Self-Learning Phase Boundaries by Active Contours

Ye-Hua Liu
*Institute for Theoretical Physics, ETH Zurich, 8093 Zurich, Switzerland**

Evert P.L. van Nieuwenburg
Institute for Quantum Information and Matter, Caltech, Pasadena, 91125 CA, USA†

We introduce a fully automatic self-learning scheme for detecting phase boundaries. This method extends the previously introduced confusion scheme for learning phase transitions, by using a cooperative network that learns to optimize the guess for the transition point. The networks together are capable of finding transition points for fully unlabeled data. This improvement allows us to efficiently study 1D and 2D parameter spaces, where for the latter we utilize an active contour model – the snake – from computer vision as a representation of the learned phase boundary. The snakes, equipped with neural networks, can learn while they move in the parameter space and thereby detect phase boundaries automatically.

Machine learning has emerged as a promising tool for condensed matter physics. Currently its wide range of applications include the detection of classical and quantum phase transitions [1–10], providing a high-quality ansatz for many-body wave functions [11–16], accelerating dynamical mean-field [17, 18] and Monte Carlo [19–26] calculations, predicting materials [27–29], and inspiring new understanding of fundamental physics, e.g. the renormalization group [30–34].

Similarly to the classification problem in machine learning, detecting phase transitions in physics is commonly approached from two angles. In the *supervised* approach, physics knowledge is used to provide answers in limiting cases (e.g. the limits of small and large magnetic fields), and the machine learner is asked to extrapolate to the transition point [1] based on an internal model that it extracted from the known limits. In the *unsupervised* approach, no such knowledge is assumed and the transition is sought by other means such as principal component analysis [2] or variational auto-encoders [7]. Both approaches have their advantages and shortcomings.

The confusion scheme proposed previously by us is a *hybrid* method [5], where no knowledge of the limiting cases is needed but the learning is still carried out in a supervised manner. Specifically, one first guesses a transition point and tries to train the machine with this guess. When the guess is correct, and hence the labels most consistent with the data, the machine learner achieves the highest performance. In this way we gain the ability to find transitions at the cost of having to repeat the training process for many guesses, which is computationally expensive.

In this work, we extend the confusion scheme by training a "guesser" together with the "learner". This leads to a fully automated scheme, where the guesser and learner cooperate and together constitute a *self-learning* method. In addition, phase transitions in 2D parameter spaces share many common aspects with computer vision, especially that of image-feature detection. The difference is that in actual images at every point (pixel) the data is represented by the color (e.g. through 4 numbers representing the red, green, blue and alpha channels), whereas in physics the data at every point in the diagram can be arbitrary measurements whose features might not be apparent to the human eyes. This inspires us to use an active contour method [35], combined with the self-learning scheme, to perform automated searching of phase boundaries in 2D phase diagrams.

METHODS

Problem setup. We consider data that can be ordered as a function of a parameter x. This is typically the case in physics and experiments, where x is the tuning parameter (e.g. magnetic field or temperature). At various values of x the data is described by $\mathbf{d}(x)$, and can be thought of as a vector of measurements at x. We describe a neural network on an abstract level as a map $\mathcal{N}$ that takes data $\mathbf{d}(x)$ and infers the probability distribution $\mathcal{N}(\mathbf{d}(x))$, representing the probabilities of $\mathbf{d}(x)$ belonging to each possible class (label). For phase-transition problems, the correct distribution varies with x discontinuously. At each x only a single probability (corresponding to the right class) is equal to unity, and the rest are zero due to normalization. Since we assume ordered data, this can be simplified by considering the probability distribution $\mathbf{L}(x)$ of labels directly on x, which for a transition point g between two phases 0 and 1 has two components $L_0(x)$ and $L_1(x)$ given by $L_0 = \Theta(g-x)$ and $L_1(x) = \Theta(x-g)$, where Θ is the Heaviside step-function.

For supervised learning, a large body of pairs $\mathbf{d}(x)$ and the corresponding correct answer $\mathbf{L}(x)$ have to be known beforehand, and the neural network $\mathcal{N}$ is trained with the goal $\mathcal{N}(\mathbf{d}(x)) \to \mathbf{L}(x)$. To achieve this goal, the set of parameters $\mathbf{W}_\mathcal{N}$ that characterize the neural network are adjusted during the training process to minimize a cost function $C[\mathcal{N}(\mathbf{d}(x)), \mathbf{L}(x)]$, quantifying the mismatch between the network's prediction and the known answer. The cost function depends implicitly

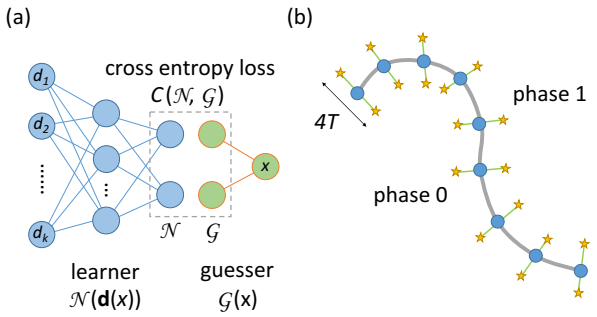

FIG. 1. (a) Schematics of the self-learning scheme for learning phase transitions sorely from the dataset $\{(x, \mathbf{d}(x))\}$, where x is the tuning parameter and $\mathbf{d}(x)$ is a vector of measurements at x. (b) Schematics of the self-learning snake. The blue circles are the snake nodes, the green lines denote normal directions at the nodes. Samples (stars) are generated in the normal direction at each node and are assigned a label according to its distance to the snake. Snakes could be open or closed, and can move to the correct phase boundary (gray line) automatically. The open snake in this figure has 9 nodes and during motion generates batches of training data with batch size $N_b = 18$ (N_b is much larger in real simulations).

on the parameters $\mathbf{W}_\mathcal{N}$ through $\mathcal{N}$, and hence can be minimized using standard gradient descent methods.

Confusion scheme. The confusion scheme was invented to work for cases in which *none* of the correct labels $\mathbf{L}$ are known. It seems that in such a case supervised training is impossible. However, if the underlying data $\mathbf{d}(x)$ can be classified into distinct classes (as is the assumption), it is intuitively clear that a consistent labelling *does* exist. Essentially by guessing the transition point g in $\mathbf{L}(x)$, and by monitoring the number of correctly classified points according to this guess, the correct value for g can be deduced. In other words, the correct value for g is that for which the assigned labels are the most consistent, and the network is *least confused* after its training.

Self-learning scheme. In the previous proposal, we search for the optimal g by a brute-force scan of the parameter space. For phase-transition problems with higher-dimensional parameter spaces, this approach is very inefficient. In this paper we introduce the guesser network $\mathcal{G}$. It performs the map $x \to \mathcal{G}(x)$, representing the the probabilities of x belonging to each possible phase. That is, the guesser has the goal of learning $\mathcal{G}(x) \to \mathbf{L}(x)$ (see Fig. 1a). The guesser is itself characterized by a set of parameters $\mathbf{W}_\mathcal{G}$ on which we wish to perform gradient descent. The overall cost function of the learner $\mathcal{N}$ and guesser $\mathcal{G}$ is now $C\left[\mathcal{N}(\mathbf{d}(x)), \mathcal{G}(x)\right]$. In this way, we have promoted the human input $\mathbf{L}$ to be an active agent $\mathcal{G}$. At every step of training, the learner $\mathcal{N}$ tries to learn the data according to the suggested labels $\mathbf{L}(x)$ obtained from the guesser. Based on the learner's performance, the guesser is then updated to provide a better set of labels, and so on.

We first assume 1D parameter space with two phases. In this case we propose a logistic-regression guesser network with one/two input/output neuron(s): $\mathcal{G}_{0,1}(x) = \sigma[s_{0,1}(x-g)/T]$, where $\sigma(x) = 1/(1+e^{-x})$ is the logistic (sigmoid) function, 0/1 denotes the first/second output neuron, and $s_{0,1} = -, +$. The guesser is hence characterized by two parameters g and T, setting respectively the guessed transition point and the sharpness of the transition. Gradient descent can be performed on g and optionally on T. In this paper we use the cross entropy cost function $C(\mathcal{N}, \mathcal{G}) = -\log\mathcal{N}\cdot\mathcal{G} - \log(1-\mathcal{N})\cdot(1-\mathcal{G})$, which is suitable for classification problems. The basic derivatives, needed in the chain rule for the final gradient on the guesser network, then read:

$$\begin{aligned}\frac{\partial C}{\partial \mathcal{G}} &= -\log\mathcal{N} + \log(1-\mathcal{N}),\\ \frac{\partial \mathcal{G}_{0,1}}{\partial g} &= -\frac{s_{0,1}}{4T\cosh^2\left[(x-g)/2T\right]},\\ \frac{\partial \mathcal{G}}{\partial T} &= \frac{x-g}{T}\frac{\partial \mathcal{G}}{\partial g},\end{aligned} \tag{1}$$

which fully determine the dynamics of the guesser: $\dot{g} = -\alpha_g \partial C/\partial g$ and $\dot{T} = -\alpha_T \partial C/\partial T$, where α_g and α_T are the learning rates for the two parameters, respectively. The dynamics of the learner follows $\dot{\mathbf{W}}_\mathcal{N} = -\alpha_\mathcal{N}\partial C/\partial \mathbf{W}_\mathcal{N}$ with another independent learning rate $\alpha_\mathcal{N}$, here the gradient is implemented by the back-propagation algorithm [36].

At this point, we could conceptually regard the guesser and learner together as one compound agent, capable of *self-learning*.

Self-learning snake. The self-learning scheme for phase transitions allows us to move to higher-dimensional parameter spaces, because now there is no need to train neural networks repetitively for every possible guess.

In the 2D case, the parametrization of the guess function is not straightforward. Inspired by the computer vision techniques for finding image features, we use an active contour model – the snake [35] – for this parametrization. A snake is a discretized curve of linked nodes (see Fig. 1b), and each node is attracted to "image forces". The nodes move under these forces (hence snake) to locate important features of the image. We emphasize here that in our case the "image forces" will be derived from physical data, not the usual colors.

Besides the image forces, there are also internal forces on the snake preventing large stretching and bending. The internal forces are derived from the energy functional $E_\text{internal} = \int_0^1 ds\,\left(\alpha|\partial_s \mathbf{v}|^2 + \beta|\partial_{ss}\mathbf{v}|^2\right)$, where $\mathbf{v}$ is the curve parametrized by $s \in [0,1)$ (closed snake) or $s \in [0,1]$ (open snake). Increasing α makes for a

more elastic snake by preventing stretching and β a more solid snake by preventing bending. The internal forces $-\nabla_{\mathbf{v}} E_{\text{internal}}$ are discretized by finite differences, which involve nearest neighbors for second-order derivatives, and next-nearest neighbors for fourth-order derivatives. Updating the snake is performed by a semi-implicit method [35]. To regularize the density of snake nodes, at each time step of motion, we move the snake nodes such that successive nodes have equal distance, all the while keeping the overall shape of the snake.

In computer vision, the external image force is local, e.g. local color intensity for lines and local color gradient for edges. In our case, the external force experienced by the snake will be derived from the machine-learning gradient Eq. (1). However the self-learning scheme depends on accessing data from both sides of the guessed transition surface (current location of the snake), at the same time. For this reason, we introduce a width of the snake (generically different at each node), denoted by the same parameter T as in the 1D case. The snake will generate (sense) samples in its vicinity within this length scale as shown in Fig. 1b. Specifically the sample points are drawn at each node perpendicularly to the snake, with distances uniformly picked in $[-2T, 2T]$. The guesser function is the same as in the 1D case, evaluated by each node in its perpendicular direction. This probing of the data within a window, and asking whether or not two distinct phases can be detected within it, is a very powerful concept that is also successfully used in Ref. 10 to scan phase diagrams.

RESULTS

Ising model. We first test our scheme on a 1D parameter space by studying the thermal phase transition of the 2D Ising model:

$$H = -J \sum_{\langle i,j \rangle} \sigma_i \sigma_j, \tag{2}$$

where the tuning parameter is the temperature $1/\beta$.

The training data are spin configurations drawn from a Monte Carlo simulation on an L by L square lattice. We select 100 temperatures uniformly from $0.1J$ to $5J$ and prepare 100 samples at each temperature. Every batch consists of $N_b = 100$ random samples, one from each temperature. The time is measured by the number of batches passed in the training process.

As shown in Fig. 2, during training, the guesser moves toward the correct transition point $1/\beta_c \sim 2.27J$, and decreases the width T such that the learning is sharper and sharper. We found for this method to work, we have to preprocess each configuration by flipping all its spins when the net magnetization $\sum_i \sigma_i$ is negative, this corresponds to a manual removal of the Z_2 symmetry.

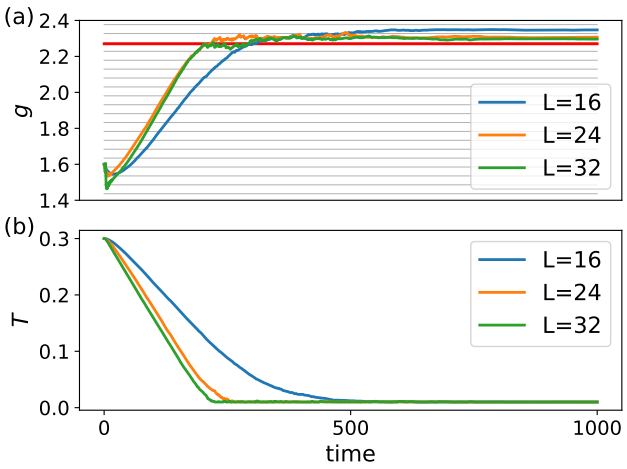

FIG. 2. Self-learning the Ising transition. (a) Starting from a lower guess of the transition point, the gradient on the guesser pushes it to move up. The red line marks the exact transition point and the gray lines the temperatures (in the range of the figure) for generating Monte Carlo samples. (b) During training, the width T decreases, meaning the combined self-learner could distinguish the two phases sharper. Training on samples from larger lattices is faster and more accurate. Parameters: batch size $N_b = 100$, number of input neurons L^2, number of hidden neurons L^2, number of output neurons 2; initial learning rates $\alpha_{\mathcal{N}} = 0.1, \alpha_g = 0.025, \alpha_T = 0.003$, decay rate 0.995; dropout keep probability 0.8, $L_2 = 0.0001$. We have set a lower bound for the width $T > 0.01$.

Bose-Hubbard model. As a first example for self-learning phase transitions in 2D parameter spaces, we choose the Bose-Hubbard model:

$$H = -J \sum_{\langle i,j \rangle} (b_i^\dagger b_j + b_j^\dagger b_i) + \sum_i \left[\frac{U n_i (n_i - 1)}{2} - \mu n_i \right]. \tag{3}$$

Regarding U as the energy unit, for each chemical potential μ, the model has a quantum phase transition from Mott insulating state to superfluid state when the hopping J is increased. A useful indicator of this phase transition, unknown to the initial snake, is the average hopping $\langle K \rangle$ where $K = \sum_{\langle i,j \rangle} (b_i^\dagger b_j + b_j^\dagger b_i)$. The critical point J_c reaches local maxima when the system is at commensurate fillings, corresponding to half-integers μ/U. A phase diagram of this system results in the series of well-known Mott-lobes.

We use the mean-field theory developed in Ref. 37 to generate vector data $\mathbf{F}(J, \mu)$, where F_n with $n = 0, 1, \ldots, n_{\max}$ denotes the amplitude for having n bosons per site. We choose a cutoff of $n_{\max} = 79$.

We target the third Mott lobe between $\mu/U = 2, 3$, and the snake successfully captures the phase boundary as shown in Fig. 3. In this case the phase boundary touches the boundary of the parameter space, so we

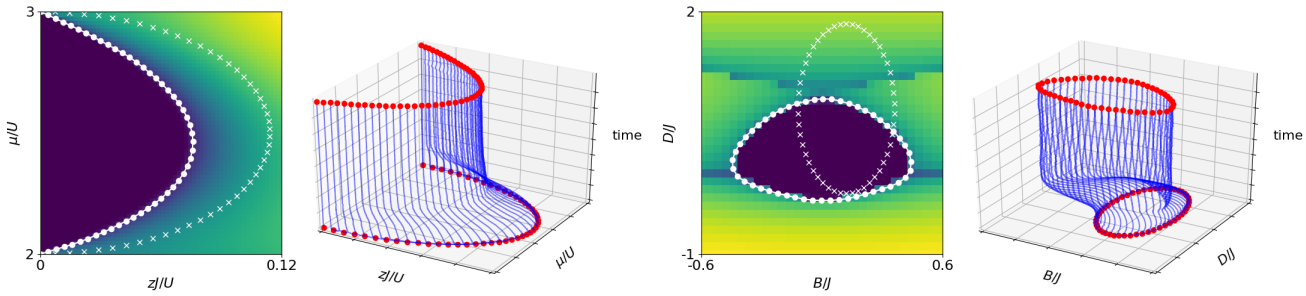

FIG. 3. Self-learning the Mott insulator to superfluid transition in the Bose Hubbard model (left panels), and the topological transition in the spin-1 antiferromagnetic Heisenberg chain with anisotropy and magnetic field (right panels). Color plots (purple to yellow goes from zero to nonzero values) show the average hopping for the Bose Hubbard model, and the difference between the largest two eigenvalues of the reduced density matrix for half of the Heisenberg chain. For the Bose Hubbard model we create a large open snake with head and tail fixed at integer chemical potentials $\mu/U = 2, 3$. In this case the snake shrinks and stops at the correct phase boundary. For the Heisenberg model, we create a large and offset closed snake, who moves, rotates, shrinks, and finally stays at the Haldane pocket. Parameters for snakes: number of nodes 50; dynamic width at each node is initialized to $T = 0.06$ (normalized by the ranges of parameters) and clipped to $T \in [0.03, 0.08]$ during motion; regularizations $\alpha = 0.002$, $\beta = 0.4$, $\gamma = 0.25$ (see Ref. 35 for details); batch size $N_b = 1500$. Parameters for the fully connected neural networks: number of input neurons 80, number of hidden neurons 80, number of output neurons 2; initial learning rates $\alpha_{\mathcal{N}} = 0.01$, $\alpha_g = 0.0008$, $\alpha_T = 0.0002$, decay rate 0.9999; dropout keep probability 0.8, $L_2 = 0.0001$.

use open snakes with fixed head and tail at known transition points. The snake's motion is then restricted to shrinking or expanding. It is important to emphasize here that we have used knowledge of only two points in this phase diagram along the $J = 0$ axis, and that the underlying data is not the average hopping as shown in the background, but the vector data $\mathbf{F}(J, \mu)$ mentioned above. Additionally, we have tested that the snake is capable of finding the lobe from an initially circular (periodic) configuration (cf. the next example on the Heisenberg model).

Heisenberg model. We now move to a quantum phase transition beyond mean-field theory. We choose the spin-1 antiferromagnetic Heisenberg chain with anisotropy and transverse magnetic field:

$$H = J \sum_i \mathbf{S}_i \cdot \mathbf{S}_{i+1} + \sum_i \left[D(S_i^z)^2 - BS_i^x \right]. \qquad (4)$$

In the 2D parameter space B/J and D/J, this model has a pocket named the Haldane phase – a topologically non-trivial phase – around zero magnetic field and anisotropy [38]. This transition can be detected by a change in the degeneracy structure of the entanglement spectrum, but the initial snake is unaware of this.

For the training data, we simulate an infinite chain with translational invariance using iTEBD [39] and record the eigenvalues of the reduced density matrix up to bond dimension $m = 80$ for the calculation. We also include measurements of the (staggered) magnetization at every point in the phase diagram.

The result of learning with the entanglement spectrum data is shown in Fig. 3. In this model the phase boundary is closed and located near the center of the parameter space. For this reason we use a closed (periodic) snake (such a snake is a good initial choice in all cases). Compared with the previous example, the snake's motion now also contains translation and rotation.

If the snake is fed only magnetization data, the resulting phase boundary depends on the initial position of the snake. In particular, if the snake is initialized near the central Haldane pocket, the snake neatly wraps around it as before. This is most likely due to the snake distinguishing the Haldane phase from other phases (i.e. those with an antiferromagnetic order parameter). We remark that the initial configuration of the snake needs to overlap with (at least) two different phases, so that it is able to probe a gradient. This is the largest drawback of snakes (as it is also for their use in computer vision), but can be overcome relatively easily by scaling/moving the snake.

DISCUSSION

In this paper we improved the previously introduced confusion scheme to a level that allows it to self-consistently find transition points. By training two networks (the guesser and learner) together, we have constructed an effective self-learner capable of detecting phase transitions automatically. This increase in efficiency allows us to also explore parameter spaces beyond 1D, where we utilized the snake model from computer vision to extract phase boundaries. Our method is in spirit similar to the actor-critic scheme for reinforcement learning [40] and the adversarial training scheme for generative models [41]. Although we have not presented

such results, it is entirely feasible to keep the guesser and learner networks separate and describe them with their own cost functions, rather than using the combined cost function and resulting gradients in Eqs. (1).

The major limitation for the snake is the need for an initial state that has overlap with the desired features to be detected. This was true for their use in computer vision, and also applies to our self-learning snake. The network governing the snake seems to empirically always pick out the *most obvious* features, and hence needs a most obvious feature to start out with. In applications to phase diagrams, we have the clear advantage of some known extreme limits at which we can fix the snake.

Ch'ng et al. have proposed to train neural networks deeply inside the known phases and then use the trained neural networks to extrapolate the whole phase diagram [3]. Such a method is, compared to our method, simpler and faster. However, as discussed in Ref. 3, the data for supervised training have to be carefully chosen, otherwise interpolation of the phase boundary could be qualitatively incorrect (i.e. the interpolated and/or extrapolated phase boundaries may depend sensitively on the data included in the training). On the contrary, since our model always learns its surroundings, it can actively explore much larger area in the parameter space. For general phase transition problems, one could use both methods complementarily.

The authors thank Lei Wang, S. Huber and S. Trebst for reading the manuscript and helpful suggestions. Y.-H.L. is supported by ERC Advanced Grant SIMCOFE. E.v.N. gratefully acknowledges financial support from the Swiss National Science Foundation (SNSF) through grant P2EZP2-172185. The authors used the `TensorFlow` [42] package for machine learning.

* liuye@phys.ethz.ch
† evert.v.nieuwenburg@gmail.com

[1] Juan Carrasquilla and Roger G Melko, "Machine learning phases of matter," Nat. Phys. **13**, 431–434 (2017), arXiv:1605.01735.
[2] Lei Wang, "Discovering phase transitions with unsupervised learning," Phys. Rev. B **94**, 195105 (2016), arXiv:1606.00318.
[3] Kelvin Ch'ng, Juan Carrasquilla, Roger G. Melko, and Ehsan Khatami, "Machine Learning Phases of Strongly Correlated Fermions," arXiv:1609.02552.
[4] Peter Broecker, Juan Carrasquilla, Roger G. Melko, and Simon Trebst, "Machine learning quantum phases of matter beyond the fermion sign problem," arXiv:1608.07848.
[5] Evert P. L. van Nieuwenburg, Ye-Hua Liu, and Sebastian D. Huber, "Learning phase transitions by confusion," Nat. Phys. **13**, 435–439 (2017), arXiv:1610.02048.
[6] Frank Schindler, Nicolas Regnault, and Titus Neupert, "Probing many-body localization with neural networks," arXiv:1704.01578.
[7] Sebastian Johann Wetzel and Manuel Scherzer, "Machine Learning of Explicit Order Parameters: From the Ising Model to SU(2) Lattice Gauge Theory," arXiv:1705.05582.
[8] Sebastian Johann Wetzel, "Unsupervised learning of phase transitions: from principal component analysis to variational autoencoders," arXiv:1703.02435.
[9] Tomoki Ohtsuki and Tomi Ohtsuki, "Deep Learning the Quantum Phase Transitions in Random Two-Dimensional Electron Systems," J. Phys. Soc. Japan **85**, 123706 (2016), arXiv:1610.00462.
[10] P. Broecker, F. Assaad, and S. Trebst, in preparation (2017).
[11] Giuseppe Carleo and Matthias Troyer, "Solving the quantum many-body problem with artificial neural networks," Science **355**, 602–606 (2017), arXiv:1606.02318.
[12] Dong-Ling Deng, Xiaopeng Li, and S. Das Sarma, "Exact Machine Learning Topological States," arXiv:1609.09060.
[13] Jing Chen, Song Cheng, Haidong Xie, Lei Wang, and Tao Xiang, "On the Equivalence of Restricted Boltzmann Machines and Tensor Network States," arXiv:1701.04831.
[14] Xun Gao and Lu-Ming Duan, "Efficient Representation of Quantum Many-body States with Deep Neural Networks," arXiv:1701.05039.
[15] Giacomo Torlai, Guglielmo Mazzola, Juan Carrasquilla, Matthias Troyer, Roger Melko, and Giuseppe Carleo, "Many-body quantum state tomography with neural networks," arXiv:1703.05334.
[16] Zi Cai, "Approximating quantum many-body wave-functions using artificial neural networks," arXiv:1704.05148.
[17] Louis-François Arsenault, Alejandro Lopez-Bezanilla, O. Anatole von Lilienfeld, and Andrew J. Millis, "Machine learning for many-body physics: The case of the Anderson impurity model," Phys. Rev. B **90**, 155136 (2014), arXiv:1408.1143.
[18] Louis-François Arsenault, O Anatole von Lilienfeld, and Andrew J Millis, "Machine learning for many-body physics: efficient solution of dynamical mean-field theory," arXiv:1506.08858.
[19] Lei Wang, "Can Boltzmann Machines Discover Cluster Updates?" arXiv:1702.08586.
[20] Li Huang, Yi-Feng Yang, and Lei Wang, "Recommender engine for continuous-time quantum Monte Carlo methods," Phys. Rev. E **95**, 031301 (2017), arXiv:1612.01871.
[21] Li Huang and Lei Wang, "Accelerated Monte Carlo simulations with restricted Boltzmann machines," Phys. Rev. B **95**, 035105 (2017), arXiv:1610.02746.
[22] Hiroyuki Fujita, Yuya. O. Nakagawa, Sho Sugiura, and Masaki Oshikawa, "Construction of Hamiltonians by machine learning of energy and entanglement spectra," arXiv:1705.05372.
[23] Junwei Liu, Yang Qi, Zi Yang Meng, and Liang Fu, "Self-learning Monte Carlo method," Phys. Rev. B **95**, 041101 (2017), arXiv:1610.03137.
[24] Junwei Liu, Huitao Shen, Yang Qi, Zi Yang Meng, and Liang Fu, "Self-Learning Monte Carlo Method in Fermion Systems," arXiv:1611.09364.
[25] Xiao Yan Xu, Yang Qi, Junwei Liu, Liang Fu, and Zi Yang Meng, "Self-Learning Determinantal Quantum Monte Carlo Method," arXiv:1612.03804.
[26] Yuki Nagai, Huitao Shen, Yang Qi, Junwei Liu,

and Liang Fu, "Self-Learning Monte Carlo Method: Continuous-Time Algorithm," arXiv:1705.06724.
[27] Felix A. Faber, Alexander Lindmaa, O. Anatole von Lilienfeld, and Rickard Armiento, "Machine Learning Energies of 2 Million Elpasolite (ABC_2D_6) Crystals," Phys. Rev. Lett. **117**, 135502 (2016), arXiv:1508.05315.
[28] Ghanshyam Pilania, Chenchen Wang, Xun Jiang, Sanguthevar Rajasekaran, and Ramamurthy Ramprasad, "Accelerating materials property predictions using machine learning," Sci. Rep. **3**, 2810 (2013).
[29] Albert P. Bartok, Sandip De, Carl Poelking, Noam Bernstein, James Kermode, Gabor Csanyi, and Michele Ceriotti, "Machine Learning Unifies the Modelling of Materials and Molecules," arXiv:1706.00179.
[30] Olivier Landon-Cardinal and David Poulin, "Practical learning method for multi-scale entangled states," New J. Phys. **14**, 085004 (2012), arXiv:1204.0792.
[31] Cédric Bény, "Deep learning and the renormalization group," arXiv:1301.3124.
[32] Pankaj Mehta and David J. Schwab, "An exact mapping between the Variational Renormalization Group and Deep Learning," arXiv:1410.3831.
[33] Henry W. Lin, Max Tegmark, and David Rolnick, "Why does deep and cheap learning work so well?" arXiv:1608.08225.
[34] Maciej Koch-Janusz and Zohar Ringel, "Mutual Information, Neural Networks and the Renormalization Group," arXiv:1704.06279.
[35] Michael Kass, Andrew Witkin, and Demetri Terzopoulos, "Snakes: Active contour models," Int. J. Comput. Vis. **1**, 321–331 (1988).
[36] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams, "Learning representations by back-propagating errors," Nature **323**, 533–536 (1986).
[37] Werner Krauth, Michel Caffarel, and Jean-Philippe Bouchaud, "Gutzwiller wave function for a model of strongly interacting bosons," Phys. Rev. B **45**, 3137–3140 (1992).
[38] Frank Pollmann, Ari M. Turner, Erez Berg, and Masaki Oshikawa, "Entanglement spectrum of a topological phase in one dimension," Phys. Rev. B **81**, 064439 (2010), arXiv:0910.1811.
[39] G. Vidal, "Classical Simulation of Infinite-Size Quantum Lattice Systems in One Spatial Dimension," Phys. Rev. Lett. **98**, 070201 (2007).
[40] Vijay R Konda and John N Tsitsiklis, "Actor-Critic Algorithms," SIAM J. Control Optim. **42**, 1143–1166 (2003).
[41] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, "Generative Adversarial Networks," arXiv:1406.2661.
[42] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," arXiv:1603.04467.