U(1)-symmetric recurrent neural networks for quantum state reconstruction

Stewart Morawetz [●],[1] Isaac J. S. De Vlugt,[1,2] Juan Carrasquilla,[3,1] and Roger G. Melko[1,2]

[1]*Department of Physics and Astronomy, University of Waterloo, Ontario, Canada N2L 3G1*
[2]*Perimeter Institute for Theoretical Physics, Waterloo, Ontario, Canada N2L 2Y5*
[3]*Vector Institute, MaRS Centre, Toronto, Ontario, Canada M5G 1M1*

Generative models are a promising technology for the enhancement of quantum simulators. These machine learning methods are capable of reconstructing a quantum state from experimental measurements, and can aid in the calculation of physical observables. In this paper, we employ a recurrent neural network (RNN) to reconstruct the ground state of the spin-$1/2$ XY model, a prototypical Hamiltonian explored in trapped ion simulators. We explore its performance after enforcing a U(1) symmetry, which was recently shown by Hibat-Allah *et al.* [Phys. Rev. Res. **2**, 023358 (2020)] to preserve the autoregressive nature of the RNN. By studying the reconstruction of the XY model ground state from projective measurement data, we show that imposing U(1) symmetry on the RNN significantly increases the efficiency of learning, particularly in the early epoch regime. We argue that this performance increase may result from the tendency of the enforced symmetry to alleviate vanishing and exploding gradients, which helps stabilize the training process. Thus, symmetry-enforced RNNs may be particularly useful for applications of quantum simulators where a rapid feedback between optimization and circuit preparation is necessary, such as in hybrid classical-quantum algorithms.

I. INTRODUCTION

Advances in the fabrication and control of quantum devices is progressing to the point where high-quality measurement data is attainable from extremely pure quantum states of tens or hundreds of qubits. A growing body of work has demonstrated the ability of technologies adopted from machine learning to provide approximate reconstructions of quantum states from practically sized data sets on such noisy intermediate-scale quantum (NISQ) devices [1–4]. A typical machine learning approach involves the use of generative models to learn and represent a multiqubit quantum state. The most well-studied generative model in this context is the restricted Boltzmann machine (RBM), which has been used to obtain high-quality reconstructions of a variety of pure and mixed states from both synthetic and experimental data [5–7].

Despite their success, one disadvantage of RBMs well-known to the machine learning community is that they do not provide a tractable distribution estimator—in other words, a normalized representation of the quantum state. This has motivated physicists to explore powerful alternatives in the so-called autoregressive models [8–11], such as recurrent neural networks (RNNs) or attention-based transformers [12]. Autoregressive models, where the total (joint) distribution of the variables is decomposed into a product of conditional probabilities, are capable of learning a normalized representation of the quantum state. Additionally, they provide a tractable method capable of efficiently producing perfectly independent samples, e.g., for use in calculating estimators of physical observables.

Beyond normalization, another important aspect of improving the performance of any quantum algorithm is the exploitation of quantum symmetries. More specifically, Hamiltonians that are invariant under a symmetry group will have conserved quantities which can be exploited to significantly simplify the task of computing the ground state wave function. A well-known example of a performance increase comes from the implementation of the U(1) symmetry of particle number (charge) or magnetization sector. We will focus on a simple model Hamiltonian invariant under U(1) symmetry operations in this paper, the spin-$1/2$ XY model, which has recently been engineered in trapped ion experiments [13–15]. The Hamiltonian of the model is

$$H = -J \sum_{\langle ij \rangle} \left(S_i^x S_j^x + S_i^y S_j^y \right), \tag{1}$$

where $\langle ij \rangle$ denotes nearest-neighbor pairs of a one-dimensional lattice with N sites, and $\mathbf{S}_i$ is the spin-$1/2$ operator on site i. The U(1) symmetry of the Hamiltonian is manifest as a block diagonal structure, and a conservation law for the total magnetization $S^z = \sum_i S_i^z$ of the ground state.

As recognized by Hibat-Allah *et al.* [16], it is possible to implement a U(1) symmetry on an RNN wave function while preserving its autoregressive property, through a projection operator on the conditional probability distributions that are the output of the RNN cell (described in detail below). In Ref. [16], the symmetry-conservation projection was employed in an RNN used as a variational wave function. In that context, the RNN cost function is defined by the expectation value of the Hamiltonian—a calculation which is improved significantly by the implementation of U(1) symmetry. In the present work, we explore the implementation of this symmetry in an RNN used for data-driven state reconstruction. In order to do so, we generate training and testing data from the

Hamiltonian equation (1), and train the parameters of an RNN to optimize a negative log-likelihood. By comparing and contrasting to both RBMs and RNNs without any implementation of symmetry, we present evidence that RNNs constrained to obey the U(1) symmetry result in a significant speedup in the time required to train to low infidelity and relative energy difference, particularly in the early epoch training regime.

II. RECURRENT NEURAL NETWORKS

The RNN is a powerful autoregressive model built for modeling sequential data, which has been discussed in many other works; we will leave the reader to peruse references relevant for its application to machine learning problems in industry applications [17]. The RNN was introduced into the field of quantum state reconstruction in Ref. [8], and as an explicit wave-function representation for variational optimization in Ref. [16]. In this section, we offer a brief background of RNNs for quantum state reconstruction as used in the present study.

A. RNN architecture

The goal of an RNN is to reconstruct an unknown target probability distribution $q(\boldsymbol{\sigma})$ by interpreting instances of an input data set $\mathcal{D} = \{\boldsymbol{\sigma}\}$, sampled from $q(\boldsymbol{\sigma})$ as sequential data. In this paper, $\boldsymbol{\sigma}$ is a vector of size N containing the occupation number of each site i, which we relate to the eigenvalues of the spin operator $S_i^z = \pm 1/2$ through $\sigma_i = 1/2 - S_i^z$.

In the context of quantum state reconstruction, the target distribution is the square of the ground state (GS) of Eq. (1),

$$q(\boldsymbol{\sigma}) = |\langle \boldsymbol{\sigma} | \psi_{\mathrm{GS}} \rangle|^2 = |\psi_{\mathrm{GS}}(\boldsymbol{\sigma})|^2. \tag{2}$$

Setting $J = 1$ in Eq. (1), the Hamiltonian satisfies the well-known Perron-Frobenius theorem [18]. Therefore, the ground state is sign-free and can be written as the square root of the target distribution,

$$\psi_{\mathrm{GS}}(\boldsymbol{\sigma}) = \sqrt{q(\boldsymbol{\sigma})}. \tag{3}$$

Since the wave function has no complex phase, it can be reconstructed from projective measurements drawn from the S^z basis only. For use in the RNN, each $\boldsymbol{\sigma}$ is formatted as a one-hotted configuration comprising of N "sequential" qubits. Specifically, $\boldsymbol{\sigma} \equiv (\sigma_1, \sigma_2, \ldots, \sigma_N)$ with $\sigma_i \in \{0, 1\}$ or $\boldsymbol{\sigma}_i \in \{(1, 0), (0, 1)\}$ in the binary and one-hotted representations, respectively. Throughout this work, we adopt the convention that spin-up refers to $\sigma_i = 0$ and spin-down refers to $\sigma_i = 1$.

The RNN defines an autoregressive model for $p(\boldsymbol{\sigma})$ using the chain rule of probabilities,

$$p(\boldsymbol{\sigma}) = p(\sigma_1) p(\sigma_2 | \sigma_1) \cdots p(\sigma_N | \sigma_{N-1}, \ldots, \sigma_1). \tag{4}$$

The building block of an RNN is a recurrent cell [17]. A "vanilla" RNN consists of a nonlinear activation function f that maps a hidden vector $\boldsymbol{h}_{i-1}$ of dimension d_h and an input vector $\boldsymbol{\sigma}_{i-1}$ of dimension $d_v = 2$ to a different hidden vector $\boldsymbol{h}_i$. Specifically,

$$\boldsymbol{h}_i = f(W\boldsymbol{\sigma}_{i-1} + U\boldsymbol{h}_{i-1} + \boldsymbol{b}), \tag{5}$$

where tunable parameters $\boldsymbol{\theta}$ of the RNN are given by the weight matrices $W \in \mathbb{R}^{d_h \times 2}$, $U \in \mathbb{R}^{d_h \times d_h}$, and bias $\boldsymbol{b} \in \mathbb{R}^{d_h}$.
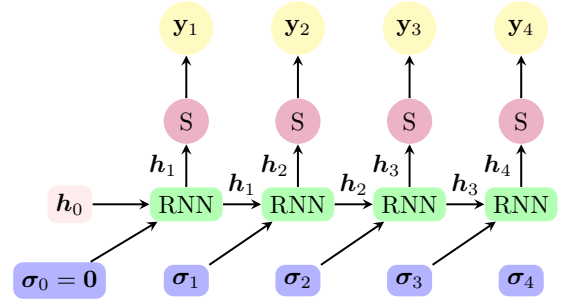

FIG. 1. A diagram of the RNN architecture. At the ith recurrent step, the RNN cell takes the previous hidden and input vectors $\boldsymbol{h}_{i-1}$ and $\boldsymbol{\sigma}_{i-1}$, and outputs the next hidden vector $\boldsymbol{h}_i$. This is then passed to a fully connected softmax layer S, which makes a prediction $\boldsymbol{y}_i = [p(\sigma_i = 0), p(\sigma_i = 1)]$ for the next input $\boldsymbol{\sigma}_i$. $\boldsymbol{h}_i$ and $\boldsymbol{\sigma}_i$ are passed to the next RNN recurrent cell in order to compute $\boldsymbol{h}_{i+1}$. This is repeated until there are output vectors $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots$, and $\boldsymbol{y}_N$ corresponding to probabilities $p(\sigma_1)$, $p(\sigma_2 | \sigma_1), \ldots$, and $p(\sigma_N | \sigma_{N-1}, \ldots, \sigma_1)$, respectively.

The initial values of the hidden and input vectors $\boldsymbol{h}_0$ and $\boldsymbol{\sigma}_0$ are fixed to the zero vector and $(1, 0)$, respectively.

A linear layer then maps each output hidden vector $\boldsymbol{h}_i$ to an output vector $\boldsymbol{y}_i \in \mathbb{R}^2$, where

$$\boldsymbol{y}_i = S(V\boldsymbol{h}_i + \boldsymbol{c}), \tag{6}$$

S is the softmax function,

$$S(v_j) = \frac{\exp(v_j)}{\sum_i \exp(v_i)},$$

and additional tunable RNN parameters are given by the weight matrix $V \in \mathbb{R}^{d_h \times 2}$ and bias $\boldsymbol{c} \in \mathbb{R}^2$. This entire structure is shown diagrammatically for $N = 4$ in Fig. 1.

This vanilla RNN cell architecture has been shown not to lend itself well to learning longer sequences of data, as gradients tend to explode or vanish [19]. To mitigate this, we use the slightly modified gated recurrent unit (GRU) cell, which has demonstrated stronger performance with learning long sequences of data [8,16,20]. Additional details of this architecture can be found in Appendix B.

For the purposes of quantum state reconstruction, the output $\boldsymbol{y}_i$ in Eq. (6) is interpreted as a probability vector corresponding to the ith qubit having $\sigma_i = 0$ or 1 given all of the previous inputs $\sigma_{<i} = \sigma_{i-1}, \sigma_{i-2}, \ldots, \sigma_1$,

$$\boldsymbol{y}_i = [p(\sigma_i = 0 | \sigma_{<i}), \ p(\sigma_i = 1 | \sigma_{<i})].$$

Then, Eq. (4) is calculated as

$$p(\boldsymbol{\sigma}) = \prod_{i=1}^{N} \boldsymbol{y}_i \cdot \boldsymbol{\sigma}_i, \tag{7}$$

and the RNN wave function is given by

$$\psi_{\mathrm{RNN}}(\boldsymbol{\sigma}) = \sqrt{p(\boldsymbol{\sigma})}. \tag{8}$$

The goal of the RNN is to find a $p(\boldsymbol{\sigma})$ which approximates the unknown target distribution $q(\boldsymbol{\sigma})$ as accurately as possible, given only the data set $\mathcal{D}$. In order to achieve this, the RNN is trained by tuning the parameters $\boldsymbol{\theta}$ (the weights and biases) in such a way as to minimize the negative log-likelihood (NLL),
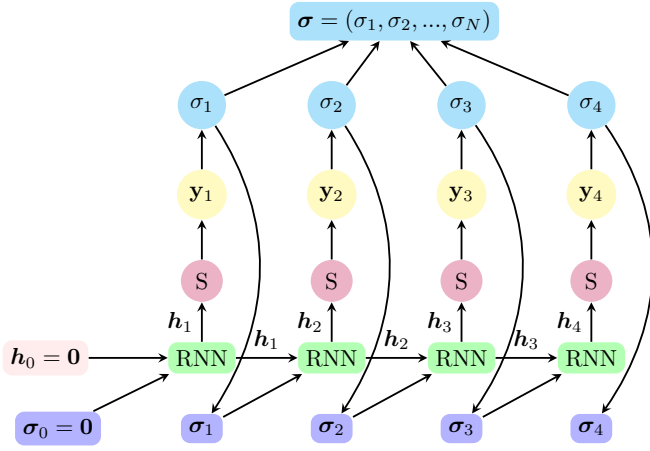
FIG. 2. A diagrammatic depiction of sampling from an RNN for $N = 4$ spins. At each recurrent step, the RNN uses the output $\boldsymbol{y}_i$ as a probability distribution from which to sample the next input spin, σ_i. This is then one-hotted and fed into the next RNN recurrent cell to calculate the proceeding spin value σ_{i+1}, and so on. In this way, the RNN can generate uncorrelated samples $\boldsymbol{\sigma}$ autoregressively.

which is given by

$$\text{NLL} = -\frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{\sigma} \in \mathcal{D}} \ln p(\boldsymbol{\sigma}). \tag{9}$$

This NLL thereby defines the loss landscape of the RNN. For all RNNs trained in this work, stochastic gradient decent (SGD) is employed to perform the optimization numerically.

B. Sampling from RNNs and symmetry enforcement

In order to discuss the implementation of symmetry in RNN training, it is instructive to first consider how one draws samples from the RNN probability distribution $p(\boldsymbol{\sigma})$. Figure 2 illustrates the general process of sampling from an RNN. At each recurrent step, the output probability distribution $\boldsymbol{y}_i$ is sampled to obtain the input σ_i to the proceeding RNN recurrent unit. In this way of autoregressive sampling, the RNN can generate perfectly uncorrelated samples of $p(\boldsymbol{\sigma})$ in a straightforward and efficient manner. While this efficiency is a main advantage of this generative model architecture, it is clear that the N samples of σ_i thus produced obey no global constraints, such as the conservation of S^z magnetization (e.g., $S^z = \sum_i S_i^z = 0$, or $\sum_i \sigma_i = N/2$ in the binary representation) required in the ground state of Eq. (1).

Although the implementation of symmetries is important for the performance of any quantum algorithm, they have not yet been implemented in generative models used for quantum state reconstruction. For the widely used RBM (see Appendix A), implementing symmetries is not possible in the block Gibbs sampling algorithm, without significant modifications to conventional contrastive divergence (CD$_k$) [21]. In contrast, as shown by Hibat-Allah *et al.*, the autoregressive property of an RNN gives it the ability for discrete symmetries to be implemented naturally in the sampling algorithm [16]. In the variational setting, where the RNN parameters are optimized based on the expectation value of the Hamiltonian, the authors of Ref. [16] concluded that imposing symmetry improved the accuracy of the ground state thus obtained. Here,

we ask if a similar performance improvement may be possible in the setting where the RNN is asked to reconstruct the ground state wave function based on measurement data.

Note first, if drawing samples of projective measurements in the S^z basis that conserve zero magnetization, there must be exactly $N/2$ spin-up and $N/2$ spin-down states for any even chain length. Treating the lattice sites as an arbitrary sequence, this means that once the cumulative number of either up spins or down spins in the chain reaches $N/2$ we know the remaining spins *must* be of the opposite spin. Mathematically, we can describe this following Ref. [16]. At every point in the sequential chain, denote $N_\uparrow$ and $N_\downarrow$ as the cumulative number of up and down spins until that point, respectively. Then, for the original network outputs $\boldsymbol{y}_i = (p_{i,0}, p_{i,1}) \equiv [p(\sigma_i = 0), p(\sigma_i = 1)]$, modify them as follows. Define the two modified probabilities

$$\widetilde{p}_{i,0} = p_{i,0} \times \Theta\left(\frac{N}{2} - N_\uparrow\right) \tag{10}$$

and

$$\widetilde{p}_{i,1} = p_{i,1} \times \Theta\left(\frac{N}{2} - N_\downarrow\right), \tag{11}$$

where

$$\Theta(x) = \begin{cases} 0, & x \leqslant 0 \\ 1, & x > 0 \end{cases}$$

is the Heaviside step function. Finally, renormalize to get the modified output

$$\widetilde{\boldsymbol{y}}_i = \frac{(\widetilde{p}_{i,0}, \widetilde{p}_{i,1})}{\widetilde{p}_{i,0} + \widetilde{p}_{i,1}}. \tag{12}$$

This ensures that if there are already $N/2$ up (down) spins, the probability of predicting another up (down) spin is exactly 0. We are then guaranteed that the network will predict down (up) spins for all the remaining sites, thereby remaining exclusively in the $S^z = 0$ sector (Fig. 3).

Training works in exactly the same way with this modification with the exception that the RNN probability distribution $p(\boldsymbol{\sigma})$ is replaced by a modified, symmetry-enforcing distribution. Also note that for spins on which the symmetry enforcement is applied, these will not contribute to the cost function, and therefore gradient information, since they have probability 1. We anticipate that the effect of the projector on the training of the RNN is to alleviate the potential vanishing and exploding of the noisy gradients and an ensuing stabilization of the training process. This can be intuitively understood from the sum-of-products structure of the gradient of the loss function [22]. We first note that the projectors effectively shorten the length of the sequences that are used in the calculation of the gradient, which eliminates some factors from their calculation that appear multiplicatively [22]. Due to this multiplicative structure, the shortening of the sequences effectively stabilizes the training by ameliorating the uncontrolled amplification or suppression of the gradients. We note that the effective shortening of the sequences induces an effect similar to the improvements brought by the truncated backpropagation (TB) algorithm [23], one of the most most practical and successful methods for training RNNs. In TB, a long sequence of K steps is split into m shorter sequences of length l (such that $K = m \times l$), each of which is used
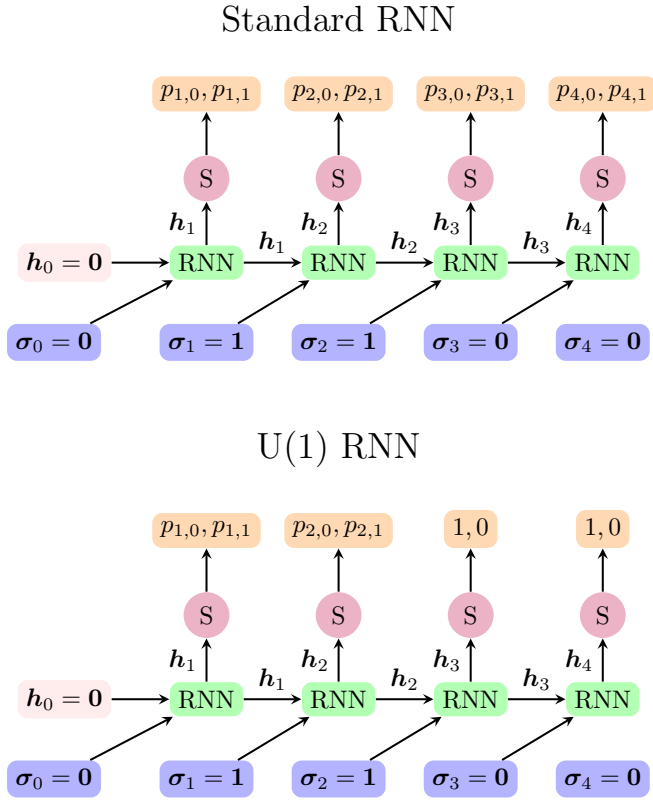
Standard RNN

U(1) RNN

FIG. 3. Comparison of a standard RNN and one in which U(1) symmetry is enforced. Here, $p_{i,j}$ refers to the probability $p(\sigma_i = j)$ with $j \in \{0, 1\}$. In this example with training sample $\boldsymbol{\sigma} = (1, 1, 0, 0)$, the "standard" RNN will not impose any restrictions on the output distribution, and therefore might give a nonzero probability of a symmetry-violating configuration. By contrast, the symmetry-enforcing network counts the first two down spins—the fixed initial input spin $\boldsymbol{\sigma}_0 = \mathbf{0}$ does not count—and then the output is modified to guarantee that the next two spins will be up. Hence $p_{3,0} = p_{4,0} = 1$ and $p_{3,1} = p_{4,1} = 0$.

as a separate training example. While traditional TB leads to the loss of potentially important correlations in the data due to the mean-field nature of the algorithm, our approach exactly accounts for the correlations due to the magnetization conservation without losing information.

III. RESULTS

In this section, we perform numerical experiments designed to test the effectiveness of the symmetry-imposed RNN in reconstructing the ground state wave function of the spin-1/2 *XY* model given by Eq. (1). The ground state of the *XY* model can be efficiently calculated in one dimension using the density matrix renormalization group (DMRG) [24–26], and the input data set $\mathcal{D}$ can then be generated via the sampling algorithm of Ferris and Vidal [27]. For all system sizes, we produce a training data set of $|\mathcal{D}| = 20\,000$ independent samples drawn from the DMRG. Note that we adopt the convention that spin-up refers to $\sigma_i = 0$ and spin-down refers to $\sigma_i = 1$, i.e., $\sigma_i = 1/2 - S_i^z$. Each training sample is then prepared as a one-hotted vector where $\sigma_i \in \{0, 1\}$ is mapped to $\boldsymbol{\sigma}_i \in \{(1, 0), (0, 1)\}$ for each lattice site i.

TABLE I. Hyperparameters used for training RNNs.

Hyperparameter	Value
Hidden units	100
Random seed	1
Learning rate	0.001
Batch size	50

A. Physical observables as training metrics

An advantage of using RNNs for the reconstruction of quantum states is that the quality of reconstruction can be assessed by calculating physical observables (such as the energy) during training, and comparing this to its target or "exact" value (calculated in this case by DMRG) [28–30]. The expectation value $\langle \hat{H} \rangle_{\psi_{\mathrm{RNN}}}$ can be approximated by an energy estimator E_{RNN}, calculated from a data set $\mathcal{S}$ autoregressively generated from the RNN,

$$E_{\mathrm{RNN}} \approx \frac{1}{|\mathcal{S}|} \sum_{\boldsymbol{\sigma} \in \mathcal{S}} \sum_{\boldsymbol{\sigma}'} \frac{\psi_{\mathrm{RNN}}(\boldsymbol{\sigma}')}{\psi_{\mathrm{RNN}}(\boldsymbol{\sigma})} H_{\boldsymbol{\sigma}\boldsymbol{\sigma}'}, \tag{13}$$

where $H_{\boldsymbol{\sigma}\boldsymbol{\sigma}'} = \langle \boldsymbol{\sigma} | \hat{H} | \boldsymbol{\sigma}' \rangle$. This expression is equivalent to the *local observable* formalism used in variational Monte Carlo.

Since all training data sets $\mathcal{D}$ are generated via DMRG, it is convenient to use the results of these calculations as our target or "exact" values for physical observables. We choose to monitor the absolute difference in E_{RNN} and the energy from DMRG (per site),

$$\varepsilon = \frac{E_{\mathrm{RNN}} - E_{\mathrm{DMRG}}}{N}, \tag{14}$$

for all systems studied in this work.

Additionally, in our case the fidelity between the DMRG wave function [taken to be the target ground state given in Eq. (3)] and the RNN wave function [Eq. (8)], can also be monitored during training. The fidelity is given by

$$\mathcal{F} = |\langle \psi_{\mathrm{GS}} | \psi_{\mathrm{RNN}} \rangle|^2, \tag{15}$$

and can be calculated directly using the wave-function coefficients provided by the DMRG simulation.

B. Numerical results

Hyperparameters for the RNNs trained in this work can be found in Table I. As a means of benchmarking performance, the RNN will be compared to the familiar RBM (see Appendix A). For the RBM, we choose standard CD_k Gibbs sampling given its simplicity, widespread use, and efficiency [1,2,8,28–38]. Hyperparameters used to train the RBMs in this work are presented in Table II of Appendix A.

We begin by studying the accuracy of the RNN reconstruction of the *XY* model ground state, using as metrics the energy and fidelity. In the following results, this estimator Eq. (13) is calculated using $|\mathcal{S}| = 10^4$ projective measurements in the S^z basis, drawn from each generative model (the RBM or RNN).

In Fig. 4, we illustrate the relative energy difference ε, Eq. (14), and infidelity $(1 - \mathcal{F})$ for $N = 4$ and $N = 10$ using an RBM, a conventional RNN, and an RNN with symmetry enforcement in training [labeled U(1)-RNN]. Results are

TABLE II. Hyperparameters used for training RBMs.

Hyperparameter	Value
n_h	10 [$N = 2$], 50 [$N = 4$], 100 [$N > 4$]
Random seeds	7777 [$N = 2$], 9999 [$N = 4, 30, 50$],
	2222 [$N = 6$], 1234 [$N = 8, 10, 20$],
	1357 [$N = 16, 40$]
Learning rate	0.01×0.999^t, where t is epoch number
Positive batch size	100
Negative batch size	200
k Gibbs steps for Γ	100

plotted as a function of training epoch, where one epoch is defined as a pass over the entire data set $\mathcal{D}$ (with a batch size defined in Table I). In these plots, the RBM clearly does not reach the same reconstruction quality as the RNNs. Further, the RBMs were trained out to 20 000 epochs; however, we see very little improvement past 2000 epochs. Thus, we conclude that the RBMs trained in this work cannot reach a reconstruction quality comparable to the RNNs in reasonable compute time. Further discussion of this, and results on larger system sizes, is relegated to Appendix A.

The results of Fig. 4 show that the U(1)-RNN is significantly faster (i.e., requires fewer epochs) in achieving lower infidelities and ε compared to a conventional RNN without symmetry imposed. Importantly, in the limit of long training time, we find that reconstruction quality in both RNNs is comparable, given the uncertainties involved. We explore this for larger system sizes, $N = 30, 40$, and 50, in Fig. 5. Here, it is clear that in each case, the U(1)-RNN reaches a small relative energy difference ($\varepsilon < 0.003$) in significantly fewer training epochs than the conventional RNN. As in the case of smaller N, a comparable energy difference is eventually achieved by
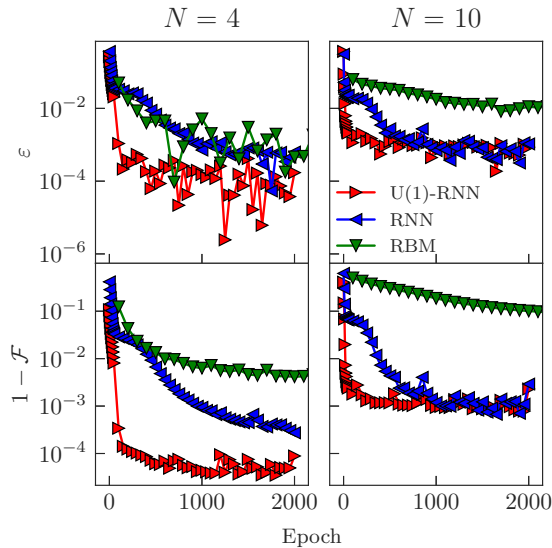

FIG. 4. Comparing the energy difference and infidelity during training for $N = 4$ and 10 systems using a recurrent neural network with [U(1)-RNN] and without (RNN) imposing the symmetry, and an RBM. Error bars are omitted for plots of ε due to the semilogarithmic scale, but are on the order of 10^{-3} or smaller.
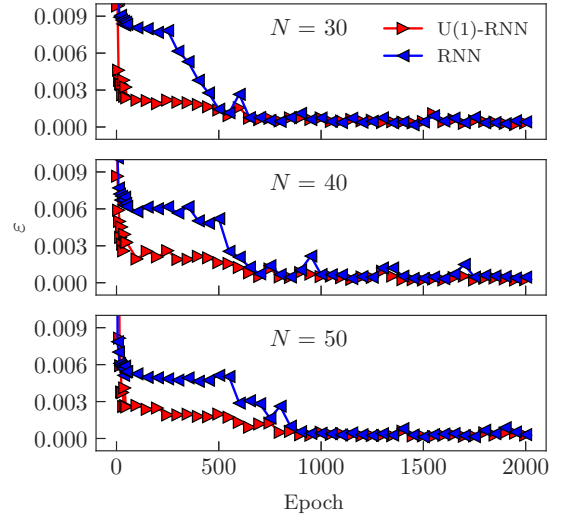

FIG. 5. Comparing the energy difference during training for $N = 30$, 40, and 50 systems using a recurrent neural network with [U(1)-RNN] and without [RNN] imposing the symmetry. Errors are on the order of 10^{-3} or smaller.

the conventional RNN (e.g., for 500 or more epochs in Fig. 4). After the two models achieve approximately the same energy, further training (e.g., past 500 epochs) appears to affect both RNNs similarly, to within fluctuations.

To further explore the correlation between the implementation of symmetry enforcement, in Fig. 6 we illustrate the fraction of samples generated by the standard RNN as a function of training, for $N = 10$. Interestingly, after the first few initial epochs of training, the conventional RNN experiences a plateau where 10%–20% of configurations fall outside of the $S^z = 0$ magnetization sector. This number drops to zero near epoch 500, which corresponds to the point where the conventional and U(1)-enforced RNNs achieve approximately the same energy and fidelity in Fig. 4. Additionally, as illustrated by the inset of Fig. 6, the plateau of $S^z \neq 0$ samples corresponds in a higher NLL [Eq. (9)] for the conventional RNN as compared to the U(1)-enforced version.
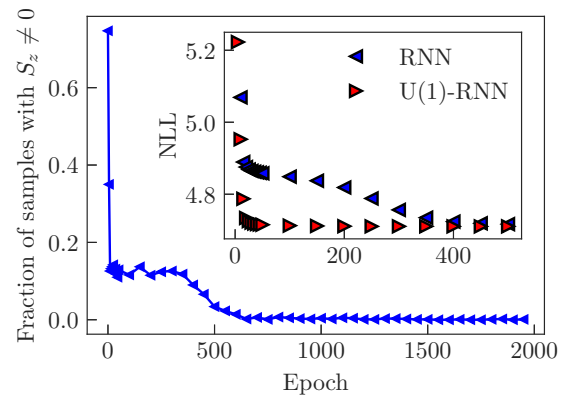

FIG. 6. The fraction of samples that a conventional RNN generates outside of the $S_z = 0$ sector for training an $N = 10$ system. The intermediate plateau coincides with a plateau in fidelity and energy difference (see Fig. 4). Inset: a comparison of the negative log-likelihood Eq. (9) between the conventional and U(1)-enforced RNNs.
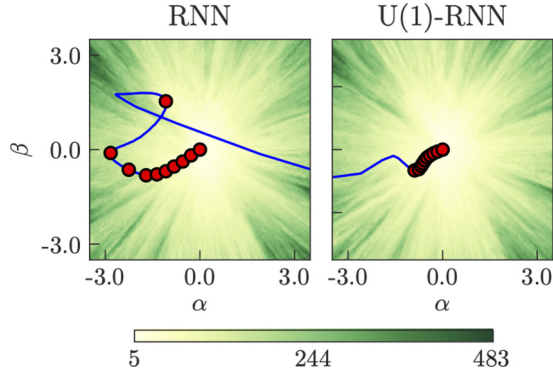
FIG. 7. Comparing the loss landscape on a randomly chosen plane in parameter space for the U(1)-RNN and the conventional RNN, training on data from an $N = 10$ system. The line traces the evolution of the network parameters in training, with the dots evenly spaced every 200 epochs. The first dot illustrated in each path occurs at epoch 200. The optimum $\boldsymbol{\theta}^*$ found in training sits at the origin. The evolution of infidelity and ϵ for these parameters appear in Fig. 4.

As a final illustration of how the symmetry enforcement affects training, we visualize the loss landscape near the optimum for each RNN. Building on earlier works [39–41], we visualize our landscapes by randomly choosing two directions $\boldsymbol{\delta}$ and $\boldsymbol{\eta}$ in the very high-dimensional space of parameters, where each component of the vectors is sampled from a normal distribution, $\mathcal{N}(0, 1)$. With these, we plot the function

$$f(\alpha, \beta) = L(\boldsymbol{\theta}^* + \alpha\boldsymbol{\delta} + \beta\boldsymbol{\eta}), \qquad (16)$$

where $\boldsymbol{\theta}^*$ are the optimal parameters obtained at the end of training, and α, β are scaling parameters used to explore the loss function L on the plane defined by $\boldsymbol{\theta}^*$, $\boldsymbol{\delta}$, and $\boldsymbol{\eta}$.

Figure 7 shows the above-defined cross section of the loss landscape for an RNN with and without symmetry imposed, and the corresponding path that the RNN parameters traverse during training. The system size is $N = 10$, with corresponding infidelity and energy differences plotted in Fig. 4. In Fig. 7, the line traces the evolution of the RNN parameters during training, with dots spaced every 200 epochs. Although we note that the loss landscapes are different in each case, it is clear that imposing symmetry leads to a more stable training where the RNN parameters approach its respective optimal value in significantly fewer epochs. Thus, enforcing the U(1) symmetry causes that optimum to be reached much more quickly than the case where no symmetry is imposed in training.

IV. DISCUSSION

In this paper, we have explored the effect of imposing symmetry in a RNN used for quantum state reconstruction. The system considered is the ground state wave function of the spin-1/2 XY model on an open-boundary chain of length N. This model is directly relevant to current trapped ion experiments [13–15], which can prepare the XY Hamiltonian on chain lengths of tens of sites or more, and produce a variety of measurements suitable for data-driven state reconstruction. Due to the U(1) symmetry of the Hamiltonian, the ground state magnetization in the S^z basis is a conserved quantity. It

is this conservation law that we impose in the autoregressive sampling procedure of the RNN. This is done through a projection operator that constrains the RNN output after the point in the sequence for which the symmetry-induced constraints fully determine the remaining spin states.

The RNNs in this work are used in the context of state reconstruction from data. In order to benchmark their performance, we use our knowledge of the exact ground state wave function (obtained via DMRG) to track the fidelity and relative energy difference as a function of training epoch. Our main observation is that the imposition of the U(1) symmetry constraint significantly improves the speed at which RNNs are able to learn good approximations to the ground state of the XY model. Specifically, such RNNs require fewer training epochs to reach relatively low infidelities and energy differences compared to a conventional RNN without U(1) symmetry imposed. By examining the tendency of conventional RNNs to produce samples outside of the $S^z = 0$ magnetization sector, it is clear that the slower training performance in that case correlates to a significant fraction of samples produced outside of this sector. Upon further training (past approximately 500 epochs in our case), the fidelity and energy metrics of the RNNs with and without symmetry begin to converge, which corresponds to the point where the conventional RNN begins to produce $S^z = 0$ configurations only. After this point is reached in training, both RNNs have similar performance with additional training epochs. We hypothesize that this acceleration arises from the effective reduction of the length of the sequences used in the backpropagation algorithm. Since these sequences are constructed multiplicatively, shortening them can reduce any uncontrolled amplification or suppression of the gradients. The resulting stabilization in the gradient evaluation is consistent with our observations of the path traversed by the parameters through the loss landscape, during the training of the RNN with and without U(1) symmetry.

It will be interesting to extend our study to quantum ground states with negative or complex expansion coefficients, and to systems motivated by other NISQ hardware [42]. For example, we anticipate that the exploitation of symmetries will also aid the simulation of quantum chemistry on NISQ hardware, since these systems are naturally endowed with a charge conservation symmetry such as the one studied in this work. Symmetry imposition could be especially beneficial in strategies that use neural-network estimators to increase the precision of measurements of physical observables in quantum chemistry [43]. Here, local fermionic observables are transformed into measurements over the entire set of qubits in a quantum computer via the Jordan-Wigner transformation, which increases the statistical errors of important observables significantly. Finally, speedups in early epoch training like those identified in this work may be particularly compelling when combined with NISQ applications that require rapid reconstruction of estimators in an iterative optimization, such as variational hybrid quantum-classical algorithms [44].

APPENDIX A: RBM

1. Theory

The RBM is a stochastic neural network comprising two layers: a *visible* layer $\boldsymbol{\sigma}$ and a *hidden* layer $\boldsymbol{h}$. The network contains parameters $\boldsymbol{\lambda} = (\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c})$, where $\boldsymbol{W}$ is a weight matrix connecting each node in the visible layer to each node in the hidden layer, and $\boldsymbol{b}$ and $\boldsymbol{c}$ are external bias fields for the visible and hidden layer, respectively. The RBM defines a probability distribution over its two layers,

$$p_{\boldsymbol{\lambda}}(\boldsymbol{\sigma}, \boldsymbol{h}) = \frac{e^{-E_{\boldsymbol{\lambda}}(\boldsymbol{\sigma}, \boldsymbol{h})}}{Z_{\boldsymbol{\lambda}}}, \tag{A1}$$

where

$$E_{\boldsymbol{\lambda}}(\boldsymbol{\sigma}, \boldsymbol{h}) = -\boldsymbol{\sigma}^{\mathsf{T}} \boldsymbol{W} \boldsymbol{h} - \boldsymbol{\sigma}^{\mathsf{T}} \boldsymbol{b} - \boldsymbol{h}^{\mathsf{T}} \boldsymbol{c}, \tag{A2}$$

and normalization

$$Z_{\boldsymbol{\lambda}} = \sum_{\boldsymbol{\sigma}, \boldsymbol{h}} e^{-E_{\boldsymbol{\lambda}}(\boldsymbol{\sigma}, \boldsymbol{h})}. \tag{A3}$$

The hidden layer is commonly comprised of n_h binary nodes, while in the context of physical systems of N spin-1/2 particles the visible layer corresponds to the N particles.

The marginal distribution over the visible layer is obtained by summing over the hidden layer:

$$p_{\boldsymbol{\lambda}}(\boldsymbol{\sigma}) = \sum_{\boldsymbol{h}} p_{\boldsymbol{\lambda}}(\boldsymbol{\sigma}, \boldsymbol{h}) = \frac{e^{-\mathcal{E}_{\boldsymbol{\lambda}}(\boldsymbol{\sigma})}}{Z_{\boldsymbol{\lambda}}}, \tag{A4}$$

where

$$\mathcal{E}_{\boldsymbol{\lambda}}(\boldsymbol{\sigma}) = -\boldsymbol{\sigma}^{\mathsf{T}} \boldsymbol{b} - \sum_{j=1}^{n_h} \ln \left[1 + \exp \left(c_j + \sum_{i=1}^{N} W_{ij} \sigma_i \right) \right]. \tag{A5}$$

An RBM can be trained to reconstruct a target distribution $q(\boldsymbol{\sigma})$ by minimizing the Kullback-Leibler (KL) divergence,

$$\mathrm{KL}_{\boldsymbol{\lambda}} = \sum_{\boldsymbol{\sigma}} q(\boldsymbol{\sigma}) \ln \left(\frac{q(\boldsymbol{\sigma})}{p_{\boldsymbol{\lambda}}(\boldsymbol{\sigma})} \right). \tag{A6}$$

The gradients of the KL divergence with respect to all RBM parameters $\boldsymbol{\lambda}$ can be approximated as

$$\nabla_{\boldsymbol{\lambda}} \mathrm{KL}_{\boldsymbol{\lambda}} \approx \langle \nabla_{\boldsymbol{\lambda}} \mathcal{E}_{\boldsymbol{\lambda}}(\boldsymbol{\sigma}) \rangle_{\mathcal{D}} - \langle \nabla_{\boldsymbol{\lambda}} \mathcal{E}_{\boldsymbol{\lambda}}(\boldsymbol{\sigma}) \rangle_{p_{\boldsymbol{\lambda}}}, \tag{A7}$$

where $\mathcal{D}$ is a dataset of samples from q,

$$\langle \nabla_{\boldsymbol{\lambda}} \mathcal{E}_{\boldsymbol{\lambda}}(\boldsymbol{\sigma}) \rangle_{\mathcal{D}} = \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{\sigma} \in \mathcal{D}} \nabla_{\boldsymbol{\lambda}} \mathcal{E}_{\boldsymbol{\lambda}}(\boldsymbol{\sigma}), \tag{A8}$$

termed the *positive phase* of the gradient, and

$$\langle \nabla_{\boldsymbol{\lambda}} \mathcal{E}_{\boldsymbol{\lambda}}(\boldsymbol{\sigma}) \rangle_{p_{\boldsymbol{\lambda}}} \approx \frac{1}{|\Gamma|} \sum_{\boldsymbol{\sigma}^{(k)} \in \Gamma} \nabla_{\boldsymbol{\lambda}} \mathcal{E}_{\boldsymbol{\lambda}}(\boldsymbol{\sigma}^{(k)}), \tag{A9}$$

termed the *negative phase* of the gradient. Here, Γ denotes samples generated from the RBM via contrastive divergence (CD_k) block-Gibbs sampling. Starting from an initial configuration $\boldsymbol{\sigma}^{(0)}$ from $\mathcal{D}$, one can sample a hidden layer configuration $\boldsymbol{h}^{(0)}$ via

$$p_{\boldsymbol{\lambda}}(h_j = 1 | \boldsymbol{\sigma}) = S \left(c_j + \sum_i W_{ij} \sigma_i \right), \tag{A10}$$

where S is the softmax function. From $\boldsymbol{h}^{(0)}$, we may sample a new visible configuration $\boldsymbol{\sigma}^{(1)}$ via

$$p_{\boldsymbol{\lambda}}(\sigma_j = 1 | \boldsymbol{h}) = S \left(b_i + \sum_j W_{ij} h_j \right). \tag{A11}$$

Continuing in this alternating fashion k times, we end with a configuration $\boldsymbol{\sigma}^{(k)}$.

The gradients of $\mathcal{E}_{\boldsymbol{\lambda}}$ with respect to all RBM parameters are as follows:

$$\frac{\partial \mathcal{E}_{\boldsymbol{\lambda}}(\boldsymbol{\sigma})}{\partial W_{ij}} = -p_{\boldsymbol{\lambda}}(h_j = 1 | \boldsymbol{\sigma}) \sigma_i, \tag{A12a}$$

$$\frac{\partial \mathcal{E}_{\boldsymbol{\lambda}}(\boldsymbol{\sigma})}{\partial c_j} = -p_{\boldsymbol{\lambda}}(h_j = 1 | \boldsymbol{\sigma}), \tag{A12b}$$

$$\text{and} \quad \frac{\partial \mathcal{E}_{\boldsymbol{\lambda}}(\boldsymbol{\sigma})}{\partial b_i} = -\sigma_i. \tag{A12c}$$

2. RBMs for quantum state reconstruction

Given a target quantum state, we wish to reconstruct an RBM as outlined in Sec. A 1 may be employed to reconstruct it if the target state has no sign structure and can therefore be written as

$$\psi(\boldsymbol{\sigma}) = \sqrt{q(\boldsymbol{\sigma})}. \tag{A13}$$

Given Eq. (A4), the RBM defines a trial state

$$\psi_{\boldsymbol{\lambda}}(\boldsymbol{\sigma}) = \sqrt{p_{\boldsymbol{\lambda}}(\boldsymbol{\sigma})}. \tag{A14}$$

Therefore, in reconstructing $q(\boldsymbol{\sigma})$ we also reconstruct the target state $\psi(\boldsymbol{\sigma})$.

All RBMs trained in this work were written using the QUCUMBER package [29] and employed SGD to perform the optimization of the hyperparameters in Table II. Note that one gradient update is comprised over many smaller updates over minibatches of $\mathcal{D}$. The size of the minibatches used to calculate Eq. (A8) is called the *positive batch size*, while $|\Gamma|$ in Eq. (A9) is called the *negative batch size*.

Six different random seeds to initialize the RBM parameters were also investigated. For all RBM results in this work, we chose the random seed that gave the best results in terms of the training metric [Eqs. (14) and (15)]. Regarding the hidden layer size n_h in the RBMs, it was observed in preliminary
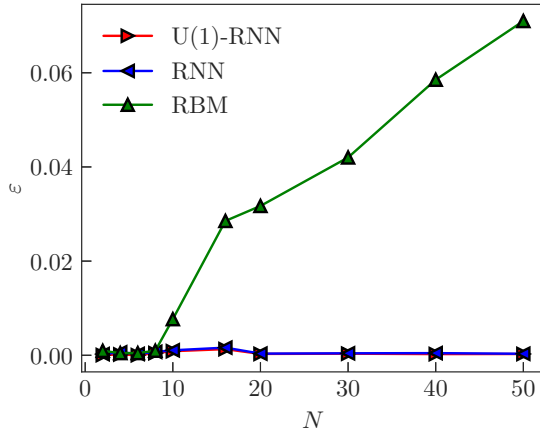
FIG. 8. Comparing the energy difference for larger generative models. After 2000 epochs of training were completed, 10^4 samples were generated from the U(1)-RNN, RNN, and RBM to respectively calculate ε. All networks were sufficiently converged at this time and error bars are smaller than the plot markers. After this time in the epoch regime, the RNN and U(1)-RNN yield nearly identical values of ε. In contrast, the RBM's ε is drastically higher than both of the RNNs for chain lengths greater than 8.

calculations that hidden layer sizes in Table II gave the best numerically stable results.

In Fig. 8, we show a plot of the energy difference across several different system sizes after 2000 epochs of training, i.e., where the optimal parameters $\boldsymbol{\theta}^*$ have been reached, for the U(1)-RNN, RNN, and RBM. For RNNs, the energy difference at this time shows minimal improvement regardless of whether symmetry was imposed or not. What is interesting here is the drastically different behavior of the RBM. For RBMs in this work, it was observed that hidden layer sizes greater than 100 yielded worse results (i.e., larger ε and infidelity) than what is reported. Not only this, but with larger hidden layer sizes often came less stable numerics; the training metrics did not come to a stable equilibrium. To try and ameliorate the training metrics and stabilize the numerics when larger hidden layer sizes were used, a quicker learning rate decay was employed in order to weaken the gradient updates. Unfortunately, this also did not bear any fruit.

Although exact properties of the ground state of the *XY* model can be extracted from RBMs analytically [47,48], the observation of the difficulty in training RBMs with widely used and robust methods for reconstructing the ground state of the *XY* model shows that the optimization problem is nontrivial for larger system sizes. A similar result was reported in Ref. [28]. With relatively little pretraining effort, we were able to achieve much better training metrics and numerical stability with RNNs. To summarize, RNNs seem to be much easier to optimize than RBMs in this context.

APPENDIX B: GATED RECURRENT NETWORKS

Due to the nature of their architecture, basic RNNs have trouble learning long-distance dependencies between qubits since the gradients with respect to RNN parameters tend to either explode or vanish. To overcome this, we employ the GRU introduced in Ref. [20]. These process data in a similar way to RNNs, defining

$$z_i = \sigma(W_z\boldsymbol{\sigma}_{i-1} + U_z\boldsymbol{h}_{i-1} + \boldsymbol{b}_z),$$

$$r_i = \sigma(W_r\boldsymbol{\sigma}_{i-1} + U_r\boldsymbol{h}_{i-1} + \boldsymbol{b}_r),$$

$$\hat{\boldsymbol{h}}_i = \tanh[W_h\boldsymbol{\sigma}_{i-1} + U_h(\boldsymbol{r}_i \odot \boldsymbol{h}_{i-1}) + \boldsymbol{b}_h],$$

$$\boldsymbol{h}_i = (1 - z_i) \odot \boldsymbol{h}_{i-1} + z_i \odot \hat{\boldsymbol{h}}_i,$$

where σ refers to the sigmoid function, and $\odot$ denotes elementwise multiplication. The new hidden state is an interpolation between the previous hidden state $\boldsymbol{h}_{i-1}$ and a proposed hidden state $\hat{\boldsymbol{h}}_i$, weighted by the update gate z_i. The reset gate $\boldsymbol{r}_i$ controls to what extent the proposed hidden state "remembers" the previous hidden state. So if $\boldsymbol{r}_i = 0$, the unit acts as though this is the first element of the sequence, making the proposed state "forget" the part of the sequence already encoded in $\boldsymbol{h}_{i-1}$. The output layer remains unchanged from the basic RNN.

The matrices $W_{h,r,z}$ and $U_{h,r,z}$ and biases $\boldsymbol{b}_{h,r,z}$ are all filled with trainable parameters. For hidden units with dimension n_h and inputs with dimension n_v, then $W_{z,r,h} \in \mathbb{R}^{n_h \times n_v}$, $U_{z,r,h} \in \mathbb{R}^{n_h \times n_h}$, and $\boldsymbol{b}_{z,r,h} \in \mathbb{R}^{n_h}$. The rest of the training procedure remains unchanged from how it operates in a regular RNN.

[1] G. Torlai and R. G. Melko, Learning thermodynamics with Boltzmann machines, Phys. Rev. B **94**, 165134 (2016).

[2] G. Torlai, G. Mazzola, J. Carrasquilla, M. Troyer, R. Melko, and G. Carleo, Neural-network quantum state tomography, Nat. Phys. **14**, 447 (2018).

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

[4] J. Carrasquilla, Machine learning for quantum matter, Adv. Phys.: X **5**, 1797528 (2020).

[5] G. Torlai, B. Timar, E. P. L. van Nieuwenburg, H. Levine, A. Omran, A. Keesling, H. Bernien, M. Greiner, V. Vuletić, M. D. Lukin *et al.*, Integrating Neural Networks with a Quantum Simulator for State Reconstruction, Phys. Rev. Lett. **123**, 230504 (2019).

[6] R. G. Melko, G. Carleo, J. Carrasquilla, and J. I. Cirac, Restricted Boltzmann machines in quantum physics, Nat. Phys. **15**, 887 (2019).

[7] G. Torlai and R. G. Melko, Machine-learning quantum states in the NISQ era, Annu. Rev. Condens. Matter Phys. **11**, 325 (2020).

[8] J. Carrasquilla, G. Torlai, R. G. Melko, and L. Aolita, Reconstructing quantum states with generative models, Nat. Mach. Intell. **1**, 155 (2019).

[9] D. Wu, L. Wang, and P. Zhang, Solving Statistical Mechanics Using Variational Autoregressive Networks, Phys. Rev. Lett. **122**, 080602 (2019).

[10] O. Sharir, Y. Levine, N. Wies, G. Carleo, and A. Shashua, Deep Autoregressive Models for the Efficient Variational Simulation of Many-Body Quantum Systems, Phys. Rev. Lett. **124**, 020503 (2020).

[11] C. Roth, Iterative retraining of quantum spin models using recurrent neural networks, arXiv:2003.06228.

[12] P. Cha, P. Ginsparg, F. Wu, J. Carrasquilla, P. L. McMahon, and Eun-Ah Kim, Attention-based quantum tomography, arXiv:2006.12469.

[13] P. Richerme, Zhe-Xuan Gong, A. Lee, C. Senko, J. Smith, M. Foss-Feig, S. Michalakis, A. V. Gorshkov, and C. Monroe, Non-local propagation of correlations in quantum systems with long-range interactions, Nature (London) **511**, 198 (2014).

[14] P. Jurcevic, B. P. Lanyon, P. Hauke, C. Hempel, P. Zoller, R. Blatt, and C. F. Roos, Quasiparticle engineering and entanglement propagation in a quantum many-body system, Nature (London) **511**, 202 (2014).

[15] C. Monroe, W. C. Campbell, L.-M. Duan, Z.-X. Gong, A. V. Gorshkov, P. Hess, R. Islam, K. Kim, N. Linke, G. Pagano *et al.*, Programmable quantum simulations of spin systems with trapped ions, Rev. Mod. Phys. **93**, 25001 (2021).

[16] M. Hibat-Allah, M. Ganahl, L. E. Hayward, R. G. Melko, and J. Carrasquilla, Recurrent neural network wave functions, Phys. Rev. Res. **2**, 023358 (2020).

[17] Z. C. Lipton, J. Berkowitz, and C. Elkan, A critical review of recurrent neural networks for sequence learning, arXiv:1506.00019.

[18] P. Tarazaga, M. Raydan, and A. Hurman, Perron–Frobenius theorem for matrices with some negative entries, Linear Algebra Appl. **328**, 57 (2001).

[19] Y. Bengio, P. Simard, and P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Trans. Neural Netw. **5**, 157 (1994).

[20] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, On the properties of neural machine translation: Encoder–decoder approaches, in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation* (Association for Computational Linguistics, Doha, Qatar, 2014), pp. 103–111.

[21] G. E. Hinton, Training products of experts by minimizing contrastive divergence, Neural Comput. **14**, 1711 (2002).

[22] R. Pascanu, T. Mikolov, and Y. Bengio, On the difficulty of training recurrent neural networks, in *Proceedings of the 30th International Conference on Machine Learning (ICML '13)* (PMLR, Atlanta, GA, 2013), pp. 1310–1318.

[23] R. J. Williams and J. Peng, An efficient gradient-based algorithm for on-line training of recurrent network trajectories, Neural Comput. **2**, 490 (1990).

[24] ITensor Library (version 2.0.11), http://itensor.org.

[25] S. R. White, Density-matrix algorithms for quantum renormalization groups, Phys. Rev. B **48**, 10345 (1993).

[26] S. R. White, Density Matrix Formulation for Quantum Renormalization Groups, Phys. Rev. Lett. **69**, 2863 (1992).

[27] A. J. Ferris and G. Vidal, Perfect sampling with unitary tensor networks, Phys. Rev. B **85**, 165146 (2012).

[28] I. J. S. De Vlugt, D. Iouchtchenko, E. Merali, Pierre-Nicholas Roy, and R. G. Melko, Reconstructing quantum molecular rotor ground states, Phys. Rev. B **102**, 035108 (2020).

[29] M. J. S. Beach, I. De Vlugt, A. Golubeva, P. Huembeli, B. Kulchytskyy, X. Luo, R. G. Melko, E. Merali, and G. Torlai, QuCumber: wavefunction reconstruction with neural networks, SciPost Phys. **7**, 9 (2019).

[30] D. Sehayek, A. Golubeva, M. S. Albergo, B. Kulchytskyy, G. Torlai, and R. G. Melko, Learnability scaling of quantum states: Restricted Boltzmann machines, Phys. Rev. B **100**, 195125 (2019).

[31] C. Tosh, Mixing rates for the alternating Gibbs sampler over restricted Boltzmann machines and friends, in *Proceedings of the 33rd International Conference on Machine Learning, ICML 2016* (JMLR.org, 2016), pp. 840–849.

[32] Q. Wang, X. Gao, K. Wan, F. Li, and Z. Hu, A novel restricted Boltzmann machine training algorithm with fast Gibbs sampling policy, Math. Probl. Eng. **2020**, 4206457 (2020).

[33] M. Á. Carreira-Perpiñan and G. Hinton, On contrastive divergence learning, in *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics* (Society for Artificial Intelligence and Statistics, 2005), pp. 33–40.

[34] G. Carleo, M. Troyer, G. Torlai, R. G. Melko, J. Carrasquilla, and G. Mazzola, Neural-network quantum states, Bull. Am. Phys. Soc. **2018**, 10 (2018).

[35] G. E. Hinton, A practical guide to training restricted Boltzmann machines, in *Neural Networks: Tricks of the Trade*, edited by G. Montavon, G. B. Orr, and K.-R. Müller (Springer, Berlin/Heidelberg, 2012), Vol. 7700, p. 599.

[36] R. Salakhutdinov, A. Mnih, and G. Hinton, Restricted Boltzmann machines for collaborative filtering, in *Proceedings of the 24th International Conference on Machine Learning* (Association for Computing Machinery, New York, 2007), pp. 791.

[37] T. Giacomo, Augmenting Quantum Mechanics with Artificial Intelligence, PhD thesis, University of Waterloo, 2018.

[38] G. Torlai and R. G. Melko, Latent Space Purification Via Neural Density Operators, Phys. Rev. Lett. **120**, 240503 (2018).

[39] I. J. Goodfellow, O. Vinyals, and A. M. Saxe, Qualitatively characterizing neural network optimization problems, arXiv:1412.6544.

[40] D. J. Im, M. Tao, and K. Branson, An empirical analysis of the optimization of deep network loss surfaces, arXiv:1612.04010.

[41] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, Visualizing the loss landscape of neural nets, arXiv:1712.09913.

[42] J. Preskill, Quantum computing in the NISQ era and beyond, Quantum **2**, 79 (2018).

[43] G. Torlai, G. Mazzola, G. Carleo, and A. Mezzacapo, Precise measurement of quantum observables with neural-network estimators, Phys. Rev. Res. **2**, 022060(R) (2020).

[44] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, The theory of variational hybrid quantum-classical algorithms, New J. Phys. **18**, 023023 (2016).

[45] J. D. Hunter, Matplotlib: A 2D graphics environment, Comput. Sci. Eng. **9**, 90 (2007).

[46] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith *et al.*, Array programming with NumPy, Nature (London) **585**, 357 (2020).

[47] E. Rrapaj and A. Roggero, Exact representations of many body interactions with RBM neural networks, Phys. Rev. E **103**, 013302 (2021).

[48] D. L. Deng, X. Li, and S. Das Sarma, Quantum Entanglement in Neural Network States, Phys. Rev. X **7**, 021021 (2017).

[49] www.vectorinstitute.ai/#partners.