Adversarial Machine Learning Phases of Matter

Si Jiang, 1, * Sirui Lu, 1,2, * and Dong-Ling Deng 1,3, |

1Center for Quantum Information, IIIS, Tsinghua University, Beijing 100084, People’s Republic of China
2Max-Planck-Institut für Quantenoptik, Hans-Kopfermann-Str. 1, D-85748 Garching, Germany
3Shanghai Qi Zhi Institute, 41th Floor, AI Tower, No. 701 Yunjin Road, Xuhui District, Shanghai 200232, China

We study the robustness of machine learning approaches 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 perturbations: adding a tiny amount of carefully crafted noises into the original legitimate examples will cause the classifiers to make incorrect predictions at a notably high confidence level. Through the lens of activation maps, we find that some important underlying physical principles and symmetries remain to be adequately captured for classifiers with even near-perfect performance. This explains why adversarial perturbations exist for fooling these classifiers. In addition, we find that, after adversarial training the classifiers will become more consistent with physical laws and consequently more robust to certain kinds of adversarial perturbations. Our results provide valuable guidance for both theoretical and experimental future studies on applying machine learning techniques to condensed matter physics.

Machine learning is currently revolutionizing many technological areas of modern society, ranging from image/speech recognition to content filtering on social networks and 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 [13–30]. Following these approaches, notable proof-of-principle experiments with different platforms [31–34], including electron spins in diamond nitrogen-vacancy (NV) centers [31], doped CuO2 [34], and cold atoms in optical lattices [32, 33], 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 [35–41], it has been shown that machine learning models can be surprisingly vulnerable to adversarial perturbations if the dimension of the data is high enough—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. [42], where adding a small adversarial perturbation, though unnoticeable to human eyes, will cause the classifier to misclassify 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 to obtain the adversarial perturbations. To better understand why these adversarial examples can fool the classifier in the physics context, we open up the neural network and use an idea borrowed from the machine learning community, called activation map [43, 44], to study how the classifier infers different phases of matter. Further, we show that an adversarial training-based defense strategy improves classifiers’ ability to resist specific perturbations and how well the underlying physical principles are captured. Our results shed light on the fledgling field of machine-learning applications in condensed matter physics, which may provide an important paradigm for future theoretical and experimental studies as the field matures.

To begin with, we introduce the main ideas of adversar-
Adversarial examples in machine learning, which involves the generation and defense of adversarial examples [35–38]. Adversarial examples are instances with small intentionally crafted perturbations to cause the classifier make incorrect predictions. Under the supervised learning scenario, we have a training data set with labels $D_n = \{ (x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\}$, a classifier $h(\cdot; \theta)$ and a loss function $L$ to evaluate the classifier’s performance. Adversarial examples generation task can be reduced to an optimization problem: searching for a bounded perturbation that maximizes the loss function [45]:

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

In the machine learning literature, a number of methods have been proposed to solve the above optimization problem, along with corresponding defense strategies [39, 46–49]. We employ some of these methods and one general defense strategy, adversarial training, on two concrete examples: one concerns the conventional paramagnetic/ferromagnetic phases with a two-dimensional classical Ising model [14, 16, 17]; the other involves topological phases with experimental raw data generated by a solid-state quantum simulator [31].

The ferromagnetic Ising model.—The first example we consider involves the ferromagnetic Ising model defined on a 2D square lattice: $H_{\text{Ising}} = -J \sum_{i,j} \sigma_i \sigma_j$, where the Ising variables $\sigma_i = \pm 1$ 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 $T_c = 2/\ln(1 + \sqrt{2}) \approx 2.366$ [50], between a high-temperature paramagnetic phase and a low-temperature ferromagnetic phase. In the context of machine learning phases of matter, different pioneering approaches, including these based on supervised learning [16], unsupervised learning [14], or a confusion scheme combining both [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.

To study the robustness of these introduced machine learning approaches to adversarial perturbations, we first train a powerful classifier which has comparable performance with the ones shown in [16]. After training, the network can successfully classify data from the test set with a high accuracy larger than $97\%$ [45]. Then we try to obtain adversarial perturbations to attack this seemingly ideal classifier. It is natural to consider discrete attack in this scenario since the spin configuration in Ising model can be only discretely changed as spin flips. We apply the differential evolution algorithm (DEA) [51] to the Monte Carlo sampled spin configurations and obtain the corresponding adversarial perturbations. A concrete example found by DEA is illustrated in Fig. 2(a–b). Initially, the legitimate example shown in (a) is in the ferromagnetic phase, which has magnetization $M = |\sum_1^N \sigma_i|/N = 0.791$ and the classifier classifies it into the correct phase with confidence $72\%$. DEA obtains an adversarial example shown in (b) by flipping only a single spin, which is located in the red circle. This new spin configuration has almost the same magnetization $M = 0.789$ and should still belong to the ferromagnetic phase, but the classifier misclassifies it into the paramagnetic phase with confidence $60\%$. If we regard $H_{\text{Ising}}$ 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, as shown in the Supplementary Material [45].

To understand why this adversarial example crafted with tiny changes leads to misclassification, we dissect the classifier by estimating each position’s importance to the final prediction, which we call the activation map of the classifier [45]. In Fig. 2(c) we depict the activation map for ferromagnetic phase. It is evident that the classifier makes prediction mainly based on positions with large activation values (dark colors). The position where the adversarial spin flip happens in Fig. 2(b) has an activation value $3.28$, which is the forth largest among all 900 positions. Then we enumerate all positions with activation values larger than $2.6$ and find that single
spin flips, which changes the contribution to ferromagnetic phase from positive to negative, can all lead to misclassification [45]. The values at different positions for the activation map is found to be nonuniform, which contradicts to the physical knowledge that each spin contributes equally to the order parameter $M$. This explains why the classifier is vulnerable to these particular spin flips. We remark that in the traditional machine learning realm of classifying daily-life images (such as images of cats and dogs), such an explanation is unattainable due to the absence of a sharply defined “order parameter”.

**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 [52] and are described by nonlocal topological invariants [53, 54], 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, 55] or unsupervised [22, 56–66] learning paradigms, have been proposed recently and some of them been demonstrated in proof-of-principle experiments [31, 32].

The obtaining of adversarial examples is more challenging, since the topological invariants capture the global properties of the systems and are insensitive to local perturbations. We consider a three-band model for 3D chiral topological insulators (TIs) [67, 68]:

$$H_{TI} = \sum_{k \in BZ} \Psi_k^\dagger H_k \Psi_k,$$

where $\Psi_k = (c_{k,\mu}, c_{k,\sigma}, c_{k,-\sigma})^T$ with $c_{k,\mu}$ the fermion creation operator at momentum $k = (k_x, k_y, k_z)$ in the orbital (spin) state $\mu = -1, 0, 1$ and the summation is over the Brillouin zone (BZ); $H_k = \lambda_1 \sin k_x + \lambda_2 \sin k_y + \lambda_6 \sin k_z - \lambda_7 (\cos k_x + \cos k_y + \cos k_z + h)$ denotes the single-particle Hamiltonian, with $\lambda_{1,2,6,7}$ being four traceless Gell-Mann matrices [67].

The topological properties for each band can be characterized by a topological invariant $\chi(m) [69]$ and it is straightforward to obtain that $\chi(m) = 0, 1,$ 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 NV center and a demonstration of the supervised learning approach to topological phases has been reported [31]. 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.

We first train a 3D CNN with numerically simulated data. The training curve is shown in Fig. 3(a) and the accuracy on validation data saturates at a high value ($\approx 99\%$) [45]. After the training, we fix the model parameters of the CNN and utilize the Fast Gradient Sign Method (FGSM) [48], projected gradient descent (PGD) [48] and Momentum Iterative Method (MIM) [49] to generate adversarial perturbations [70]. Fig. 3(b) shows the confidence probabilities of the classification of a sample with $\chi(m) = 0$ as functions of the MIM iterations. From this figure, $P(\chi(m) = 0)$ decreases rapidly as the iteration number increases and converges to a small value ($\approx 2\%$) after about eight iterations. Meanwhile, $P(\chi(m) = 1)$ increases rapidly and converges to a large value ($\approx 98\%$), indicating a misclassification of the classifier—after about eight iterations, the sample originally from the category $\chi(m) = 0$ is misclassified to belong to the category $\chi(m) = 1$ with a confidence level $\approx 98\%$ [45]. We note that a direct calculation of the topological invariant through integration confirms that $\chi(m) = 0$ for this adversarial example, indicating that the tiny perturbation would not affect the traditional methods.

It is more challenging to figure out why the topological phases classifier is so vulnerable to the adversarial perturbation. Since these convolutional kernels are repeatedly applied to different spatial windows, we have limited method to calculate the importance of each location, as we do with an Ising classifier. We study the activation maps of all convolutional kernels in the first convolutional layer. We find that the sixth kernel has totally different activation patterns for topologically trivial and nontrivial phases, which acts as a strong indicator for the classifier to distinguish these phases [45]. Specifically, the activation patterns for $\chi(m) = 0, 1$ phases are illustrated in Fig. 3(c-d). We then compare the sixth kernel’s activation maps of the legitimate sample and the adversarial example. As shown in Fig. 3(e-f), we can clearly see that the tiny adversarial perturbation makes the activation map much more correlated with the $\chi(m) = 1$ ones, which gives the classifier high confidence to assert that the adversarial example...
belongs to the $\chi^{(m)} = 1$ phase [45]. This explains why adversarial examples can deceive the classifier.

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 [71–73]. The data distributions in the scenarios of learning phases of matter typically satisfy the $W_2$ Talagrand transportation-cost inequality, thus a phase classifier could be adversarially deceived with high probability [74].

**Adversarial training.**—In adversarial machine learning, a number of countermeasures against adversarial examples have been developed [75, 76]. Adversarial training, whose essential idea is to first generate a substantial amount of adversarial examples and then retrain the classifier with both original data and crafted data, 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. Partial results are plotted in Fig. 4(a-b). While the classifier’s performance on legitimate examples maintains intact, the test accuracy on adversarial examples increases significantly (at the end of the adversarial training, it increases from about 60% to 98%). This result indicates that the retrained classifier is immune to the adversarial examples generated by the corresponding attacks.

We also study how adversarial training can make the classifiers grasp physical quantities more thoroughly. As shown in Fig. 2(d), the activation map of the Ising model classifier becomes much flatter after adversarial training (the standard deviation of different positions is reduced from 0.88 to 0.20), which indicates that after adversarial learning the output of the classifier is more consistent with the physical order parameter of magnetization where each spin contributes equally, and hence more robust to adversarial perturbations. This is also reflected by the fact that after adversarial training the classifier can identify the phase transition point more accurately, as shown in Fig. 4(c-d).

**Discussion and conclusion.**—We mention that the adversarial training method is useful only on adversarial examples which are crafted on the original classifier. The defense may not work for black-box attacks [77, 78], 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 [79]. In the future, it would be interesting and desirable to find other defense strategies to strengthen the robustness of phase classifiers to adversarial perturbations. In addition, an experimental demonstration of adversarial learning phases of matter together with defense strategies would also be an important step towards reliable practical applications of machine learning in physics.

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 reinforcement learning) and other type of phases are possible. Through two concrete examples, we have demonstrated explicitly that typical phase classifiers based on deep neural networks are extremely vulnerable to tiny adversarial perturbations. We have studied the explainability of adversarial examples and demonstrated that adversarial training significantly improves the robustness of phase classifiers by assisting the model to learn underlying physical principle and symmetries. 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, Peter
Zoller, Xiaopeng Li, Mucong Ding, Rainer Blatt, Zico Kolter, Juan Carrasquilla, and Peter Shor for helpful discussions. This work was supported by the start-up fund from Tsinghua University (Grant No. 53330300319).

* 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. 14, 447 (2018).
[6] 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).
[7] Y.-Z. You, Z. Yang, and X.-L. Qi, “Machine learning spatial geometry from entanglement features,” Phys. Rev. B 97, 045153 (2018).
[8] D.-L. Deng, X. Li, and S. Das Sarma, “Machine learning topological states,” Phys. Rev. B 96, 195145 (2017).
[9] D.-L. Deng, “Machine learning detection of bell nonlocality in quantum many-body systems,” Phys. Rev. Lett. 120, 240402 (2018).
[10] D.-L. Deng, X. Li, and S. Das Sarma, “Quantum entanglement in neural network states,” Phys. Rev. X 7, 021021 (2017).
[11] X. Gao and L.-M. Duan, “Efficient representation of quantum many-body states with deep neural networks,” Nat. Commun. 8, 662 (2017).
[12] R. G. Melko, G. Carleo, J. Carrasquilla, and J. I. Cirac, “Restricted Boltzmann machines in quantum physics,” Nat. Phys. 15, 887 (2019).
[13] K. Ch’ng, J. Carrasquilla, R. G. Melko, and E. Khatami, “Machine learning phases of strongly correlated fermions,” Phys. Rev. X 7, 031038 (2017).
[14] L. Wang, “Discovering phase transitions with unsupervised learning,” Phys. Rev. B 94, 195105 (2016).
[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] F. Broecker, J. Carrasquilla, R. G. Melko, and S. Trebst, “Machine learning quantum phases of matter beyond the fermion sign problem,” Sci. Rep. 7, 8823 (2017).
[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. Rodríguez-Nieva and M. S. Scheurer, “Identifying topological order through unsupervised machine learning,” Nat. Phys. 7, 790 (2017).
[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] P. Suchsland and S. Wessel, “Parameter diagnostics of phases and phase transition learning by neural networks,” Phys. Rev. B 97, 174435 (2018).
[26] T. Ohtsuki and T. Ohtsuki, “Deep learning the quantum phase transitions in random two-dimensional electron systems,” J. Phys. Soc. Japan 85, 123706 (2016).
[27] T. Ohtsuki and T. Ohtsuki, “Deep learning the quantum phase transitions in random electron systems: Applications to three dimensions,” J. Phys. Soc. Japan 86, 044708 (2017).
[28] T. Ohtsuki and T. Mano, “Drawing phase diagrams of random quantum systems by deep learning the wave functions,” J. Phys. Soc. Japan 89, 022001 (2020).
[29] E. Greplova, A. Valenti, G. Boschung, F. Schäfer, N. Lörch, and S. Huber, “Unsupervised identification of topological order using predictive models,” New J. Phys. 22, 045003 (2020).
[30] R. A. Vargas-Hernández, J. Sous, M. Berciau, and R. V. Krems, “Extrapolating quantum observables with machine learning: Inferring multiple phase transitions from properties of a single phase,” Phys. Rev. Lett. 121, 255702 (2018).
[31] 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.-M. Duan, “Machine learning topological phases with a solid-state quantum simulator,” Phys. Rev. Lett. 122, 210503 (2019).
[32] B. S. Rem, N. Käming, M. Tarnowski, L. Asteria, N. Flaschner, C. Becker, K. Sengstock, and C. Weitenberg, “Identifying quantum phase transitions using artificial neural networks on experimental data,” Nat. Phys. 15, 917 (2019).
[33] A. Bohrdt, C. S. Chiu, G. Ji, M. Xu, D. Greif, M. Greiner, B. S. Rem, N. Käming, M. Tarnowski, L. Asteria, N. Flaschner, 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.-M. Duan, “Machine learning topological phases with a solid-state quantum simulator,” Phys. Rev. Lett. 122, 210503 (2019).
[34] B. S. Rem, N. Käming, M. Tarnowski, L. Asteria, N. Flaschner, C. Becker, K. Sengstock, and C. Weitenberg, “Identifying quantum phase transitions using artificial neural networks on experimental data,” Nat. Phys. 15, 921 (2019).
[35] Y. Zhang, A. Mesaros, K. Fujita, S. Edkins, M. Hamidian, K. Ch’ng, H. Eisaki, S. Uchida, J. S. Davis, E. Khatami, et al., “Classifying snapshots of the doped hubbard model with machine learning,” Nat. Phys. 15, 921 (2019).
[36] B. Biggio and F. Roli, “Wild patterns: Ten years after the rise of adversarial machine learning,” Pattern Recognition 84, 317 (2018).
[37] 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.
[38] Y. Vorobyevich and M. Kantarcioğlu, “Adversarial machine learning,” Synthesis Lectures on Artificial Intelligence and Machine Learning 12, 1 (2018).
[39] D. J. Miller, Z. Xiang, and G. Kesidis, “Adversarial learning targeting deep neural network classification: A comprehensive review of defenses against attacks,” Proceedings of the IEEE 108, 402 (2020).
[40] I. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” in International Conference on Learning Representations (2015).
[40] N. Liu and P. Wittek, “Vulnerability of quantum classification to adversarial perturbations,” Phys. Rev. A 101, 062331 (2020).
[41] L. Schmidt, S. Santurkar, D. Tsipras, K. Talwar, and A. Madry, “Adversarially robust generalization requires more data,” in Advances in Neural Information Processing Systems (2018).
[42] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” in International Conference on Learning Representations (2014).
[43] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, “Object detectors emerge in deep scene cnns,” in International Conference on Learning Representations (2015).
[44] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, “Learning deep features for discriminative localization,” in IEEE Conference on Computer Vision and Pattern Recognition (2016) pp. 2921–2929.
[45] See Supplemental Material at [URL will be inserted by publisher] for details on different methods to obtain adversarial examples, the derivation and explanation of activation maps, and for more numerical results on both adversarial examples generation and adversarial training.
[46] 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).
[47] S. Das and P. N. Suganthan, “Differential evolution: A survey of the state-of-the-art,” IEEE Transactions on Evolutionary Computation 15, 4 (2010).
[48] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” (2019), arXiv:1706.06083 [stat.ML].
[49] 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.
[50] L. Onsager, “Crystal statistics. i. a two-dimensional model with an order-disorder transition,” Phys. Rev. 76, 17 (1949).
[51] M. E. Fisher, “On a two-dimensional model of a ferromagnet in an external field,” Phys. Rev. Lett. 21, 773 (1968).
[52] J. Su, D. V. Vargas, and K. Sakurai, “One pixel attack for fooling deep neural networks,” IEEE Transactions on Evolutionary Computation (2019).
[53] E. M. Lifshitz and L. P. Pitaevskii, Statistical physics: theory of condensed state, Vol. 9 (Elsevier, 2013).
[54] X.-L. Qi and S.-C. Zhang, “Topological insulators and superconductors,” Rev. Mod. Phys. 83, 1057 (2011).
[55] M. Z. Hasan and C. L. Kane, “Colloquium: Topological insulators,” Rev. Mod. Phys. 82, 3045 (2010).
[56] Y. Zhang, R. G. Melko, and E.-A. Kim, “Machine learning z2 quantum spin liquids with quasiparticle statistics,” Phys. Rev. B 96, 245119 (2017).
[57] L.-W. Yu and D.-L. Deng, “Unsupervised learning of non-hermitian topological phases,” Phys. Rev. Lett. 126 (2021).
[58] M. S. Scheurer and R.-J. Slager, “Unsupervised machine learning and band topology,” Phys. Rev. Lett. 124 (2020).
[59] Y. Long, J. Ren, and H. Chen, “Unsupervised manifold clustering of topological phononics,” Phys. Rev. Lett. 124 (2020).
[60] A. Lidiak and Z. Gong, “Unsupervised machine learning of quantum phase transitions using diffusion maps,” Phys. Rev. Lett. 125 (2020).
[61] Y. Che, C. Gneiting, T. Liu, and F. Nori, “Topological quantum phase transitions retrieved through unsupervised machine learning,” Phys. Rev. B 102 (2020).
[62] K. Fukushima, S. S. Funai, and H. Iida, “Featuring the topology with the unsupervised machine learning,” (2019), arXiv:1908.00281 [cs.LG].
[63] F. Schäfer and N. Lörch, “Vector field divergence of predictive model output as indication of phase transitions,” Phys. Rev. E 99, 062107 (2019).
[64] O. Balabanov and M. Granath, “Unsupervised learning using topological data augmentation,” Phys. Rev. Research 2 (2020).
[65] C. Alexandrou, A. Athenodorou, C. Chrysostomou, and S. Paul, “The critical temperature of the 2d-ising model through deep learning autoencoders,” Eur. Phys. J. B 93, 226 (2020).
[66] E. Greplová, A. Valenti, G. Boschung, F. Schafer, N. Lorch, and S. Huber, “Unsupervised identification of topological order using predictive models,” New Journal of Physics 22, 045003 (2020).
[67] J. Arnold, F. Schäfer, M. Žonda, and A. U. J. Lode, “Interpretable and unsupervised phase classification,” Phys. Rev. Research 3 (2021).
[68] T. Neupert, L. Santos, S. Ryu, C. Chamon, and C. Mudry, “Noncommutative geometry for three-dimensional topological insulators,” Phys. Rev. B 86, 035125 (2012).
[69] 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).
[70] For each band, the topological invariant $\chi^{(n)} (\text{here } l = m, u)$ denotes the lower, middle and upper bands, respectively, can be written as an integral in the 3D momentum space:

$$\chi^{(n)} = \frac{1}{4\pi^2} \int_{BZ} \epsilon^{\mu\nu\tau} \mathcal{A}_\mu \partial_\nu \mathcal{A}_\tau \, d^3k,$$

where $\epsilon^{\mu\nu\tau}$ is the Levi-Civita symbol with $\mu, \nu, \tau \in \{x, y, z\}$, and the Berry connection is $A^{(n)}_{\mu} = (\psi_k^{(n)}) \partial_\nu |\psi_k^{(n)}\rangle$ with $|\psi_k^{(n)}\rangle$ denoting the Bloch state for the $\eta$ band. The topological invariants for each band are related as $\chi^{(s)} = \chi^{(l)} = \chi^{(u)} / 4$ for the three-band chiral topological insulator model studied in this paper.

[71] N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy, et al., “Technical report on the cleverhans v2.1.0 adversarial examples library,” (2018), arXiv:1610.00768 [cs.LG].
[72] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, “Robustness may be at odds with accuracy,” in International Conference on Learning Representations (2019).
[73] A. Fawzi, H. Fawzi, and O. Fawzi, “Adversarial vulnerability for any classifier,” in Advances in Neural Information Processing Systems (2018) pp. 1178–1187.
[74] J. Gilmer, L. Metz, F. Faghri, S. S. Schoenholz, M. Raghun, M. Wattenberg, and I. Goodfellow, “Adversarial spheres,” (2018), arXiv:1801.02774 [cs.CV].
[75] E. Dohmatob, “Generalized no free lunch theorem for adversarial robustness,” in Proeceedings of the 36th International Conference on Machine Learning (2019) pp. 1646–1654.
[76] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, “Adversarial attacks and defenses: A survey,” (2018), arXiv:1810.00069 [cs.LG].
[77] 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).
[78] 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.
[79] N. Papernot, P. McDaniel, and I. Goodfellow, “Transferability in machine learning: from phenomena to black-box attacks using adversarial samples,” arXiv:1605.07277 (2016).
[80] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, “Ensemble adversarial training: Attacks and defenses,” arXiv:1705.07204 (2017).
Supplementary Material for: Vulnerability of Machine Learning Phases of Matter

I. METHODS FOR GENERATING ADVERSARIALPERTURBATIONS

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 \( \mathcal{D} = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\} \) and the task of obtaining adversarial examples reduces to solving the following optimization problem:

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

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 [46, 47], which is a population based optimization algorithm for solving complex multi-modal optimization problems and has recently been used for generating one-pixel adversarial perturbations to fool deep neural networks in image recognition [51]. For the continuous attack scenario, we use a number of attacking methods, including fast gradient sign method (FGSM) [39, 48], projected gradient descent (PGD) [48] and momentum iterative method (MIM) [49].

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 [70] 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.

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 [46, 47]. 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 “blackbox” setting to generate adversarial examples for the ferromagnetic Ising model [51], 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, based on prior samples. Particularly, for each component \( X_i'(s) \) of the children \( X'_i \), we have the following generation rule:

\[
p_s \sim U(0, 1),
\]

\[
X_i'(s) = \begin{cases} [X_j + M(X_k - X_l)](s), & p_s \leq P \\ X_i(s), & p_s > P \end{cases}
\]

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 \( [n]/\{i\} \), \( P \) is called the crossover probability. 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 \( (a, b, s)^N \), where \((a, b)\) is the flipped spins’ positions, \( s \) is the spin after the reversing (which is restricted to be either 0 or 1), and \( N \) is the number of changed 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.
Algorithm 1 The Differential Evolution Algorithm

**Input** A legitimate sample \((\bar{x}, y)\), the trained model \(h(\cdot; \theta)\), the loss function \(L\).

**Input** The iteration number \(T\), the population size \(n\), the mutation factor \(M\), the crossover probability \(P\), the number \(N\) of spins to flip.

**Output** An adversarial example \(\tilde{x}^*\).

1: Set the position bound \(B\) to be the shape of \(\bar{x}\)
2: Randomly generate perturbation \(X_i = \mathcal{R}_{[a, b]}(a, b, s)\) with \((a, b) \in B\) and \(s = 1 - \bar{x}(a, b)\) for \(i = 1, 2, \ldots, n\)
3: Get adversarial sample \(\tilde{x}_i^*\) by adding \(X_i\) on \(\bar{x}\) for \(i = 1, 2, \ldots, n\)
4: for \(t = 1, 2, \ldots, T\) do
5: for \(i = 1, 2, \ldots, n\) do
6: Randomly pick distinct \(j, k, l \in [n]/\{i\}\)
7: Generate children: \(X'_i = X_j + M(X_k - X_l)\)
8: for \(s = 1, 2, \ldots, N\) do
9: Randomly pick \(p\) from \(U(0, 1)\)
10: if \(p > P\) then
11: \(X_i(s) = X_i(s)\)
12: end if
13: end for
14: end for
15: if \(L(h(\bar{x}^*_i; \theta), y) > L(h(\bar{x}^*_j; \theta), y)\) then
16: \(X_i = X'_i\)
17: \(\tilde{x}^*_i = \tilde{x}^*_i\)
18: end if
19: end for
20: end for
21: Find the \(X_p\) among \(\{X_1, X_2, \ldots, X_n\}\) that has the highest confidence probability for the wrong classification category, apply \(X_p\) to \(\bar{x}\) to get \(\tilde{x}^*\)
22: return \(\tilde{x}^*\)

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 [39, 48]. 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 \(\bar{x}\) to generate the adversarial sample \(\tilde{x}^*\). Since we know all parameters of the model, we can compute the fastest increasing direction on the position \(\bar{x}\), which is just the gradient of the loss function: \(\nabla_x L(h(\bar{x}; \theta), y)\). The FGM is a one-step attack which perturbs \(\bar{x}\) along the direction of the gradient with one particular stepsize:

\[
\delta_{\text{FGM}} = \max_{\bar{x} \in \Delta} (\nabla_x L(h(\bar{x}; \theta), y), \delta).
\]  

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

If we take \(l_{\infty}\)-norm bound, we get a simple rule for obtaining the adversarial perturbation via FGM:

\[
\delta_{\text{FGM}} = \epsilon \cdot \text{sign}(\nabla_x L(h(\bar{x}; \theta), y)).
\]  

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 \(\bar{x}\) be bounded by \([-1, 1]\). If the adversarial sample has components exceeding this bound, FGM simply change the value of this component to be the value of either \(x_{\text{min}}\) or \(x_{\text{max}}\). A pseudocode representation of the fast gradient sign method is shown in Algorithm 2.

Algorithm 2 Fast Gradient Sign Method

**Input** The trained model \(h(\cdot; \theta)\), the loss function \(L\), the legitimate sample \((\bar{x}, y)\).

**Input** The perturbation strength \(\epsilon\), upper and lower bound \(x_{\text{min}}, x_{\text{max}}\).

**Output** An adversarial example \(\tilde{x}^*\).

1: Input \(\bar{x}\) into \(h\) to get \(\nabla_x L(h(\bar{x}; \theta), y)\)
2: for each component \(\tilde{x}(\bar{x})\) of \(\bar{x}\) do
3: \(\delta_i = \epsilon \cdot \text{sign}(\nabla_x L(h(\bar{x}; \theta), y)(i))\)
4: \(\tilde{x}(\bar{x}) = \bar{x}(\bar{x}) + \delta_i\)
5: if \(\tilde{x}(\bar{x}) > x_{\text{max}}\) then
6: \(\tilde{x}(\bar{x}) = x_{\text{max}}\)
7: end if
8: if \(\tilde{x}(\bar{x}) < x_{\text{min}}\) then
9: \(\tilde{x}(\bar{x}) = x_{\text{min}}\)
10: end if
11: end for
12: return \(\tilde{x}^*\)

C. Projected gradient descent method

As shown in Eq. (S5), 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 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 [48]. 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

\[
\tilde{x}_{t+1} = \pi_C(\tilde{x}_t + \alpha \cdot \text{sign}(\nabla_x L(h(\theta, \tilde{x}_t), y))),
\]  

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

### Algorithm 3 Projected Gradient Descent Method

**Input** The trained model $h(\cdot, \theta)$, loss function $L$, the legitimate sample $(\bar{x}, y)$.

**Input** The perturbation strength $\epsilon$, iteration number $T$, upper and lower bound $x_{\min}, x_{\max}$.

**Output** An adversarial example $\bar{x}^*$.

1. $\bar{x}_0 = \bar{x}$
2. $\alpha = \frac{\epsilon}{T}$
3. for $i = 1, \ldots, T$ do
   4. Input $\bar{x}_{i-1}$ into $h$ to get $\nabla_x L(h(\bar{x}_{i-1}; \theta), y)$
   5. for every component $\bar{x}_{i-1}(j)$ of $\bar{x}_{i-1}$ do
   6. $\delta_j = \alpha \cdot \text{sign}(\nabla_x L(h(\bar{x}_{i-1}; \theta), y)(j))$
   7. $\bar{x}_i(j) = \bar{x}_{i-1}(j) + \delta_j$
   8. if $\bar{x}_i(j) > x_{\max}$ then
   9. $\bar{x}_i(j) = \pi_{\max}(\bar{x}_i(j)) = 2x_{\max} - \bar{x}_i(j)$
   10. end if
   11. if $\bar{x}_i(j) < x_{\min}$ then
   12. $\bar{x}_i(j) = \pi_{\min}(\bar{x}_i(j)) = 2x_{\min} - \bar{x}_i(j)$
   13. end if
   14. end for
   15. end for
   16. return $\bar{x}^* = \bar{x}_T$

### Algorithm 4 Momentum Iterative Method

**Input** The trained model $h(\cdot, \theta)$, loss function $L$, the legitimate sample $(\bar{x}, y)$.

**Input** The perturbation strength $\epsilon$, iteration number $T$, decay factor $\mu$, upper and lower bound $x_{\min}, x_{\max}$.

**Output** An adversarial example $\bar{x}^*$.

1. $\bar{x}_0 = \bar{x}$, $a_0 = 0$
2. $\alpha = \frac{\epsilon}{T}$
3. for $t = 1, \ldots, T$ do
   4. Input $\bar{x}_{t-1}$ into $h$ to get $\nabla_x L(h(\bar{x}_{t-1}; \theta), y)$
   5. $a_t = \mu \cdot a_{t-1} + \frac{\nabla_x L(h(\bar{x}_{t-1}; \theta), y)}{||\nabla_x L(h(\bar{x}_{t-1}; \theta), y)||}$
   6. for each component $\bar{x}_{t-1}(j)$ of $\bar{x}_{t-1}$ do
   7. $\delta_j = \alpha \cdot \text{sign}(a_t)$
   8. $\bar{x}_t(j) = \bar{x}_{t-1}(j) + \delta_j$
   9. if $\bar{x}_t(j) > x_{\max}$ then
   10. $\bar{x}_t(j) = \pi_{\max}(\bar{x}_t(j)) = 2x_{\max} - \bar{x}_t(j)$
   11. end if
   12. if $\bar{x}_t(j) < x_{\min}$ then
   13. $\bar{x}_t(j) = \pi_{\min}(\bar{x}_t(j)) = 2x_{\min} - \bar{x}_t(j)$
   14. end if
   15. end for
   16. return $\bar{x}^* = \bar{x}_T$

## 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 analysis of adversarial examples. In addition, we provide more numerical simulation results for both the examples of the Ising model 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 input data is the equilibrium spin configurations sampled from Monte Carlo simulations, same as in Ref. [16]. We use 0 and 1 to represent whether the spin is up or down. The lattice size is fixed to be $30 \times 30$, and therefore the input data $\bar{x}$ 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
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.

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(a) 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 ferromagnetic 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 ferromagnetic 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) of 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 $H_{\text{Ising}}$ 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 Sec. I, to generate adversarial examples. Partial of our results are shown in

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 5067th 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 $90\%$. 


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 $||\delta||_\infty \leq 0.1$.

| Position | Original spin | Value in AM | $P'(\text{ferro})$ |
|----------|---------------|-------------|-------------------|
| 18       | $-$           | $-2.751$    | 0.479             |
| 95       | $-$           | $-2.606$    | 0.464             |
| 108      | $-$           | $-2.736$    | 0.462             |
| 116      | $-$           | $-2.651$    | 0.440             |
| 229      | $-$           | $-3.503$    | 0.402             |
| 251      | $+$           | $-3.554$    | 0.902             |
| 289      | $-$           | $-2.727$    | 0.461             |
| 337      | $-$           | $-3.296$    | 0.446             |
| 356      | $-$           | $-2.867$    | 0.451             |
| 396      | $-$           | $-2.830$    | 0.461             |
| 445      | $-$           | $-2.931$    | 0.424             |
| 521      | $-$           | $-2.671$    | 0.881             |
| 534      | $-$           | $-3.277$    | 0.396             |
| 561      | $-$           | $-2.610$    | 0.481             |
| 768      | $+$           | $-2.767$    | 0.880             |
| 783      | $-$           | $-2.637$    | 0.445             |
| 829      | $+$           | $-2.765$    | 0.892             |

TABLE S1. The result of single spin flips. The original spin configuration is classified as ferromagnetic phase with confidence $P_{\text{ferro}} = 72\%$. The “position” is the spin index on the $30 \times 30$ lattice. The “original spin” represents the original spin direction in the legitimate example. The “value in AM” represents the position’s corresponding value in the activation map. “$P'(\text{ferro})$” means the classifier’s confidence to identify the sample into ferromagnetic phase after the a single spin flip on the original legitimate example. From the table, we can find that spin changes to have the different sign to the corresponding value in AM will dramatically decrease the confidence of being ferromagnetic phase.

Fig. S3. In Fig. S3(a), we randomly choose a sample from the paramagnetic phase, which is plotted in Fig. S3(b). At 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.

In Fig. 2(a) of the main text, we briefly introduced the idea of the classifier’s activation map on the Ising model, now we explain in detail about how we get this activation map. Since the classifier structure we used for the Ising model is rather simple, the inference of the classifier can be strictly written as the following:

$$P = \text{softmax}(W_2 \cdot \text{sigmoid}(W_1 \cdot x + b_1) + b_2)$$

$$\sim (W_2 \cdot W_1) x$$  \hspace{1cm} (S9)

where $x$ is the input Ising configuration with shape 900 × 1, $W_1$ is the first layer weight with shape 100 × 900, and $b_1$ is the first layer bias with shape 100 × 1. Similarly, $W_2$ is the second layer weight with shape 2 × 100 and $b_2$ is the second layer bias with shape 2 × 1. We can see that $P_1$, which is the first element of final prediction $P$ and represents the confidence of being ferromagnetic phase, is proportional to $(W_2 \cdot W_1 \cdot x)_1$. Therefore, we denote $(W_2 \cdot W_1)_1\ldots$, which is the fist row of $W_2 \cdot W_1$, as the activation map $W_{\text{ferro}}$. $W_{\text{ferro}}$ has shape 1 × 900 and approximately represents $\partial P_{\text{ferro}}/\partial x$, denoting the weight of each spin on final prediction to the ferromagnetic phase. The spin with the same sign as the corresponding position’s value in activation map will contributes to being classified as the ferromagnetic phase and vice versa. This explains why a single spin flip can change the classifier’s prediction dramatically, either increase or decrease. Using the legitimate example shown in main text Fig. 2(a), we try single spin flips on all positions in the activation map with value larger than 2.6
FIG. S4. (a) 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 $-\pi, 0, 4\pi/5$ are displayed. (b) An adversarial example obtained by FGM, which only differs with the sample in (a) by a tiny perturbation and has average fidelity 0.997. (c)(d)(e)(f) The activation maps (AM) of the sixth kernel in the first convolutional layer under different settings. (c) The average AM on all samples in the test set with $\chi^{(m)} = 0$. (d) The average AM on $\chi^{(m)} = 1$. (e) The AM obtained by taking the legitimate sample (a) as the input to the original classifier and (f) is taking the adversarial example (b) as input. (g)(h) are similar to (e)(f) but change the original classifier into the classifier after adversarial training.

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. S5(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 $\rho$ as $[31]$:

$$\rho = \frac{1}{3} (I + \sqrt{3}b \cdot \vec{\lambda}), \quad (S10)$$

where $\vec{\lambda}$ is a vector consists of the eight Gell-Mann matrices, $I$ and list the result in Table S1. We can find that single spin flips which make their signs change to the opposite of values in the AM can make the classifier incorrectly identifies the example into paramagnetic phase (i.e., $P'(\text{ferro}) < 0.5$). This result, combined with the highly non-uniform activation map, show that the classifier’s prediction relies heavily on only several particular spins. This does not agree with the physically defined order parameter $M = \frac{\sum_{\chi} \sigma_{\chi}}{N}$ where each spin contributes equally, indicating the model does not fully capture the underlying physical principles.
is the three-by-three identity matrix, and \( b = (b_1, b_2, \cdots, b_8) \) with \( b_i = \frac{1}{2} \sqrt{3 \text{tr} (\rho_A)} \). 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. S5(b). The performance of the training process is shown in Fig. S5(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 (\( \approx 0.05\)). 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 \( \| \delta \|_\infty \leq 0.2 \).

In order to obtain Fig. 3(b) of the main text, we randomly choose a sample from the category with topological invariant \( \chi^{(m)} = 1 \). 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 \( \chi^{(m)} = 0 \) after about three iterations. The confidence probability for the misclassification approaches 98\% after four iterations and begins to saturate at this value. The original legitimate sample is shown in Fig. S4(a) and Fig. S4(b) plots its corresponding adversarial example obtained by MIM. As discussed in Eq. (S10), each density matrix is represented by a vector \( b \) of length eight. For ease of visualization, in Fig. S4(a-b) we plot only the first component of \( b \), namely \( b_1 \), for each momentum point. We mention that one can also use the experimental data obtained recently in Ref. [31] with a solid-state simulator to generate adversarial examples. This is also observed in our numerical simulations.

To demonstrate why this adversarial example can mislead the powerful topological phase classifier, we study the activation map of each kernel in the first convolutional layer. We find that the sixth kernel has totally different activation patterns on topologically trivial and nontrivial phases, implying the classifier uses this kernel as a strong indicator for different phases. To verify this statement in depth, we calculate the correlations between the sixth kernel’s activation maps on \( \chi^{(m)} = 0,1,-2 \) phases. The metric we use is Normalized Cross Correlation (NCC) \( \rho \), which is expressed as the following:

\[
\rho(I_1, I_2) = \frac{\sigma(I_1, I_2)}{\sqrt{D(I_1) \cdot D(I_2)}}
\]

where \( I_1, I_2 \) are two activation maps, \( D(\cdot) \) is the variance and \( \sigma(\cdot, \cdot) \) is the covariance. It is easy to verify that \( |\rho| \leq 1 \) and larger \( \rho \) indicates stronger correlation. The result is shown in Table S2. We can find that the activation map of topologically nontrivial phases (\( \chi^{(m)} = 1, -2 \)) are highly correlated (\( \rho = 0.971 \)), but have little correlation with topologically trivial phase (\( \rho = 0.110, 0.244 \) respectively).

Then the functionality of adversarial example can be explained by the increase of correlation with the activation maps of incorrect phases. In Fig. 2(c) of the main text we show one layer of the activation map of the sixth kernel, we now show the layer with \( k_z = -\pi, 0, 4\pi/5 \) in Fig. S4(c-f), corresponding to the layers in Fig. S4(a-b). We can clearly see that the activation map of adversarial example (\( I_{adv} \)) becomes more correlated with average activation map of \( \chi = 1 (I_{adv}) \). This can also be verified from Table S2, the NCC with \( I_{adv} \) changes from 0.235 to 0.689. This means that the sixth kernel’s behaviour on the adversarial example is more similar as the \( \chi = 1 \) phase’s, which makes the classifier do the incorrect prediction.

### III. 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 [48]. 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(a-b) plots the results of the adver-
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.

In Fig. 2(d) of the main text we present that the activation map of the Ising model classifier becomes much flatter after the adversarial training, which accords with the symmetry of calculating $M$. This results in the effectiveness of the adversarial training on identifying the phase transition point: In Fig. 4(c-d) in the main text we demonstrate the Ising model classifier’s prediction curves on different $M$ and different spin configurations before and after adversarial training. Since we only consider the $30 \times 30$ lattice, we divide $M$ from 0 to 1 into 450 intervals and in each interval we randomly generate 100 samples. We generate samples according to $M$ instead of $T$, which is in order to elaborate the classifier’s behaviors around the small range of the transition point. From the Fig. 4(c-d) in the main text we can see that the classifier presents similar predictions on samples with the same $M$, and the confidence of being the ferromagnetic phase goes down as $M$ decreases. This indicates that the classifier has learnt the correct classification rule according to the order parameter $M$.

In Fig. S6 we show the relation between the confidence of being the ferromagnetic phase and $M$ by averaging 100 different spin configurations on each $M$ interval. We can clearly see that after adversarial training, the prediction curve becomes sharper and smoother, and the transition point identified at the confidence threshold $P_{\text{ferro}} = 0.5$ moves closer to the theoretical critical temperature $T_c$. This implies that the classifier can identify the phase transition point more precisely after adversarial training.

Besides the Ising model, We show that adversarial training can also help the topological phase classifier to make better inference. From table S2, we find that the activation map of the sixth kernel on topologically trivial phase ($I_{\chi=0}$) becomes less correlated with topologically nontrivial ones ($I_{\chi=1,-2}$). In addition, the activation maps between $I_{\chi=1}$ and $I_{\chi=-2}$ also become less correlated. This indicates that the kernel begins to learn how to distinguish two topologically nontrivial phases. In Fig. S4(g-h), we plot the sixth kernel’s activation map on the legitimate and adversarial examples after the adversarial training. We can find that $I_{\text{adv}}$ is much less correlated with $I_{\chi=1}$, which can also be verified by table S2. All these results imply that the classifier learned a more robust convolutional kernel, which can better distinguish three different topological phases, to do inferring.

The effectiveness of the adversarial training can be also derived from another view: although one cannot expect a universal defense strategy that is able to make the phase classifiers robust to all types of adversarial perturbations, the defense against certain adversarial attack can help the model better learn the physical principle behind the problem. To support our claim, we try the discrete attack on the Ising model classifier after the adversarial training against continuous attack (FGSM in this example). The result is shown in Fig. S7. We generate adversarial examples based on the test set with the Monte Carlo method on 41 different temperatures. Each temperature has 250 samples. Under each temperature, we enumerate one pixel and two pixels’ flip on all 250 samples and find the ratio of samples that can make the model do misclassification. We find that after adversarial training, the refined classifier is more robust to discrete spins flip, and the peaks move closer to the phase transition point.
FIG. S7. One pixel and two pixels attacks’ results on the original classifier and refined classifier after adversarial training against FGSM attacks. (a)(b) show the results of the one pixel attack. It is clear that the model is more vulnerable near critical temperatures $T_c$. After adversarial training, adversarial samples ratio decreases a lot and the peak moves to $T_c$ exactly, which means the model becomes more robust. (c)(d) show the results of the two pixels attack and a similar conclusion can be drawn.