for different phases. From a more theoretical computer science perspective, the vulnerability of the phase classifiers can be understood as a consequence of the strong "No Free Lunch" theorem—there exists an intrinsic tension between adversarial robustness and generalization accuracy [49–51]. The data distributions in the scenarios of learning phases of matter typically satisfy the so-called W_2 Talagrand transportation-cost inequality, thus any phase classifier could be adversarially deceived with high probability [52].

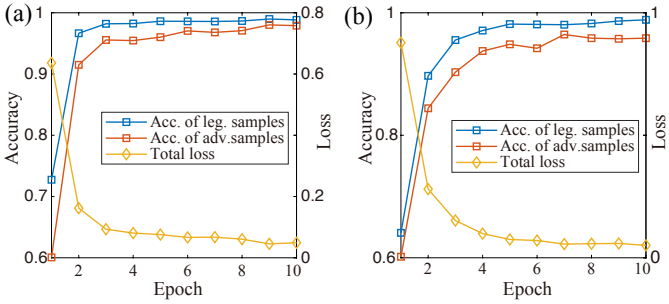

FIG. 4. Strengthening the robustness of phase classifiers (the 3D CNN) by adversarial training [41]. (a) We first numerically generate adequate adversarial examples with the FGSM method, and then retain the CNN with both the legitimate and crafted data. The blue (red) line shows the test accuracy on legitimate (adversarial) data and the "total loss" indicates the loss function for the total data set containing both legitimate and adversarial data. (b) Similar adversarial training for the defense of the PGD attack.

Adversarial training. —In the field of adversarial machine learning, a number of countermeasures against adversarial examples have been developed [53, 54]. Training with adversarial examples is one of these countermeasures to make the classifiers more robust. Here, in order to study how it works for machine learning phases of matter, we apply adversarial training to the 3D CNN classifier used in classifying topological phases. The essential idea is to first generate a substantial amount of adversarial examples and then retrain the targeted classifier with both the original data and the crafted data. Partial of our results is plotted in Fig. 4. From this figure, it is evident that after adversarial training, the test accuracy of the classifier increases significantly (at the end of the training, the test accuracies for both the legitimate and adversarial data are larger than 98%), indicating that the retrained classifier is immune to the adversarial examples generated by the corresponding attacks.

It is worthwhile to mention that the adversarial training method is useful only on adversarial examples which are crafted on the original classifier, such as the FGSM and PGD attacks discussed here. The defense may not work for black-box attacks [55, 56], where an adversary generates malicious examples on a locally trained substitute model. To deal with the transferred black-box attack, one may explore the recently proposed ensemble adversarial training method that retrain the classifier with adversarial examples generated from multiple sources [57]. We also note that one *cannot* expect a universal defense strategy that is able to make the phase classifiers robust to all types of adversarial perturbations, as one method that block a certain kind of attacks will inevitably leave another vulnerability open to other types of attacks which exploit the underlying defense mechanism.

Discussion and conclusion.—Recently, Liu and Wittek have studied the vulnerability of quantum classification from a more theoretical perspective and demonstrated that a perturbation by an amount scaling inversely with the Hilbert dimension of the quantum system should be sufficient to cause a misclassification [58], which is a fundamental feature of quantum classifications in high-dimensional spaces due to the concentration of measure phenomenon [59]. Yet, in practice how to find out all possible adversarial perturbations and show the inverse scaling in learning phases of matter still remains unclear. In addition, we believe that there might be a deep connection between the existence of adversarial perturbations in deep learning and the phenomenon of orthogonality catastrophe in quantum many-body physics [60, 61], where adding an arbitrarily weak local perturbation will make the finial ground state orthogonal to the original one in the thermodynamic limit. However, future investigations are required to firmly establish this connection.

In summary, we have studied the robustness of machine learning approaches in classifying different phases of matter. Our discussion is mainly focused on supervised learning based on deep neural networks, but its generalization to other types of learning models (such as unsupervised learning or support vector machines) and other type of phases are possible and straightforward. Through two concrete examples, we have demonstrated explicitly that typical phase classifiers based on deep neural networks are extremely vulnerable to adversarial examples. Adding a tiny amount of carefully-crafted noises or even just changing a single pixel may cause the classifier to make erroneous predictions at a surprisingly high confidence level. In addition, through adversarial training, we have shown that the robustness of phase classifiers to specific types of adversarial perturbations can be significantly improved. Our results reveal a novel vulnerability aspect for the growing field of machine learning phases of matter, which would benefit future studies across condensed matter physics, machine learning, and artificial intelligence.

We thank Christopher Monroe, John Preskill, Nana Liu, Peter Wittek, Ignacio Cirac, Roger Colbeck, Yi Zhang, Xiaopeng Li, Mucong Ding, Rainer Blatt, Zico Kolter, and Peter Shor for helpful discussions. This work was supported by the start-up fund from Tsinghua University (Grant No. 53330300319) and the National Thousand-Young-Talents Program.

* These authors contributed equally to this work.
† dldeng@tsinghua.edu.cn

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature **521**, 436 (2015).
[2] M. Jordan and T. Mitchell, "Machine learning: Trends, perspectives, and prospects," Science **349**, 255 (2015).
[3] S. D. Sarma, D.-L. Deng, and L.-M. Duan, "Machine learning meets quantum physics," Physics Today **72**, 48 (2019).
[4] G. Carleo and M. Troyer, "Solving the quantum many-body problem with artificial neural networks," Science **355**, 602 (2017).
[5] G. Torlai, G. Mazzola, J. Carrasquilla, M. Troyer, R. Melko, and G. Carleo, "Neural-network quantum state tomography," Nat. Phys. , 1 (2018).
[6] K. Ch'ng, J. Carrasquilla, R. G. Melko, and E. Khatami, "Machine learning phases of strongly correlated fermions," Phys. Rev. X **7**, 031038 (2017).
[7] Y. Nomura, A. S. Darmawan, Y. Yamaji, and M. Imada, "Restricted boltzmann machine learning for solving strongly correlated quantum systems," Phys. Rev. B **96**, 205152 (2017).
[8] L. Wang, "Discovering phase transitions with unsupervised learning," Phys. Rev. B **94**, 195105 (2016).
[9] Y.-Z. You, Z. Yang, and X.-L. Qi, "Machine learning spatial geometry from entanglement features," Phys. Rev. B **97**, 045153 (2018).
[10] D.-L. Deng, X. Li, and S. Das Sarma, "Machine learning topological states," Phys. Rev. B **96**, 195145 (2017).
[11] D.-L. Deng, "Machine learning detection of bell nonlocality in quantum many-body systems," Phys. Rev. Lett. **120**, 240402 (2018).
[12] D.-L. Deng, X. Li, and S. Das Sarma, "Quantum entanglement in neural network states," Phys. Rev. X **7**, 021021 (2017).
[13] X. Gao and L.-M. Duan, "Efficient representation of quantum many-body states with deep neural networks," Nat. Commu. , 662 (2017).
[14] R. G. Melko, G. Carleo, J. Carrasquilla, and J. I. Cirac, "Restricted boltzmann machines in quantum physics," Nature Physics , 1 (2019).
[15] Y. Zhang and E.-A. Kim, "Quantum loop topography for machine learning," Phys. Rev. Lett. **118**, 216401 (2017).
[16] J. Carrasquilla and R. G. Melko, "Machine learning phases of matter," Nat. Phys. **13**, 431 (2017).
[17] E. P. van Nieuwenburg, Y.-H. Liu, and S. D. Huber, "Learning phase transitions by confusion," Nat. Phys. **13**, 435 (2017).
[18] P. Broecker, J. Carrasquilla, R. G. Melko, and S. Trebst, "Machine learning quantum phases of matter beyond the fermion sign problem," Sci. Rep. **7** (2017), 10.1038/s41598-017-09098-0.
[19] S. J. Wetzel, "Unsupervised learning of phase transitions: From principal component analysis to variational autoencoders," Phys. Rev. E **96**, 022140 (2017).
[20] W. Hu, R. R. P. Singh, and R. T. Scalettar, "Discovering phases, phase transitions, and crossovers through unsupervised machine learning: A critical examination," Phys. Rev. E **95**, 062122 (2017).
[21] Y.-T. Hsu, X. Li, D.-L. Deng, and S. Das Sarma, "Machine learning many-body localization: Search for the elusive nonergodic metal," Phys. Rev. Lett. **121**, 245701 (2018).
[22] J. F. Rodriguez-Nieva and M. S. Scheurer, "Identifying topological order through unsupervised machine learning," Nature Physics , 1 (2019).
[23] P. Zhang, H. Shen, and H. Zhai, "Machine learning topological invariants with neural networks," Phys. Rev. Lett. **120**, 066401 (2018).
[24] P. Huembeli, A. Dauphin, and P. Wittek, "Identifying quantum phase transitions with adversarial neural networks," Phys. Rev. B **97**, 134109 (2018).
[25] W. Lian, S.-T. Wang, S. Lu, Y. Huang, F. Wang, X. Yuan, W. Zhang, X. Ouyang, X. Wang, X. Huang, L. He, X. Chang, D.-L. Deng, and L. Duan, "Machine learning topological phases with a solid-state quantum simulator," Phys. Rev. Lett. **122**, 210503 (2019).
[26] B. S. Rem, N. Käming, M. Tarnowski, L. Asteria, N. Fläschner,

C. Becker, K. Sengstock, and C. Weitenberg, "Identifying quantum phase transitions using artificial neural networks on experimental data," Nature Physics , 1 (2019).
[27] A. Bohrdt, C. S. Chiu, G. Ji, M. Xu, D. Greif, M. Greiner, E. Demler, F. Grusdt, and M. Knap, "Classifying snapshots of the doped hubbard model with machine learning," Nature Physics **15**, 921 (2019).
[28] Y. Zhang, A. Mesaros, K. Fujita, S. Edkins, M. Hamidian, K. Ch'ng, H. Eisaki, S. Uchida, J. S. Davis, E. Khatami, *et al.*, "Machine learning in electronic-quantum-matter imaging experiments," Nature **570**, 484 (2019).
[29] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," Pattern Recognition **84**, 317 (2018).
[30] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *Proceedings of the 4th ACM workshop on Security and artificial intelligence* (ACM, 2011) pp. 43–58.
[31] Y. Vorobeychik and M. Kantarcioglu, "Adversarial machine learning," Synthesis Lectures on Artificial Intelligence and Machine Learning **12**, 1 (2018).
[32] D. J. Miller, Z. Xiang, and G. Kesidis, "Adversarial learning in statistical classification: A comprehensive review of defenses against attacks," arXiv:1904.06292 (2019).
[33] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations* (2015).
[34] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," arXiv:1312.6199 (2013).
[35] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," arXiv:1706.06083 (2017).
[36] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018) pp. 9185–9193.
[37] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning* (MIT press, 2016).
[38] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," IEEE Transactions on Evolutionary Computation (2019), 10.1109/TEVC.2019.2890858.
[39] L. Onsager, "Crystal statistics. i. a two-dimensional model with an order-disorder transition," Phys. Rev. **65**, 117 (1944).
[40] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," arXiv:1603.04467 (2016).
[41] See Supplemental Material at [URL will be inserted by publisher] for details on different methods to obtain adversarial examples, the structure of the deep neural networks, and for more numerical results.
[42] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," Journal of global optimization **11**, 341 (1997).
[43] E. M. Lifshitz and L. P. Pitaevskii, *Statistical physics: theory of the condensed state*, Vol. 9 (Elsevier, 2013).
[44] X.-L. Qi and S.-C. Zhang, "Topological insulators and superconductors," Rev. Mod. Phys. **83**, 1057 (2011).
[45] M. Z. Hasan and C. L. Kane, "Colloquium: Topological insulators," Rev. Mod. Phys. **82**, 3045 (2010).
[46] Y. Zhang, R. G. Melko, and E.-A. Kim, "Machine learning z_2 quantum spin liquids with quasiparticle statistics," Phys. Rev. B **96**, 245119 (2017).
[47] T. Neupert, L. Santos, S. Ryu, C. Chamon, and C. Mudry, "Noncommutative geometry for three-dimensional topological insulators," Phys. Rev. B **86**, 035125 (2012).
[48] S.-T. Wang, D.-L. Deng, and L.-M. Duan, "Probe of three-dimensional chiral topological insulators in an optical lattice," Phys. Rev. Lett. **113**, 033002 (2014).
[49] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," arXiv:1805.12152 (2018).
[50] A. Fawzi, H. Fawzi, and O. Fawzi, "Adversarial vulnerability for any classifier," in *Advances in Neural Information Processing Systems* (2018) pp. 1178–1187.
[51] J. Gilmer, L. Metz, F. Faghri, S. S. Schoenholz, M. Raghu, M. Wattenberg, and I. Goodfellow, "Adversarial spheres," arXiv:1801.02774 (2018).
[52] E. Dohmatob, "Limitations of adversarial robustness: strong no free lunch theorem," arXiv:1810.04065 (2018).
[53] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," arXiv:1810.00069 (2018).
[54] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," IEEE transactions on neural networks and learning systems (2019), 10.1109/TNNLS.2018.2886017.
[55] N. Narodytska and S. Kasiviswanathan, "Simple black-box adversarial attacks on deep neural networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (IEEE, 2017) pp. 1310–1318.
[56] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," arXiv:1605.07277 (2016).
[57] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," arXiv:1705.07204 (2017).
[58] N. Liu and P. Wittek, "Vulnerability of quantum classification to adversarial perturbations," arXiv:1905.04286 (2019).
[59] M. Ledoux, *The concentration of measure phenomenon*, 89 (American Mathematical Soc., 2001).
[60] P. W. Anderson, "Infrared catastrophe in fermi gases with local scattering potentials," Phys. Rev. Lett. **18**, 1049 (1967).
[61] D.-L. Deng, J. H. Pixley, X. Li, and S. Das Sarma, "Exponential orthogonality catastrophe in single-particle and many-body localized systems," Phys. Rev. B **92**, 220201 (2015).
[62] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," IEEE transactions on evolutionary computation **15**, 4 (2010).

Supplementary Material for: Vulnerability of Machine Learning Phases of Matter

I. METHODS FOR GENERATING ADVERSARIAL PERTURBATIONS

In the main text, we have shown that the machine learning approaches to phases of matter based on deep neural networks are extremely vulnerable to adversarial examples: adding a tiny amount of carefully-crafted perturbation, which are imperceptible to human eyes, into the original legitimate data will cause the phase classifiers to make incorrect predictions with a high confidence level. Here in this section, we give more technical details on how to obtain the adversarial perturbations.

As discussed in the main text, in supervised learning the training data is labeled $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \cdots, (\mathbf{x}^{(n)}, y^{(n)})\}$ and the task of obtaining adversarial examples reduces to solving the following optimization problem:

$$\max_{\delta \in \Delta} L(h(\mathbf{x}^{(i)} + \delta; \theta), y^{(i)}). \tag{S1}$$

In the adversarial machine learning literature, a number of methods have been introduced to deal with the above optimization problem. We consider two scenarios in this paper, one is called discrete attack scenario, where the adversarial perturbations are discrete and the original legitimate samples are modified by discrete values; the other is called continuous attack scenario, where the perturbations are continuous and the original legitimate samples are modified continuously. For the discrete attack scenario, we mainly apply the differential evolution algorithm [42, 62], which is a population based optimization algorithm for solving complex multi-modal problems and has recently been used for generating one-pixel adversarial perturbations to fool deep neural networks in image recognition [38]. For the continuous attack scenario, we use a number of attacking methods, including fast gradient sign method (FGSM) [33, 35], projected gradient descent (PGD)[35] and momentum iterative method (MIM) [36].

For the case of the ferromagnetic Ising model, we apply both the discrete and continuous attacks, whereas for the case of topological phases of matter we apply only the continuous attacks. We use cleverhans [**?**] to implement FGSM, PGD, and MIM for both the Ising and chiral topological insulator cases. In each case, we produce the adversarial samples based on the origin legitimate training set. We define the success ratio as the proportion of adversarial samples that successfully fool the classifier. In the following, we briefly sketch the essential ideas for each attacking methods used in this paper. For each method, we also provide a pseudocode to clearly illustrate how it works.

Algorithm 1 The Differential Evolution Algorithm

Input A legitimate sample $(\vec{x}, y)$.
Input The iteration number T, the population size n, the number m of pixels to be changed, the mutual factor M.
Output An adversarial example $\vec{x}^*$.

```
Set the position bound B to be the shape of x⃗
Randomly generate perturbation X_i = (a, b, s)^m with (a, b) ∈ B and s = 1 − x_ab for i = 1, 2, ..., n
for t = 1, 2, ..., T do
    Get adversarial sample x⃗_i^* by X_i
    Feed x⃗_i^* to the model to get the confidence on each class.
    Set P_i (the confidence probability for the wrong classification category) as the evaluation threshold.
    Generate children by X'_i = X_j + M(X_k − X_l) with randomly chosen j, k, l.
    Get adversarial sample x⃗_i^{*'} by X'_i, feed them to the model, get P'_i.
    for i = 1, 2, ..., n do
        if P'_i > P_i then
            X_i=X'_i
        end if
    end for
end for
Find the X_p that has the highest confidence probability for the wrong classification category, apply X_p to x⃗ to get x⃗^*
return x⃗^*
```

A. Differential evolution algorithm

Differential evolution is a population based optimization algorithm and is arguably one of the most powerful stochastic real-parameter optimization algorithms in solving complex multi-modal optimization problems [42, 62]. It belongs to the general class of evolutionary algorithms and the computational steps it takes are quite similar to these taken by a standard evolutionary algorithm. Yet, unlike traditional evolutionary algorithms, the differential evolution algorithm perturbs the current generation population members with the scaled differences of randomly chosen distinct population members. More specifically, during each iteration we randomly generate a new set of candidate solutions (called children) according to the current population (parents), and then compare the children with their corresponding parents, replacing the parents if the children have higher fitness value.

We apply the differential evolution algorithm in the "black-box" setting to generate adversarial examples for the ferromagnetic Ising model [38], where we assume no prior information about the classifier's internal structures and only discrete changes of the samples could be made: we first generate some counterfeit samples by reversing a number of magnetic moments of the legitimate sample randomly. We denote these samples as $X_1, X_2, \ldots, X_n$, where n is the population size. We then feed these samples into the classifier to obtain the confidence probability for each configuration. Then we produce new counterfeit samples, which are called children,

Algorithm 2 Fast Gradient Sign Method

Input The trained model h, loss function L, the legitimate sample $(\vec{x}, y)$.
Input The perturbation bound ϵ, upper and lower bound $x_{\text{min}}, x_{\text{max}}$.
Output An adversarial example $\vec{x}^*$.

Input $\vec{x}$ into F to get $\nabla_x L(\theta, \vec{x}, y)$
for Every component i of $\vec{x}$ **do**
  $\delta_i = \epsilon \cdot \text{sign}(\nabla_x L(\theta, \vec{x}, y)_i)$
  $x_i^* = x_i + \delta_i$
  if $x_i^* > x_{\text{max}}$ **then**
    $x_i^* = x_{\text{max}}$
  end if
  if $x_i^* < x_{\text{min}}$ **then**
    $x_i^* = x_{\text{min}}$
  end if
end for
return $\vec{x}^*$

Algorithm 3 Projected Gradient Descent Method

Input The trained model h, loss function L, the legitimate sample $(\vec{x}, y)$.
Input The perturbation bound ϵ, iteration number T, upper and lower bound $x_{\text{min}}, x_{\text{max}}$.
Output An adversarial example $\vec{x}^*$.

$\vec{x_0} = \vec{x}$
$\alpha = \frac{\epsilon}{T}$
for $i = 1, \ldots, T$ **do**
  Input $\vec{x}_{i-1}$ into F to get $\nabla_x L(\theta, \vec{x}_{i-1}, y)$
  for Every component j of $\vec{x}_{i-1}$ **do**
    $\delta_j = \alpha \cdot \text{sign}(\nabla_x L(\theta, \vec{x}_{i-1}, y)_i)$
    $(x_i)_j = (x_{i-1})_j + \delta_j$
    if $(x_i)_j > x_{\text{max}}$ **then**
      $(x_i)_j = \pi_C((x_i)_j) = 2x_{\text{max}} - (x_i)_j$
    end if
    if $(x_i)_j < x_{\text{min}}$ **then**
      $(x_i)_j = \pi_C((x_i)_j) = 2x_{\text{min}} - (x_i)_j$
    end if
  end for
end for
return $\vec{x}^* = \vec{x}_T$

based on prior samples:

$$X_i' = X_j + M(X_k - X_l), \tag{S2}$$

where M is the mutual factor (larger M leads to larger search radius but take longer time to converge). j, k, l are chosen randomly from $[n]$. If these children have better performance (i.e., higher confidence probability for the wrong classification category), then we replace their corresponding parents with these children. We repeat this procedure with several iterations until it converges and the desired adversarial samples are obtained. For the particular Ising case considered in this paper, we denote every children generation as a sequence of $(a, b, s)^m$, where (a, b) is the changed pixels' positions, s is the value after the reversing (which is restricted to be either 0 or 1), and m is the number of changed pixels (spins). A pseudocode representation of the differential evolution algorithm is shown in Algorithm 1.

It is worthwhile to mention that the differential evolution algorithm cannot guarantee that the optimal solution will be obtained. It is possible that the algorithm may only yields certain local minima. In our scenario, this means that the adversarial examples we obtained may not be the most effective ones to fool the classifier.

B. Fast gradient sign method

The fast gradient sign method is a simple one-step scheme for solving Eq. (S1) and has been widely used in the adversarial machine learning community [33, 35]. Before introducing this method, let us first introduce the fast gradient method (FGM).

We work in a white-box attack setting, where full information about the classifier is assumed. Our goal is to maximize the loss function for a particular input data $\vec{x}$ to generate the adversarial sample $\vec{x}^*$. Since we know all parameters of the model, we can compute the fastest increasing direction on the position $\vec{x}$, which is just the gradient of the loss function: $\nabla_x L(\theta, \vec{x}, y)$. The FGM is a one-step attack which perturbs $\vec{x}$ along the direction of the gradient with one particular stepsize:

$$\delta_{\text{FGM}} = \max_{\delta \in \Delta} \langle \nabla_x L(\theta, \vec{x}, y), \delta \rangle. \tag{S3}$$

The perturbation is constrained within l_p-norm bound: $\|\delta\|_p \leq \epsilon$.

If we take l_∞-norm bound, we get a simple rule for obtaining the adversarial perturbation via FGSM:

$$\delta_{\text{FGSM}} = \epsilon \cdot \text{sign}(\nabla_x L(\theta, \vec{x}, y)). \tag{S4}$$

For different problems, there are other particular perturbation bounds as well. One of the most useful bounds is the rectangular-box-like bound, where each component of the adversarial sample is bounded by some constant numbers $x_{\text{min}} \leq x \leq x_{\text{max}}$. For example, for the case of chiral topological insulators studied in this paper, we require that every component of $\vec{x}$ be bounded by $[-1, 1]$. If the adversarial sample has components exceeding this bound, FGSM simply change the value of this component to be the value of either x_{min} or x_{max}. A pseudocode representation of the fast gradient sign method is shown in Algorithm 2.

C. Projected gradient descent method

As shown in Eq. (S4), one may interpret the FGSM as a simple one-step scheme for maximizing the inner part of the saddle point formulation. With a small stepsize, FGSM may perform well. But with a large stepsize, FGSM can perform poorly since the gradient of the loss function may change significantly during this step. To deal with this problem, a more

Algorithm 4 Momentum Iterative Method

Input The trained model h, loss function L, the legitimate sample $(\vec{x}, y)$.
Input The perturbation bound ϵ, iteration number T, decay factor μ, upper and lower bound $x_{\min}, x_{\max}$.
Output An adversarial example $\vec{x}^*$.

$\vec{x}_0 = \vec{x}, a_0 = 0$
$\alpha = \frac{\epsilon}{T}$
for $t = 1, \ldots, T$ **do**
  Input $\vec{x}_{i-1}$ into F to get $\nabla_x L(\theta, \vec{x}_{t-1}, y)$
  $a_t = \mu \cdot a_{t-1} + \frac{\nabla_x L(\theta, \vec{x}_{t-1}, y)}{||\nabla_x L(\theta, \vec{x}_{t-1}, y)||}$
  for Every component j of $\vec{x}_{t-1}$ **do**
    $\delta_j = \alpha \cdot \text{sign}((a_t)_j)$
    $(x_t)_j = (x_{t-1})_j + \delta_j$
    if $(x_t)_j > x_{\max}$ **then**
      $(x_t)_j = \pi_C((x_t)_j) = 2x_{\max} - (x_t)_j$
    end if
    if $(x_t)_j < x_{\min}$ **then**
      $(x_t)_j = \pi_C((x_t)_j) = 2x_{\min} - (x_t)_j$
    end if
  end for
end for
return $\vec{x}^* = \vec{x}_T$

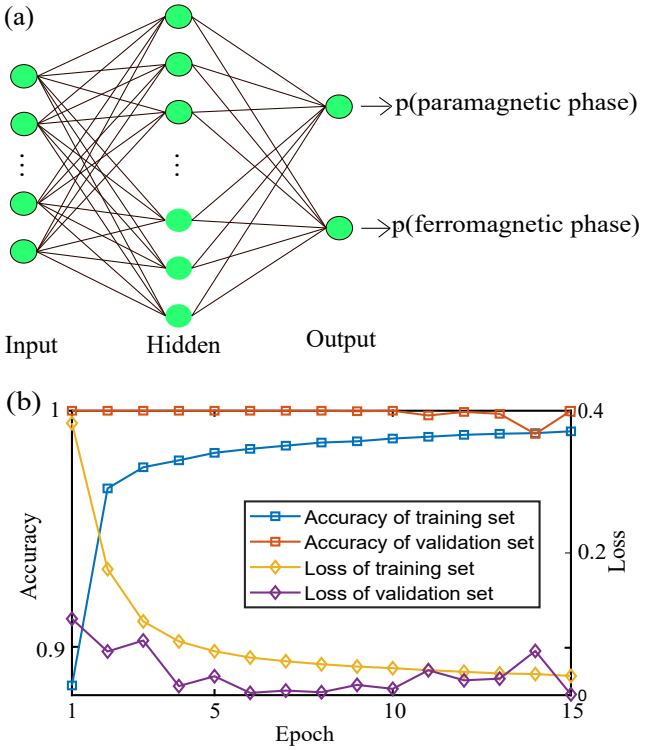

FIG. S1. Machine learning ferromagnetic/paramagnetic phases of the Ising model. (a) The classifier is a fully connected feed forward neural network. It consists of one input layer with 900 neurons which have one-to-one correspondence to the spins of the Ising model, one hidden layer with 100 sigmoid neurons, and one output layer with two softmax neurons outputting the probabilities of the paramagnetic and ferromagnetic phases. (b) The training process. The classifier is trained with numerically simulated data at 40 different temperatures from $T = 0$ to $T = 3.54$. The training set contains 90000 samples, each sample is a array with length 900. The validation set is of size 10000 and the test set is of size 10250. We use RMSprop optimizer with batch size of 256 and learning rate of 10^{-3}. The accuracy is the correct classification percentage and the loss is the value of cross-entropy.

powerful method is its multi-step variant, which is called the projected gradient descent method (PGD). The basic idea of PGD is to use FGSM methods with multiple times (T) and perform projections iteratively to enforce that the perturbation is within an appropriate region [35]. At each step, we check if the proposed update has moved out of the region, and apply a projection back if it does. So the rule for updating is

$$\vec{x}_{t+1} = \pi_C(\vec{x}_t + \alpha \cdot \text{sign}(\nabla_x L(\theta, \vec{x}_t, y))), \tag{S5}$$

where $\alpha = \frac{\epsilon}{T}$ is the stepsize and π_C is the projection operation which projects those points out of the chosen appropriate region [denoted as Δ in Eq. (S1)] back. In our scenarios, the permitted region we choose is the region that restricts every component of $\vec{x}$ to be in $[x_{\min}, x_{\max}]$, therefore, π_C is simply the projection for each component into $[x_{\min}, x_{\max}]$. A pseudocode representation for the projected gradient descent method is shown in Algorithm 3.

D. Momentum iterative method

The FGSM assumes the sign of the gradient of loss function will not change around the data point and generates an adversarial example by applying the sign of the gradient to a legitimate example only once. However, in many practical applications the assumption may not hold when the distortion is large, rendering the adversarial example generated by FGSM "under-fits" the model. On the other hand, iterative FGSM like PGD moves the counterfeit examples gradually in the direction of the sign of the gradient in each iteration and hence can easily drop into poor local extremums and "overfit" the model. To deal with such a dilemma, one can integrate momentum into the iterative FGSM so as to stabilize update directions and escape from local extremums [36]. This is the essential idea of the momentum iterative method.

For a T iterations attack with l_∞-norm constraint ϵ, in every iteration we calculate the gradient descent direction and add the gradient descent direction in the last iteration with a decay factor μ as the accelerated velocity:

$$a_{t+1} = \mu \cdot a_t + \frac{\nabla_{x_t} L(\theta, x_t, y)}{||\nabla_{x_t} L(\theta, x_t, y)||}, \tag{S6}$$

and the rule for updating is:

$$x_{t+1} = x_t + \alpha \cdot \text{sign}(a_{t+1}), \tag{S7}$$

where $\alpha = \frac{\epsilon}{T}$ is the stepsize. A pseudocode representation for the momentum iterative method is shown in Algorithm 4.

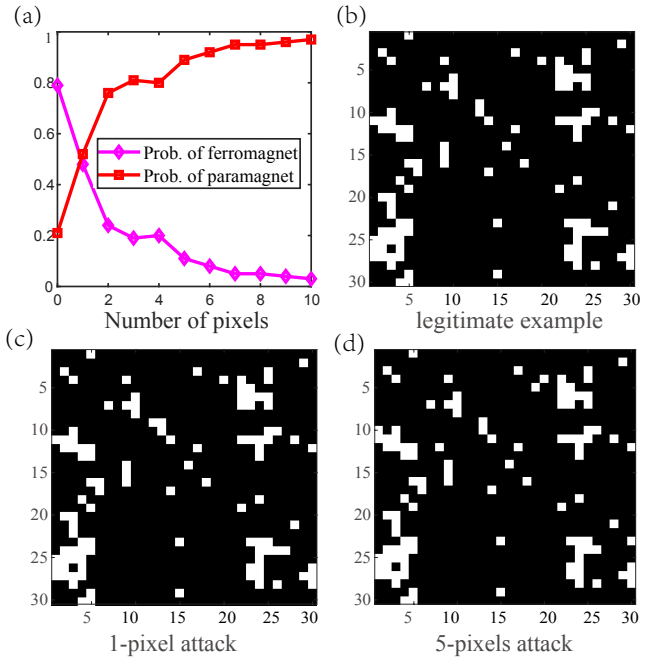

FIG. S2. Performance of the differential evolution algorithm for the case of ferromagnetic Ising model. In this figure, the population n is set to be 100 and the mutual factor M set to be 30. The algorithm stops and returns the adversarial samples when the confidence probability converges with increasing number of iterations. (a) The confidence probabilities for the ferromagnetic and paramagnetic categories versus the number of pixels that are allowed to be changed (i.e., spins allowed to be flipped). We randomly choose a legitimate sample (here the 5067^{th} sample in the validation set) which is correctly classified by the neural network to belong to the ferromagnetic phase with confidence 80%. (b) The legitimate sample from the ferromagnetic phase. Here, each small square corresponds to a spin and black (white) color means the corresponding spin points down (up). (c) An adversarial sample with one pixel changed. The classifier misclassifies this modified sample into the paramagnetic category with confidence 52%. (d) An adversarial sample with five pixels changed. The classifier misclassifies this modified sample into the paramagnetic phase with confidence 90%.

II. MORE DETAILS ON THE TWO CONCRETE EXAMPLES

In this section, we provide more technical details on the neural network structures of the classifiers, the training process, and the defense strategy. In addition, we provide more numerical simulation results for both the examples of the Ising and the chiral topological insulator.

A. The ferromagnetic Ising model

In the main text, we have shown that adding a tiny amount of adversarial perturbation as small as a single pixel can lead the classifier to misclassify a spin-configuration image from the ferromagnetic phase into the paramagnetic category. In this example, our phase classifier is a fully connected feed-forward neural network, which is composed of an input layer with 900 neurons, a hidden layer with 100 sigmoid neurons, and an analogous output layer with two sigmoid neurons, as shown in Fig. S1(a). The neural network is implemented with TensorFlow [40]. The input data is the equilibrium spin configurations sampled from Monte Carlo simulations, same as in Ref. [**?**]. We use 0 and 1 to represent whether the spin is up or down. The lattice size is fixed to be 30×30, and therefore the input data $\vec{x}$ are $\{0, 1\}$ arrays with length 900. The training and validation sets are both numerically generated with Monte Carlo simulations [16], and their sizes are 90000 and 10000, respectively. We use the RMSprop as the optimizer with batch size of 256 and the learning rate is set to be 10^{-3}. In Fig. S1(b), we plot the results for the training process. From this figure, it is clear that the accuracy increases (the loss decreases) as the number of epochs increases, and after 15 epochs the network can successfully classify samples from the validation/test set with a high accuracy larger than 97%.

After the training process, we fix the parameters of the classifier and utilize different methods to generate adversarial examples. The first method we use is the differential evolution algorithm, which is a discrete attack method. Fig.2 (b) in the main text plots the classification probabilities for ferromagnetic and paramagnetic phases as a function of the iteration number. It is clear that this algorithm is very powerful and after around two iterations, the classifier begins to misclassify the samples. Fig. 2(c) of the main text gives an adversarial example that only differs with the original legitimate one by a single pixel. Intuitively, if we modify the original sample by flipping more spins, the confidence probability for the classifier to misclassify the modified sample will increase. This is also verified in our numerical simulations and partial of our results are shown in Fig. S2. In Fig. S2(a), we randomly choose a legitimate sample from the paramagnetic phase , which is shown in Fig. S2(b). Without changing any pixel (flipping a spin), the classifier will correctly identify the sample as from the paramagnetic phase with a confidence level $\approx 80\%$. However, this confidence probability will decrease rapidly as the number of pixels that are allowed to change increases. This is clearly demonstrated in S2(a). Fig. S2(c) and Fig. S2(d) plot two adversarial examples with one and five pixels of the original sample (Fig. S2(b)) changed, respectively. For this particular legitimate sample, changing one pixel (five pixels) will lead the classifier to misclassify it with confidence 52% (90%). We note that in order to obtain Fig.2 (b) and Fig. 2 (c) in the main text, the hyper parameters we used are the same as these in Fig. S2.

As discussed in the main text, we may also regard H_{Ising} as a quantum Hamiltonian and the input data to be the local magnetization, and thus we allow the input data to be continuously modified. In this case, we can use different methods, such as FGSM, PGD, and MIM discussed in section I, to generate adversarial examples. Partial of our results are shown in Fig. S3. In Fig. S3(a), we randomly choose a sample from the paramagnetic phase, which is plotted in Fig. S3(b). At

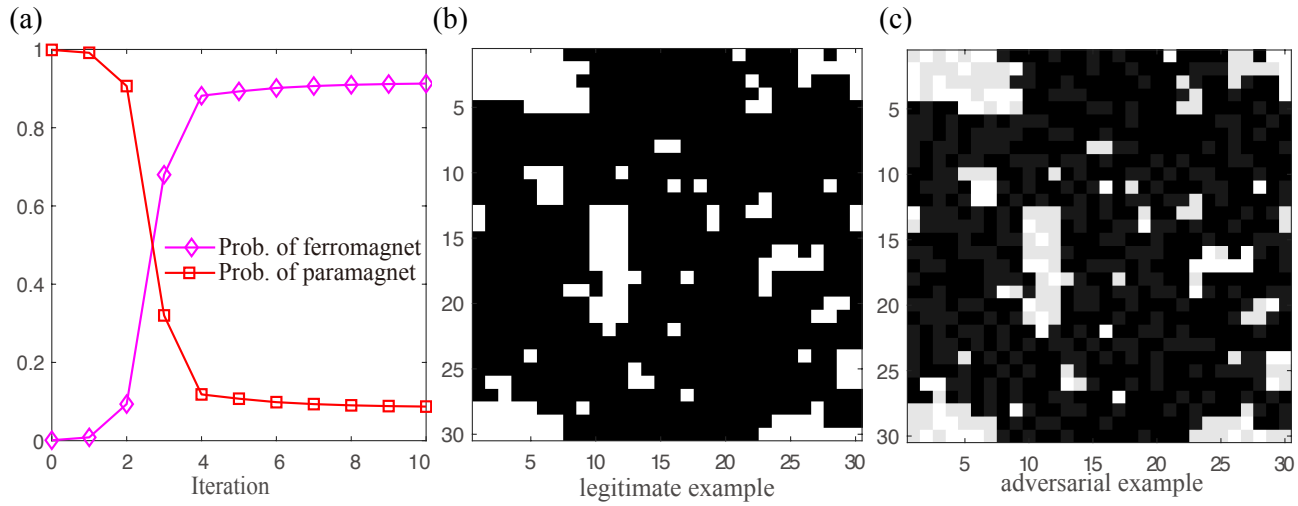

FIG. S3. Performance of the momentum iterative method (MIM) for the case of Ising model. (a) The classification probabilities of the ferromagnetic and paramagnetic phases as a function of the iteration number.It is clear that after around three iterations, the classifier will begin to misclassify the samples, and after ten iterations the slightly modified samples will be identified as belongs to the ferromagnetic category with confidence $\geq 90\%$. (b) A randomly chosen legitimate sample from the paramagnetic phase. (c) An adversarial example obtained by MIM, which is slightly different from the original sample. Here, the perturbation is restricted to be within $\|\delta\|_\infty \leq 0.1$.

the beginning, the classifier can correctly identify this sample as in the paramagnetic category with confidence $\geq 99\%$. We then use MIM to modify the original sample and after around three iterations, the classifier will begin to make incorrect predictions, and after ten iterations it will misclassify the sample to be in the ferromagnetic phase with confidence $\geq 90\%$. Fig. S3(c) shows the corresponding adversarial example obtained by MIM after ten iterations.

B. Topological phases of matter

For the example of topological phases of matter, the classifier we consider is a 3D convolutional neural network (CNN), as shown in Fig. S4 (a). It consists of two 3D convolution layers, a 3D max pooling layer, a dropout layer with rate 0.4 to avoid overfitting, and a flattening layer connected with two fully-connected layers with 0.55 dropout. The output layer is a softmax layer outputting the probability for the three possible topological phases. We use the RMSprop as the optimizer with batch size of 128. The loss function is chosen to be the cross-entropy. The learning rate is set to be 10^{-3}.

In our scenario, the input data are the density matrices on a $10 \times 10 \times 10$ momentum grid and we express each density matrix ρ as [25]:

$$\rho = \frac{1}{3}(\mathbb{I} + \sqrt{3}\mathbf{b} \cdot \vec{\lambda}), \tag{S8}$$

where $\vec{\lambda}$ is a vector consists of the eight Gell-Mann matrices, $\mathbb{I}$ is the three-by-three identity matrix, and $\mathbf{b} = (b_1, b_2, \cdots, b_8)$ with $b_i = \frac{1}{2}\sqrt{3}\mathrm{tr}(\rho\lambda_i)$. Therefore, in this representation of the density matrices each sample of the input data has the form $10 \times 10 \times 10 \times 8$, which can be regarded as $10 \times 10 \times 10$ pixels image with 8 color channels. To train the network, we numerically generate 5001 samples as the training set and 2001 samples as the validation set with parameter h varied uniformly from -5 to 5. The training process is shown in Fig. S4(b). The performance of the training process is shown in Fig. S4(b). From this figure, the training accuracy for the training set increases rapidly at the beginning and then saturates at a high value ($\approx 99\%$), whereas the loss for the training set decrease rapidly at the beginning and then saturate at a small value (≈ 0.05). This indicates that the classifier performs remarkably well on the legitimate samples.

After the training was done, we use three different methods, namely FGSM, PGD, and MIM, to generate adversarial examples. We find that all these methods work notably well and can generate adversarial examples with success ratio larger than 76% (i.e., for more than 76% of the legitimate samples, these methods can successfully output the corresponding adversarial examples) with the perturbation bounded by $||\delta||_\infty \leq 0.2$. Partial of our results are plotted in Fig. 3 (b-d) in the main text. In order to obtain Fig. 3(b), we randomly choose a sample from the category with topological invariant $\lambda^{(m)}/\pi = 1$. At the beginning, the classifier can successfully identify this sample with almost unit confidence probability ($\approx 99.5\%$). Here, we use MIM to generate adversarial perturbations with restriction $||\delta||_\infty \leq 0.05$. The classifier will begin to misclassify the slightly modified sample to be in the category $\lambda^{(m)}/\pi = 0$ after about three iterations. The confidence probability for the misclassification approaches 98% after four iterations and begins to saturate at this value. Fig.3(c) in the main text plots the original legitimate sample chosen in Fig. 3(c) and Fig. 3(d) plots its corresponding adversarial example obtained by MIM. As discussed in Eq. (S8), each density matrix is represented by a vector $\mathbf{b}$ of length eight. For easy visualization, in Fig.

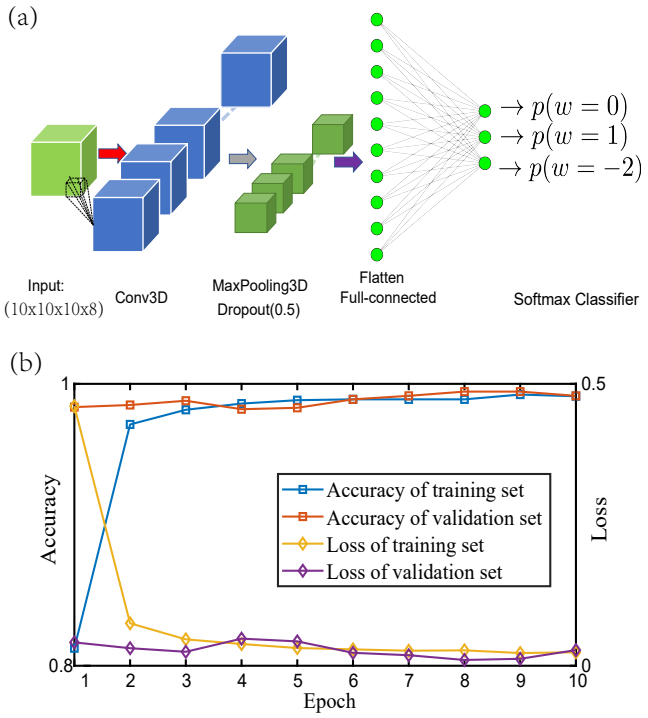

FIG. S4. Learning topological phases with a 3D convolutional neural network (CNN). (a) The structure of the CNN phase classifier. (b) The training process. Here, the loss function is chosen to be the cross-entropy. It is clear that after ten epochs, the classifier can successfully identify the samples from both the training and validation sets with accuracy $\approx 99\%$.

3(c-d) in the main text we plot only the first component of $\mathbf{b}$, namely b_1, for each momentum point. We mention that one can also use the experimental data obtained recently in Ref. [25] with a solid-state simulator to generate adversarial examples. This is also observed in our our numerical simulations.

C. Adversarial training

In order to increase the robustness of the deep neural networks to adversarial perturbations, a number of methods have been developed in the adversarial machine learning literature. The simplest and most straightforward one is adversarial training [35]. Its essential idea is to first generate a substantial amount of adversarial examples with certain attacking methods and then retrain the classifier with both the original legitimate data and the crafted data. After retraining, the classifier will be more immune to the corresponding attacks and its robustness to the adversarial perturbations will be enhanced.

In the main text, Fig. 4 plots the results of the adversarial training to defense the FGSM and PGD attacks for the 3D CNN. In order to obtain this figure, we use 5001 legitimate samples and their corresponding 5001 adversarial samples as the training set for retraining the network. After every epoch we calculate the accuracy of legitimate samples and adversarial samples, respectively. The loss is calculated on both legitimate and adversarial samples. We mention that the adversarial samples used for training are different in every epoch. In each epoch, we use the current model and legitimate samples to generate adversarial samples, using both legitimate and adversarial samples to train the model, and then in the next epoch, we generate new adversarial samples by the model with updated parameters. From the figure, it is clear that the accuracy for both the legitimate samples and adversarial samples increase as the number of epochs increase, and saturate at notable values larger than 0.96. This demonstrates that the retrained classifier is indeed much more robust to the adversarial perturbations generated by the corresponding attacking methods.