Tutorial

How To Use Neural Networks To Investigate Quantum Many-Body Physics

Juan Carrasquilla[1] and Giacomo Torlai[2,3,*]

[1] *Vector Institute, MaRS Centre, Toronto, Ontario M5G 1M1, Canada*
[2] *AWS Center for Quantum Computing, Pasadena, California 91125, USA*
[3] *Center for Computational Quantum Physics, Flatiron Institute, New York, New York 10010, USA*

(Received 26 January 2021; published 12 November 2021)

Over the past few years, machine learning has emerged as a powerful computational tool to tackle complex problems in a broad range of scientific disciplines. In particular, artificial neural networks have been successfully used to mitigate the exponential complexity often encountered in quantum many-body physics, the study of properties of quantum systems built from a large number of interacting particles. In this article, we review some applications of neural networks in condensed matter physics and quantum information, with particular emphasis on hands-on tutorials serving as a quick start for a newcomer to the field. The prerequisites of this tutorial are basic probability theory and calculus, linear algebra, basic notions of neural networks, statistical physics, and quantum mechanics. The reader is introduced to supervised machine learning with convolutional neural networks to learn a phase transition, unsupervised learning with restricted Boltzmann machines to perform quantum tomography, and the variational Monte Carlo method with recurrent neural networks for approximating the ground state of a many-body Hamiltonian. For each algorithm, we briefly review the key ingredients and their corresponding neural-network implementation, and show numerical experiments for a system of interacting Rydberg atoms in two dimensions.

DOI: 10.1103/PRXQuantum.2.040201

CONTENTS

[*] gttorlai@amazon.com; This work was done before Giacomo Torlai joined Amazon.

I. INTRODUCTION

"Quantum many-body physics" refers to the mathematical framework to study the collective behavior of large numbers of interacting particles. The emerging cooperative phenomena that result from seemingly simple interactions can produce an astounding variety of phases of matter, such as conventional metals and magnetically ordered states, as well as unanticipated states, including high-temperature superconductivity, strange metals, and spin liquids [1]. In addition to naturally occurring quantum systems, many-body physics studies synthetic quantum matter (e.g., ultracold atoms, superconducting qubits, and trapped ions), which simultaneously reveals new phenomena in highly controlled laboratory settings and advances the development of quantum computers and other quantum information processing devices.

In spite of the simplicity of the physical laws that govern such multiparticle quantum objects, the theoretical and experimental analysis of these systems confront us with complexities that are ultimately rooted in the dimensionality explosion associated with the exponential scaling of the size of the space where quantum many-body states live. Traditionally, the study of many-body systems is performed with the help of tools designed to circumvent this complexity and to produce a succinct, low-dimensional description that captures the essential aspects of a quantum system. Such descriptions arise from the analysis of data generated in a wide range of theoretical, computational, and experimental devices. These include numerical simulations of model Hamiltonians based on quantum Monte Carlo methods [2–4] or variational algorithms [5–7], but also experimental arrays of complex electronic structure images obtained from spectroscopic imaging scanning tunneling microscopy, or measurements of quantum states prepared on a physical quantum computing platform [8–11].

Machine learning, already explored as a tool in several research areas in physics [12], offers a set of alternative approaches to the study of quantum many-body systems in experiments and numerical simulations [13,14]. The resurgence of activity at the intersection between physics and machine learning is in part due to a series of scientific breakthroughs in computer vision and natural language processing. Such progress has led to a burst of research where neural networks have been repurposed to tackle fundamental questions in condensed matter physics, quantum computing, statistical physics, and atomic, molecular, and optical physics. Machine learning, and in particular deep neural networks, has been used to identify phases of matter in numerical simulations and experiments [15–36], to increase the performance of Monte Carlo simulations [37–46], to accurately describe the state of classical [47] and quantum [48–63] systems, to develop novel quantum control strategies [64–70], to perform quantum tomography [71–90], to accelerate density functional theory calculations [91–97], to develop and elucidate renormalization group analyses [98–103], and to devise quantum error correction protocols [104–118], in addition to many other examples [119–130].

Such an explosion of activity indicates that machine learning techniques may soon become commonplace in quantum many-body physics research, both in experiments and in numerical simulations. These clear trends call for the development of resources to stimulate researchers to familiarize themselves with the wealth of concepts, intuition, algorithms, hardware, software, and research culture entailed by the adoption of machine learning and neural networks in physics research. Here we take a step forward in this direction and develop a set of hands-on tutorials focused on a set of recent prototypical examples of applications of neural-network technology to problems in statistical physics, condensed matter physics, and quantum computing.

A. Outline

This article is organized as follows. Starting with a preliminary discussion, we introduce in Sec. A some fundamental concepts in machine learning and neural networks. In Sec. B we present a concise description of the physical system studied in our numerical experiments, a two-dimensional (2D) array of interacting Rydberg atoms. In Sec. III we discuss our first application, the classification of phases of matter with supervised machine learning of projective measurement data using a convolutional neural network (CNN), and demonstrate it on the quantum phase transition in the Rydberg atoms. In Sec. IV we introduce quantum state tomography (QST), and show how this problem can be phrased as an unsupervised machine learning task. Using the restricted Boltzmann machine (RBM), we show quantum tomography of the Rydberg ground states, as well as of the ground state of a small molecule from qubit measurement data. In Sec. V we present the simulation of the ground state of a many-body Hamiltonian using the variational Monte Carlo (VMC) method with a recurrent-neural-network (RNN) wave function. For each of these applications, we also show the key components of the underlying software, with full code tutorials available in an external repository [131].

II. PRELIMINARIES

A. Machine learning with neural networks

Artificial intelligence is the scientific discipline that deals with the theory and development of computer programs with the ability to perform complex tasks that are usually performed by humans, such as visual perception, game playing, speech recognition, decision-making, and translation between languages. Before its current widespread adoption, artificial intelligence first saw early success solving problems that are relatively straightforward to formalize in an abstract way. The solutions to this breed of problems are typically described by a list of very precise formal rules that computers can process efficiently. As a remarkable example, computers have been beating humans at playing chess since 1997, in part because chess involves a large set of formal rules.

Modern machine learning, instead, deals with the challenge of automatizing the solution of real-world tasks that may be easy for humans to process but that are hard to formally describe by simple rules. These techniques have spurred a recent revolution where algorithms trained using data have started to match the ability of humans to recognize objects in an image, decipher speech, or translate text into multiple languages, which are tasks that are difficult to formalize and articulate through simple rules.

A key element behind these recent developments can be largely traced back to a series of breakthroughs in the development of powerful neural-network models, where data are processed through the sequential combination of multiple nonlinear layers [132]. Such models solve a fundamental problem in learning real-world tasks—namely, the problem of automatically extracting knowledge from raw noisy data rather than relying on hard-coded knowledge directly inscribed in the algorithms by a human. Neural networks automatize the construction of sets of increasingly complex representations of the data, which can be understood as the computational disentangling of complex concepts (e.g., an object in a cluttered image) from simpler concepts (e.g., pixel values and basic shapes such as edges). These representations, in turn, lead to solutions to learning tasks with unprecedented success.

For practical purposes, machine learning algorithms can be divided into the categories of supervised, unsupervised, and reinforcement learning, all of which have found applications to quantum many-body systems [13]. While there is no formal difference between some of the algorithms in these categories when expressed in the language of probability [132,133], such a division is often used as a way to specify the details of the algorithms, the training setup, and the structure of the datasets involved.

Supervised learning tasks aim at predicting a target output vector $\boldsymbol{y}$ associated with input vector $\boldsymbol{x}$, both of which can be discrete or continuous. The training data are thus a list of pairs of input-output tuples $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{M}$, where target output conveys that such a vector corresponds to the ideal output given the input vector [133]. Starting with a training dataset with M entries, the learning algorithm outputs a function $\hat{\boldsymbol{y}} = f(\boldsymbol{x})$ that estimates the output values for unseen input vectors $\boldsymbol{x}$. Examples of supervised learning include classification, where the objective is to assign each input vector to one of a set of discrete categories, and the task of regression, where the output is a vector with continuous entries. Examples of classification and regression include, respectively, the problem of recognizing images of handwritten digits and the problem of determining the orbits of bodies around the Sun from astronomical data.

Unsupervised learning deals with the learning tasks where the training data are composed of a set of input vectors without a corresponding target output [133]. These algorithms are typically used to discover hidden structure in the datasets. Examples of tasks in unsupervised learning problems include clustering, where the objective is to discover groups of similar examples within the data, density estimation, where the objective is to estimate the underlying probability distribution associated with the data, and low-dimensional visualization of high-dimensional data algorithms, which depict complex data in two or three dimensions while trying to retain key spatial characteristics in the original data.

Finally, reinforcement learning develops algorithms dealing with the problem of discovering actions that maximize a numerical reward signal [134]. The learning algorithms are not necessarily directly exposed to examples of optimal actions. Instead, they must discover them by a process similar to a guided trial and error. Reinforcement learning augmented by deep neural networks has successfully learned policies from high-dimensional sensory input for game playing, achieving human-level performance in several challenging games, including Atari 2600 [135] as well as the board game Go [136]. Likewise, reinforcement learning has been applied to the control of quantum systems [67,137] as well as to the optimization of quantum error correction codes [110,114,115], one key ingredient in the development of fault-tolerant quantum computers. Because of the specific choice of many-body problems we entertain in this article, we do not discuss reinforcement learning techniques.

To follow this tutorial, a basic knowledge of machine learning is recommended. This includes fundamental properties of neural networks, basic Monte Carlo techniques, gradient-based optimization methods, and an elementary understanding of model overfitting and generalization. For a pedagogical and comprehensive introduction to these concepts, we suggest the reviews in Refs. [138–140], specifically phrased in the context of quantum physics.

B. Rydberg atoms in two dimensions

In the following sections, we detail how to repurpose machine learning algorithms based on neural networks to tackle some problems encountered in quantum many-body physics. We focus on a many-body system composed of interacting Rydberg atoms [141,142], which we showcase in the various numerical experiments and hands-on code tutorials. Because of the high control and manipulation that can be achieved in experiments [143], these platforms have revealed themselves to be extremely useful to investigate a broad range of applications, including Ising-like quantum magnetism [9,144–146], quantum dynamics [147–149], spin liquid physics [10], and quantum computing [150,151].

Our choice of physical systems is motivated by several reasons. First, quantum hardware based on neutral atoms provides a highly programmable platform for analog quantum simulations, and it is being increasingly explored for quantum information processing. Second, these systems are engineered in a way that each atom can be found only either in the atomic ground state or in a highly excited (Rydberg) state. Because the system is effectively described by a qubit wave function, the techniques involved are relatively simplified. Third, the Rydberg atom platform provides access to large volumes of data in the form of accurate projective measurements that open the door to studying the physical properties of the system through machine learning algorithms. Finally, as explained below, the Rydberg Hamiltonian has the very special property that its ground-state wave function is real and positive in the atomic occupation number basis. Under this assumption, the definition of a neural-network wave function is also simplified. We will, however, describe the more generic case of a wave function with a sign structure in Sec. C.

We consider a square array with linear dimension L containing $N = L^2$ atoms. Each atom is described by a local Hilbert space spanned by the states $\{|g\rangle, |e\rangle\}$, referring respectively to the atomic ground state and the highly excited Rydberg state. The atoms are subjected to a uniform laser drive with Rabi frequency Ω and detuning δ, and they interact with one another via the van der Waals potential $V(x) \approx r^{-6}$ at short distances. The resulting many-body Hamiltonian is

$$\hat{H} = -\Omega \sum_{\boldsymbol{r}} \hat{S}^x(\boldsymbol{r}) - \delta \sum_{\boldsymbol{r}}^{N} \hat{\Pi}(\boldsymbol{r}) + \frac{1}{2} \sum_{\boldsymbol{r},\boldsymbol{r}}^{\prime} V(\boldsymbol{r}-\boldsymbol{r}')\hat{\Pi}(\boldsymbol{r})\hat{\Pi}(\boldsymbol{r}'), \tag{1}$$

where $\hat{\Pi}(\boldsymbol{r}) = |e\rangle\langle e|_{\boldsymbol{r}}$ is the projector onto the Rydberg state at position $\boldsymbol{r}$, $\hat{S}^x(\boldsymbol{r}) = \frac{1}{2}\hat{\sigma}^x(\boldsymbol{r})$ are spin-$\frac{1}{2}$ operators, and $V(\boldsymbol{r}-\boldsymbol{r}') = V_0/\|\boldsymbol{r}-\boldsymbol{r}'\|^6$ is the van der Waals potential between atoms at positions $\boldsymbol{r}$ and $\boldsymbol{r}'$. In the following, we assume $\Omega = 1$ MHz.

The phase diagram for the ground state of the Rydberg Hamiltonian is dictated by the mechanism of Rydberg blockade, a constraint that prevents two atoms at sufficiently short distances from being simultaneously excited to the Rydberg states. We can characterize the phase diagram in terms of the detuning δ and the interaction strength V_0. On the square lattice, several different orders have been detected by numerical simulations [152]. Here we specifically focus on the Z_2 transition between a disordered phase at large and negative detuning, where all atoms are found in the ground state, and an ordered phase at large and positive detuning, where the system is found in one of the two symmetry-broken Néel states characterized by a checkerboard pattern in the atomic occupation number [Fig. 1(a)].

We perform numerical simulations of the ground state of Hamiltonian (1) using the density matrix renormalization group (DMRG) [5,6,153] implemented using the software package ITensor [154]. We adopt a matrix product state (MPS) variational wave function $|\Psi\rangle$ with a snakelike geometry as shown in Fig. 1(b). We fix the interaction strength to $V_0 = 3$ MHz, and retain up to the third-nearest-neighbor interactions. For several values of the detuning $\delta \in \{-5, 5\}$ MHz, we run DMRG to find an approximation of the ground state, using a singular value decomposition cutoff of 10^{-10} and a target energy accuracy of 10^{-5}. To certify convergence to the ground state, each run is repeated for different initializations of the starting MPS.

We show the results of the simulations for an 8×8 array with open boundary conditions in Figs. 1(c)–1(f). We plot as a function of the detuning the ground-state energy per site $E_0/N = \langle\Psi_0|\hat{H}|\Psi_0\rangle/N$ and the staggered magnetization $\langle\mathcal{N}\rangle = N^{-1}\sum_{\boldsymbol{r}}(-1)^{x+y}\langle\hat{S}^z(\boldsymbol{r})\rangle$, which can be used to detect Néel order. Whenever all atoms are in the ground state, $\langle\mathcal{N}\rangle \approx 0$, while for an ordered state with a checkerboard pattern, one has $\langle\mathcal{N}\rangle \approx 0.5$. We also show the average occupation number in momentum space,

$$n(\boldsymbol{k}) = \frac{1}{\sqrt{N}} \sum_{\boldsymbol{r}} e^{i\boldsymbol{k}\cdot\boldsymbol{r}}\langle\hat{n}(\boldsymbol{r})\rangle, \tag{2}$$

where $\hat{n}(\boldsymbol{r}) = \frac{1}{2}(1 - 2\hat{S}^z(\boldsymbol{r}))$. We observe a peak at $\boldsymbol{k} = (\pi, \pi)$ for large detuning $\delta = 4$ MHz [Fig. 1(e)], and a featureless state at negative detuning (not shown).

The two phases of the Rydberg atoms are separated by a second-order quantum phase transition at a critical point δ_c. We can extract an approximation of δ_c by measuring the energy gap $\Delta = |E_0 - E_1|$ between the ground state and the first excited state $|\Psi_1\rangle$. We compute E_1 by running DMRG on the Hamiltonian $\hat{H}' = \hat{H} + \omega|\Psi\rangle\langle\Psi|$, where ω is an energy penalty. From the energy gap curve, we estimate the detuning where $\Delta \approx 0$ to be $\delta_c \approx 1.3$ MHz. This approximate value is sufficient for the purpose of this article, although a more systematic scaling study with appropriate boundary conditions (to minimize finite-size effects) should be performed to accurately determine the critical point and critical exponents of the transition.

Once we have solved for the ground states of the Rydberg Hamiltonian, the corresponding MPSs can be used to generate data to train the neural networks for the different applications. In this case, the data consist of projective measurements in the atomic occupation number basis $|\boldsymbol{\sigma}\rangle = |\sigma_1, \ldots, \sigma_N\rangle$, where $\sigma_j = 0$ and $\sigma_j = 1$ refer, respectively, to the jth atom being in the ground state and the jth atom being in the Rydberg state. Given a wave function $|\Psi\rangle$, the probability to observe an atomic pattern $\boldsymbol{\sigma}$ following a measurement is given simply by the Born rule $P(\boldsymbol{\sigma}) = |\langle\boldsymbol{\sigma}|\Psi\rangle|^2$. Because of the intrinsic one-dimensional geometry of a MPS, it is possible to efficiently sample the probability distribution $P(\boldsymbol{\sigma})$ by using the chain rule of probabilities. Moreover, the sampling is exact in the sense that the samples are completely independent of one another [155].

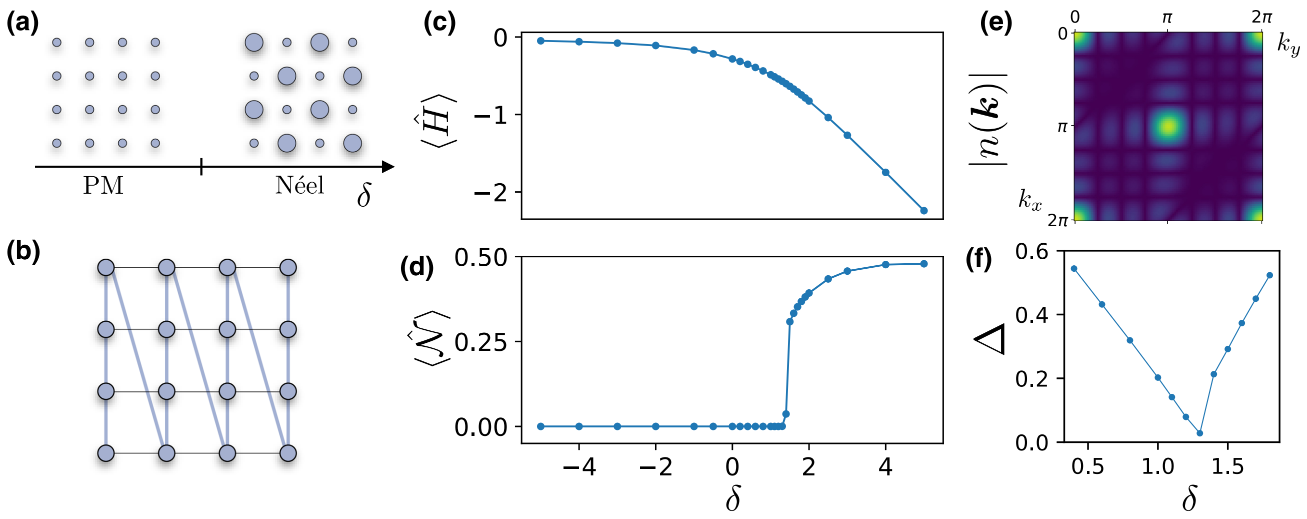

FIG. 1. Rydberg atoms in a two-dimensional square array. (a) The phase diagram at a fixed value of the interaction $V = 3$ MHz and $\Omega = 1$ MHz. At large and negative detuning, the system is in a disordered phase with all atoms in the ground state. At large and positive detuning, the atoms are found in a checkerboard pattern with Néel order. (b) Snakelike geometry of the MPS path along the square lattice, used for the DMRG simulations. Ground-state energy (c) and the staggered magnetization (Néel order) (d) as a function of the detuning δ for an 8×8 array ($V = 3$ MHz). (e) Absolute value of the average occupation number in momentum space $|n(\boldsymbol{k})|$ deep into the Z_2 ordered phase ($\delta = 4$ MHz), showing a peak at $\boldsymbol{k} = (\pi, \pi)$, a signature of antiferromagnetic order. (f) Energy gap Δ between the ground state and the first excited state, detecting a quantum phase transition at detuning $\delta \approx 1.3$. PM, paramagnetic.

III. LEARNING A QUANTUM PHASE TRANSITION

An important task in condensed matter physics and statistical physics is to characterize different phases of matter and the associated phase transitions between them. Typically, phases of matter are described in terms of simple real-space patterns and their associated order parameters, which are theoretically understood using the Landau symmetry-breaking paradigm [1]. While a wide array of theoretical and experimental tools to study interacting quantum systems have been constructed in relation to these patterns, there is an increasing set of states of matter whose theoretical and experimental understanding eludes the Landau symmetry-breaking paradigm. The characterization of these phases may rely on, for example, out-of-equilibrium properties of the system as in the many-body localized phase [156,157] or on topological invariants in topological phases and spin liquids [1,158,159].

Machine learning provides an alternative route to the characterization of phases of matter and their associated phase transitions in a semiautomated fashion without the direct use of manually specified real-space patterns and/or other signatures, provided that a sufficiently large training set is available. In its simplest form [15], given the existence of a classical or quantum phase transition between two phases in a physical system, one can use supervised learning to attempt to classify experimental or numerical snapshots of the phases of matter separated by the transition. This task can be achieved using most classification algorithms, for example, those based on a neural network or a support vector machine [133], trained on snapshots of two phases of matter labeled according to the corresponding phase from which the snapshot originated.

The architectural choice of the classifier is flexible, but a natural strategy is to take into account the structural properties of the problem such as the symmetries of the physical system and its spatial dimensionality. These choices are important since they impact the computational complexity of the learning algorithms, where encoding the knowledge of the physical problem typically increases the overall complexity of the algorithm. For instance, we might be interested in classifying numerically generated snapshots of a large two-dimensional array of spins governed by an Ising model in two different phases, for which a natural choice is a 2D CNN [132], which accounts for the 2D structure and locality of the datasets.

Although here we explore only the simplest supervised learning strategy [15], we stress that machine learning approaches to studying phases and phase transitions have been significantly expanded and they do not require precise knowledge of the location of the critical point [16,160], can be fully automatized, and can discover ordered phases [15,18,161], topological phases [16,162,163], and phases such as the many-body localized phase, which is characterized by its dynamical properties [24,164,165].

The nature of the snapshots used to train the learning algorithms is vastly flexible, and hence these strategies are of wide applicability, and can include numerically generated configurations visited during a classical or quantum Monte Carlo simulation of the physical system

[15,16,18–20,26,166], entanglement spectra [16,24], correlation matrices [167,168], tensors in a MPS [168], numerically generated projective measurements [169], high-resolution real-space snapshots of complex many-body systems obtained with quantum gas microscopes for ultracold atoms [29,170], single-shot experimental momentum-space density images of ultracold quantum gases [30], and spectroscopic imaging scanning tunneling microscopy data [28].

A. Convolutional neural networks and their training

Below we explore learning a quantum phase transition in a 2D array of interacting Rydberg atoms using projective measurements. Because of the 2D arrangement of the atoms, each measurement—obtained from a numerical simulation—can be interpreted as an "image" whose pixels depict the state of each atom after the measurement. Known to be extremely effective image classifiers [132], we use a convolutional classifier, which readily takes advantage of the structure of the classification task since it exploits the 2D spatial arrangement of the Rydberg atoms and their locality, as well as the approximate translation invariance of the system.

CNNs use a mathematical operation called "convolution" to process information for data that have a natural gridlike topology [132]. A 2D convolutional layer implements the operation

$$\mathsf{h}^{(q)}_{i,j,k} = F\left(\sum_{l,m_y,m_x} \mathsf{h}^{(q-1)}_{l,j+m_y,k+m_x} \mathsf{K}^{(q)}_{i,l,m_y,m_x}\right) := F\left(\mathsf{K}^{(q)} * \mathsf{h}^{(q-1)}\right),$$

where the trainable kernel $\mathsf{K}^{(q)}_{i,l,m_y,m_x}$ at layer q specifies the connection strength between a unit in channel i of the output and a unit in channel l of the input, with a spatial offsets of m_y rows (labeled y direction) and m_x columns (labeled x direction) between the output and the input variables. The dimensions of the array $\mathsf{K}^{(q)}_{i,l,m,n}$ are I_{out}, L_{input}, M_y, and M_x, which correspond to the number of output channels, the number of input channels, the dimension of the filter in the vertical direction, and the dimension of the filter in the horizontal direction, respectively. The activation at layer q consists of elements $\mathsf{h}^{(q)}_{l,j,k}$, where j and k label vertical and horizontal directions, respectively, and l specifies the channel. The activation units are labeled by q, where $q = 0$ corresponds to the raw projective measurement data. Finally, the nonlinear function $F(x)$, which in our examples is typically a rectified linear unit (ReLU) $F(x) = \max(0, x)$, is applied element-wise to each of the components of its input. A convolutional neural network equipped with two convolutional layers is schematically shown in Fig. 2.

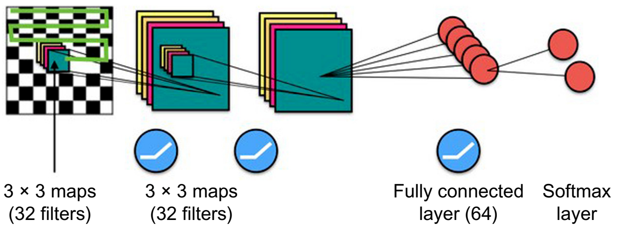

FIG. 2. A convolutional neural network. The elements of the input $\mathsf{h}^{(0)}_{l,j,k}$ correspond to the outcome of a projective measurement on the Rydberg system. The first operation is a convolutional layer with $M_y \times M_x = 3 \times 3$ kernels with $I_{\text{out}} = 32$ output channels and $L_{\text{input}} = 1$. This kernel is convolved with an input configuration with $N = 8 \times 8$ Rydberg atoms. Likewise, the second operation corresponds to a convolutional layer with $M_y \times M_x = 3 \times 3$ kernels with $I_{\text{out}} = 32$ output channels and $L_{\text{input}} = 32$. The output of the second convolutional layer is flattened and fed to a FC layer with a ReLU activation, followed by another FC layer with a softmax activation, which produces the prediction outcome.

Followed by the convolutional layers, a CNN typically processes information using sets of fully connected (FC) layers that implement a matrix-vector operation followed by a nonlinearity F as

$$\mathsf{h}^{(q)}_{i} = F\left(\sum_{l} \mathsf{h}^{(q-1)}_{l} \mathsf{K}^{(q)}_{i,l} + \mathsf{b}^{(q)}_{i}\right), \tag{3}$$

where the trainable parameters of the FC layer are the kernel $\mathsf{K}^{(q)}_{i,l}$ and the bias vector $\mathsf{b}^{(q)}_{i}$. To feed the output of a convolutional layer $\mathsf{h}^{(q-1)}_{l,j,k}$ to a FC layer, the array is reshaped or "flattened" to $\mathsf{h}'^{(q-1)}_{l}$ so that all the original components are packed into a one-dimensional array with dimension L_{FC}. The last two layers of the CNN in Fig. 2 correspond to two fully connected layers with a ReLU and a softmax nonlinearity, respectively. The softmax function S is given by

$$\mathrm{S}(\boldsymbol{v}) = \frac{\exp \boldsymbol{v}}{\sum_i \exp v_i}. \tag{4}$$

where v_i are the components of a vector $\boldsymbol{v}$ and the exponential function acts element-wise on the components of the vector. The input to the CNN and its trainable parameters are real, so the outcome of the softmax layer can be interpreted as a probability distribution since $0 \le S(v_i) \le 1$ and $\sum_i S(v_i) = 1$.

Finally, we we interpret our CNN as a model for the conditional probability of assigning a phase of matter $y = 0, 1$ to a projective measurement outcome $\boldsymbol{\sigma} = \mathsf{h}^{(0)}$, i.e., $P_{\boldsymbol{\theta}}(y|\boldsymbol{\sigma})$, where $\boldsymbol{\theta}$ encompasses all the trainable parameters of the CNN. The conditional probability is given by

$$P_{\boldsymbol{\theta}}(y|\boldsymbol{\sigma}) = S\left[b_y^{(4)} + \sum_m \mathsf{K}_{y,m}^{(4)} F\left(b_m^{(3)} + \sum_l \mathsf{K}_{m,l}^{(3)}\,\text{flatten}\left(F\left[\mathsf{K}^{(2)} * F\left(\mathsf{K}^{(1)} * \boldsymbol{\sigma}\right)\right]\right)_l\right)\right], \tag{5}$$

where the function flatten$()_l$ is the lth component of a vector that arises from reshaping the incoming argument of the function to a one-dimensional array.

To estimate the parameters of the CNN we use the maximum likelihood principle, where the parameters of a statistical model are selected by assigning high probability to the observed data. For a dataset with observations $\{\boldsymbol{\sigma}_n, y_n\}_{n=1}^M$, where $y_n = 0, 1$ label the phase of matter from which a projective measurement $\boldsymbol{\sigma}_n$ was taken from, the likelihood assigned by the model to the dataset can be written as

$$p(\boldsymbol{y}|\boldsymbol{\theta}) = \prod_{n=1}^{M} P_{\boldsymbol{\theta}}(y_n|\boldsymbol{\sigma}_n)^{y_n}[1 - P_{\boldsymbol{\theta}}(y_n|\boldsymbol{\sigma}_n)]^{1-y_n}, \tag{6}$$

where $\boldsymbol{y} = (y_1, \ldots, y_M)$. Instead of attempting to maximize the likelihood, it is convenient to define a loss function by taking the negative logarithm of the likelihood, which gives the cross-entropy

$$E(\boldsymbol{\theta}) = -\ln(p(\boldsymbol{y}|\boldsymbol{\theta})) = \sum_{n=1}^{M}\{y_n \ln(P_{\boldsymbol{\theta}}(y_n|\boldsymbol{\sigma}_n)) + (1 - y_n)\ln[1 - P_{\boldsymbol{\theta}}(y_n|\boldsymbol{\sigma}_n)]\}. \tag{7}$$

To train the model, we minimize $E(\boldsymbol{\theta})$ using gradient descent techniques [133]. While it is possible to evaluate the gradients of $E(\boldsymbol{\theta})$ with respect to the parameters $\boldsymbol{\theta}$ in the CNN analytically using the chain rule, a more convenient and less error-prone approach is to use automatic differentiation (AD), which is a set of techniques to numerically evaluate the derivative of a function specified by a computer program. A complete survey detailing AD can be found in Ref. [171].

In addition, instead of using the entire dataset in the calculation of $E(\boldsymbol{\theta})$ and its gradients, we use smaller batches of data of size $M_{\text{batch}} < M$, which means that the gradients used during optimization become stochastic since they fluctuate from batch to batch. In the examples below we use $N_{\text{batch}} = 32$. The gradient update rule used in our examples is a modified version of the usual gradient descent method called "Adam" [172].

1. Code walk-through

We demonstrate supervised learning with a CNN to learn the quantum phase transition in the Rydberg array using the machine learning software library TensorFlow [173]. We first generate training data by sampling the MPS wave functions obtained from DMRG at different detunings δ. These data are then divided into a training set (used to update the neural-network parameters) and a test set (used to validate the performance of the model). Each dataset consists of a list of atomic occupation patterns $\boldsymbol{\sigma}$ and their "phase label" y.

We begin by importing the required functionalities and loading the data. Since we are using a CNN with a two-dimensional geometry, the atomic configurations in the training and test datasets need to be appropriately reshaped from the one-dimensional MPS structure.

```
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models

# linear dimensions of the system
Lx = 8
Ly = 8

# training set
train_config = np.loadtxt("xtrain.txt")
train_label = np.loadtxt("ytrain.txt",dtype=np.
    uint8)

# test set
test_config = np.loadtxt("xtest.txt")
test_label  = np.loadtxt("ytest.txt",dtype=np.
    uint8)

# reshaping the training/test configurations
train_config = np.reshape(train_config,(
    train_config.shape[0],Lx,Ly,1))
test_config = np.reshape(test_config,(
    test_config.shape[0],Lx,Ly,1))

# reshaping the training/test labels
test_label  = np.reshape(test_label,(test_label.
    shape[0],1))
train_label = np.reshape(train_label,(
    train_label.shape[0],1))

# Names for the phases
class_names = ["disordered", "ordered"]
```

Next we define the neural-network architecture by combining layers predefined in TensorFlow. After initialization, we proceed to implement the model of Eq. (5). First, the raw input data are processed by two stacked convolutional layers, each one with 3×3 filters and 32 channels using rectified linear units. The output of the second CNN layer is fed to two stacked fully connected layers with 64 hidden units and two output units. These last units correspond to model output for the disordered and ordered phase. This set of hyperparameters is set using a scaling experiment until convergence (see Sec. VI).

```
# initialize the model
model = models.Sequential()

# first convolutional layer
model.add(layers.Conv2D(32, (3, 3),
          activation = "relu",
          input_shape = (Lx, Ly, 1)))

# second convolutional layer
model.add(layers.Conv2D(32, (3, 3),
          activation="relu"))

# flatten the output
model.add(layers.Flatten())

# dense layer with 64 output units
model.add(layers.Dense(64, activation='relu'))

# dense layer with two output units
model.add(layers.Dense(2))
```

Once the architecture is defined, the model can be compiled by adding the cost function [i.e., the cross-entropy in Eq. (7)] and the optimizer to update the model parameters, the Adam optimizer [172]. Internally, TensorFlow builds a computational graph containing each operation being executed from the input state to the final output of the architecture. The gradients of the cost function with respect to each network parameter are then evaluated using AD. At compilation time, we can also add a metric to be monitored during training, in this case the classification accuracy (i.e., the fraction of the test set samples that are being classified correctly).

```
# Compiling the model
model.compile(optimizer='adam',
  loss=tf.keras.losses.
    SparseCategoricalCrossentropy(from_logits=
    True),
  metrics=['accuracy'])
```

The model is now ready to be trained using the Rydberg data for a set number of epochs, which is the number of passes of the entire training dataset the machine learning algorithm has completed. After training is complete, we can evaluate the model on the held-out test data to quantify its accuracy.

```
# training the model
history = model.fit(
  train_config,
  train_label,
  epochs=5,
  validation_data = (test_config, test_label))

# evaluate the overall model
test_loss,test_acc = model.evaluate(test_config,
     test_label, verbose=2)
```

We show the results of the training in Fig. 3 evaluated on various test sets at different detuning values δ. We plot the average output signal for the two output neurons, i.e., an estimate of $f(y,\delta) = \sum_{\sigma} |\Psi(\boldsymbol{\sigma})|^2 P_{\theta}(y|\boldsymbol{\sigma})$, on the topmost dense layer. When the detuning is large and negative, the disordered neuron saturates to 1, and the ordered

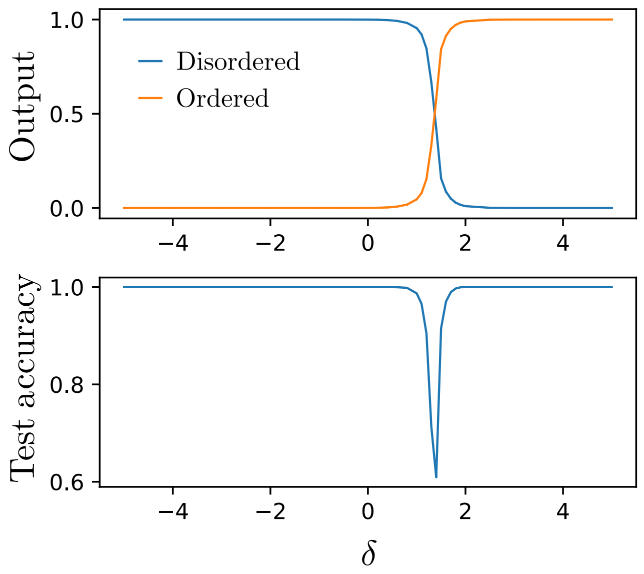

FIG. 3. Learning the quantum phase transition in the Rydberg-atom array. Top: Output signal for the two units in the topmost dense layer of the neural-network architecture as a function of the detuning, corresponding to Eq. (5). Bottom: Accuracy on the test dataset, showing a dip near the critical point. This estimator is calculated by comparing the prediction of the CNN with the correct labels assigned to the samples in the test set.

neuron is nearly 0, while the signals reverse at large and positive detuning. We can use the crossing point between the two curves to detect the critical point. We also show the accuracy in the test set, which shows a dip near the critical point. This is when the neural network is most uncertain about assigning a phase label to any given atomic configuration.

IV. QUANTUM STATE TOMOGRAPHY

Quantum characterization, verification, and validation is a framework for algorithms and routines used to assess the quality and the performance of experimental quantum hardware and characterize its components [174]. The workflow underlying these algorithms is inherently data driven: appropriate measurement data are first collected from the quantum device under examination, and are then processed by an algorithm running on a classical computer. Depending on the degree of complexity of the algorithm, different amounts of information can be gained. This could be a single figure of merit, such as the average error rate for a set of quantum gates [175–177], or the fidelity (or a proxy thereof) between a quantum state prepared by the hardware and a desired reference state [178–180]. One may be also interested in retrieving the full quantum state generated by a device. This procedure—the reconstruction of an unknown quantum state from measurement data—is called "quantum state tomography" [181–188].

There are two assumptions in QST: the ability to prepare many identical copies of the quantum state ϱ of interest and the ability to repeatedly perform measurements on it. The set of measurements is, in general, described by positive-operator-valued measures (POVMs) $\boldsymbol{\mathcal{M}} = \{\Pi_k\}$ [189]. The set $\boldsymbol{\mathcal{M}}$ is said to be *informationally complete* (IC) if it spans the full Hilbert space. Von Neumann measurements in the Pauli bases are an example of informationally (over)complete measurements, where for a single qubit $\boldsymbol{\mathcal{M}}$ contains the six rank-1 projectors into the eigenstates of the Pauli matrices. For an IC set, any quantum state can be uniquely identified by the probabilities of the measurements in $\boldsymbol{\mathcal{M}}$, as specified by the Born rule $p(k) = \mathrm{Tr}\, \varrho \Pi_k$.

The simplest method to perform QST is *linear inversion*, which reconstructs the quantum state by simply inverting the Born rule:

$$\varrho = (A^\dagger A)^{-1} A^\dagger \hat{p}, \tag{8}$$

where each row in the matrix A is a vectorized POVM element $|\Pi_k\rangle\rangle$, and $\hat{p}$ is the empirical measurement probability. One issue with linear inversion is that the reconstructed density operator $\boldsymbol{\rho}$ is not necessarily positive, although negative eigenvalues can be appropriately removed to produce a positive state that is closest to the output of linear inversion [187]. A more powerful approach, but also more computationally intensive, is *maximum likelihood estimation* [182–184], where the state $\boldsymbol{\rho}$ is found by minimizing the likelihood function for the observed data under the constraint $\boldsymbol{\rho} \geq 0$.

Traditional QST algorithms based on linear inversion or maximum likelihood suffer a complexity that scales exponentially with the number of qubits or particles involved. This exponential scaling stems from two reasons. First, the representation of the quantum state $\boldsymbol{\rho}$, which is inevitably exponential in the system size. Second, the *sample complexity*; that is, the number of measurements that need to be collected in an experiment to achieve a faithful reconstruction of the quantum state. Typically, statistics from an IC set are required to get a good fit, and the size thereof scales exponentially with the number of qubits, with provable upper and lower bounds [190,191]. For these reasons, traditional QST has remained limited to quantum systems containing only a small number of particles [192].

Several algorithms to overcome this severe complexity have been proposed over the last decade. Notable examples include compressed sensing tomography [193–196], permutationally invariant tomography [197,198], and tensor-network tomography [199–202], which rely, respectively, on the sparsity, translational invariance, and low entanglement of the target quantum state. More recently, a new framework built on neural networks and unsupervised learning was proposed [71], and is based on the assumption that most physical states of interest typically contain some degree of *structure* (i.e., correlations, symmetries, etc.), in the sense that they can be described using a reduced number of parameters (much smaller than the dimension of the Hilbert space). The general idea is to leverage the capability of unsupervised machine learning to autonomously identify such structure in raw data, and compress it using a neural-network representation of the quantum state.

In what follows, we focus on pure quantum states, and discuss the extension to mixed states at the end of this section. We consider a system of N qubits (or any other two-level system) described by a wave function $|\Phi\rangle$ with amplitudes $\Phi(\boldsymbol{\sigma}) = \langle \boldsymbol{\sigma} | \Phi \rangle$ in an appropriate reference basis $|\boldsymbol{\sigma}\rangle = |\sigma_1, \ldots, \sigma_N\rangle$ ($\sigma_j \in \{0, 1\}$). To circumvent the scalability issue of standard QST, we adopt a compact representation of a wave function expressed in terms of a neural network [50]. The resulting *neural-network wave function* is simply a highly nonlinear parametric function of the basis states $\psi_{\boldsymbol{\theta}}(\boldsymbol{\sigma})$, where $\boldsymbol{\theta}$ is a set of parameters (e.g., weights and biases). Several types of neural networks have been successfully implemented to perform QST, including feed-forward neural networks [76,78], variational autoencoders [72], generative adversarial networks [82], recurrent neural networks [75,89], and transformers [84]. Here we examine the restricted Boltzmann machine because of its simplicity, high expressive power, and natural interpretation in describing correlated multivariate probability distributions.

A. The restricted Boltzmann machine

The restricted Boltzmann machine is an energy-based model introduced in the early 1980s for generative modeling [203,204], and is built on a connection between cognitive science and statistical mechanics [205–207]. The RBM features two layers of stochastic binary units: a visible layer $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots)$ and a hidden layer $\boldsymbol{h} = (h_1, h_2, \ldots)$, containing, respectively, N and n_h neurons (or units). The two layers in the RBM are fully connected by a symmetric weight matrix $\boldsymbol{W}$, with no intralayer connections (hence its *restricted* nature). The visible units are used to represent the variables relevant to the specific problem at hand, such as the pixel values in an image (or the computational basis states for qubits). The size of the hidden layer is a natural control parameter for the representational power of the model. Since RBMs are universal function approximators [208], they can capture any discrete distribution provided the number of hidden units is sufficiently large (possibly exponential in the number of visible units).

The RBM associates with each configuration of the visible and hidden layers $(\boldsymbol{\sigma}, \boldsymbol{h})$ the energy

$$E_{\boldsymbol{\theta}}(\boldsymbol{\sigma}, \boldsymbol{h}) = -\sum_j \sum_i W_{ij} h_i \sigma_j - \sum_j b_j \sigma_j - \sum_i c_i h_i, \tag{9}$$

where $\boldsymbol{\theta} = (\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c})$ is the set of parameters, and we also introduced biases $\boldsymbol{b}$ and $\boldsymbol{c}$ for the visible and hidden units, respectively. Given this energy functional, the (stochastic) RBM units are distributed according to the Boltzmann distribution at temperature $\beta = 1$:

$$p_{\boldsymbol{\theta}}(\boldsymbol{\sigma}, \boldsymbol{h}) = Z_{\boldsymbol{\theta}}^{-1} e^{-E_{\boldsymbol{\theta}}(\boldsymbol{\sigma}, \boldsymbol{h})}, \tag{10}$$

where the partition function is

$$Z_{\boldsymbol{\theta}} = \sum_{\boldsymbol{\sigma}, \boldsymbol{h}} e^{-E_{\boldsymbol{\theta}}(\boldsymbol{\sigma}, \boldsymbol{h})}. \tag{11}$$

Importantly, because the network architecture is restricted, we can trace out the hidden layer explicitly, obtaining the marginal probability distribution over the visible space:

$$p_{\boldsymbol{\theta}}(\boldsymbol{\sigma}) = \sum_{\boldsymbol{h}} p_{\boldsymbol{\theta}}(\boldsymbol{\sigma}, \boldsymbol{h}) = Z_{\boldsymbol{\theta}}^{-1} e^{\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{\sigma})}, \tag{12}$$

where we defined an "effective energy"

$$\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{\sigma}) = \sum_j b_j \sigma_j + \sum_i \left(1 + e^{\sum_j W_{ij}\sigma_j + c_i}\right). \tag{13}$$

The main purpose of the RBM is generative modeling, which is the task of learning a representation of an unknown probability distribution from data, allowing the neural network to produce new data points. In other words, the RBM training attempts to discover low-dimensional features in the data to allow generalization beyond the finite-size dataset.

Let us consider a dataset $\mathcal{D} = \{\boldsymbol{\sigma}_k\}$ with underlying (unknown) probability distribution $q(\boldsymbol{\sigma})$. The RBM can be trained using unsupervised learning to minimize the distance between the two distributions. Such distance measure is typically expressed in terms of the Kullback-Leibler divergence [209]:

$$D_{\mathrm{KL}}(q|p_{\boldsymbol{\theta}}) = \sum_{\boldsymbol{\sigma}} q(\boldsymbol{\sigma}) \log \frac{q(\boldsymbol{\sigma})}{p_{\boldsymbol{\theta}}(\boldsymbol{\sigma})}, \tag{14}$$

with $D_{\mathrm{KL}}(q|p_{\boldsymbol{\theta}}) > 0$ for all $q, p_{\boldsymbol{\theta}}$ and $D_{\mathrm{KL}}(q|p_{\boldsymbol{\theta}}) = 0$ if and only if $p_{\boldsymbol{\theta}} = q$.

The exponentially large sum over the full configuration space is approximated using the available data, leading to the cost function

$$\mathcal{C}(\boldsymbol{\theta}) = -\frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{\sigma} \in \mathcal{D}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\sigma}) - H_{\mathcal{D}}, \tag{15}$$

where $|\mathcal{D}|$ is the size of the dataset. Up to a constant dataset entropy term $H_{\mathcal{D}}$, the Kullback-Leibler divergence reduces simply to the negative logarithm of the likelihood function $\mathcal{L}(\mathcal{D}|p_{\boldsymbol{\theta}})$.

The RBM can be trained using one of the many flavors of gradient descent. It is possible to show that the gradients of the cost function are given by

$$\nabla_{\boldsymbol{\theta}} \mathcal{C}(\boldsymbol{\theta}) = \langle \nabla_{\boldsymbol{\theta}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{\sigma}) \rangle_{p_{\boldsymbol{\theta}}} - \langle \nabla_{\boldsymbol{\theta}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{\sigma}) \rangle_{\mathcal{D}}, \tag{16}$$

where the gradients $\nabla_{\boldsymbol{\theta}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{\sigma})$ can be computed exactly for any sample $\boldsymbol{\sigma}$. We see that the gradients $\nabla_{\boldsymbol{\theta}} \mathcal{C}(\boldsymbol{\theta})$ have two components. There is the average over the data points $\langle \nabla_{\boldsymbol{\theta}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{\sigma}) \rangle_{\mathcal{D}}$, which is fast to compute. In contrast, the average over the model distribution

$$\langle \nabla_{\boldsymbol{\theta}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{\sigma}) \rangle_{p_{\boldsymbol{\theta}}} = \frac{1}{Z_{\boldsymbol{\theta}}} \sum_{\boldsymbol{\sigma}} e^{\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{\sigma})} \nabla_{\boldsymbol{\theta}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{\sigma}) \tag{17}$$

requires the partition function, whose calculation is, in general, intractable. However, this expectation value can be approximated using the Monte Carlo method by drawing N_S samples from the model distribution $\{\boldsymbol{\sigma}_i\} \sim p_{\boldsymbol{\theta}}(\boldsymbol{\sigma})$:

$$\langle \nabla_{\boldsymbol{\theta}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{\sigma}) \rangle_{\mathcal{D}} \approx \frac{1}{N_S} \sum_{i=1}^{N_S} \nabla_{\boldsymbol{\theta}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{\sigma}_i). \tag{18}$$

This is the most computationally intensive step of the training, and depending on the specific distribution to be learned, advanced Monte Carlo algorithms may be required to collect sufficiently uncorrelated samples.

B. Reconstruction of Rydberg atoms

Now that we have introduced the main features of the RBM and its training, we are ready to explore its use for QST. As a first application, we examine the reconstruction of Rydberg-atom wave functions. An important property of the ground-state wave function $|\Phi\rangle$ of the Rydberg Hamiltonian (1) is that it is positive in the occupation number basis $\Phi(\boldsymbol{\sigma}) \geq 0$ (where $\sigma_j = 0$ and $\sigma_j = 1$ refer to the ground and excited states). This property follows directly from the representation of the Hamiltonian in this basis, in which all of its off-diagonal elements can be gauged to be negative (i.e., the Hamiltonian is *stoquastic* in this basis [210]).

The positivity of the target state implies that we may parametrize the neural-network wave function simply as $\psi_{\boldsymbol{\theta}}(\boldsymbol{\sigma}) = \sqrt{p_{\boldsymbol{\theta}}(\boldsymbol{\sigma})}$ for any normalized probability distribution $p_{\boldsymbol{\theta}}(\boldsymbol{\sigma})$. Here we choose the RBM probability distribution [Eq. (12)], where the visible units correspond to the atomic occupations. Moreover, because the wave function is positive, measurement data from a single basis are sufficient to characterize the state. This means that the QST problem, under these assumptions, is equivalent to unsupervised learning of projective measurement data in the atomic occupation number basis. The tomographic reconstruction of the quantum state is done by iteratively changing the RBM parameters to minimize the cost function

$$\mathcal{C}(\boldsymbol{\theta}) = -\frac{1}{|\mathcal{D}|}\sum_{\boldsymbol{\sigma}\in\mathcal{D}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\sigma}). \tag{19}$$

When convergence in the training is reached, the taught RBM can be used to estimate various properties of interest. For a generic observable $\hat{\mathcal{O}}$, the expectation value on the RBM wave function reduces to an average over the RBM distribution:

$$\langle\hat{\mathcal{O}}\rangle = \frac{\langle\psi_{\boldsymbol{\theta}}|\hat{\mathcal{O}}|\psi_{\boldsymbol{\theta}}\rangle}{\langle\psi_{\boldsymbol{\theta}}|\psi_{\boldsymbol{\theta}}\rangle} = \frac{\sum_{\boldsymbol{\sigma}}\psi_{\boldsymbol{\theta}}(\boldsymbol{\sigma})\langle\boldsymbol{\sigma}|\hat{\mathcal{O}}|\psi_{\boldsymbol{\theta}}\rangle}{\sum_{\boldsymbol{\sigma}} p_{\boldsymbol{\theta}}(\boldsymbol{\sigma})} \tag{20}$$

$$= \frac{1}{Z_{\boldsymbol{\theta}}}\sum_{\boldsymbol{\sigma}} p_{\boldsymbol{\theta}}(\boldsymbol{\sigma})\mathcal{O}_{\text{loc}}(\boldsymbol{\sigma}), \tag{21}$$

where we introduced the so-called local observable

$$\mathcal{O}_{\text{loc}}(\boldsymbol{\sigma}) = \frac{\langle\boldsymbol{\sigma}|\hat{\mathcal{O}}|\psi_{\boldsymbol{\theta}}\rangle}{\langle\boldsymbol{\sigma}|\psi_{\boldsymbol{\theta}}\rangle}. \tag{22}$$

The expectation value in Eq. (21) can then be approximated with a Markov chain using Monte Carlo sampling, similarly to the evaluation of the gradients. The evaluation of the local observable $\mathcal{O}_{\text{loc}}(\boldsymbol{\sigma})$ remains efficient as long as the matrix representation of $\mathcal{O}$ in the reference basis is sufficiently sparse. This measurement procedure is also useful for monitoring different observables during training, such as average densities and correlation functions. These metrics can be used to assess convergence, since in general the calculation of the Kullback-Leibler divergence is intractable as it also requires estimation of the partition function $Z_{\boldsymbol{\theta}}$.

1. Code walk-through

We perform QST on the Rydberg-atom data using the PYTHON package NetKet [211]. First, we import the library and define the relevant parameters for the numerical experiments. For instance, we consider the reconstruction of a square array with linear size $L = 8$ (containing $N = 64$ spins), with Hamiltonian parameters $V = 3.0$ MHz, $\Omega = 1.0$ MHz, and $\delta = 2.0$ MHz.

```python
from mpi4py import MPI
import netket as nk

# Rydberg Hamiltonian parameters
L = 8         # linear size
V = 3.0       # Van der Waals interaction
Omega = 1.0   # Rabi frequency
delta = 2.0   # detuning
```

We then define the lattice structure and the Hilbert space for the neural-network wave function, and load the training data from a file:

```python
# define the lattice structure
square_lattice = nk.graph.Hypercube(
    length=L,
    n_dim=2,
    pbc=False)

# build the Hilbert space
hilbert = nk.hilbert.Qubit(graph=square_lattice)

# load the training data
data_path = ".../path_to_data"
rotations, samples, bases = LoadData(hilbert,
    data_path)
```

Since we are doing training in a single measurement basis, the arrays `bases` and `rotations` are “trivial.” Otherwise, these variables would contain, respectively, an integer encoding of each distinct basis and its associated (local) unitary rotations.

Next we define the main components of the QST algorithm: the neural-network wave function, the sampler used to approximate the gradients, and the optimizer for the parameter updates.

```python
# Neural-network wavefunction
rbm = nk.machine.RbmSpinReal(hilbert=hilbert,
    alpha=1)

# Monte Carlo sampler
sa = nk.sampler.MetropolisLocal(machine=rbm)

# Optimizer
op = nk.optimizer.AdaDelta(rho=0.95, epscut =
    1.0e-7)

# Initialize tomography object
qst = nk.Qsr(
        sampler = sa,
        optimizer = op,
        n_samples_data = 1000,
        n_samples = 2000,
        rotations = rotations,
        samples = samples,
        bases = bases,
        sr = None)
```

In the definition of the RBM (with real-valued network parameters), the parameter $\alpha = n_h/N$ represents the density of hidden units. We use the AdaDelta optimizer [212] and a Metropolis sampler using simple single-spin flips. The tomography parameters `n_samples_data` and `n_samples` refer, respectively, to the number of training samples and the number of samples drawn from the model distribution to compute the gradients for a single parameter update (i.e., for batch gradient descent). We refer the reader to the NetKet documentation for additional details. Finally, we generate the Rydberg Hamiltonian to be measured during the learning, and run the QST for a fixed number of training iterations (`epochs`).

```python
# define observable for measurements
H = generatehamiltonian(hilbert, L, L, V, Omega,
     delta)
qst.add_observable(H, "H")

# run quantum state tomography
for ep in qst.iter(epochs):
    obs = qst.get_observable_stats()
```

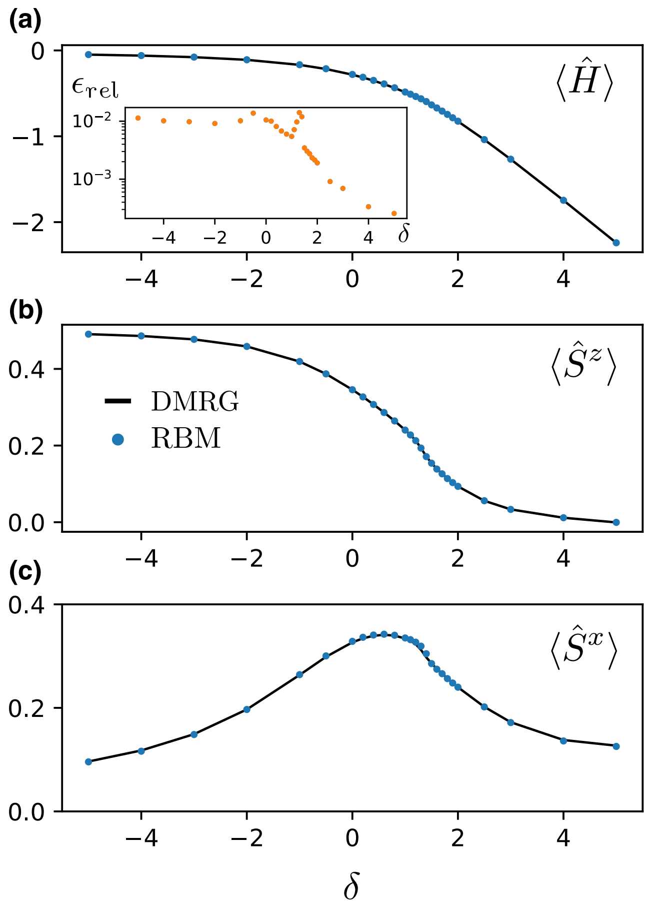

FIG. 4. Quantum state tomography of an 8×8 array of Rydberg atoms with unsupervised learning of single-shot atomic occupation data. We compare various observables measured using the MPSs obtained from DMRG (black line) and the neural-network wave functions learned from data generated at different values of the detuning δ. We plot the average energy per spin (a) and the average magnetization along the z axis (b) and x axis (c). The inset in (a) shows the relative error in the energy $\epsilon_{\mathrm{rel}} = |E_{\mathrm{RBM}} - E_{\mathrm{DMRG}}|/|E_{\mathrm{DMRG}}$. Error bars estimated via standard deviations are too small to be visible.

We perform QST on datasets of projective measurements generated using the MPS obtained from DMRG at different detunings. Each dataset contains 10^5 measurements. We train each RBM separately using the hyperparameters reported above, and measure at each training iteration various observables of interest. We show the results in Fig. 4, where we plot the average energy per spin $\langle \hat{H} \rangle / N$ and the average magnetizations $\langle \hat{S}^{z/x} \rangle = \sum_j \langle \hat{S}_j^{z/x} \rangle / N$ after the training has converged. Each data point is obtained by our averaging the expectation values of the observables over the last 100 iterations of the training. The reconstruction shows overall good agreement with the exact values computed with the MPS wave functions.

C. Reconstruction of a molecular wave function

We now move to the more general case of a wave function that is nonpositive or complex valued. To accommodate this different setup, we first need to modify the variational ansatz to allow for complex-valued amplitudes $\psi_{\boldsymbol{\theta}}(\boldsymbol{\sigma})$. One way to achieve this is to use the RBM to parametrize the probability distribution in the reference basis (as before) and couple it with an additional RBM that parametrizes the phases, $\psi_{\boldsymbol{\theta}\,\boldsymbol{\mu}}(\boldsymbol{\sigma}) = \sqrt{p_{\boldsymbol{\theta}}(\boldsymbol{\sigma})} e^{i \log p_{\boldsymbol{\mu}}(\sigma)}$ [71]. A different strategy, which we adopt here, is to cause the weights and biases to have complex values [50]. For the latter representation, we cannot interpret the RBM as a probabilistic graphical model anymore.

The presence of phases in the target wave function leads to an additional overhead in the measurement requirements. Projective measurements in the computational basis do not carry enough information to uniquely identify the state. To obtain information about the phases, measurements in additional bases are required. To account for this, we can write the training data set as $\mathcal{D} = \{\boldsymbol{x}_k\}$, where each single-shot measurement is given by $\boldsymbol{x} = (\boldsymbol{\tau}, \boldsymbol{\sigma})$, with $\boldsymbol{\tau}$ and $\boldsymbol{\sigma}$ referring, respectively, to the measurement basis and the binary measurement outcome. For example, for the case of Pauli measurements, the data point $\boldsymbol{x} = (\sigma_1^x = 0, \sigma_2^z = 1)$ refers to a measurement of qubit 1 (qubit 2) in the eigenbasis of the Pauli X (Pauli Z) operator.

The learning algorithm proceeds in a similar fashion to the case of a positive wave function, reducing to the optimization of the negative log-likelihood cost function:

$$\mathcal{C}(\boldsymbol{\theta}) = -\frac{1}{|\mathcal{D}|} \sum_{k=1}^{|\mathcal{D}|} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_k). \qquad (23)$$

The important difference is that to evaluate measurement probabilities in bases other than the reference one, we need to appropriately rotate the neural-network wave function. We can write this measurement probability as

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}_k) = \frac{|\langle \boldsymbol{\sigma}_k | \boldsymbol{U}(\boldsymbol{\tau}_k) | \psi_{\boldsymbol{\theta}} \rangle|^2}{\langle \psi_{\boldsymbol{\theta}} | \psi_{\boldsymbol{\theta}} \rangle}, \qquad (24)$$

where $\boldsymbol{U}(\boldsymbol{\tau})$ refers to the unitary transformation that rotates the reference basis into the measurement basis $\boldsymbol{\tau}$. We also assume local measurements, leading to the factorization $\boldsymbol{U}(\boldsymbol{\tau}) = \bigotimes_{j=1}^N U(\tau_j)$ into single-qubit unitary rotations $U(\tau_j)$. For the example described above, with $\boldsymbol{x} = (\sigma_1^x = 0, \sigma_2^z = 1)$, the neural-network probability is (up to a normalization factor) $p_{\boldsymbol{\theta}}(\boldsymbol{x}) \propto |\langle 01 | H_1 \otimes \mathbb{1}_2 | \psi_{\boldsymbol{\theta}} \rangle|^2$, where H is the Hadamard gate (i.e., the rotation into the σ^x basis). For a full derivation of a reconstruction with generic measurement bases, we refer the reader to Ref. [80].

We show neural-network QST of a nonpositive wave function for the case of the electronic ground state of a small molecule. The Hamiltonian in second quantization is given by

$$\hat{H} = \sum_{\alpha,\beta} t_{\alpha\beta}\hat{c}_\alpha^\dagger \hat{c}_\beta + \frac{1}{2}\sum_{\alpha,\beta,\gamma,\delta} u_{\alpha\beta\gamma\delta}\hat{c}_\alpha^\dagger \hat{c}_\beta^\dagger \hat{c}_\gamma \hat{c}_\delta, \tag{25}$$

where $\hat{c}^\dagger$ and $\hat{c}$ are fermionic creation and annihilation operators, and $t_{\alpha\beta}$ and $u_{\alpha\beta\gamma\delta}$ are electronic integrals. This Hamiltonian can be mapped into a qubit Hamiltonian using one of several mappings (i.e., Jordan-Wigner, Bravyi-Kitaev, etc.):

$$\hat{H} = \sum_k c_k \hat{P}_k, \tag{26}$$

where c_k are interaction coefficients and $\hat{P}_k$ are operators that belong to the N-qubit Pauli group.

We specifically look at the beryllium hydride molecule (BeH_2) in the STO-3G basis, mapped to $N = 6$ qubits [213]. To generate the training data, we first obtain the full ground-state wave function $|\Phi\rangle$ with exact diagonalization of the qubit Hamiltonian. Given the ground state, we can generate measurement data by sampling the full probability distribution obtained from the Born rule. A single measurement is obtained by our first selecting a measurement basis $\boldsymbol{\tau}$, then rotating the wave function accordingly $|\Phi_{\boldsymbol{\tau}}\rangle = \boldsymbol{U}(\boldsymbol{\tau})|\Phi\rangle$, and finally sampling the measurement outcome $\boldsymbol{\sigma} \sim P_{\boldsymbol{\tau}}(\boldsymbol{\sigma}) = |\langle\boldsymbol{\sigma}|\Psi_{\boldsymbol{\tau}}\rangle|^2$. We choose the measurement bases according to the Pauli operators $\hat{P}_k$ appearing in the Hamiltonian, each one being selected randomly among this set.

We show the results of the QST experiment in Fig. 5, where we plot the fidelity between the neural-network wave function and the target wave function during the training. The ability to faithfully learn this class of wave functions from limited measurement data has important implications for simulations of quantum chemistry with near-term hardware (e.g., using a variational quantum eigensolver approach [214]). This is due to the variational property of parametric quantum states (more details are given in Sec. V). Measuring the energy from a chemistry quantum simulation requires the independent measurement of a (typically) large number of noncommuting Pauli operators, which inevitably leads to a large variance in the energy estimators [213]. In contrast, by using the measurement data to train a RBM wave function instead, energy estimators with extremely low variance can be produced (provided the training is sufficiently accurate). This, however, introduces a systematic bias due to the training imperfection on limited-size datasets and the RBM approximation. The trade-off between variance and bias was studied in Ref. [80], which demonstrated a substantial decrease in measurement overhead for sufficiently pure quantum states.

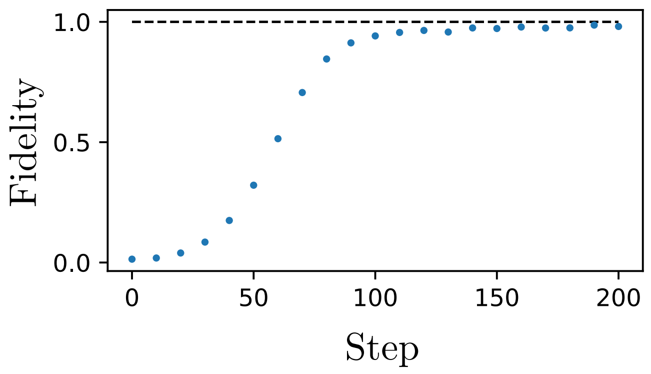

FIG. 5. Quantum state tomography of the beryllium hydride molecule. We show the fidelity $\mathcal{F} = |\langle\Phi|\psi_{\boldsymbol{\theta}}\rangle|^2$ between the neural-network wave function $\psi_{\boldsymbol{\theta}}$ and the exact molecular wave function Φ at each iteration during training.

D. Discussion

We have shown that for wave functions whose amplitudes can be gauged to be real and positive, QST is equivalent to unsupervised learning of projective measurement data in a single basis. For more general wave functions with a sign structure or complex-valued amplitudes, measurements in multiple bases are required to reconstruct the phases. These are processed by the neural network by appropriately rotating the parametrized wave function with a unitary $\boldsymbol{U}(\boldsymbol{\tau}) = \bigotimes_{j=1}^N U(\tau_j)$ composed of single-qubit basis rotations $U(\tau_j)$. In practice, this operation entails an exponential cost in the number of nontrivial rotations $U(\tau_j) \neq \mathbb{1}_j$, which means that only a small fraction of any IC set of bases can be use to train the neural network. Nevertheless, this reduced amount of information may be enough to reconstruct sufficiently structured quantum states. For example, a ground state of a local gapped Hamiltonian can be identified by the statistics of projective measurements in a set of bases corresponding to the decomposition of the Hamiltonian in the Pauli group.

An important assumption that was made is the purity of the quantum state under reconstruction, which is violated in any practical setting. There are cases where an approximate pure state reconstruction of an experimental quantum state may be justified, and could still provide valuable insights [77]. However, for benchmarking and noise characterization tasks, one needs to reconstruct the full density matrix. This can be achieved by introducing a neural-network parametrization of a density operator $\rho_{\boldsymbol{\theta}}(\boldsymbol{\sigma}, \boldsymbol{\sigma}')$, which is trained in an analogous way. The positivity of $\rho_{\boldsymbol{\theta}}$ can be enforced using a purification scheme, where $\rho_{\boldsymbol{\theta}}$ is purified by additional latent units in the neural network [73]. An iterative procedure to discover the dominant eigenstates of density operators using RBMs has also been proposed [81].

A different approach for reconstructing generic mixed quantum states consists in directly parametrizing the probability distribution $p(\boldsymbol{\alpha}) = \mathrm{Tr}\, \varrho \Pi_{\boldsymbol{\alpha}}$ of an IC POVM set $\{\Pi_{\boldsymbol{\alpha}}\}$ [75]. In contrast the purification scheme, this approach does not require unitary rotations, and thus removes the exponential complexity associated with processing data from arbitrary local bases. This, however, comes at the cost of possibly violating the positivity of the learned density matrix, since this cannot be enforced at the level of the POVM distribution parametrization.

Finally, in the examples shown above, we always know the exact target quantum state. This clearly facilitates the validation of the reconstruction. In a more general setting, where the state under reconstruction it not known, convergence in the reconstruction needs to be assessed heuristically using finite-size scaling experiments [79]. Once a given representative metric is chosen (e.g., the energy or correlation functions), its estimator should be evaluated by asymptotically increasing the number of parameters in the neural network, as well as the number of training samples.

V. VARIATIONAL GROUND-STATE OPTIMIZATION

The variational principle in quantum mechanics states that the expectation value of the Hamiltonian of a physical system of interest over any valid wave function is always greater than or equal to the ground-state energy of the system. This principle indicates that a strategy for finding an approximation to the ground-state energy of a system is to start from a parametrized wave function and vary its parameters until it yields the minimum possible energy. Here we specifically consider the VMC method [7], where the expectation value of the energy (used for optimizing the trial state) is evaluated by Monte Carlo sampling.

Historically, the choice of the wave function, which is critical to the success of the algorithm, has been motivated in close connection to a physical understanding of the problem, e.g., from approximate mean-field solutions supplemented with some form of additional correlations (such as a Jastrow factor [7]). More recently, motivated by their representation power, efficiency, and generality, neural networks have been explored as trial wave functions [50]. In particular, and as we explore below, RNNs are naturally well suited to the study of systems exhibiting strong correlations, such as those arising in the study of classical and quantum systems, which are prevalent in condensed matter physics and statistical physics [47,61,62,75]. Furthermore, the ability to draw unbiased samples from a RNN probability distribution greatly empowers VMC approaches, as it removes the need for sophisticated sampling algorithms to obtain sufficiently small autocorrelation times.

A. Recurrent neural networks

A RNN models a probability distribution $p(\boldsymbol{\sigma}) = p(\sigma_1, \ldots, \sigma_N)$ using a sequential structure according to the chain rule of probabilities:

$$p(\boldsymbol{\sigma}) = p(\sigma_1) p(\sigma_2 \mid \sigma_1) \prod_{j=3}^{N} p(\sigma_j \mid \boldsymbol{\sigma}_{-j}), \tag{27}$$

where $\boldsymbol{\sigma}_{-j} = (\sigma_1, , \ldots, \sigma_{j-2}, \sigma_{j-1})$. While specifying every conditional $p(\sigma_j | \boldsymbol{\sigma}_{-j})$ gives a full characterization of any possible distribution $p(\boldsymbol{\sigma})$, the computational resources to store and manipulate such a representation grow exponentially with system size N. To alleviate this problem, a recurrent neural network parametrizes the conditional probability distributions $p(\sigma_j \mid \boldsymbol{\sigma}_{-j})$ at any *time step* j. The elementary building block of a RNN is a recurrent cell ϑ [Fig. 6(a)]. In its simplest form, a "vanilla" recurrent cell is a nonlinear function that maps the direct sum (or concatenation) of an incoming hidden vector $\boldsymbol{h}_{j-1} \in \mathbb{R}^{n_h}$ [Fig. 6(a)] and an input $x_j = \sigma_{j-1}$ to an output hidden vector $\boldsymbol{h}_j$ such that

$$\boldsymbol{h}_j = f\left(W[\boldsymbol{h}_{j-1}; \sigma_{j-1}] + \boldsymbol{b}\right), \tag{28}$$

where f is a nonlinear activation function acting elementwise on the components of its argument and $[\boldsymbol{h}_{j-1}; \sigma_{j-1}]$ is the concatenation operation. The variational parameters of this simple RNN are given by the weight matrix $W \in \mathbb{R}^{n_h \times (n_h+2)}$, the bias vector $\boldsymbol{b} \in \mathbb{R}^{n_h}$, and the vectors $\boldsymbol{h}_0$ and σ_0 that initialize the recursion. The vector σ_{j-1} is a one-hot encoding of the input configuration such that, for example, $\sigma_{j-1} = (1, 0), (0, 1)$ for a configuration where the input is either 0 or 1, respectively. The goal of the internal state vector $\boldsymbol{h}_j$ is to encode the dependency of the conditional $p(\sigma_j \mid \boldsymbol{\sigma}_{-j})$ on all the previous variables $\boldsymbol{\sigma}_{-j}$. Thus, in our setting, the internal state vector provides the mechanism through which the RNN ansatz encodes correlations and entanglement in the quantum state, since, for example, setting $\boldsymbol{h}_j = 0$ results in a product state. As implied below, the dimension n_h of $\boldsymbol{h}_j$ determines the number of parameters in the RNN and its expressive power.

Instead of the vanilla RNN in Eq. (28), we specifically consider gated recurrent units (GRUs) [215]. The GRU was introduced to solve the vanishing and exploding gradient issues of RNNs encountered during the learning process, where the neural network's weights receive updates proportional to the partial derivative of the error function with respect to the current weight. In a vanilla RNN, the gradients can become vanishingly small or extremely large, effectively preventing the weight from changing or experiencing numerical overflow, respectively [216]. A schematic of a GRU is shown in [Fig. 6(b)]. Given a *visible* input state σ_{j-1} and a vector $\boldsymbol{h}_{j-1}$, the GRU at time step j processes them according to the sequence of operations and outputs an updated latent vector $\boldsymbol{h}_j$. The first two

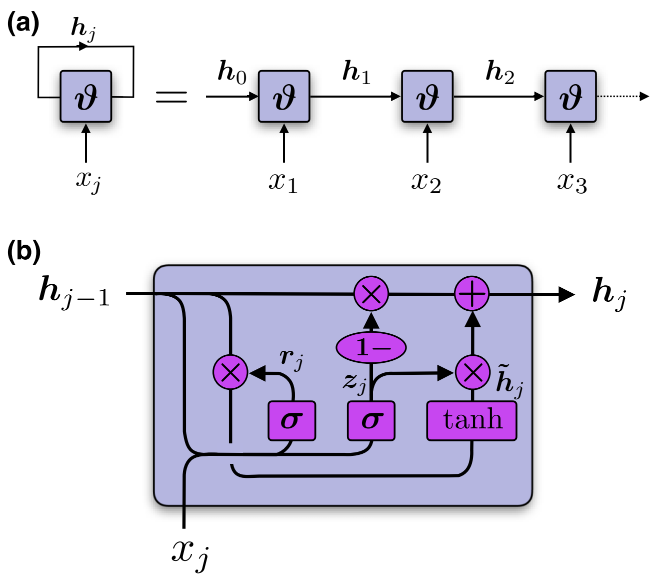

FIG. 6. Recurrent neural network. (a). A generic RNN cell (left) and its unrolling in time to process sequenced data (right). (b) The gated recurrent unit, and the set of operations used to process the input latent state $\boldsymbol{h}_{j-1}$ and visible state x_j to generate a new latent state $\boldsymbol{h}_j$. Lines joining together means concatenation, while the circles are element-wise operations.

operations are the so-called *update gate* and *reset gate*:

$$\boldsymbol{z}_j = \text{sig}\Big(\boldsymbol{W}_z[\boldsymbol{h}_{j-1}; \sigma_{j-1}] + \boldsymbol{b}_z\Big), \tag{29}$$

$$\boldsymbol{r}_j = \text{sig}\Big(\boldsymbol{W}_r[\boldsymbol{h}_{j-1}; \sigma_{j-1}] + \boldsymbol{b}_r\Big), \tag{30}$$

where $\text{sig}(x) = (1 + e^{-x})^{-1}$ is the sigmoid function, and $\boldsymbol{W}_z, \boldsymbol{W}_r \in \mathbb{R}^{n_h \times (n_h+2)}$, and $\boldsymbol{b}_z, \boldsymbol{b}_r \in \mathbb{R}^{n_h}$ are variational parameters. These gates are used to control how much information about previous time steps is kept encoded in the latent vector. Intuitively, if the reset gate is close to 0, the hidden state is forced to omit the information encoded in the previous hidden state and utilize the current input only. This effectively encourages a more compact and efficient utilization of the capacity of the RNN since the hidden state can drop irrelevant information as needed. On the other hand, the update gate controls how much information from the previous hidden state will carry over to the current hidden state.

Next, given the input visible state σ_j at the current time step, an internal latent state is created according to

$$\tilde{\boldsymbol{h}}_j = \tanh\Big(\tilde{\boldsymbol{W}}[\boldsymbol{r}_j \odot \boldsymbol{h}_{j-1}, \sigma_j] + \tilde{\boldsymbol{b}}\Big), \tag{31}$$

where $\boldsymbol{a} \odot \boldsymbol{b}$ is the element-wise multiplication, and $\tilde{\boldsymbol{W}} \in \mathbb{R}^{n_h \times (n_h+2)}$ and $\tilde{\boldsymbol{b}} \in \mathbb{R}^{n_h}$ are new sets of parameters. The new latent vector (output of the GRU) is generated as

$$\boldsymbol{h}_j = (\boldsymbol{1} - \boldsymbol{z}_j) \odot \boldsymbol{h}_{j-1} + \boldsymbol{z}_j \odot \tilde{\boldsymbol{h}}_j \tag{32}$$

and is sent to the GRU at time step $(j+1)$. Finally, the conditionals are computed using a softmax layer:

$$p(\sigma_j \mid \boldsymbol{\sigma}_{-j}) = S\left(U\boldsymbol{h}_j + \boldsymbol{c}\right), \tag{33}$$

where $U \in \mathbb{R}^{2\times n_h}$ and $\boldsymbol{c} \in \mathbb{R}^2$ are the variational parameters of the softmax layer.

Given the above parametrization of a probability distribution $p(\boldsymbol{\sigma})$, we can now promote RNNs to quantum mechanical wave functions $\psi(\boldsymbol{\sigma})$. As noted in the QST examples, we stress that *stoquastic* many-body Hamiltonians have ground states $|\Psi\rangle$ with strictly real and positive amplitudes in the standard computational basis [210]. This class of states can be represented in terms of probability distributions,

$$|\Psi\rangle = \sum_{\boldsymbol{\sigma}} \psi(\boldsymbol{\sigma})\,|\boldsymbol{\sigma}\rangle = \sum_{\boldsymbol{\sigma}} \sqrt{p(\boldsymbol{\sigma})}\,|\boldsymbol{\sigma}\rangle. \tag{34}$$

For such family of quantum states, which includes the ground states of the Rydberg system considered in this work, it is natural to try to approximate $p(\boldsymbol{\sigma})$ with a RNN.

B. Variational Monte Carlo simulation of Rydberg atoms

The goal of the VMC method is to iteratively optimize an ansatz wave function to approximate ground states of local Hamiltonians [7]. The VMC method uses a trial wave function $|\Psi_{\boldsymbol{\theta}}\rangle$ endowed with parameters $\boldsymbol{\theta}$. Here we consider a GRU-RNN wave function. Crucially, we exploit the fact that the RNN wave function allows efficient sampling from the square of the amplitudes of $|\Psi_{\boldsymbol{\theta}}\rangle$.

The VMC method iteratively optimizes the expectation value of the energy

$$E \equiv \frac{\langle \Psi_{\boldsymbol{\theta}}|\hat{H}|\Psi_{\boldsymbol{\theta}}\rangle}{\langle \Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle}. \tag{35}$$

The minimization is done using the gradient descent method or any variant of it. Since the RNN wave function is normalized such that $\langle \Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle = 1$, the expectation value in Eq. (35) can be written as

$$\begin{aligned} E = \langle \Psi_{\boldsymbol{\theta}}|\hat{H}|\Psi_{\boldsymbol{\theta}}\rangle &= \sum_{\boldsymbol{\sigma}} |\psi_{\boldsymbol{\theta}}(\boldsymbol{\sigma})|^2 \sum_{\boldsymbol{\sigma}'} H_{\boldsymbol{\sigma}\boldsymbol{\sigma}'} \frac{\psi_{\boldsymbol{\theta}}(\boldsymbol{\sigma}')}{\psi_{\boldsymbol{\theta}}(\boldsymbol{\sigma})} \\ &\equiv \sum_{\boldsymbol{\sigma}} |\psi_{\boldsymbol{\theta}}(\boldsymbol{\sigma})|^2 E_{\text{loc}}(\boldsymbol{\sigma}) \\ &\approx \frac{1}{N_S} \sum_{\boldsymbol{\sigma}\sim|\psi_{\boldsymbol{\theta}}(\boldsymbol{\sigma})|^2} E_{\text{loc}}(\boldsymbol{\sigma}), \end{aligned} \tag{36}$$

which represents a sample average of the local energy $E_{\text{loc}}(\boldsymbol{\sigma})$. The gradients $\partial_{\boldsymbol{\theta}} E$ can be similarly written as

$$\partial_{\boldsymbol{\theta}} E = \sum_{\boldsymbol{\sigma}} |\psi_{\boldsymbol{\theta}}(\boldsymbol{\sigma})|^2 \frac{\partial_{\boldsymbol{\theta}} \psi^*_{\boldsymbol{\theta}}(\boldsymbol{\sigma})}{\psi^*_{\boldsymbol{\theta}}(\boldsymbol{\sigma})} E_{\text{loc}}(\boldsymbol{\sigma}) + \text{c.c.} \tag{37}$$

An optimization step involves drawing N_S samples $\{\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)}, \ldots, \boldsymbol{\sigma}^{(N_S)}\}$ from $|\psi_{\boldsymbol{\theta}}(\boldsymbol{\sigma})|^2$, followed by an estimation of the energy gradients

$$\partial_{\boldsymbol{\theta}} E \approx \frac{2}{N_S} \text{Re} \left(\sum_{i=1}^{N_S} \frac{\partial_{\boldsymbol{\theta}} \psi^*_{\boldsymbol{\theta}}(\boldsymbol{\sigma}^{(i)})}{\psi^*_{\boldsymbol{\theta}}(\boldsymbol{\sigma}^{(i)})} E_{\text{loc}}(\boldsymbol{\sigma}^{(i)}) \right) \tag{38}$$

using automatic differentiation [171] and updating the parameters according to

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \partial_{\boldsymbol{\theta}} E \tag{39}$$

with a small learning rate α. Instead of this simple gradient descent rule, we can also use the Adam optimizer [172] to implement the parameter updates.

The stochastic evaluation of the gradients in Eq. (38) implies that these may exhibit high variance, which can potentially slow down the convergence of the algorithm. This problem can be alleviated through the introduction of a term in Eq. (38) that helps reduce the variance of the gradients [61]:

$$\begin{aligned} \partial_{\boldsymbol{\theta}} E &\approx \frac{2}{N_S} \text{Re} \left(\sum_{i=1}^{N_S} \frac{\partial_{\boldsymbol{\theta}} \psi^*_{\boldsymbol{\theta}}(\boldsymbol{\sigma}^{(i)})}{\psi^*_{\boldsymbol{\theta}}(\boldsymbol{\sigma}^{(i)})} \left(E_{\text{loc}}(\boldsymbol{\sigma}^{(i)}) - E \right) \right) \\ &= \frac{2}{N_S} \text{Re} \left(\sum_{i=1}^{N_S} \partial_{\boldsymbol{\theta}} \log \psi^*_{\boldsymbol{\theta}}(\boldsymbol{\sigma}^{(i)}) \left(E_{\text{loc}}(\boldsymbol{\sigma}^{(i)}) - E \right) \right). \end{aligned} \tag{40}$$

This estimator has improved variance, stabilizes the convergence of the algorithm, and is unbiased [61]. In the limit where $E_{\text{loc}}(\boldsymbol{\sigma}^{(i)}) \approx E$ near convergence, the variance of the gradients $\partial_{\boldsymbol{\theta}} E$ goes to zero as opposed to the nonzero variance of the gradients in Eq. (38).

1. Code walk-through

We provide an implementation of the VMC method using a RNN wave function based on the TensorFlow library. We first define all the relevant parameters of the Rydberg Hamiltonian and the RNN training.

```
# Hamiltonian parameters
Lx = 4       # linear size in x direction
Ly = 4       # linear size in y direction
N = Lx*Ly    # total number of atoms
V = 7.0      # Van der Waals interaction
Omega = 1.0 # Rabi frequency
delta = 1.0 # detuning

# RNN-VMC parameters
lr = 0.001      # learning rate
nh = 32         # number of hidden units
ns = 500        # number of samples
epochs = 1000   # training iterations
seed = 1234     # seed of RNG

# initialize the RNN wavefunction
vmc = VariationalMonteCarlo(
        Lx, Ly, V, Omega, delta,
        nh, lr, epochs, seed)
```

The RNN parameters are the learning rate for the optimizer, the number of hidden units in the GRU cell, the number of samples used to approximate the energy (and its gradients) at each training iteration, and the total number of epochs. The VMC module is then initialized accordingly.

We can now perform the VMC simulation by training the RNN parameters. For a given number of epochs, we first sample the RNN distribution to generate `ns` samples. These can be used to evaluate the cost function of the optimization and its gradients, computed here using the AD functionalities from TensorFlow. The parameters are then updated according to the gradients.

```
# training loop
for n in range(epochs):
    # sample the RNN wavefunction
    samples, _ = vmc.sample(ns)

    # evaluate the loss function in AD mode
    with tf.GradientTape() as tape:
        logpsi = vmc.logpsi(samples)
        eloc = vmc.localenergy(samples, logpsi)
        Eo = tf.stop_gradient(tf.reduce_mean(
    eloc))

        loss = tf.reduce_mean(2.0*tf.multiply(
    logpsi, tf.stop_gradient(eloc)) - 2.0*Eo*
    logpsi)

    # compute the gradients
    gradients = tape.gradient(loss, vmc.
    trainable_variables)

    # update the parameters
    vmc.optimizer.apply_gradients(zip(gradients,
     vmc.trainable_variables))

    # get average energy
    avg_E = np.mean(eloc.numpy())
```

We show in Fig. 7 the results of VMC simulations for an 8×8 array of Rydberg atoms. In Fig. 7(a), we plot the average total energy at each training iteration for a RNN wave function with a GRU containing $n_h = 25$ and $n_h = 100$ hidden units. As expected, increasing the number

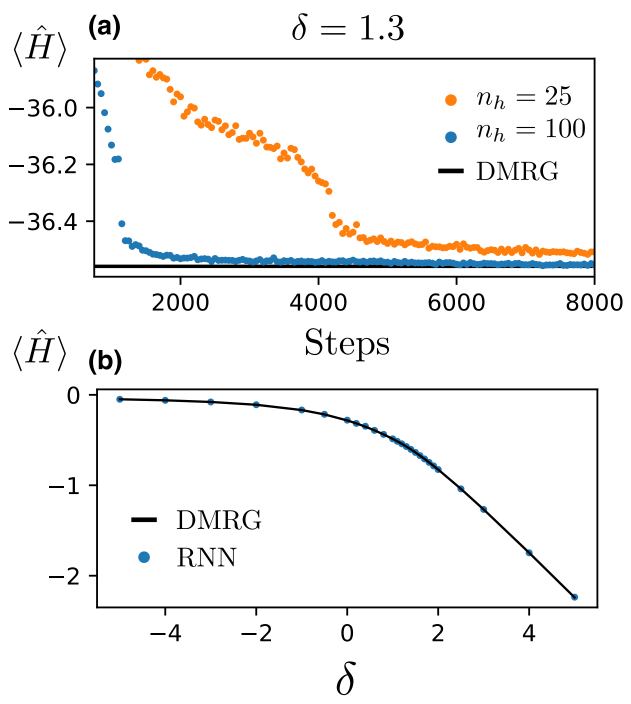

FIG. 7. Variational Monte Carlo simulation of the Rydberg Hamiltonian. (a) Average energy as a function of the training step near the phase transition at $\delta = 1.3$. We show RNN wave functions with $n_h = 25$ and $n_h = 100$ hidden units, compared with the result obtained from DMRG. The first few VMC iterations are not plotted to increase readability of the convergence transient. (b) Average energy over the full phase diagram for a RNN wave function with $n_h = 100$, in comparison with DMRG.

of hidden units in the RNN leads to increased accuracy in the training. In Fig. 7(b), we plot the average energy after convergence as a function of the detuning using $n_h = 100$, and compare it with the values obtained from DMRG.

VI. FREQUENTLY ASKED QUESTIONS

In this section, we expand on a brief set of questions typically encountered during the training and evaluation of machine learning models in the context of quantum physics.

A. What neural-network architecture should I use?

The architecture and characteristics of the neural network depend directly on the features of the problem at hand, such as whether the degrees of freedom represented are subject to geometric constraints or are constrained by a set of symmetries. For instance, a CNN is naturally well suited in capturing two-dimensional systems with translational invariance [15]. Several other symmetries have also been implemented in neural-network representations of wave functions, including U(1) charge conservation [89], SU(2) spin symmetry [217], and gauge symmetries [218,219]. Similarly, one may choose to adopt an autoregressive neural network (such as the RNN) for problems where sampling may be a bottleneck in the algorithm, as it often is in VMC simulation of strongly interacting quantum matter and spin glasses [47,61,220].

B. How do I certify convergence in the training?

There is no general answer to this question, but there are some guidelines that one should follow when implementing data-driven algorithms for quantum problems. There are usually two complexity scales involved in training a neural-network model: the representational power of the model (i.e., the number of network parameters) and the sample complexity [i.e., how much data one needs to learn the underlying features of a state with small error (for VMC simulations, one can interpret the number of Monte Carlo samples per gradient step as data)]. The assessment of convergence is problem specific, and depends on whether there are tractable estimators to probe the convergence. For VMC simulation, for example, the value of the energy is a natural estimator of convergence, and energy variance can be use to estimate how close the parametric wave function is to an energy eigenstate, where zero variance indicates that the algorithm has found an exact eigenstate.

In contrast, in QST tasks the optimization objective (e.g., the Kullback-Leibler divergence) is not accessible when using models with intractable densities (i.e., the RBM). In this scenario, one should instead monitor other observables that are relevant to the problem at hand, such as local densities or magnetizations, or various types of correlation functions. A direct comparison between the model prediction and the measurement data can be obtained using observables that are diagonal in the measurement basis. For both of the above, convergence can be assessed by performing a scaling analysis of a relevant observables of choice against both the number of measurements and the number of network parameters [79].

VII. CONCLUSIONS

In this article, we presented applications of machine learning algorithms based on neural networks to quantum many-body physics. We focused on numerical demonstrations and provided hands-on code tutorials based on open-source software [154,173,211] with the goal of facilitating learning for researchers new to the field and accelerating the adoption of machine learning in quantum physics.

We showcased distinct machine learning paradigms, implemented with different neural-network architectures. As a test bed for the numerical experiments, we chose a system of interacting Rydberg atoms arranged in a two-dimensional square array. First, we demonstrated supervised learning of atomic occupation data, which were generated from ground states of the Rydberg Hamiltonian obtained using the density matrix renormalization group. Using a convolutional neural network trained on labeled data, we showed how to learn the quantum phase transition between a disordered phase and the antiferromagnetically ordered phase.

We presented unsupervised learning of unlabeled data using the restricted Boltzmann machine. We showed that for pure quantum states with real and positive amplitude (e.g., the Rydberg ground states), this procedure is equivalent to quantum state tomography based on a neural-network representation of a quantum state. Using atomic occupation data, we trained Boltzmann machines to learn the ground states of the Rydberg Hamiltonian. We also described a simple extension of this approach to learn quantum states with a sign structure, and showed a demonstration in the context of simulation of chemistry with quantum computers, where we learned the ground state of a molecule using qubit measurements.

The final application we explored is the Monte Carlo optimization of a variational wave function to estimate the ground state of a many-body Hamiltonian. We parametrized a wave function using a recurrent neural network and trained its parameters to lower both the expectation value and the variance of the Rydberg Hamiltonian.

It is becoming increasingly evident that machine learning is full of thrilling opportunities and conceptual advances with great potential to energize computational and experimental physics. As these notions continue to spread through the research landscape of strongly correlated quantum matter and quantum information science, we hope this tutorial will provide a useful first step into the expanding domain of artificial intelligence for the study of quantum many-body systems.

ACKNOWLEDGMENTS

The numerical simulations were performed at the Simons Foundation Super-Computing Center using the following software libraries: ITensor [154] and PastaQ [221] (for the simulation of the Rydberg atoms and data generation), TensorFlow [173] (for supervised learning and variational Monte Carlo simulations), and NetKet [211] (for quantum tomography). The Flatiron Institute is supported by the Simons Foundation. J.C. acknowledges support from the Natural Sciences and Engineering Research Council of Canada, the Shared Hierarchical Academic Research Computing Network, Compute Canada, the Google Quantum Research Award program, and the CIFAR AI Chairs program.

[1] Wen Xiao Gang, *Quantum Field Theory of Many-Body Systems: from the Origin of Sound to an Origin of Light and Electrons* (Oxford University Press, Oxford, 2007).
[2] David Ceperley and Berni Alder, Quantum monte carlo, Science **231**, 555 (1986).
[3] H. G. Evertz, The loop algorithm, Adv. Phys. **52**, 1 (2003).
[4] Anders W. Sandvik, Stochastic series expansion method with operator-loop update, Phys. Rev. B **59**, R14157 (1999).
[5] Steven R. White, Density Matrix Formulation for Quantum Renormalization Groups, Phys. Rev. Lett. **69**, 2863 (1992).
[6] Steven R. White, Density-matrix algorithms for quantum renormalization groups, Phys. Rev. B **48**, 10345 (1993).
[7] Federico Becca and Sandro Sorella, *Quantum Monte Carlo Approaches for Correlated Systems* (Cambridge University Press, 2017).
[8] Frank Arute *et al.*, Observation of separated dynamics of charge and spin in the Fermi-Hubbard model, arXiv:2010.07965 (2020).
[9] Sepehr Ebadi *et al.*, Quantum Phases of Matter on a 256-Atom Programmable Quantum Simulator, arXiv:2012.12281 (2020).
[10] Giulia Semeghini *et al.*, Probing Topological Spin Liquids on a Programmable Quantum Simulator, arXiv:2104.04119 (2021).
[11] K. J. Satzinger *et al.*, Realizing topologically ordered states on a quantum processor, arXiv:2104.01180 (2021).
[12] Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, Maria Schuld, Naftali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborová, Machine learning and the physical sciences, Rev. Mod. Phys. **91**, 045002 (2019).
[13] Juan Carrasquilla, Machine learning for quantum matter, Adv. Phys.: X **5**, 1797528 (2020).
[14] Giacomo Torlai and Roger G. Melko, Machine-learning quantum states in the nisq era, Annu. Rev. Condens. Matter Phys. **11**, 325 (2020).
[15] Juan Carrasquilla and Roger G. Melko, Machine learning phases of matter, Nat. Phys. **13**, 431 (2017).
[16] Evert P. L. van Nieuwenburg, Ye-Hua Liu, and Sebastian D. Huber, Learning phase transitions by confusion, Nat. Phys. **13**, 435 (2017).
[17] Giacomo Torlai and Roger G. Melko, Learning thermodynamics with boltzmann machines, Phys. Rev. B **94**, 165134 (2016).
[18] Lei Wang, Discovering phase transitions with unsupervised learning, Phys. Rev. B **94**, 195105 (2016).
[19] Kelvin Ch'ng, Juan Carrasquilla, Roger G. Melko, and Ehsan Khatami, Machine Learning Phases of Strongly Correlated Fermions, Phys. Rev. X **7**, 031038 (2017).
[20] Peter Broecker, Juan Carrasquilla, Roger G. Melko, and Simon Trebst, Machine learning quantum phases of matter beyond the fermion sign problem, Sci. Rep. **7**, 8823 (2017).
[21] Yi Zhang and Eun-Ah Kim, Quantum Loop Topography for Machine Learning, Phys. Rev. Lett. **118**, 216401 (2017).
[22] Dong-Ling Deng, Xiaopeng Li, and S. Das Sarma, Machine learning topological states, Phys. Rev. B **96**, 195145 (2017).
[23] Frank Schindler, Nicolas Regnault, and Titus Neupert, Probing many-body localization with neural networks, Phys. Rev. B **95**, 245134 (2017).

[24] Yi-Ting Hsu, Xiao Li Dong-Ling Deng, and S. Das Sarma, Machine Learning Many-Body Localization: Search for the Elusive Nonergodic Metal, Phys. Rev. Lett. **121**, 245701 (2018).
[25] Patrick Huembeli, Alexandre Dauphin, and Peter Wittek, Identifying quantum phase transitions with adversarial neural networks, Phys. Rev. B **97**, 134109 (2018).
[26] Jonas Greitemann, Ke Liu, and Lode Pollet, Probing hidden spin order with interpretable machine learning, Phys. Rev. B **99**, 060404 (2019).
[27] Ke Liu, Jonas Greitemann, and Lode Pollet, Learning multiple order parameters with interpretable machines, Phys. Rev. B **99**, 104410 (2019).
[28] Yi Zhang, A. Mesaros, K. Fujita, S. D. Edkins, M. H. Hamidian, K. Ch'ng, H. Eisaki, S. Uchida, J. C. Séamus Davis, Ehsan Khatami, and Eun-Ah Kim, Machine learning in electronic-quantum-matter imaging experiments, Nature **570**, 484 (2019).
[29] Annabelle Bohrdt, Christie S. Chiu, Geoffrey Ji, Muqing Xu, Daniel Greif, Markus Greiner, Eugene Demler, Fabian Grusdt, and Michael Knap, Classifying Snapshots of the Doped Hubbard Model with Machine Learning, arXiv:1811.12425 (2018).
[30] Benno S. Rem, Niklas Käming, Matthias Tarnowski, Luca Asteria, Nick Fläschner, Christoph Becker, Klaus Sengstock, and Christof Weitenberg, Identifying Quantum Phase Transitions using Artificial Neural Networks on Experimental Data, arXiv:1809.05519 (2018).
[31] Wenqian Lian, Sheng-Tao Wang, Sirui Lu Yuanyuan Huang, Fei Wang, Xinxing Yuan, Wengang Zhang, Xiaolong Ouyang, Xin Wang, Xianzhi Huang, Li He Xiuying Chang, Dong-Ling Deng, and Luming Duan, Machine Learning Topological Phases with a Solid-State Quantum Simulator, Phys. Rev. Lett. **122**, 210503 (2019).
[32] Fulvio Flamini, Nicolò Spagnolo, and Fabio Sciarrino, Visual assessment of multi-photon interference, Quantum Sci. Technol. **4**, 024008 (2019).
[33] Taira Giordani, Alessia Suprano, Emanuele Polino, Francesca Acanfora, Luca Innocenti, Alessandro Ferraro, Mauro Paternostro, Nicolò Spagnolo, and Fabio Sciarrino, Machine Learning-Based Classification of Vector Vortex Beams, Phys. Rev. Lett. **124**, 160401 (2020).
[34] Frank Schäfer and Niels Lörch, Vector field divergence of predictive model output as indication of phase transitions, Phys. Rev. E **99**, 062107 (2019).
[35] Eliska Greplova, Agnes Valenti, Gregor Boschung, Frank Schafer, Niels Lorch, and Sebastian D. Huber, Unsupervised identification of topological phase transitions using predictive models, New J. Phys. **22**, 045003 (2020).
[36] Julian Arnold, Frank Schäfer, Martin Žonda, and Axel U. J. Lode, Interpretable and unsupervised phase classification, arXiv:2010.04730 (2020).
[37] Li Huang and Lei Wang, Accelerated monte carlo simulations with restricted Boltzmann machines, Phys. Rev. B **95**, 035105 (2017).
[38] Junwei Liu, Yang Qi, Zi Yang Meng, and Liang Fu, Self-learning monte carlo method, Phys. Rev. B **95**, 041101 (2017).
[39] Xiao Yan Xu, Yang Qi, Junwei Liu, Liang Fu, and Zi Yang Meng, Self-learning quantum monte carlo method in interacting fermion systems, Phys. Rev. B **96**, 041119 (2017).
[40] E. M. Inack, G. E. Santoro, L. Dell'Anna, and S. Pilati, Projective quantum monte carlo simulations guided by unrestricted neural network states, Phys. Rev. B **98**, 235145 (2018).
[41] T. Parolini, E. M. Inack, G. Giudici, and S. Pilati, Tunneling in projective quantum monte carlo simulations with guiding wave functions, Phys. Rev. B **100**, 214303 (2019).
[42] S. Pilati, E. M. Inack, and P. Pieri, Self-learning projective quantum monte carlo simulations guided by restricted Boltzmann machines, Phys. Rev. E **100**, 043301 (2019).
[43] B. McNaughton, M. V. Milošević, A. Perali, and S. Pilati, Boosting monte carlo simulations of spin glasses using autoregressive neural networks, Phys. Rev. E **101**, 053312 (2020).
[44] M. S. Albergo, G. Kanwar, and P. E. Shanahan, Flow-based generative models for markov chain monte carlo in lattice field theory, Phys. Rev. D **100**, 034515 (2019).
[45] Kim A. Nicoli, Shinichi Nakajima, Nils Strodthoff, Wojciech Samek, Klaus-Robert Müller, and Pan Kessel, Asymptotically unbiased estimation of physical observables with neural samplers, Phys. Rev. E **101**, 023304 (2020).
[46] Kim A. Nicoli, Christopher J. Anders, Lena Funcke, Tobias Hartung, Karl Jansen, Pan Kessel, Shinichi Nakajima, and Paolo Stornati, Estimation of Thermodynamic Observables in Lattice Field Theories with Deep Generative Models, Phys. Rev. Lett. **126**, 032001 (2021).
[47] Dian Wu, Lei Wang, and Pan Zhang, Solving Statistical Mechanics Using Variational Autoregressive Networks, Phys. Rev. Lett. **122**, 080602 (2019).
[48] J. Androsiuk, L. Kulak, and K. Sienicki, Neural network solution of the schroedinger equation for a two-dimensional harmonic oscillator, Chem. Phys. **173**, 377 (1993).
[49] I. E. Lagaris, A. Likas, and D. I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Networks **9**, 987 (1998).
[50] Giuseppe Carleo and Matthias Troyer, Solving the quantum many-body problem with artificial neural networks, Science **355**, 602 (2017).
[51] Zi Cai and Jinguo Liu, Approximating quantum many-body wave functions using artificial neural networks, Phys. Rev. B **97**, 035116 (2018).
[52] Di Luo and Bryan K. Clark, Backflow Transformations via Neural Networks for Quantum Many-Body Wave Functions, Phys. Rev. Lett. **122**, 226401 (2019).
[53] David Pfau, James S. Spencer, Alexander G. D. G. Matthews, and W. M. C. Foulkes, Ab initio solution of the many-electron schrödinger equation with deep neural networks, Phys. Rev. Res. **2**, 033429 (2020).
[54] Jan Hermann, Zeno Schatzle, and Frank Noe, Deep-neural-network solution of the electronic Schrödinger equation, Nat. Chem. **12**, 891 (2020).
[55] Michael J. Hartmann and Giuseppe Carleo, Neural-Network Approach to Dissipative Quantum Many-Body Dynamics, Phys. Rev. Lett. **122**, 250502 (2019).
[56] Alexandra Nagy and Vincenzo Savona, Variational Quantum Monte Carlo Method with a Neural-Network Ansatz

for Open Quantum Systems, Phys. Rev. Lett. **122**, 250501 (2019).
[57] Filippo Vicentini, Alberto Biella, Nicolas Regnault, and Cristiano Ciuti, Variational Neural-Network Ansatz for Steady States in Open Quantum Systems, Phys. Rev. Lett. **122**, 250503 (2019).
[58] Nobuyuki Yoshioka and Ryusuke Hamazaki, Constructing neural stationary states for open quantum many-body systems, Phys. Rev. B **99**, 214306 (2019).
[59] Kenny Choo, Antonio Mezzacapo, and Giuseppe Carleo, Fermionic neural-network states for ab-initio electronic structure, Nat. Commun. **11**, 2368 (2020).
[60] Or Sharir, Yoav Levine, Noam Wies, Giuseppe Carleo, and Amnon Shashua, Deep Autoregressive Models for the Efficient Variational Simulation of Many-Body Quantum Systems, Phys. Rev. Lett. **124**, 020503 (2020).
[61] Mohamed Hibat-Allah, Martin Ganahl, Lauren E. Hayward, Roger G. Melko, and Juan Carrasquilla, Recurrent neural network wave functions, Phys. Rev. Res. **2**, 023358 (2020).
[62] Christopher Roth, Iterative Retraining of Quantum Spin Models Using Recurrent Neural Networks, arXiv:2003.06228 (2020).
[63] Agnes Valenti, Eliska Greplova, Netanel H. Lindner, and Sebastian D. Huber, Correlation-Enhanced Neural Networks as Interpretable Variational Quantum States, arXiv:2103.05017 (2021).
[64] Marin Bukov, Alexandre G. R. Day, Dries Sels, Phillip Weinberg, Anatoli Polkovnikov, and Pankaj Mehta, Reinforcement Learning in Different Phases of Quantum Control, Phys. Rev. X **8**, 031086 (2018).
[65] Thomas Fösel, Petru Tighineanu, Talitha Weiss, and Florian Marquardt, Reinforcement Learning with Neural Networks for Quantum Feedback, Phys. Rev. X **8**, 031084 (2018).
[66] Alexandre G. R. Day, Marin Bukov, Phillip Weinberg, Pankaj Mehta, and Dries Sels, Glassy Phase of Optimal Quantum Control, Phys. Rev. Lett. **122**, 020601 (2019).
[67] Murphy Yuezhen Niu, Sergio Boixo, Vadim N. Smelyanskiy, and Hartmut Neven, Universal quantum control through deep reinforcement learning, npj Quantum Inf. **5**, 1 (2019).
[68] Jiahao Yao, Lin Lin, and Marin Bukov, Reinforcement Learning for Many-Body Ground State Preparation based on Counter-Diabatic Driving, arXiv:2010.03655 (2020).
[69] Luuk Coopmans, Di Luo, Graham Kells, Bryan K. Clark, and Juan Carrasquilla, Protocol Discovery for the Quantum Control of Majoranas by Differential Programming and Natural Evolution Strategies, arXiv:2008.09128 [cond-mat, physics:physics, physics:quant-ph] (2020).
[70] Frank Schafer, Michal Kloc, Christoph Bruder, and Niels Lorch, A differentiable programming method for quantum control, Machine Learning: Sci. Technol. **1**, 035009 (2020).
[71] Giacomo Torlai, Guglielmo Mazzola, Juan Carrasquilla, Matthias Troyer, Roger Melko, and Giuseppe Carleo, Neural-network quantum state tomography, Nat. Phys. **14**, 447 (2018).
[72] Andrea Rocchetto, Edward Grant, Sergii Strelchuk, Giuseppe Carleo, and Simone Severini, Learning hard quantum distributions with variational autoencoders, npj Quantum Inf. **4**, 28 (2018).
[73] Giacomo Torlai and Roger G. Melko, Latent Space Purification via Neural Density Operators, Phys. Rev. Lett. **120**, 240503 (2018).
[74] Yihui Quek, Stanislav Fort, and Hui Khoon Ng, Adaptive Quantum State Tomography with Neural Networks, arXiv:1812.06693 (2018).
[75] Juan Carrasquilla, Giacomo Torlai, Roger G. Melko, and Leandro Aolita, Reconstructing quantum states with generative models, Nat. Machine Intelligence **1**, 155 (2019).
[76] Adriano Macarone Palmieri, Egor Kovlakov, Federico Bianchi, Dmitry Yudin, Stanislav Straupe, Jacob D. Biamonte, and Sergei Kulik, Experimental neural network enhanced quantum tomography, npj Quantum Inf. **6**, 20 (2020).
[77] Giacomo Torlai, Brian Timar, Evert P. L. van Nieuwenburg, Harry Levine, Ahmed Omran, Alexander Keesling, Hannes Bernien, Markus Greiner, Vladan Vuletić, Mikhail D. Lukin, Roger G. Melko, and Manuel Endres, Integrating Neural Networks with a Quantum Simulator for State Reconstruction, Phys. Rev. Lett. **123**, 230504 (2019).
[78] Tao Xin, Sirui Lu, Ningping Cao, Galit Anikeeva, Dawei Lu, Jun Li, Guilu Long, and Bei Zeng, Local-measurement-based quantum state tomography via neural networks, npj Quantum Inf. **5**, 109 (2019).
[79] Dan Sehayek, Anna Golubeva, Michael S. Albergo, Bohdan Kulchytskyy, Giacomo Torlai, and Roger G. Melko, Learnability scaling of quantum states: Restricted Boltzmann machines, Phys. Rev. B **100**, 195125 (2019).
[80] Giacomo Torlai, Guglielmo Mazzola, Giuseppe Carleo, and Antonio Mezzacapo, Precise measurement of quantum observables with neural-network estimators, Phys. Rev. Res. **2**, 022060 (2020).
[81] Abhijeet Melkani, Clemens Gneiting, and Franco Nori, Eigenstate extraction with neural-network tomography, Phys. Rev. A **102**, 022412 (2020).
[82] Shahnawaz Ahmed, Carlos Sánchez Muñoz, Franco Nori, and Anton Frisk Kockum, Quantum State Tomography with Conditional Generative Adversarial Networks, arXiv:2008.03240 (2020).
[83] E. S. Tiunov, V. V. Tiunova (Vyborova), A. E. Ulanov, A. I. Lvovsky, and A. K. Fedorov, Experimental quantum homodyne tomography via machine learning, Optica **7**, 448 (2020).
[84] Peter Cha, Paul Ginsparg, Felix Wu, Juan Carrasquilla, Peter L. McMahon, and Eun-Ah Kim, Attention-based Quantum Tomography, arXiv:2006.12469 (2020).
[85] Marcel Neugebauer, Laurin Fischer, Alexander Jäger, Stefanie Czischek, Selim Jochim, Matthias Weidemüller, and Martin Gärttner, Neural-network quantum state tomography in a two-qubit experiment, Phys. Rev. A **102**, 042604 (2020).
[86] Isaac J. S. De Vlugt, Dmitri Iouchtchenko, Ejaaz Merali, Pierre-Nicholas Roy, and Roger G. Melko, Reconstructing quantum molecular rotor ground states, Phys. Rev. B **102**, 035108 (2020).
[87] Alistair W. R. Smith, Johnnie Gray, and M. S. Kim, Efficient Approximate Quantum State Tomography with Basis Dependent Neural-Networks, arXiv:2009.07601 (2020).

[88] Giacomo Torlai, Christopher J. Wood, Atithi Acharya, Giuseppe Carleo, Juan Carrasquilla, and Leandro Aolita, Quantum process tomography with unsupervised learning and tensor networks, arXiv:2006.02424 (2020).
[89] Stewart Morawetz, Isaac J. S. De Vlugt, Juan Carrasquilla, and Roger G. Melko, U(1) symmetric recurrent neural networks for quantum state reconstruction, arXiv:2010.14514 (2020).
[90] Shahnawaz Ahmed, Carlos Sánchez Munoz, Franco Nori, and Anton Frisk Kockum, Classification and reconstruction of optical quantum states with deep neural networks, arXiv:2012.02185 (2020).
[91] John C. Snyder, Matthias Rupp, Katja Hansen, Klaus-Robert Müller, and Kieron Burke, Finding Density Functionals with Machine Learning, Phys. Rev. Lett. **108,** 253002 (2012).
[92] John C. Snyder, Matthias Rupp, Katja Hansen, Leo Blooston, Klaus-Robert Müller, and Kieron Burke, Orbital-free bond breaking via machine learning, J. Chem. Phys. **139,** 224104 (2013).
[93] Li Li, Thomas E. Baker, Steven R. White, and Kieron Burke, Pure density functional for strong correlation and the thermodynamic limit from machine learning, Phys. Rev. B **94,** 245129 (2016).
[94] Felix Brockherde, Leslie Vogt, Li Li, Mark E. Tuckerman, Kieron Burke, and Klaus-Robert Müller, Bypassing the kohn-sham equations with machine learning, Nat. Commun. **8,** 872 (2017).
[95] Kevin Ryczko, David A. Strubbe, and Isaac Tamblyn, Deep learning and density-functional theory, Phys. Rev. A **100,** 022512 (2019).
[96] Javier Robledo Moreno, Giuseppe Carleo, and Antoine Georges, Deep Learning the Hohenberg-Kohn Maps of Density Functional Theory, Phys. Rev. Lett. **125,** 076402 (2020).
[97] M. Michael Denner, Mark H. Fischer, and Titus Neupert, Efficient learning of a one-dimensional density functional theory, Phys. Rev. Res. **2,** 033388 (2020).
[98] Pankaj Mehta and David J. Schwab, An exact mapping between the Variational Renormalization Group and Deep Learning, arXiv:1410.3831 (2014).
[99] Maciej Koch-Janusz and Zohar Ringel, Mutual information, neural networks and the renormalization group, Nat. Phys. **14,** 578 (2018).
[100] Satoshi Iso, Shotaro Shiba, and Sumito Yokoo, Scale-invariant feature extraction of neural network and renormalization group flow, Phys. Rev. E **97,** 053304 (2018).
[101] Shuo-Hui Li and Lei Wang, Neural Network Renormalization Group, Phys. Rev. Lett. **121,** 260601 (2018).
[102] Hong-Ye Hu, Shuo-Hui Li, Lei Wang, and Yi-Zhuang You, Machine learning holographic mapping by neural network renormalization group, Phys. Rev. Res. **2,** 023369 (2020).
[103] Jui-Hui Chung and Ying-Jer Kao, Neural Monte Carlo Renormalization Group, arXiv:2010.05703 (2020).
[104] Giacomo Torlai and Roger G. Melko, Neural Decoder for Topological Codes, Phys. Rev. Lett. **119,** 030501 (2017).
[105] Stefan Krastanov and Liang Jiang, Deep neural network probabilistic decoder for stabilizer codes, Sci. Rep. **7,** 11003 (2017).
[106] Savvas Varsamopoulos, Ben Criger, and Koen Bertels, Decoding small surface codes with feedforward neural networks, Quantum Sci. Technol. **3,** 015004 (2017).
[107] Paul Baireuther, Thomas E. O'Brien, Brian Tarasinski, and Carlo W. J. Beenakker, Machine-learning-assisted correction of correlated qubit errors in a topological code, Quantum **2,** 48 (2018).
[108] Christopher Chamberland and Pooya Ronagh, Deep neural decoders for near term fault-tolerant experiments, Quantum Sci. Technol. **3,** 044002 (2018).
[109] Nikolas P. Breuckmann and Xiaotong Ni, Scalable neural network decoders for higher dimensional quantum codes, Quantum **2,** 68 (2018).
[110] Hendrik Poulsen Nautrup, Nicolas Delfosse, Vedran Dunjko, Hans J. Briegel, and Nicolai Friis, Optimizing quantum error correction codes with reinforcement learning, Quantum **3,** 215 (2019).
[111] Ye-Hua Liu and David Poulin, Neural Belief-Propagation Decoders for Quantum Error-Correcting Codes, Phys. Rev. Lett. **122,** 200501 (2019).
[112] Nishad Maskara, Aleksander Kubica, and Tomas Jochym-O'Connor, Advantages of versatile neural-network decoding for topological codes, Phys. Rev. A **99,** 052351 (2019).
[113] Agnes Valenti, Evert van Nieuwenburg, Sebastian Huber, and Eliska Greplova, Hamiltonian learning for quantum error correction, Phys. Rev. Res. **1,** 033092 (2019).
[114] Philip Andreasson, Joel Johansson, Simon Liljestrand, and Mats Granath, Quantum error correction for the toric code using deep reinforcement learning, Quantum **3,** 183 (2019).
[115] Ryan Sweke, Markus Kesselring, Evert van Nieuwenburg, and J. Eisert, Reinforcement learning decoders for fault-tolerant quantum computation, Machine Learning: Sci. Technol. **2,** (2020).
[116] Xiaotong Ni, Neural network decoders for large-distance 2D toric codes, Quantum **4,** 310 (2020).
[117] Amarsanaa Davaasuren, Yasunari Suzuki, Keisuke Fujii, and Masato Koashi, General framework for constructing fast and near-optimal machine-learning-based decoder of the topological stabilizer codes, Phys. Rev. Res. **2,** 033399 (2020).
[118] David Fitzek, Mattias Eliasson, Anton Frisk Kockum, and Mats Granath, Deep q-learning decoder for depolarizing noise on the toric code, Phys. Rev. Res. **2,** 023230 (2020).
[119] Yi-Zhuang You, Zhao Yang, and Xiao-Liang Qi, Machine learning spatial geometry from entanglement features, Phys. Rev. B **97,** 045153 (2018).
[120] Alireza Seif, Kevin A. Landsman, Norbert M. Linke, Caroline Figgatt, C. Monroe, and Mohammad Hafezi, Machine learning assisted readout of trapped-ion qubits, J. Phys. B: At., Mol. Opt. Phys. **51,** 174006 (2018).
[121] Alexey A. Melnikov, Hendrik Poulsen Nautrup, Mario Krenn, Vedran Dunjko, Markus Tiersch, Anton Zeilinger, and Hans J. Briegel, Active learning machine learns to create new quantum experiments, Proc. Natl. Acad. Sci. **115,** 1221 (2018).
[122] Vedran Dunjko and Hans J. Briegel, Machine learning & artificial intelligence in the quantum domain: A review of recent progress, Rep. Prog. Phys. **81,** 074001 (2018).

[123] Kai-Wen Zhao, Wen-Han Kao, and Kai-Hsin Wu, and Ying-Jer Kao, Generation of ice states through deep reinforcement learning, Phys. Rev. E **99**, 062106 (2019).

[124] Juan Carrasquilla, Di Luo, Felipe Pérez, Ashley Milsted, Bryan K. Clark, Maksims Volkovs, and Leandro Aolita, Probabilistic Simulation of Quantum Circuits with the Transformer, arXiv:1912.11052 (2019).

[125] Stavros Efthymiou, Matthew J. S. Beach, and Roger G. Melko, Super-resolving the Ising model with convolutional neural networks, Phys. Rev. B **99**, 075113 (2019).

[126] Raban Iten, Tony Metger, Henrik Wilming, Lídia del Rio, and Renato Renner, Discovering Physical Concepts with Neural Networks, Phys. Rev. Lett. **124**, 010508 (2020).

[127] E. Flurin, L. S. Martin, S. Hacohen-Gourgy, and I. Siddiqi, Using a Recurrent Neural Network to Reconstruct Quantum Dynamics of a Superconducting Qubit from Physical Observations, Phys. Rev. X **10**, 011006 (2020).

[128] Koji Hashimoto, Hong-Ye Hu, and Yi-Zhuang You, Neural ODE and Holographic QCD, arXiv:2006.00712 (2020).

[129] Di Luo, Zhuo Chen, Juan Carrasquilla, and Bryan K. Clark, Autoregressive Neural Network for Simulating Open Quantum Systems via a Probabilistic Formulation, arXiv:2009.05580 (2020).

[130] Axel U. J. Lode, Rui Lin, Miriam Büttner, Luca Papariello, Camille Lévêque, R. Chitra, Marios C. Tsatsos, Dieter Jaksch, and Paolo Molignini, Optimized Observable Readout from Single-shot Images of Ultracold Atoms via Machine Learning, arXiv:2010.14510 (2020).

[131] https://github.com/gtorlai/neuralnetworks-for-quantum.

[132] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning* (MIT Press, 2016).http://www.deeplearningbook.org

[133] Christopher M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)* (Springer-Verlag, Berlin, Heidelberg, 2006).

[134] Richard S. Sutton and Andrew G. Barto, *Introduction to Reinforcement Learning* (MIT Press, Cambridge, MA, USA, 1998), 1st ed.

[135] Volodymyr Mnih *et al.*, Human-level control through deep reinforcement learning, Nature **518**, 529 (2015).

[136] David Silver *et al.*, Mastering the game of go with deep neural networks and tree search, Nature **529**, 484 (2016).

[137] Marin Bukov, Alexandre G. R. Day, Dries Sels, Phillip Weinberg, Anatoli Polkovnikov, and Pankaj Mehta, Reinforcement Learning in Different Phases of Quantum Control, Phys. Rev. X **8**, 031086 (2018).

[138] Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre G. R. Day, Clint Richardson, Charles K. Fisher, and David J. Schwab, A high-bias, low-variance introduction to machine learning for physicists, Phys. Rep. **810**, 1 (2019).

[139] Titus Neupert, Mark H. Fischer, Eliska Greplova, Kenny Choo, and Michael Denner, Introduction to Machine Learning for the Sciences, arXiv:2102.04883 (2021).

[140] Florian Marquardt, Machine Learning and Quantum Devices, arXiv:2101.01759 (2021).

[141] Paul Fendley, K. Sengupta, and Subir Sachdev, Competing density-wave orders in a one-dimensional hard-boson model, Phys. Rev. B **69**, 075106 (2004).

[142] Antoine Browaeys and Thierry Lahaye, Many-body physics with individually controlled rydberg atoms, Nat. Phys. **16**, 132 (2020).

[143] Manuel Endres, Hannes Bernien, Alexander Keesling, Harry Levine, Eric R. Anschuetz, Alexandre Krajenbrink, Crystal Senko, Vladan Vuletic, Markus Greiner, and Mikhail D. Lukin, Atom-by-atom assembly of defect-free one-dimensional cold atom arrays, Science **354**, 1024 (2016).

[144] P. Schauß, J. Zeiher, T. Fukuhara, S. Hild, M. Cheneau, T. Macrì, T. Pohl, I. Bloch, and C. Gross, Crystallization in Ising quantum magnets, Science **347**, 1455 (2015).

[145] Henning Labuhn, Daniel Barredo, Sylvain Ravets, Sylvain de Léséleuc, Tommaso Macrì, Thierry Lahaye, and Antoine Browaeys, Tunable two-dimensional arrays of single Rydberg atoms for realizing quantum Ising models, Nature **534**, 667 (2016).

[146] Pascal Scholl, Michael Schuler, Hannah J. Williams, Alexander A. Eberharter, Daniel Barredo, Kai-Niklas Schymik, Vincent Lienhard, Louis-Paul Henry, Thomas C. Lang, Thierry Lahaye, Andreas M. Läuchli, and Antoine Browaeys, Programmable quantum simulation of 2D antiferromagnets with hundreds of Rydberg atoms, arXiv:2012.12268 (2020).

[147] Hannes Bernien, Sylvain Schwartz, Alexander Keesling, Harry Levine, Ahmed Omran, Hannes Pichler, Soonwon Choi, Alexander S Zibrov, Manuel Endres, Markus Greiner, Vladan Vuletić, and Mikhail D Lukin, Probing many-body dynamics on a 51-atom quantum simulator, Nature **551**, 579 (2017).

[148] Alexander Keesling, Ahmed Omran, Harry Levine, Hannes Bernien, Hannes Pichler, Soonwon Choi, Rhine Samajdar, Sylvain Schwartz, Pietro Silvi, Subir Sachdev, Peter Zoller, Manuel Endres, Markus Greiner, Vladan Vuletic, and Mikhail D. Lukin, Quantum kibble-zurek mechanism and critical dynamics on a programmable Rydberg simulator, Nature **568**, 207 (2019).

[149] Dolev Bluvstein, Ahmed Omran, Harry Levine, Alexander Keesling, Giulia Semeghini, Sepehr Ebadi, Tout T. Wang, Alexios A. Michailidis, Nishad Maskara, Wen Wei Ho, Soonwon Choi, Maksym Serbyn, Markus Greiner, Vladan Vuletic, and Mikhail D. Lukin, Controlling many-body dynamics with driven quantum scars in Rydberg atom arrays, arXiv:2012.12276 (2020).

[150] Harry Levine, Alexander Keesling, Giulia Semeghini, Ahmed Omran, Tout T. Wang, Sepehr Ebadi, Hannes Bernien, Markus Greiner, Vladan Vuletić, Hannes Pichler, and Mikhail D. Lukin, Parallel Implementation of High-Fidelity Multiqubit Gates with Neutral Atoms, Phys. Rev. Lett. **123**, 170503 (2019).

[151] Iris Cong, Sheng-Tao Wang, Harry Levine, Alexander Keesling, and Mikhail D. Lukin, Hardware-Efficient, Fault-Tolerant Quantum Computation with Rydberg Atoms, arXiv:2105.13501 (2021).

[152] Rhine Samajdar, Wen Wei Ho, Hannes Pichler, Mikhail D. Lukin, and Subir Sachdev, Complex Density Wave Orders and Quantum Phase Transitions in a Model of Square-Lattice Rydberg Atom Arrays, Phys. Rev. Lett. **124**, 103601 (2020).

[153] Ulrich Schollwoeck, The density-matrix renormalization group in the age of matrix product states, Ann. Phys. **326**, 96 (2011). january 2011 Special Issue
[154] Matthew Fishman, Steven R. White, and E. Miles Stoudenmire, The ITensor Software Library for Tensor Network Calculations, arXiv:2007.14822 (2020).
[155] Andrew J. Ferris and Guifre Vidal, Perfect sampling with unitary tensor networks, Phys. Rev. B **85**, 165146 (2012).
[156] D. M. Basko, I. L. Aleiner, and B. L. Altshuler, Metal–insulator transition in a weakly interacting many-electron system with localized single-particle states, Ann. Phys. **321**, 1126 (2006).
[157] Dmitry A. Abanin, Ehud Altman, Immanuel Bloch, and Maksym Serbyn, Colloquium: Many-body localization, thermalization, and entanglement, Rev. Mod. Phys. **91**, 021001 (2019).
[158] Lucile Savary and Leon Balents, Quantum spin liquids: A review, Rep. Prog. Phys. **80**, 016502 (2016).
[159] J. Knolle and R. Moessner, A field guide to spin liquids, Annu. Rev. Condens. Matter Phys. **10**, 451 (2019).
[160] Peter Broecker, Fakher F. Assaad, and Simon Trebst, Quantum phase recognition via unsupervised machine learning, arXiv:1707.00663 (2017).
[161] Sebastian J. Wetzel, Unsupervised learning of phase transitions: From principal component analysis to variational autoencoders, Phys. Rev. E **96**, 022140 (2017).
[162] Joaquin F. Rodriguez-Nieva and Mathias S. Scheurer, Identifying topological order through unsupervised machine learning, Nat. Phys. **15**, 790 (2019).
[163] Eran Lustig, Or Yair, Ronen Talmon, and Mordechai Segev, Identifying Topological Phase Transitions in Experiments Using Manifold Learning, Phys. Rev. Lett. **125**, 127401 (2020).
[164] Wen-Jia Rao, Machine learning the many-body localization transition in random spin systems, J. Phys.: Condens. Matter **30**, 395902 (2018).
[165] Jordan Venderley, Vedika Khemani, and Eun-Ah Kim, Machine Learning Out-Of-Equilibrium Phases of Matter, Phys. Rev. Lett. **120**, 257204 (2018).
[166] Xiao-Yu Dong, Frank Pollmann, and Xue-Feng Zhang, Machine learning of quantum phase transitions, Phys. Rev. B **99**, 121104 (2019).
[167] Yuan-Hong Tsai, Meng-Zhe Yu, and Yu-Hao Hsu, and Ming-Chiang Chung, Deep learning of topological phase transitions from entanglement aspects, Phys. Rev. B **102**, 054512 (2020).
[168] Korbinian Kottmann, Patrick Huembeli, Maciej Lewenstein, and Antonio Acín, Unsupervised Phase Discovery with Deep Anomaly Detection, Phys. Rev. Lett. **125**, 170603 (2020).
[169] A. Berezutskii, M. Beketov, D. Yudin, Z. Zimborás, and J. D. Biamonte, Probing criticality in quantum spin chains with neural networks, J. Phys.: Complexity **1**, 03LT01 (2020).
[170] Ehsan Khatami, Elmer Guardado-Sanchez, Benjamin M. Spar, Juan Felipe Carrasquilla, Waseem S. Bakr, and Richard T. Scalettar, Visualizing strange metallic correlations in the two-dimensional fermi-hubbard model with artificial intelligence, Phys. Rev. A **102**, 033326 (2020).
[171] Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind, Automatic differentiation in machine learning: A survey, J. Mach. Learn Res. **18**, 1 (2018).
[172] Diederik P. Kingma and Jimmy Ba, Adam: A Method for Stochastic Optimization, arXiv:1412.6980 (2014).
[173] Martín Abadi *et al.*, TensorFlow: Large-scale machine learning on heterogeneous systems (2015), software available from tensorflow.org.
[174] Jens Eisert, Dominik Hangleiter, Nathan Walk, Ingo Roth, Damian Markham, Rhea Parekh, Ulysse Chabaud, and Elham Kashefi, Quantum certification and benchmarking, Nat. Rev. Phys. **2**, 382 (2020).
[175] E. Knill, D. Leibfried, R. Reichle, J. Britton, R. B. Blakestad, J. D. Jost, C. Langer, R. Ozeri, S. Seidelin, and D. J. Wineland, Randomized benchmarking of quantum gates, Phys. Rev. A **77**, 012307 (2008).
[176] Easwar Magesan, Jay M. Gambetta, and Joseph Emerson, Characterizing quantum gates via randomized benchmarking, Phys. Rev. A **85**, 042311 (2012).
[177] Robin Harper, Ian Hincks, Chris Ferrie, Steven T. Flammia, and Joel J. Wallman, Statistical analysis of randomized benchmarking, Phys. Rev. A **99**, 052350 (2019).
[178] Steven T. Flammia and Yi-Kai Liu, Direct Fidelity Estimation from few Pauli Measurements, Phys. Rev. Lett. **106**, 230501 (2011).
[179] Leandro Aolita, Christian Gogolin, Martin Kliesch, and Jens Eisert, Reliable quantum certification of photonic state preparations, Nat. Commun. **6**, 8498 (2015).
[180] M. Gluza, M. Kliesch, J. Eisert, and L. Aolita, Fidelity Witnesses for Fermionic Quantum Simulations, Phys. Rev. Lett. **120**, 190501 (2018).
[181] K. Vogel and H. Risken, Determination of quasiprobability distributions in terms of probability distributions for the rotated quadrature phase, Phys. Rev. A **40**, 2847 (1989).
[182] Z. Hradil, Quantum-state estimation, Phys. Rev. A **55**, R1561 (1997).
[183] J. Řeháček, Z. Hradil, and M. Ježek, Iterative algorithm for reconstruction of entangled states, Phys. Rev. A **63**, 040303 (2001).
[184] Daniel F. V. James, Paul G. Kwiat, William J. Munro, and Andrew G. White, Measurement of qubits, Phys. Rev. A **64**, 052312 (2001).
[185] Miroslav Ježek, Jaromír Fiurášek, and Zdeněk Hradil, Quantum inference of states and processes, Phys. Rev. A **68**, 012305 (2003).
[186] Robin Blume-Kohout, Optimal, reliable estimation of quantum states, New J. Phys. **12**, 043034 (2010).
[187] John A. Smolin, Jay M. Gambetta, and Graeme Smith, Efficient Method for Computing the Maximum-Likelihood Quantum State from Measurements with Additive Gaussian Noise, Phys. Rev. Lett. **108**, 070502 (2012).
[188] Christopher Granade, Christopher Ferrie, and Steven T. Flammia, Practical adaptive quantum tomography, New J. Phys. **19**, 113017 (2017).
[189] Michael A. Nielsen and Isaac L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, 2010).
[190] Ryan O'Donnell and John Wright, Efficient quantum tomography, arXiv:1508.01907 (2015).
[191] Jeongwan Haah, Aram W. Harrow, Zhengfeng Ji, Xiaodi Wu, and Nengkun Yu, Sample-optimal tomography of quantum states, IEEE Trans. Inf. Theory **63**, 5628 (2017).

[192] H. Haffner, W. Hansel, C. F. Roos, J. Benhelm, D. Chek-al kar, M. Chwalla, T. Korber, U. D. Rapol, M. Riebe, P. O. Schmidt, C. Becher, O. Guhne, W. Dur, and R. Blatt, Scalable multiparticle entanglement of trapped ions, Nature **438**, 643 (2005).

[193] David Gross, Yi-Kai Liu, Steven T. Flammia, Stephen Becker, and Jens Eisert, Quantum State Tomography via Compressed Sensing, Phys. Rev. Lett. **105**, 150401 (2010).

[194] Steven T. Flammia, David Gross, Yi-Kai Liu, and Jens Eisert, Quantum tomography via compressed sensing: Error bounds, sample complexity and efficient estimators, New J. Phys. **14**, 095022 (2012).

[195] A. Shabani, R. L. Kosut, M. Mohseni, H. Rabitz, M. A. Broome, M. P. Almeida, A. Fedrizzi, and A. G. White, Efficient Measurement of Quantum Dynamics via Compressive Sensing, Phys. Rev. Lett. **106**, 100401 (2011).

[196] C. A. Riofrio, D. Gross, S. T. Flammia, T. Monz, D. Nigg, R. Blatt, and J. Eisert, Experimental quantum compressed sensing for a seven-qubit system, Nat. Commun. **8**, 15305 (2017).

[197] G. Toth, W. Wieczorek, D. Gross, R. Krischek, C. Schwemmer, and H. Weinfurter, Permutationally Invariant Quantum Tomography, Phys. Rev. Lett. **105**, 250403 (2010).

[198] Tobias Moroder, Philipp Hyllus, Géza Tóth, Christian Schwemmer, Alexander Niggebaum, Stefanie Gaile, Otfried Gühne, and Harald Weinfurter, Permutationally invariant state reconstruction, New J. Phys. **14**, 105001 (2012).

[199] M Cramer, M. B. Plenio, S. T. Flammia, R Somma, D Gross, S. D. Bartlett, O Landon-Cardinal, D Poulin, and Y. K. Liu, Efficient quantum state tomography, Nat. Commun. **1**, 149 (2009).

[200] T. Baumgratz, D. Gross, M. Cramer, and M. B. Plenio, Scalable Reconstruction of Density Matrices, Phys. Rev. Lett. **111**, 020401 (2013).

[201] B. P. Lanyon, C. Maier, M. Holzäpfel, T. Baumgratz, C. Hempel, P. Jurcevic, I. Dhand, A. S. Buyskikh, A. J. Daley, M. Cramer, M. B. Plenio, R. Blatt, and C. F. Roos, Efficient tomography of a quantum many-body system, Nat. Phys. **13**, 1158 (2017).

[202] Jun Wang, Zhao-Yu Han, Song-Bo Wang, Zeyang Li, Liang-Zhu Mu, Heng Fan, and Lei Wang, Scalable quantum tomography with fidelity estimation, Phys. Rev. A **101**, 032321 (2020).

[203] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski, A learning algorithm for Boltzmann machines, Cognitive Sci. **9**, 147 (1985).

[204] P. Smolensky, in *Parallel Distributed Processing*, edited by David E. Rumelhart, James L. McClelland, and CORPORATE PDP Research Group (MIT Press, Cambridge, MA, USA, 1986), Chap. Information Processing in Dynamical Systems: Foundations of Harmony Theory, p. 194.

[205] W. A. Little, The existence of persistent states in the brain, Math. Biosci. **19**, 101 (1974).

[206] W. A. Little and Gordon L. Shaw, Analytic study of the memory storage capacity of a neural network, Math. Biosci. **39**, 281 (1978).

[207] J. J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, Proc. Natl. Acad. Sci. **79**, 2554 (1982).

[208] Nicolas Le Roux and Yoshua Bengio, Representational power of restricted Boltzmann machines and deep belief networks, Neural Comput. **20**, 1631 (2008).

[209] S. Kullback and R. A. Leibler, On information and sufficiency, Ann. Math. Statist. **22**, 79 (1951).

[210] Sergey Bravyi, David P. DiVincenzo, Roberto I. Oliveira, and Barbara M. Terhal, The Complexity of Stoquastic Local Hamiltonian Problems, arXiv:quant-ph/0606140 (2006).

[211] Giuseppe Carleo, Kenny Choo, Damian Hofmann, James E. T. Smith, Tom Westerhout, Fabien Alet, Emily J. Davis, Stavros Efthymiou, Ivan Glasser, Sheng-Hsuan Lin, Marta Mauri, Guglielmo Mazzola, Christian B. Mendl, Evert van Nieuwenburg, Ossian O'Reilly, Hugo Théveniaut, Giacomo Torlai, and Alexander Wietek, NetKet: A Machine Learning Toolkit for Many-Body Quantum Systems, arXiv:1904.00031 (2019).

[212] Matthew D. Zeiler, ADADELTA: An Adaptive Learning Rate Method, arXiv:1212.5701 (2012).

[213] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M. Chow, and Jay M. Gambetta, Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets, Nature **549**, 242 (2017).

[214] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien, A variational eigenvalue solver on a photonic quantum processor, Nat. Commun. **5**, 4213 (2014).

[215] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (Association for Computational Linguistics, Doha, Qatar, 2014), p. 1724.

[216] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio, in *Proceedings of the 30th International Conference on Machine Learning*, edited by Sanjoy Dasgupta and David McAllester, Proceedings of Machine Learning Research Vol. 28 (PMLR, Atlanta, Georgia, USA, 2013), p. 1310.

[217] Tom Vieijra, Corneel Casert, Jannes Nys, Wesley De Neve, Jutho Haegeman, Jan Ryckebusch, and Frank Verstraete, Restricted Boltzmann Machines for Quantum States with Non-Abelian or Anyonic Symmetries, Phys. Rev. Lett. **124**, 097201 (2020).

[218] Di Luo, Zhuo Chen, Kaiwen Hu, Zhizhen Zhao, Vera Mikyoung Hur, and Bryan K. Clark, Gauge Invariant Autoregressive Neural Networks for Quantum Lattice Models, arXiv:2101.07243 (2021).

[219] Di Luo, Giuseppe Carleo, Bryan K. Clark, and James Stokes, Gauge equivariant neural networks for quantum lattice gauge theories, arXiv:2012.05232 (2020).

[220] Mohamed Hibat-Allah, Estelle M. Inack, Roeland Wiersema, Roger G. Melko, and Juan Carrasquilla, Variational Neural Annealing, arXiv:2101.10154 (2021).

[221] Matthew Fishman and Giacomo Torlai, PastaQ: A package for simulation, tomography and analysis of quantum computers (2020).