New Journal of Physics
The open access journal at the forefront of physics

Deutsche Physikalische Gesellschaft DPG IOP Institute of Physics

PAPER • OPEN ACCESS

Unsupervised learning of Rydberg atom array phase diagram with Siamese neural networks

To cite this article: Zakaria Patel *et al* 2022 *New J. Phys.* **24** 113021

New Journal of Physics
The open access journal at the forefront of physics

Deutsche Physikalische Gesellschaft DPG
IOP Institute of Physics

Published in partnership with: Deutsche Physikalische Gesellschaft and the Institute of Physics

OPEN ACCESS

RECEIVED
9 May 2022

REVISED
7 October 2022

ACCEPTED FOR PUBLICATION
21 October 2022

PUBLISHED
15 November 2022

PAPER

Unsupervised learning of Rydberg atom array phase diagram with Siamese neural networks

Zakaria Patel[1], Ejaaz Merali[2,3] and Sebastian J Wetzel[3,*]

[1] Department of Engineering Physics, McMaster University, Hamilton, Ontario L8S 4L8, Canada
[2] Department of Physics and Astronomy, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada
[3] Perimeter Institute for Theoretical Physics, Waterloo, Ontario N2L 2Y5, Canada
[*] Author to whom any correspondence should be addressed.

E-mail: swetzel@perimeterinstitute.ca

Keywords: artificial neural networks, phase transitions, Ising model, Rydberg array

Abstract

We introduce an unsupervised machine learning method based on Siamese neural networks (SNNs) to detect phase boundaries. This method is applied to Monte-Carlo simulations of Ising-type systems and Rydberg atom arrays. In both cases the SNN reveals phase boundaries consistent with prior research. The combination of leveraging the power of feed-forward neural networks, unsupervised learning and the ability to learn about multiple phases without knowing about their existence provides a powerful method to explore new and unknown phases of matter.

1. Introduction

Machine learning (ML) algorithms enable computers to learn from experience and generalize their gained knowledge to previously unknown settings. It is perhaps the most transformative technology of the early 21st century. The ability to recognize objects in images [1] or translate languages [2] without being explicitly programmed for this task, highlights the enormous potential of ML.

In recent years the physical sciences have adopted ML based algorithms to explore complex questions. Many methods have been designed to solve problems beyond the scope of data science, and have now the potential to revolutionize physics. The most prominent examples of promising tasks that have been tackled include finding phase transitions [3–13], reconstructing or simulating quantum systems [14–20] and rediscovering physical concepts [21–30]. All these advances are summarized in review articles directed at different audiences. A didactical review to the most modern techniques can be found in [31], a review article focused on the applications of ML to examine quantum matter [32] and a broad overview across different physical disciplines is summarized in [33].

The current manuscript contributes to the development of methods that automatize the calculation of phase diagrams with little to no human prior knowledge about the nature of the underlying phases. The subfield of automated phase recognition can be subdivided into two categories: supervised and unsupervised phase recognition.

In the first case, the operating scientists are aware of the possible phases and have a rough estimate of where these phases are positioned in the phase diagram; however, they are unsure about the exact location of the phases and the transitions between them. Supervised learning of phase transitions can be based on different ML algorithms. It was initially introduced using convolutional neural networks [3], which are to this day the most powerful and robust tools to learn accurate physical phase boundaries. There are hybrid methods that build upon purposely mislabelling phase classes [4], methods that are built upon support vector machines [24], and other powerful frameworks [7, 34–40]. These methods have demonstrated success across a wide range of physical systems, from simple spin lattices, over strongly correlated quantum systems, up to lattice gauge theories [8, 10, 11, 13, 23, 41–51].

The second category contains unsupervised phase recognition algorithms. These algorithms are useful when the researcher who is employing these tools is unaware of the underlying phase structure, meaning

they do not know about the existence or location of certain phases and thus cannot supply this information to the ML algorithm. The simplest unsupervised phase recognition scheme is based on principal component analysis [5] and the most widely used unsupervised scheme that leverages the power of artificial neural networks is based on variational autoencoders [6]. These methods have been examined, enhanced [9, 52–55] and successfully applied to many systems in physics and materials science [26, 56–66]. Compared to supervised algorithms, unsupervised methods usually have the drawback of lacking accuracy in determining phase boundaries [6], or restricting the kinds of order parameters that can be learned [5].

We introduce an unsupervised ML method to discover phase transitions based on Siamese neural networks (SNNs). Siamese networks were initially introduced for fingerprint and signature identification [67, 68]. Instead of predicting a certain class, Siamese networks predict if two inputs belong to the same class. Hence, these networks can be used for multi- or infinite class classification by comparison to anchor data points whose label is known. Although Siamese networks are very powerful, they have experienced little use in the physical sciences. So far Siamese networks have been employed to discover symmetry invariants and conserved quantities [25]. While SNNs are supervised ML algorithms, our proposed phase recognition method is unsupervised, in the sense that it does not require any phase information. This apparent contradiction is reconciled in section 3.4.

While we initially present our phase recognition method using the example of two stacked Ising models exhibiting four different phases, we demonstrate the power of this method by examining the phase diagram of the Rydberg atom array. Rydberg atom arrays are a powerful platform for experimental realizations of quantum many-body phenomena [69–71]. Neutral atoms are typically arranged via optical tweezers to construct various physical lattices at varying interaction strengths which give rise to rich phase diagrams [72]. Such systems have already been examined with the help of ML algorithms. The phase diagram has been revealed by a combined effort of unsupervised and supervised methods [73]. Experimental states have been reconstructed using neural network based tomography [74]. Ground states have been calculated [75] and simulated measurement data has been used for pre-training variational wave functions [76].

The paper is structured in the following order: we first introduce the models we are examining with our new Siamese network based framework. These models include a stacked Ising model and the Rydberg atom array. Subsequently, we describe how we prepare the input data using Monte Carlo simulations. We describe how SNNs are constructed and trained, and develop our framework to do unsupervised learning of phase transitions. Then we apply our method to both models and present the results in the form of the predicted underlying phase structure. Finally, we summarize our findings and place our method into the broader context of ML tools for phase recognition.

2. Models

2.1. Stacked Ising model

The Ising model on the square lattice is a simple, well studied, and exactly solvable model from statistical physics that exhibits a phase transition. Thus, it provides the ideal starting point for benchmarking the performance of a phase recognition algorithm. Its Hamiltonian is

$$H(S) = -J\sum_{\langle ij\rangle_{nn}} s_i s_j + h\sum_i s_i, \tag{1}$$

a function of spin configurations $S = (s_1, \ldots, s_n)$. In the following, we focus on the ferromagnetic Ising model $J = 1$ and set the external field $h = 0$.

SNN phase recognition is an algorithm that is able to detect multiple phases in an unsupervised manner. Hence, we trivially combine two Ising models by overlaying them on the same lattice where we can tune each temperature $T, \tilde{T}$, or in other words the ratios $J/k_B T$ and $\tilde{J}/k_B \tilde{T}$, independently. The combined Hamiltonian

$$H(S) = -J\sum_{\langle ij\rangle_{nn}} s_i s_j - \tilde{J}\sum_{\langle ij\rangle_{nn}} \tilde{s}_i \tilde{s}_j \tag{2}$$

acts on lattices containing two spins per site, $S = ((s_1, \tilde{s_1}), \ldots, (s_n, \tilde{s_n}))$. Since there is no interaction between them, phase transitions trivially occur at lines $(T, \tilde{T}) \approx (\cdot, 2.269)$ and $(T, \tilde{T}) \approx (2.269, \cdot)$ in the phase diagram (figure 1).

2.2. Rydberg array

A common Hamiltonian that can be implemented by Rydberg arrays has a form similar to that of a transverse field Ising model, meaning it is sign-problem free and thus amenable to simulation on classical computers. The Hamiltonian acts on a collection of atoms which individually act like two-level systems,

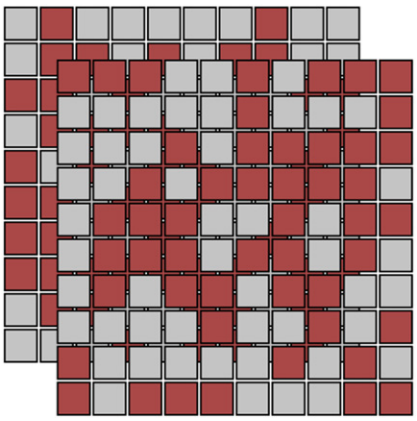

Figure 1. Combining the Ising lattices involves the simple stacking shown in this figure. The red sites indicate spin-up, and the grey sites correspond to spin-down. The two lattice configurations are independently sampled at temperatures T and $\tilde{T}$ to produce a single stacked configuration at $(T, \tilde{T})$.

having a ground-state $|g\rangle \equiv |0\rangle$ and an excited, so-called *Rydberg* state $|r\rangle \equiv |1\rangle$. The atoms are subject to a long-range interaction which is described by a van der Waals (vdW) interaction of the form $V_{ij} \sim |\mathbf{r}_i - \mathbf{r}_j|^{-6}$ that penalizes atoms that are simultaneously in the Rydberg state [77]. Additionally, the atoms are subject to coherent laser fields: a *detuning* δ which acts like a chemical potential, driving atoms into their Rydberg states, and a *Rabi oscillation* with frequency Ω which excites ground-state atoms and de-excites atoms in Rydberg states.

$$H = \sum_{i<j} V_{ij} n_i n_j - \delta \sum_i n_i + \frac{\Omega}{2} \sum_i \sigma_i^x, \tag{3}$$

where $n_i = |1\rangle\langle 1|_i$ is the occupation/number operator, and $\sigma_i^x = |1\rangle\langle 0|_i + |0\rangle\langle 1|_i$. The interaction strength is typically parametrized in terms of a *Rydberg blockade radius* R_b, which describes an effective radius within which two simultaneous Rydberg excitations are heavily penalized: $V_{ij} = \Omega(R_\mathrm{b}/|\mathbf{r}_i - \mathbf{r}_j|)^6$ [78, 79].

3. Methods

3.1. Monte-Carlo simulation

The well-known single-spin-flip Metropolis algorithm is used to generate importance sampled Monte Carlo configurations of the Ising model on a square lattice of size 20×20 with periodic boundary conditions [80]. After initializing a random lattice, we evolve the simulation for 7168 MC steps between drawing samples. It is important to note that neural networks can pick up on any residual correlations, thus relying on conventional auto-correlation measures to determine the independence of lattice configurations is not enough. We produce 92 independent configurations at each of 100 temperatures ranging from $T_\mathrm{min} = 1.53$ to $T_\mathrm{max} = 3.28$. This naturally translates to $92 \times 92 = 8464$ samples for the stacked Ising model at each temperature pair $(T, \tilde{T})$.

For the Rydberg system, we make use of a recent quantum Monte Carlo (QMC) method [79] based on the stochastic series expansion (SSE) formalism [81–83] to generate occupation basis samples of the Rydberg Hamiltonian. The QMC simulation is based on a power-iteration scheme which projects out the ground-state of the Hamiltonian [84, 85]:

$$|E_0\rangle \approx (C - H)^M |+\rangle^{\otimes N}, \tag{4}$$

where $|+\rangle$ is the positive eigenstate of σ_x, N is the number of lattice sites, C is a constant energy shift used to cure the sign-problem emerging from the diagonal part of the Hamiltonian, and M is called the projection length. We perform our simulations on a 16×16 square lattice with open boundaries at various parameter values. Unlike previous DMRG-based studies [72, 78] we do not impose a truncation on the vdW interaction. We take our projection length M to be 100 000 which we found was more than enough to accurately converge to the ground-state over the parameter sets which were simulated. For our simulations we fixed $\Omega = 1$ and performed scans over $R_\mathrm{b} \in \{1.0, 1.1, \ldots, 1.8\}$ and $\delta \in \{0.5, 0.6, \ldots, 2.9\}$.

To generate the occupation basis data for the SNNs, we first perform $N_\mathrm{MC} = 100\,000$ Monte Carlo update steps to allow the chain to reach equilibrium. We then record one sample every 10 000 steps in order

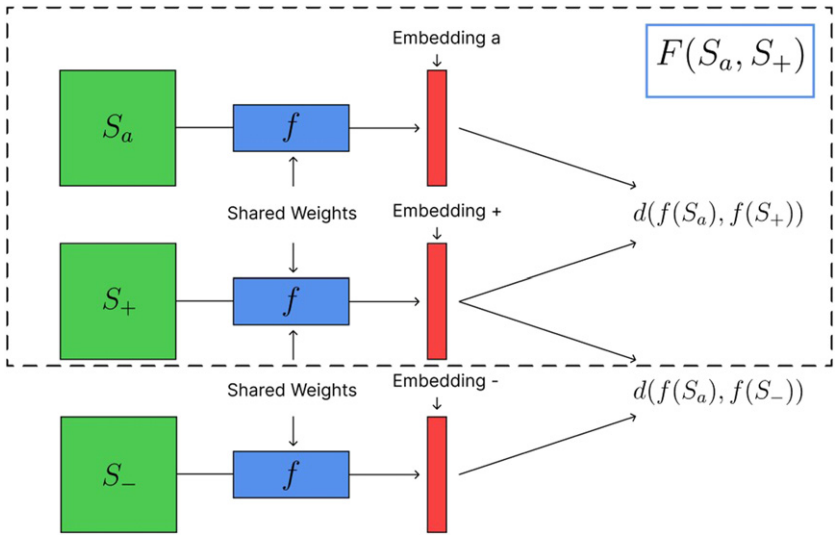

Figure 2. A SNN F takes a pair of input samples (S_i, S_j) and predicts their similarity $d(f(S_i), f(S_j))$ The training architecture effectively mimics two SNNs acting on a triplet (S_a, S_+, S_-) of samples. S_a represents the anchor input, S_+ the positive sample from the same class as the anchor, and S_- the negative sample from a different class. The shared convolutional subnetworks f map their inputs into an embedding space. The SNN is trained to minimize distances $d(f(S_a), f(S_+))$, while maximizing distances $d(f(S_a), f(S_-))$. During the inference stage, we discard one branch of this network to obtain a pairwise comparison between the anchor and a new unknown sample.

to eliminate any possible autocorrelation between successive samples[4]. Each Monte Carlo step consists of a diagonal update step, followed by a cluster update step in which all possible line-clusters are built deterministically and flipped independently according to a Metropolis condition; see [79] for further details. Additionally, at each point in parameter space (R_b, δ) we run 3 independent Markov chains. The chains are allowed to evolve until each has generated 400 samples, giving a total of 1200 (approximately) independent samples for each parameter pair.

3.2. Siamese neural networks

Artificial neural networks are directed graphs that have the ability to learn an approximation to any smooth function $f(x) = y$ given sufficiently many parameters. A neural network is built by successively applying matrix multiplications characterized by weights w_{ij}^L that are offset by biases b_i^L. $(i, j$ are neuron indices in different layers $L)$. Between subsequent matrix multiplications there is a non-linear activation function, common choices of which are sigmoid or rectified linear units (ReLUs). A neural network is trained by applying it to a data set and optimizing the network parameters to minimize a certain objective function using gradient descent.

SNNs were introduced to solve an infinite class classification problem as it occurs in finger print recognition or signature verification [67, 68]. Instead of assigning a class label to a data instance, the SNN compares two data points and determines their similarity. A solution to the infinite class classification problem is obtained by calculating the predicted similarity between labelled anchor data points and a new unlabelled data instance.

A SNN $F(S_i, S_j)$ (figure 2) consists of two identical sub-networks f (figure 3) which project an input pair into a latent embedding space. The similarity of two inputs is determined based on the distance in embedding space.

$$F(S_i, S_j) = d\big(f(S_i), f(S_j)\big) \tag{5}$$

Possible distance metrics d must be chosen according to the problem at hand, in our case we chose the average squared Euclidean distance on the unit sphere which is equivalent to the cosine distance.

$$d(f_1, f_2) = \frac{\sum_{i=0}^{N}(f_{1i} - f_{2i})^2}{N}, \tag{6}$$

Here f_{1i}, f_{2i} denotes different components in an N dimensional embedding space. Instead of training the SNN on paired training data, an effective way to train SNNs is through minimizing contrastive loss

[4] We computed integrated autocorrelation times, τ (defined such that the effective sample size is given by $N_{\text{eff}} = N_{\text{MC}}/\tau$), of equation (17) at the Fourier space points using a logarithmic binning analysis. We found that for each of these Fourier modes, autocorrelation times were in the range 1–3 after skipping 10 000 samples between each measurement.

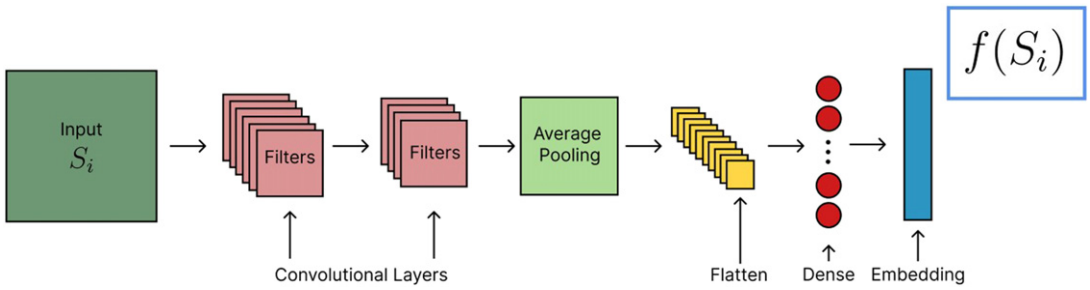

Figure 3. The architecture of the convolutional subnetwork f in the SNN F. The input S_i is first passed through 2 convolutional layers, followed by an average pooling to capture a quantity characterized by an average. The result is flattened in order to feed it to a dense layer. This dense layer is then connected to an embedding layer.

functions involving triplets (S_a, S_+, S_-) of data points

$$\mathcal{L} = \max(d(f(S_a), f(S_+)) - d(f(S_a), f(S_-)) + \alpha, 0), \tag{7}$$

where the hyperparameter α is chosen such that the neural network is prevented from learning trivial embeddings. The intuition behind α is its interpretation as a margin to encourage separation between the anchor and the negative embedding in the embedding space [86]. Minimizing $\mathcal{L}$ requires minimizing the distance between the anchor and positive sample $d(f(S_a), f(S_+))$, while maximizing the distance between the anchor and negative sample $d(f(S_a), f(S_-)$. $\max(\cdot, 0)$ prevents the neural network from learning runaway embeddings $f(S_i)_i \to \infty$.

3.3. Model architecture

The explicit model architecture for f figure 3 depends on the underlying data set. In the case of the Ising model, f consists of two 2D convolutional layers, both with stride $(1, 1)$, a kernel size of $(3, 3)$, and with 6 and 10 filters, respectively. Each layer is fed into a ReLU activation function. The ReLU activation function returns $f(x) = \max(0, x)$, introducing non-linearities into the neural network. The resulting image dimensions are $(16, 16)$. This is followed by a $(16, 16)$ average pooling layer. The output is flattened and fed to a dense layer with 10 neurons. Subsequently, we feed this output to the embedding layer, which also contains 10 neurons and a sigmoid activation function. The embedding is normalized to unit length under Euclidean norm.

The model architecture for examining the Rydberg system is similar to the Ising model. In this case, we begin with two 2D convolutional layers, both with stride (1, 1), a kernel size of (3, 3), and with 6 and 4 filters, respectively. The resulting image dimensions are (12, 12). As before, we subsequently apply an average pooling layer, but this time of dimension (12, 12). The remainder of the architecture is identical to before.

We use the Adam optimizer to train our neural network. Furthermore, our training procedure involves the early stopping callback. This technique involves training as long the loss is decreasing. If the loss is not decreasing for m epochs, training is stopped. The value of m is known as the patience. We set the maximum number of epochs to 150, which is enough to allow the callback to decide when to terminate the training process. We observe that small values of patience, around $m = 1 \ldots 3$ and $m = 6$ are sufficient for training on the Ising model and the Rydberg system, respectively. Additionally, we use a learning rate of $\alpha_{\text{lr}} = 0.001$ for the Ising model, and $\alpha_{\text{lr}} = 0.0005$ for the Rydberg system. Another hyperparameter is the margin for the contrastive loss, where we use $\alpha = 0.4$. We employ the TensorFlow and Keras libraries to implement the network, training, and callbacks.

3.4. Phase boundaries from Siamese networks

3.4.1. Supervised learning of phase transitions

Since the proposal to use supervised ML algorithms for calculating phase diagrams, the most powerful method still remains using feed-forward neural networks for the binary classification of Monte-Carlo samples [3]. In this case a neural network is trained on configurations from known parts of the phase diagram labelled by their phase. By denoting the phases with binary labels $y \in [0, 1]$, a neural network f is trained to predict the phase of a configuration S

$$f(S) = \begin{cases} \text{Phase A} \\ \\ \text{Phase B} \end{cases} . \tag{8}$$

After training, this neural network is then applied to samples from unknown parts of the phase diagram. Since these networks intrinsically learn the underlying physical features characterizing the phases like order parameters and other thermodynamic quantities [23], the predictions of the neural networks flip from one label to the other at the position of the phase transition.

The principle that guides us through the development of an unsupervised Siamese network-based scheme for phase recognition is to leverage the power of neural networks for phase classification in the supervised setting. This is done by reformulating the task of predicting phases by a neural network. A SNN takes a pair of input configurations and predicts if they are similar or different with respect to a metric imposed by the objective function during training.

$$F(S_1, S_2) = \begin{cases} \text{Same} \quad \text{phase} \\ \\ \text{Different} \quad \text{phase} \end{cases} \tag{9}$$

In this formulation a SNN can be used as a supervised algorithm for multi-phase recognition. A truly novel discovery of and unknown phase of matter cannot be made with supervised learning, since that would imply we already have some knowledge about the existence and position of such a phase. In order to do unsupervised learning of phases with SNNs the training data cannot be supplied with phase labels.

In each update step the Siamese network is trained on triplets (S_a, S_+, S_-), see figure 2, where S_a is called anchor, S_+ the positive comparison, and S_- the negative comparison. In an unsupervised setting we do not have the true phase labels at hand. However, we know that two samples from the exact same point on the phase diagram must have the same phase. Thus, we create training batches where S_a and S_+ are from the same coordinates in the phase diagram, while S_- is sampled randomly from anywhere in the phase diagram. If S_+ and S_- stem from the same phase the training signals should not interfere with the correct training signals from correctly labeled input. As it turns out, such a signal gets approximately canceled out, such that the remaining noise is subleading compared to the signal that is obtained when S_+ and S_- are from different phases. In a physical context, the noise might stem from thermal fluctuations, and the leading signal from relevant thermodynamical quantities.

Let us now examine gradient cancellations during the training of neural networks to understand why training signals from pairs where S_+ and S_- come from the same phase do not negatively impact training.

3.4.2. Gradient manipulation

The purpose of this section is to provide an understanding of how choosing suitable training signals will affect the training of SNNs. From this understanding it will be clear why we chose the architecture depicted in figure 2 to ensure the approximate cancellation of unwanted training signals.

It is important to understand how gradients update the neural network. Let us discuss this at the example of a general neural network h trained in a supervised setting to minimize the mean square error loss function on a labelled data set (X, Y). The discussion can be extended to any network in this manuscript. The effect of a single training example $(x, y) \in (X, Y)$ on the loss function is

$$L(h(x), y) = (h(x) - y)^2 \tag{10}$$

The neural network switches its prediction at the decision boundary

$$h(x) \begin{cases} < 0.5 & \text{class A} \\ \\ > 0.5 & \text{class B} \end{cases} \tag{11}$$

Neural networks are trained using backpropagation of gradients. A typical neural network contains millions of parameters that are tweaked based on backpropagation. In our particular case, however, the SNN is quite small, containing only 884 parameters. Let us focus on the gradient signal on an example weight w out of the millions of parameters characterizing a typical neural network. Let us further assume we have two identical training configurations x with opposite labels labels $y \in [0, 1]$. Each update step involves the product of the learning rate η and the inverse of the derivative of the loss function with respect to w:

$$w_{\text{new}} = -\eta \, \partial_w L(h(x), y) \tag{12}$$

The update depends on the label y:

$$\partial_w L = (h(x) - y)^2|_{h(x)=0.5} = \begin{cases} \partial_w h(x) & y = 0 \\ -\partial_w h(x) & y = 1 \end{cases} \tag{13}$$

Thus, by supplying the neural network with two similar training samples, but opposite labels in the same training step, their effect on the weights of the neural networks would approximately cancel each other out. While this combined training signal forces the neural network to be more uncertain $h(x) \to 0.5$, it will never change the prediction itself.

3.4.3. Comments on unsupervised learning of phase transitions

We emphasize that in this step we do not have access to any true phase labels and thus cannot make use of them; this is why the method is unsupervised. Something similar happens in other unsupervised methods, for example k-means-clustering, in this case the user needs to assign each sample to a cluster which would initially give the wrong label, but during training the cluster will be adjusted to a clustering solution.

Since there is still no comprehensive theory on neural network training dynamics, the above discussion lacks in mathematical rigor. Hence, in line with all other ML based phase recognition methods, the only way to convince ourselves of the capabilities of the proposed method is an empirical study by applying the method to physical systems.

4. Results

For the purpose of calculating phase diagrams, we train SNNs as outlined in the previous paragraphs for both the stacked Ising model and the Rydberg atom array. We create training triplets (S_a, S_+, S_-) containing a randomly sampled anchor configuration S_a, a positive configuration S_+ sampled from the same point in the phase diagram and a negative configuration S_- sampled from any other point in the phase diagram. After having successfully trained a SNN it is employed to perform pairwise comparisons on configurations along a specified one-dimensional slice through the phase diagram.

4.1. Adjacency comparisons

In this scheme, a configuration corresponding to a point in the phase diagram is compared to its neighbors within a certain distance. At the example of a single Ising model, this means a configuration from temperature T_n is compared to a configuration at T_{n+k}, where n is the temperature list index and k is a constant shift. The value of k can be treated as a hyperparameter. A comparison between configurations at T_n and T_{n+k} is assigned a temperature of $\frac{1}{2}(T_n + T_{n+k})$.

Our similarity measure of choice is the *normalized cosine dissimilarity* scaled to a range $[0, 1]$, where 1 corresponds to the empirically found maximal dissimilarity and 0 to the empirically determined minimal similarity. The cosine dissimilarity is related to the cosine distance via $1 - \cos(\theta)$, where

$$\cos(\theta) = \frac{f(S_n) \cdot f(S_{n+k})}{\|f(S_n)\| \|f(S_{n+k})\|} \tag{14}$$

The cosine distance is equivalent to the Euclidean distance when acting on the SNN embedding space normalized to the unit sphere ($\|f(S_i)\|^2 = 1$):

$$\|f(S_n) - f(S_{n+k})\|^2 = 2 - 2\ \cos(\theta) \tag{15}$$

Since the neural network f is designed to output positive values $f(S_n)_i \in [0, 1]$, the cosine dissimilarity takes on its minimum $1 - \cos(\theta) = 0$ for similar embeddings and its maximum $1 - \cos(\theta) = 1$ if the embeddings are dissimilar.

The result of scanning across a phase transition is depicted schematically in figure 4(c). The highest peak will indicate the SNN prediction of the phase transition.

4.2. Ising model

We examine the results of applying SNN unsupervised phase recognition to the 20×20 stacked Ising model. Analytically, the phase boundaries of the stacked Ising model in the thermodynamic limit are $(T, \tilde{T}) = (2.269, \cdot)$ and $(T, \tilde{T}) = (\cdot, 2.269)$. The phase transition temperature is prone to finite size effects as the lattice becomes smaller. If the phase transition would be calculated using the magnetization, finite size effects would distort the phase boundaries to $(T, \tilde{T}) \approx (2.36, \cdot)$ and $(T, \tilde{T}) \approx (\cdot, 2.36)$; however, it is important to note that the quantities a neural network learns might differ from the magnetization and thus experience different finite size scalings [57].

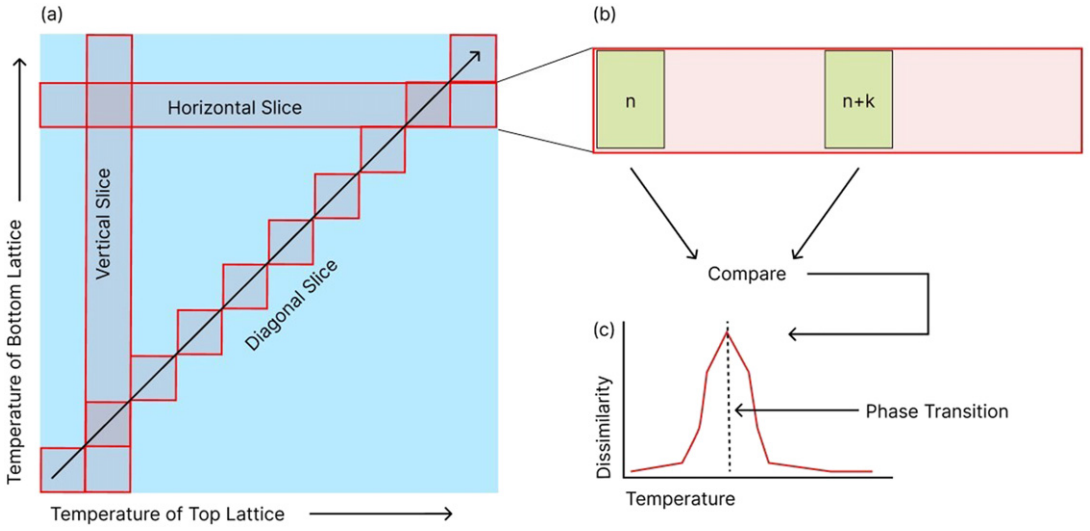

Figure 4. (a) The horizontal and vertical slices extend through the parameter grid, while the diagonal slice pierces through the center of the lattice. (b) Within each slice, we compare configurations at indices n and $n + k$ in order to (c) evaluate their dissimilarity. The temperature value at which the peak is situated in (c) corresponds to the predicted phase transition temperature.

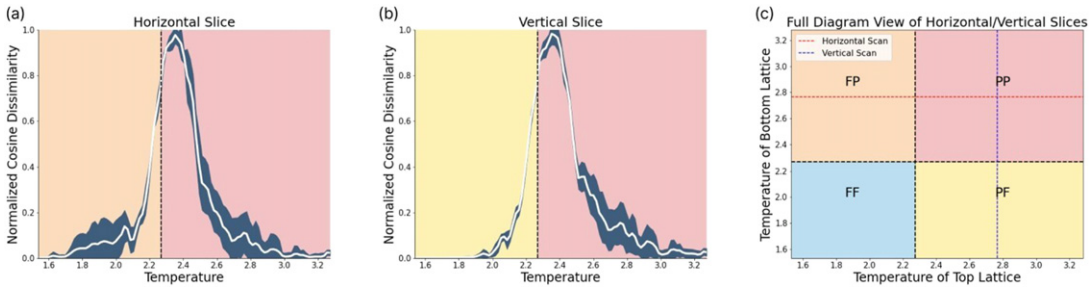

Figure 5. We use $k = 1$ to perform adjacency comparisons across the phase diagram. The blue shading represents the standard deviation over an ensemble of 5 runs. We do not see significant variability in the prediction of the phase transition. These particular slices are taken at (a) $(\cdot, \tilde{T} = 2.767)$ and (b) $(T = 2.767, \cdot)$. The peaks of both slices predict $T_c \approx 2.352$.

In order to calculate the stacked Ising model phase boundaries, we choose to perform vertical and horizontal scans across the phase diagram, where each scan is repeated five times and the standard deviation of this ensemble is displayed as uncertainty. Exemplarily, the results of two of these scans can be found in figure 5. The location of these scans is depicted in (c). (a) Reveals the phase transition from (ferromagnetic, paramagnetic) to (paramagnetic, paramagnetic) (FP to PP), while (b) displays the phase transition from PF to PP. Collecting 21 vertical and 21 horizontal scans reveals the phase boundaries of the stacked Ising model, as seen in figure 6. By comparing the horizontal (green dotted line) and vertical scans (red dotted line) with the Ising model phase boundaries in the thermodynamic limit (black dashed lines) one can observe a clear difference. The SNN predicts critical temperatures of $T_c \approx 2.352$, consistent with the finite size correction of the magnetization which indicates a phase transition at $T_c \approx 2.36$.

In order to reveal the power of our SNN based phase recognition scheme, we have to scan across diagonal slices within the phase diagram that contain more than one phase transition. For this purpose we scan diagonally across the phase diagram as depicted in figure 7. The two diagonal scans across the lattice reveal that this technique can identify more than one phase transition. The first diagonal scan is performed along a line through the center of the lattice. In this case, we see a single phase transition, as the network scans directly through the intersection of the vertical and horizontal phase lines. A closer examination of the effect of changing k can be found in figure D1. The prediction yields a phase boundary at $(T, \tilde{T})_c = (2.28, 2.28)$.

The second diagonal scan we perform is shifted, such that it crosses both the vertical and horizontal phase transition. In this case, the true phase boundaries are cut by the shifted diagonal slice at $(T, \tilde{T})_c = (1.827, 2.269)$ and $(T, \tilde{T})_c = (2.269, 2.711)$. The network is able to capture both transitions at $(T, \tilde{T})_c \approx (1.857, 2.299)$ and $(T, \tilde{T})_c \approx (2.281, 2.723)$. A closer examination is found in figure D2.

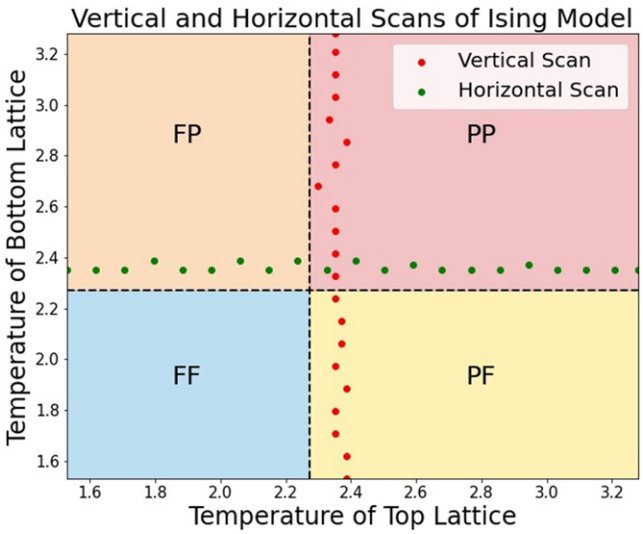

Figure 6. The phase diagram of the stacked Ising model contains four regions. Both of the sublattices of the stacked model can be in either the paramagnetic (P) or ferromagnetic (F) phase. We perform adjacency comparisons as in figures 5(a) and (b) for each point in the dotted lines indicating the SNN estimates for the phase transitions. The deviations from the true phase transitions are quantitatively consistent with finite size effects on the magnetization.

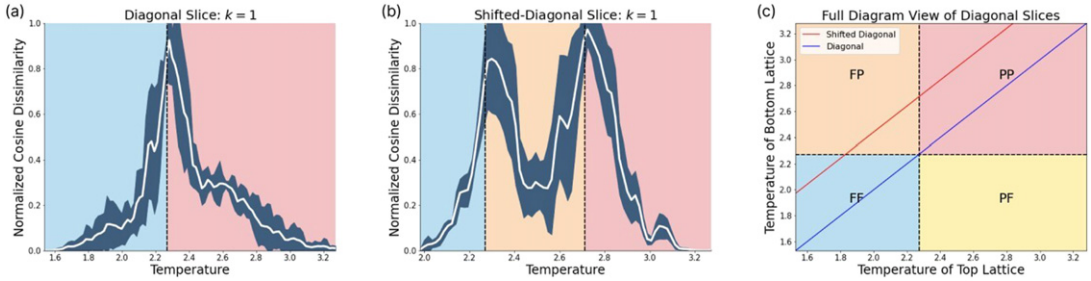

Figure 7. We perform a two different diagonal scans across phase diagram of the stacked Ising model, using $k = 1$. The adjacency comparison plot (a) depicts the result from a scan across the diagonal (blue line in (c)), while (b) depicts the same for the shifted diagonal (red line in (c)). In (c), the diagonal (blue) line only crosses the phase boundaries once in the center (FF to PP), and we correctly observe a single corresponding peak in (a). Likewise, the shifted diagonal line (red) crosses two phase boundaries (from FF to FP, then FP to PP), resulting in two corresponding peaks. 100 equally spaced points between 1.53 and 2.28 translate to a bin width of $\Delta_T \approx 0.7575$.

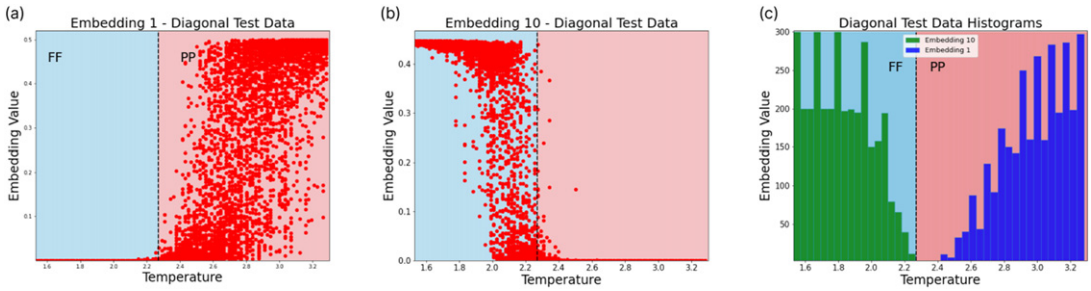

Figure 8. Latent space embeddings of the SNN are applied to configurations of the stacked Ising model. In particular, we choose our configurations from the diagonal slice, with $k = 1$, to reveal how the SNN encodes phase information. Of the 10 embedding neurons, some encode the phase information of the diagonal slice. Embedding neurons 1 and 10 are such neurons, distinguishing between the FF and PP phases in the diagonal slice. (a) and (b) Depict a separation of activity between the aforesaid regions. (c) Counts the number of instances in which a point is above the halfway line in the y-axis of (a) and (b). There is a separation in these histograms between the two phase regions, FF and PP. According to (c), embedding 1 (blue) has a preference for encoding information in the PP region, while embedding 10 (green) encodes the FF region. 100 equally spaced points between 1.53 and 2.28 translate to a bin width of $\Delta_T \approx 0.7575$.

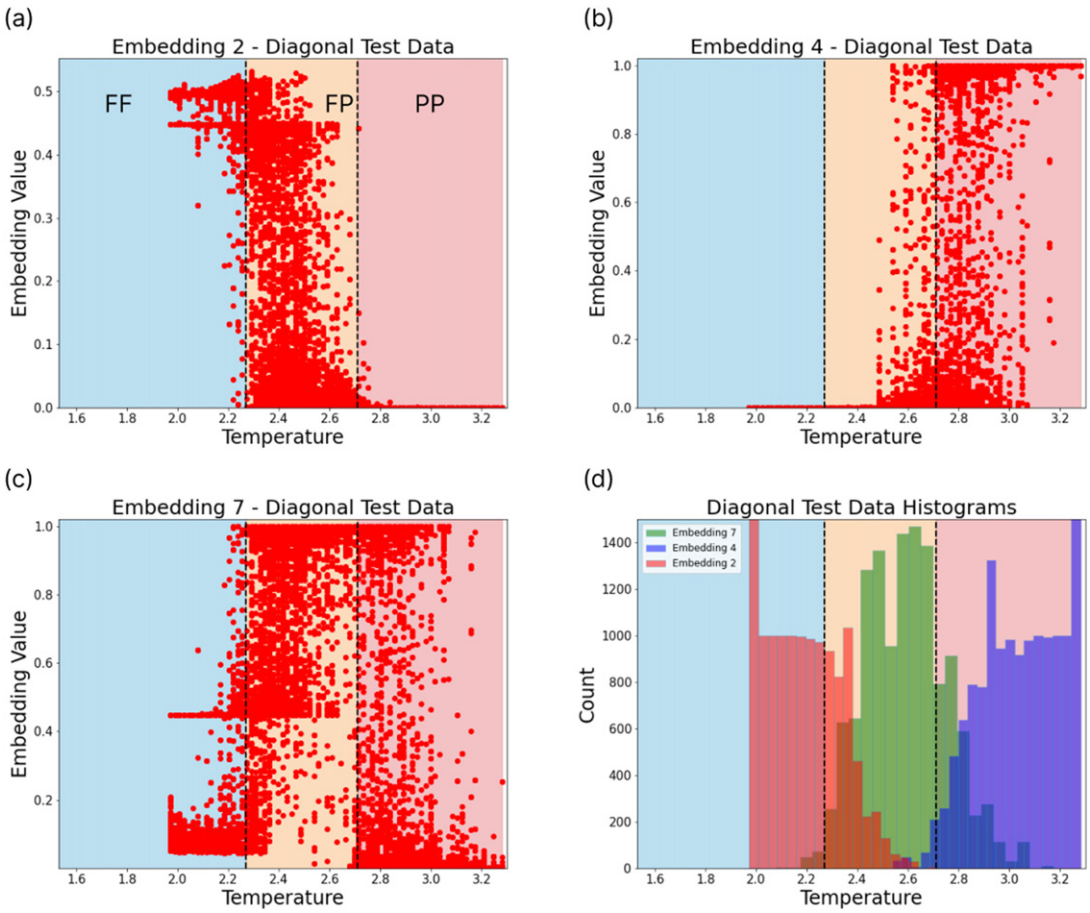

Figure 9. Latent space embeddings of the SNN are applied to configurations of the stacked Ising model belonging to the shifted diagonal slice, which contains two phase transitions. The embeddings display structures associated with each phase. While (a) appears to encode the FP phase, its corresponding histogram reveals that it encodes the FF phase. The histogram accounts for the density of points beyond the halfway line in (a), indicating that many points are clustered together in the FF phase. (c) Similarly shows the encoding of the FP phase, while (b) shows the encoding of the PP phase. The corresponding histograms are colour-coded in (d). 100 equally spaced points between 1.53 and 2.28 translate to a bin width of $\Delta_T \approx 0.7575$.

Furthermore, the embeddings shed some light on how the neural network encodes the aforesaid phase transitions. The embedding is an n-dimensional output of the SNN, mapped from the input. This is shown in red in figure 2. These embeddings may represent a vector in n-dimensional space. For example, the inputs S_a and S_+ in figure 2 produce two different vectors after they are passed through the network. To find how dissimilar these vectors are, one may use several distance metrics, such as a Euclidean or cosine distance between the two vectors. In figure 8 we see embeddings which encode phase information for the diagonal slice. These embeddings clearly separate between the two underlying phases. Embedding 1 only spikes in the PP phase, while embedding 10 activates at FF regimes. A switch between both embeddings occurs at the phase transition. Other embeddings can be found in figure C1. In the case of a single embedding, we may have noise contained in the same embedding neuron where the phase behaviour is captured. With additional embeddings, the noise may be relegated to other embedding neurons, isolating the phase transition information.

The embedding space of the shifted diagonal scan is depicted in figure 9. We observe three crucial behaviours in the embeddings. Each of the three histograms show maximum activity in three different regions. Embedding 2 encodes the FF phase, embedding 4 the PP phase, and embedding 7 encodes configurations from the FP phase.

4.3. Rydberg array

The Rydberg atom array phase diagram is not as well studied as the Ising model, so we need to compare our SNN phase boundaries to recent papers [72, 78] and evaluate the order parameters on the QMC data ourselves.

In order to identify the approximate phase boundaries we compute predictors for the phases of interest. Each phase corresponds to various peaks in the absolute value of the Fourier transform (FT) of the one-point function:

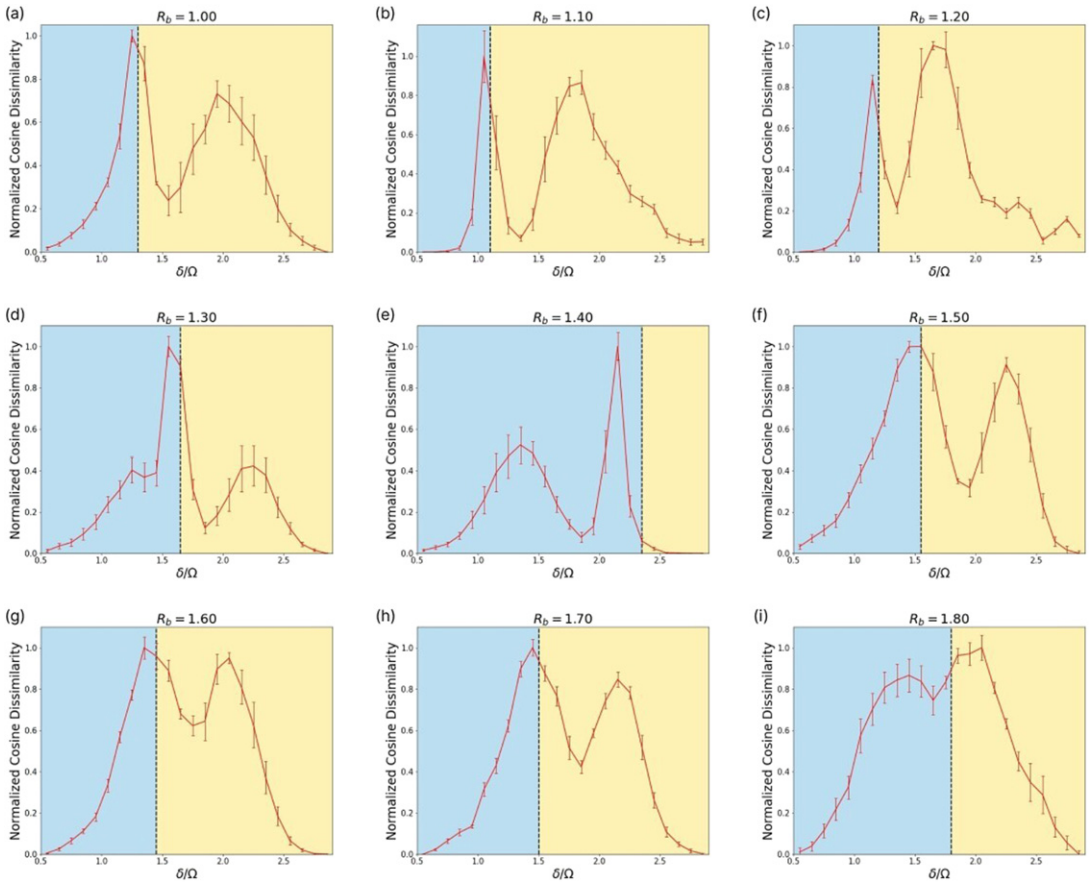

Figure 10. We apply the SNN phase recognition method to the Rydberg array in the region $R_b \in \{1.0, \ldots, 1.8\}$ and $\delta \in \{0.5, \ldots, 2.9\}$. We scan along each horizontal slice by choosing a fixed R_b, and traversing along all $\delta \in \{0.5, \ldots, 2.9\}$. We choose $k = 1$ to perform adjacency comparisons within these scans. The dissimilarity peaks when configurations at δ_i and δ_{i+1} are of different phases. The vertical lines in each plot represent the approximate phase transition at each R_b based on [78], with the two colours clearly indicating the partitioning of the regions.

$$n(\mathbf{k}) = n(k_x, k_y) = \left| \frac{1}{\sqrt{N}} \sum_j n_j \, \exp(i\mathbf{k} \cdot \mathbf{r}_j) \right| \tag{16}$$

where n_j is the Rydberg occupation of the jth site. Furthermore, we symmetrize the FT by averaging over permutations of the momentum axes:

$$\mathcal{F}(k_x, k_y) = \frac{1}{2}\left[n(k_x, k_y) + n(k_y, k_x)\right] \tag{17}$$

The phases we are interested in are the checkerboard, striated, and star phases, which are discussed in [78]. The peaks occur for the checkerboard phase at (π, π), for the striated phase at $(\pi, 0)$ and (π, π), and for the star phase at $(\pi, 0)$, and $(\pi/2, \pi)$ [78]. We compute, for each point in parameter space (R_b, δ), the symmetrized FT at these specific momenta averaged over the full 1200 sample data set given to the SNNs.

4.3.1. Testing

The Rydberg atom array phase diagram is examined in regimes where the checkerboard, striated and star phases are present. Thus, guided by [72] we focus the region $R_b \in \{1.0, \ldots, 1.8\}$ and $\delta \in \{0.5, \ldots, 2.9\}$. The training of the SNN on Rydberg QMC data is performed similar to the Ising model on each horizontal and vertical slice. Inference is performed by adjacency comparisons with step-size $k = 1$. Because the Rydberg phase diagram is coarser than the Ising model phase diagram (25 points between $\delta = 0.5$ to $\delta = 2.90$ for the Rydberg array, compared to 100 points between $T = 1.53$ and $T = 3.28$ for the Ising model), there is no reason to use $k > 1$.

All horizontal and vertical scans depicted in figures 10 and E1, respectively, are the result of an ensemble average over 5 different runs per slice. The error bars represent the standard deviation between runs. The vertical lines in each plot in figure 10 represent the approximate phase transition at each R_b based on [78]. Our observations differ from their result in the sense that we consistently observe two phase transitions in all slices (although some of these phase transition peaks are weak, as shown in figure 10). However, we also

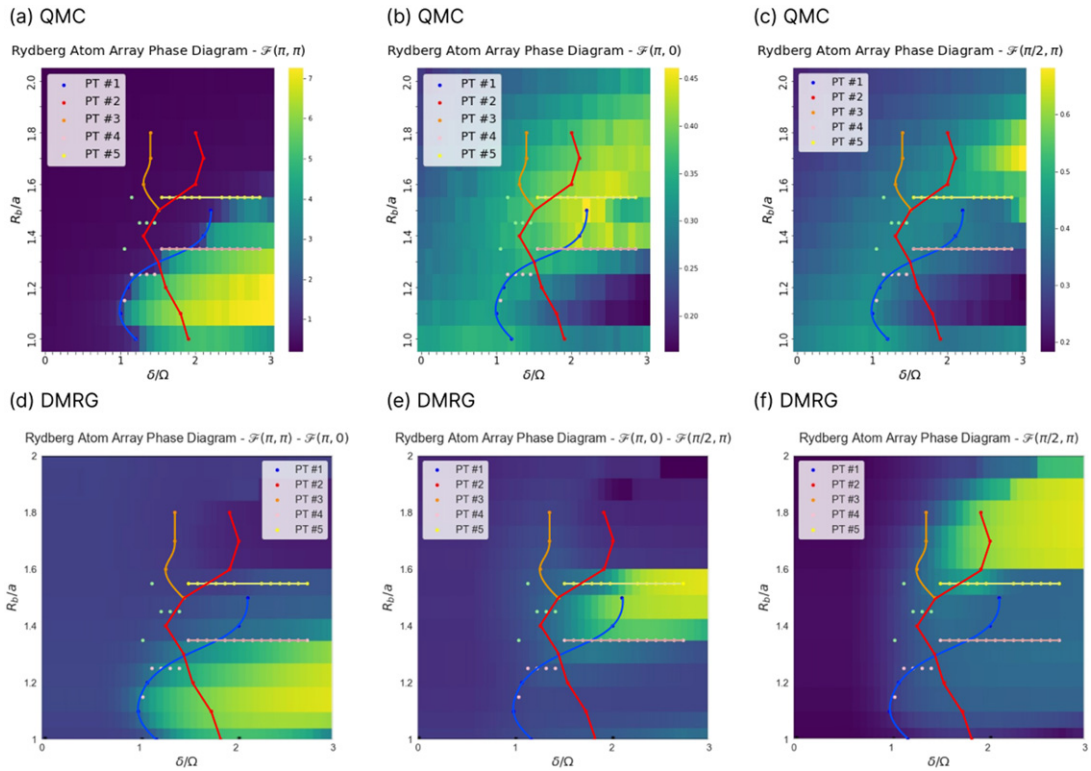

Figure 11. Phase transition lines are superposed on the Rydberg phase diagram. These lines result from connecting the peaks from figures 10 and E1 in the smoothest possible way. Doing so results in three distinct phase boundaries. In (a)–(c) we overlay these phase lines on the diagram generated via our QMC simulation, whereas (d)–(f) use the phase diagram produced by DMRG calculations taken from [72].

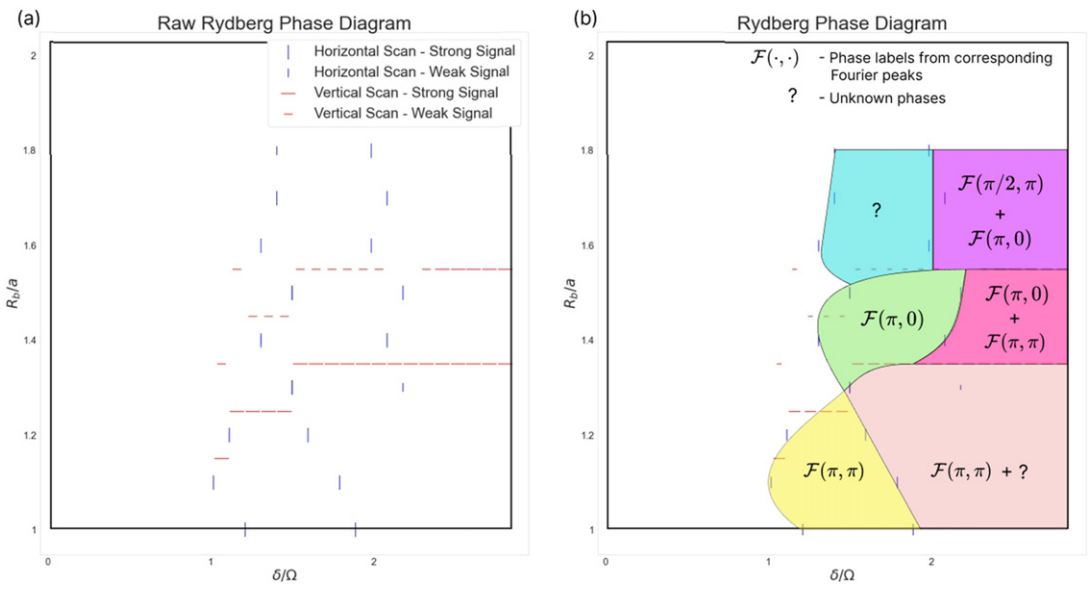

Figure 12. SNN phase recognition method applied to the Rydberg array in the region $R_b \in \{1.0, \ldots, 1.8\}$ and $\delta \in \{0.5, \ldots, 2.9\}$. (a) Raw results of SNN scans along horizontal and vertical slices collected from figures 10 and E1. Strong signals correspond to clearly visible phase transition signals, while weak signals are only considered phase transition indicators if they can be consistently identified in neighboring slices. (b) Regions in the phase diagram are separated by connecting markers in (a) as smoothly as possible, they are labelled according to the Fourier modes characterizing the phases in figures 11(a)–(c).

confirm that in all cases the SNN predicts a phase transition that coincides with the results from [78]. The secondary peaks are consistent with the boundary criticality found in [71]. The two peaks approximately match with the boundary and bulk transitions shown in figure 1 of [71] for $1.3 < R_b < 1.8$. Furthermore, the results in figure E1 reveal the striated phase in between the star and checkerboard phases, which is also consistent with [78].

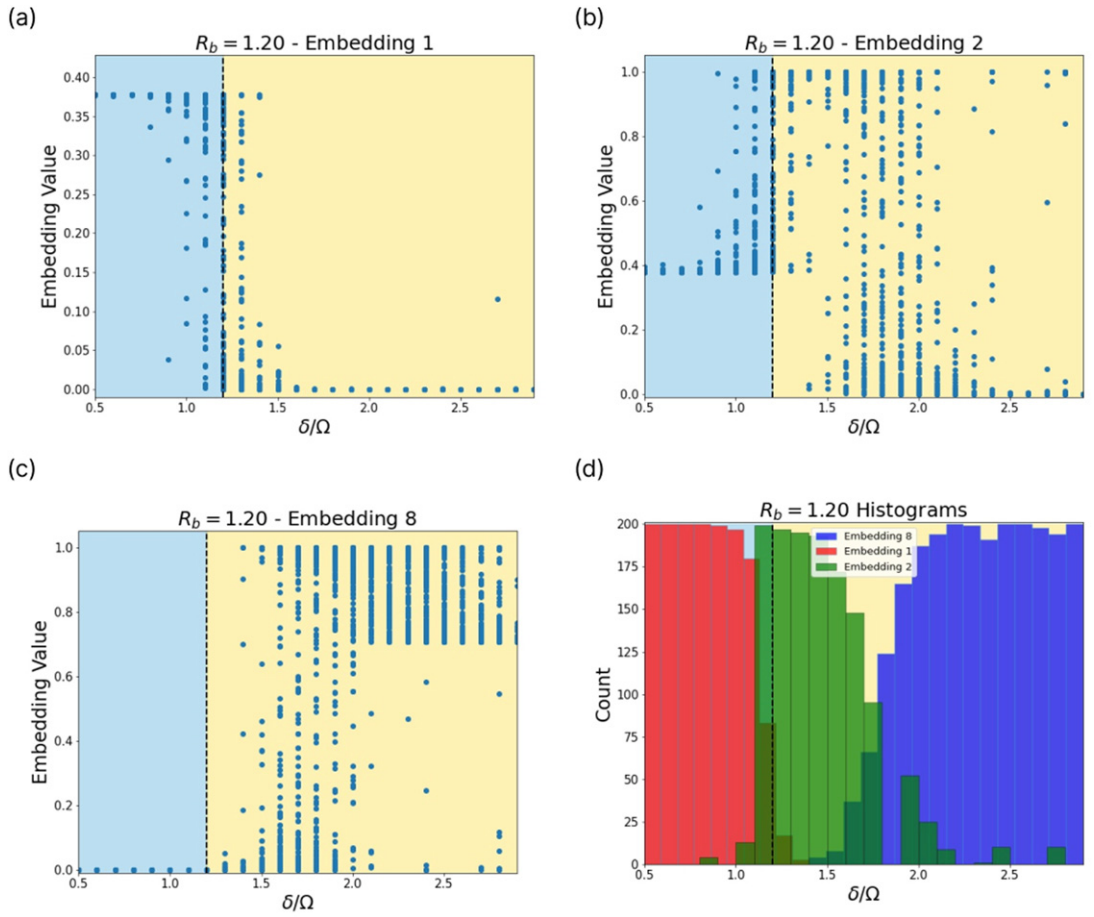

Figure 13. Latent space embeddings of the SNN are applied to configurations of the Rydberg system belonging to the horizontal slice along $R_b = 1.20$ in the checkerboard region. Of the 10 embedding neurons, some encode phase information along $R_b = 1.20$. Embeddings 1, 2 and 8 are such neurons, revealing a phase transition near $\delta = 1.20$, which is marked by the black dashed line. The phase information is markedly partitioned by this line, as can be seen in (a) and the corresponding red embedding in (d). This phase transition corresponds to the first peak in figure 10(c). The second peak in figure 10(c) is located near $R_b = 1.70$. Embedding 8 in (c) and the blue histogram in (d) reveal a phase transition near $\delta = 1.70$. (b) Shows the encoding of an intermediate phase, with the corresponding histogram in (d) coloured green.

In figures 11(a)–(c) we compare the SNN predictions with the results from evaluating order parameters on our QMC data. For each the checkerboard phase, the star phase, and the striated phase we create separate plots. The QMC data is chosen as the background, while in the foreground we connect phase transition signals from the SNN in the smoothest way possible in the form of blue, red, orange, pink and yellow lines. We first observe a perfect agreement of the checkerboard phase transitions seen in (a). The striated and star phase in figures (b) and (c) are very elusive; however, the SNN tends to capture clearer phase information than what is suggested by the evaluation of the order parameters. The red line captures the striated phase for $R_b > 1.3$. There are two distinct differences between SNN and QMC order parameter results: there is no QMC order parameter signal to explain the orange line. Further, the red line continues well within the checkerboard phase, where none of the three order parameters signal a phase transition.

We also compare our results to those obtained by [72] in figures 11(d)–(f). Again the SNN prediction of the checkerboard phase transition is in good agreement with the DMRG results. Further, the red line for $1.3 < R_b < 1.6$ is similar to their striated phase boundary. The pink and yellow lines also bound the striated phase. In addition, our orange line corresponds very well to the DMRG star phase. The DMRG star phase is much larger than the QMC star phase. However, the SNN results should mimic what is present in the QMC data. This contradiction might be resolved by two different explanations: (1) the neural network is able to extract features which are a stronger indicator for the star phase than Fourier modes. (2) There is another phase present in the QMC data that is responsible for the orange phase boundary [78], suggests possible candidates in form of rhombic, banded or staggered order.

The full independent SNN prediction of the Rydberg array phase diagram is constructed in figure 12, where (a) depicts the raw peaks of the horizontal and vertical scans in figures 10 and E1, respectively. The vertical scan slices between $\delta \in \{0.50, \ldots, 1.00\}$ do not reveal any explicit phase transition, as expected, and are therefore not included in the phase diagram. Figure 12(b) is constructed by connecting markers as smoothly as possible. This results in 7 phase regions, labelled according to figure 11.

Figure 13 provides insight into how the SNN encodes the phase information in the Rydberg system. Certain embeddings correspond to very specific regions in the phase diagram. Embeddings 1 and 2 separate in the vicinity of the first phase transition, and where embeddings 2 and 8 separate, the SNN indicates a second phase transition.

It is surprising that the SNN is able to reveal clearer phase information from the QMC data than what a direct evaluation of order parameters might indicate. The reason for this might be that the neural network is able to calculate other thermodynamic quantities which are relevant to describe the underlying physics, such as energies or susceptibilities. Further, the neural network might be able to deal better with domain boundaries that separate different regions within single samples. This observation encourages us to predict that neural networks might be able to reveal phases where conventional order parameter evaluations on MC configurations have trouble doing so. In that sense the SNN predicts the occurrence of 2 phases that are not evident in the QMC order parameter analysis, the blue and orange regions in figure 12. These findings guide us to examine these regions closer with a keen eye on revealing previously unknown physics.

5. Conclusion

We have introduced a SNN based method to detect phase boundaries in an unsupervised manner. This method does not require any physical knowledge about the nature or existence of the underlying phases. The most powerful ML tools to detect phases and phase transitions that reproduced many complicated phase diagrams are still supervised feed-forward neural networks [3]. Since a true novel discovery of phases cannot be supervised, there have been several proposals of ML methods that can be used for unsupervised phase detection. The most popular among those are PCA [5] and autoencoders [6]. These methods have lacked in performance compared to their supervised counterparts. This work presents an approach to access the power of this supervised algorithm in an unsupervised method. Inside of the SNN the subnetwork figure 3 is identical to the architecture of a supervised feed-forward neural network. Furthermore, an optimized supervised feed-forward neural network would provide a (local) minimum for the subnetwork in the SNN. In that sense the most similar method is learning by confusion [4] that relies on assigning random phase labels to a supervised ML method to identify the most optimal labels.

Our method is shown to reproduce phase diagrams when trained on Monte-Carlo configurations of the corresponding physical system. We introduced this method at the example of a model consisting of two stacked Ising models exhibiting a phase diagram of four phases. Further, we used this method to calculate the phase diagram of a Rydberg atom array which is to the most extent consistent with prior results, and shows additional signatures of unknown and coexistence phases. Further, in some regimes the SNN tends to be better at picking up phase information than order parameters applied to QMC data. As typical for neural network based phase recognition schemes, we do not have insight on what features a neural network is learning in order to calculate the phase boundaries. These quantities have been shown to be related to order parameters and other physically relevant quantities. Explicitly revealing them is difficult, but possible using methods like [23, 87].

While it remains to be seen if ML based methods will reveal a fundamentally new unknown phase, we believe SNN based phase detection has the features of a top contender with its ability to detect multiple phases in an unsupervised manner while providing a strong connection to the most powerful supervised ML methods for phase detection.

Acknowledgments

We thank Roger Melko for helpful discussions. We thank the National Research Council of Canada for their partnership with Perimeter on the PIQuIL. Research at Perimeter Institute is supported in part by the Government of Canada through the Department of Innovation, Science and Economic Development Canada and by the Province of Ontario through the Ministry of Colleges and Universities.

Data availability statement

The data that support the findings of this study are available upon reasonable request from the authors.

Appendix A. Normalized cosine dissimilarity

In this section we display the calculation of the cosine dissimilarity in figure A1.

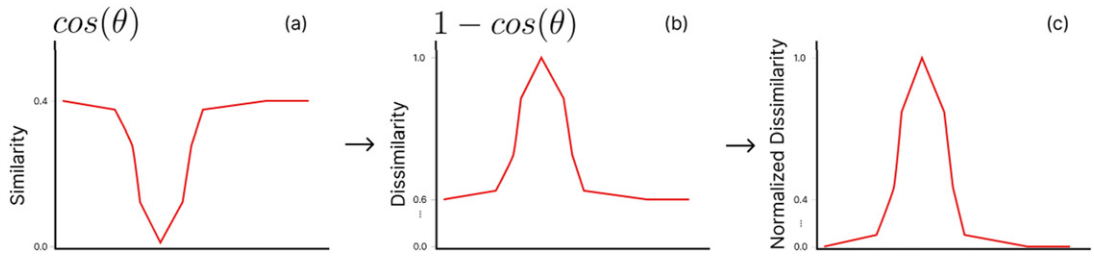

Figure A1. (a) We begin with the standard cosine similarity plot. (b) We then compute $1 - \cos(\theta)$ to obtain the cosine dissimilarity. (c) We normalize to guarantee that the plot ranges from 0 to 1.

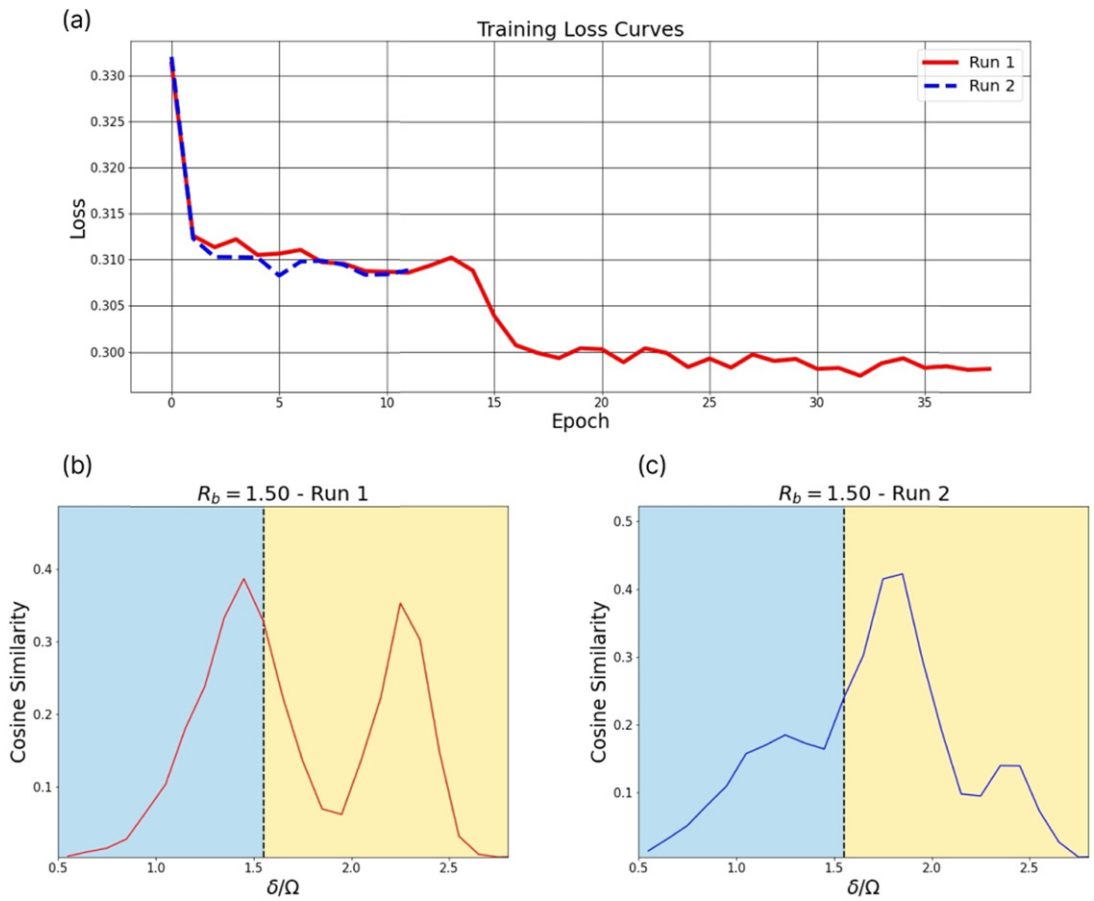

Figure B1. We train on Rydberg array configurations belonging to the slice $R_b = 1.50$, employing early stopping. (a) The blue loss curve reveals that a single peak, shown in (c), occurs when training falls prey to local minima. Training ceases at around 12 epochs in this case due to the early stopping callback. Furthermore, we observe that the red loss curve succeeds in escaping this minimum, and converges to a lower loss. This lower loss corresponds to dual peaks, shown in (b).

Appendix B. Training

B.1. General training dynamics

We observe that training is prone to getting stuck in local minima. For this reason, we may rerun the training phase several times for certain slices. In particular, the diagonal slices (both the shifted diagonal

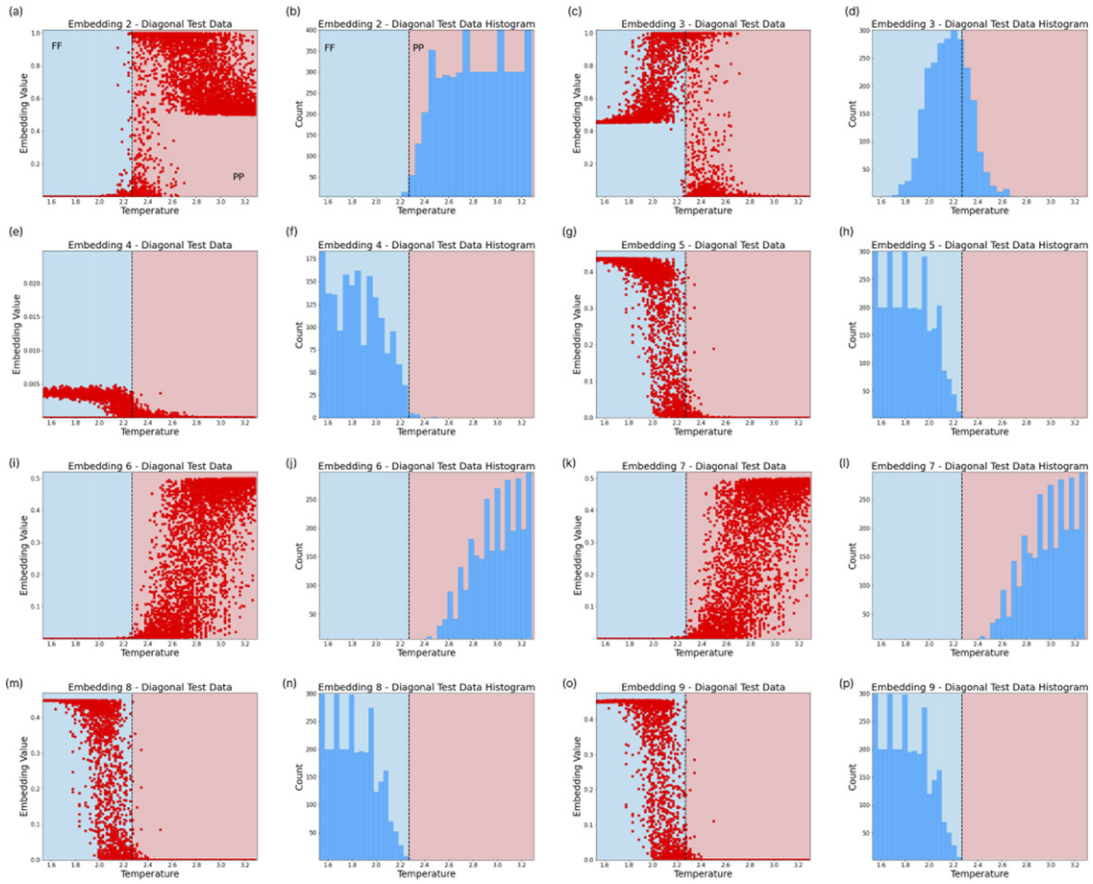

Figure C1. Latent space embeddings of the SNN are applied to configurations of the stacked Ising model belonging to the diagonal slice. This slice contains a single phase transition between FF and PP, as depicted in figure 7(c). Some neurons belonging to these embeddings encode phase information, albeit redundantly. For instance, (j) and (l) both encode the PP phase, but almost identically. Other embeddings appear unique, but still encode information about the same phase transition. (j) and (m) encode the PP and FF phases, respectively, but the both provide insight into the same phase transition depicted by the black dashed line.

and the direct diagonal) are rerun if their adjacency comparisons exhibit unexpected/sub-optimal behaviour. One may use the loss curve to identify such instances. For instance, one run may exhibit a single phase transition, while the other may exhibit 2. We opt to keep the result with the lower loss. This scenario is depicted in figure B1. The single peak result in (c) corresponds to a higher loss. We identify this as a local minimum. The red curve in (a) corresponds to the plot (b). While this run also gets stuck in a local minimum, it is able to escape upon further training. As such, we keep (b) as our result.

B.2. Ising model—training on the diagonal slice

The diagonal slice is shown in figures 4(a) and 7(c). Of the 92 Ising configurations produced at each temperature, 50 are used in the training set, and 42 in the testing set. Then, we use these to produce 100 stacked configurations at random for both the training and testing sets. For example, we choose two random configurations out of the set of 50 for the training set, and overlay them to produce a single stacked configuration.

B.3. Ising model—training on the shifted diagonal slice

The shifted diagonal slice is shown in figure 7(c). Here, we shift the slice by 25 temperature points vertically in order to cross two phase transitions. Of the 92 Ising configurations produced at each temperature, 70 are used in the training set, and 22 in the testing set. Once again, we use these to produce stacked configurations at random for both the training and testing sets. However, this time we produce 3500 stacked configurations for the training set, and 500 for the testing set. Because the diagonal slice does not contain many points from any of the three phases it crosses, we increase the amount of training data to compensate.

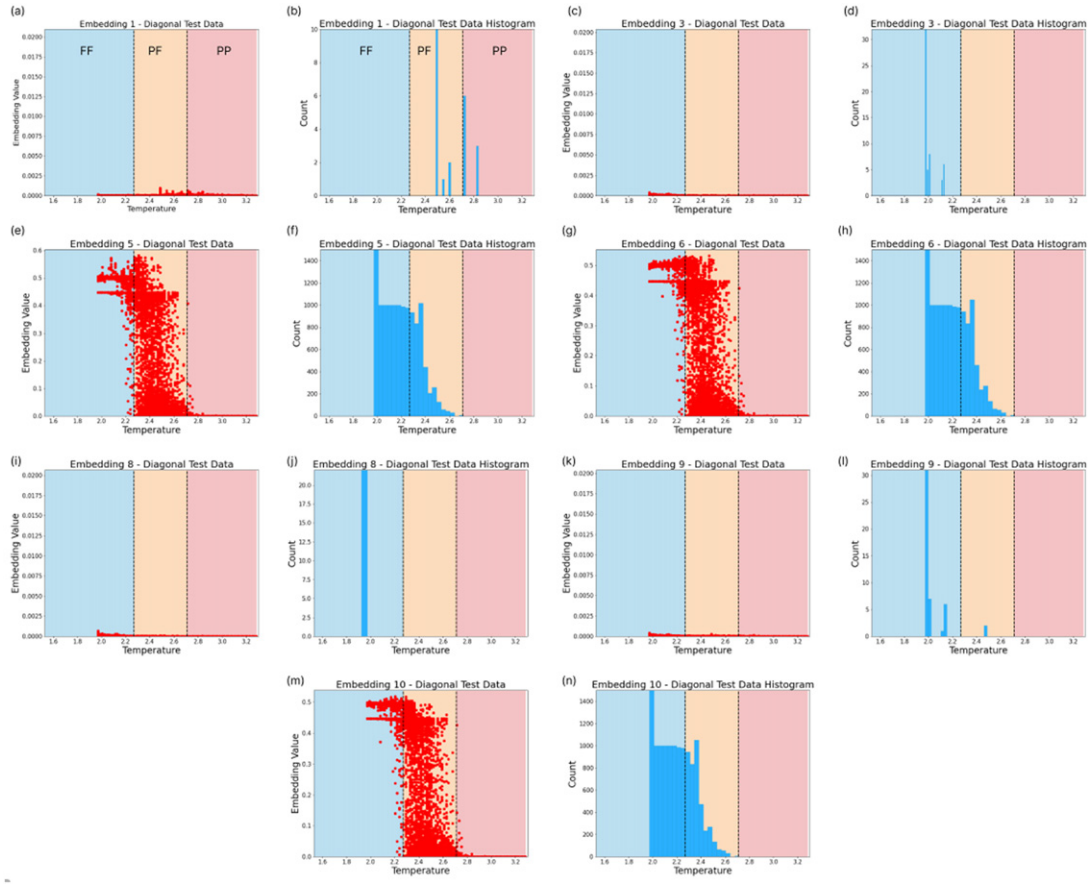

Figure C2. Latent space embeddings of the SNN are applied to configurations of the stacked Ising model belonging to the shifted diagonal slice. This slice contains three regions—FF, PF, and PP, as depicted in (a) and (b). Some neurons belonging to these embeddings encode phase information, albeit redundantly, while other neurons do not appear to encode anything at all. For instance, (e) and (g) both encode the FF phase, but almost identically. On the other hand, (a), (c), (i), and (k) do not encode any phase information.

B.4. Ising model—training on the horizontal/vertical slices

Of the 92 Ising configurations produced at each temperature, 42 are used in the training set, and 50 in the testing set. As with the diagonal slice, we then use these to produce 100 stacked configurations at random for both the training and testing sets.

B.5. Training the Rydberg system

Training on the Rydberg system is performed similar to the Ising model, i.e. slice-wise. We train over $R_b \in \{1.00, \ldots, 1.80\}$ and $\delta \in \{0.5, \ldots, 2.9\}$, with 1000 configurations per δ-step. 800 are used for training, and the remaining 200 for testing.

Appendix C. Additional embeddings

This section presents the remaining embeddings for the Ising model's diagonal and shifted-diagonal scans, as well as the $R_b = 1.20$ horizontal scan.

C.1. Ising model—diagonal embeddings

See figure C1.

C.2. Ising model—shifted diagonal embeddings

See figure C2.

C.3. Rydberg system

See figure C3.

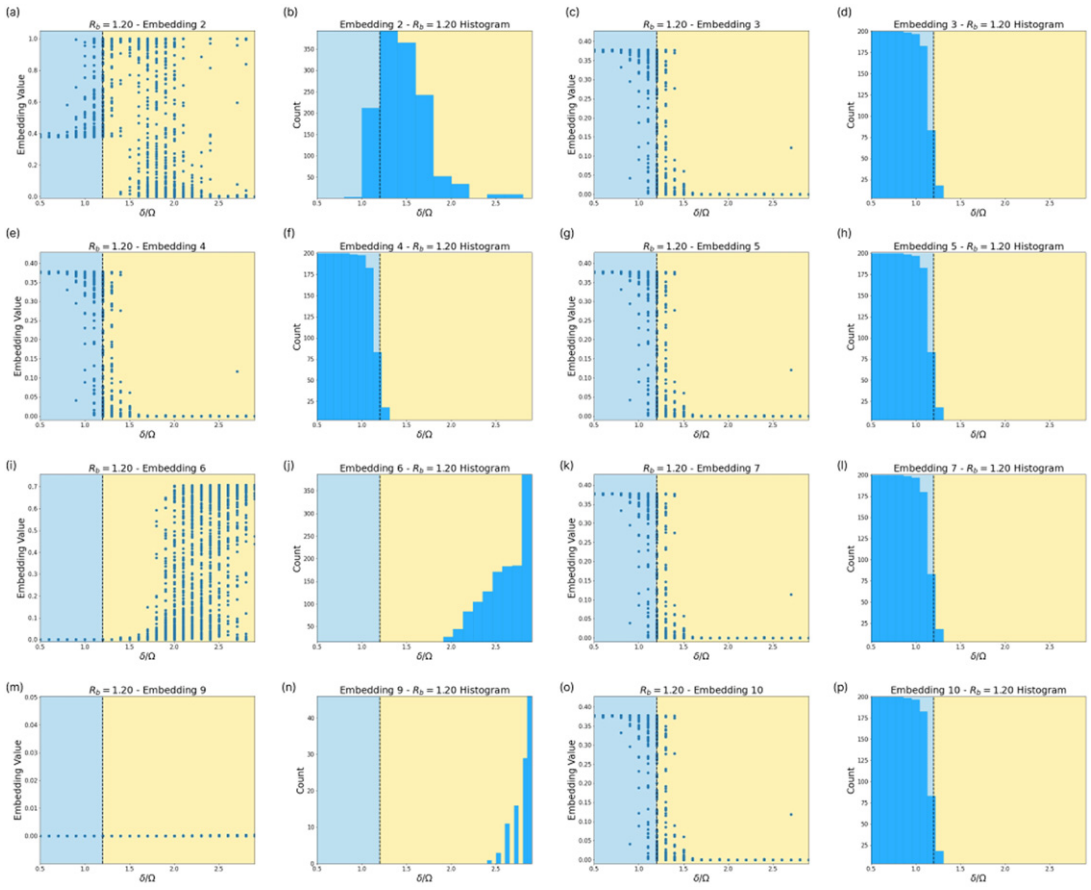

Figure C3. Latent space embeddings of the SNN are applied to configurations of the Rydberg system belonging to the horizontal slice along $R_b = 1.20$ in the checkerboard region. Most neurons encode redundant information. For instance, (a) and (c) encode information related to the same phase, while (c) and (f) exhibit identical phase structures. We also observe that embedding 9 in (m) encodes no phase information.

Appendix D. Additional adjacency comparisons

This section contains further examines the effect of the hyperparameters k and α on the adjacency comparison.

D.1. Effect of k

We vary k through values 1, 5, and 10 in order to examine its impact on the adjacency comparison. A higher k indicates a larger separation along the slice between two configurations subject to comparison (i.e. they are more dissimilar). As such, we expect smoother adjacency plots when using higher values of k, as the SNN is more easily able to distinguish between configurations that are far apart. Indeed, figure D1 shows a gradual smoothing of the adjacency plots with higher k (figure D3).

D.2. Effect of α

We vary α through values $\{0.1, 0.2, \ldots, 0.6\}$ to choose a good value of the hyperparameter.

Appendix E. Additional Rydberg phase scans

We perform additional scans over the Rydberg phase diagram. Notably, figures E2 and E3 are obtained by training on Rydberg arrays under *periodic* boundary conditions, in order to avoid the possibility of boundary criticality [71] contaminating the results. For the periodic system, the long-range interactions are modified to sum over periodic replicas. We find that no Ewald summation [88] is necessary as the vDW interactions decay quite rapidly, hence convergence is obtained with a modest 49 replicas in total (i.e. 3 along each of the cardinal directions). The SSE simulation is then run using this re-summed interaction matrix. Comparing against recent Tensor network based phase diagram calculations [89, 90], we see good agreement between the sharpest peaks in figures E2 and 2(a) of the aforementioned reference. For $R_b > 1.6$,

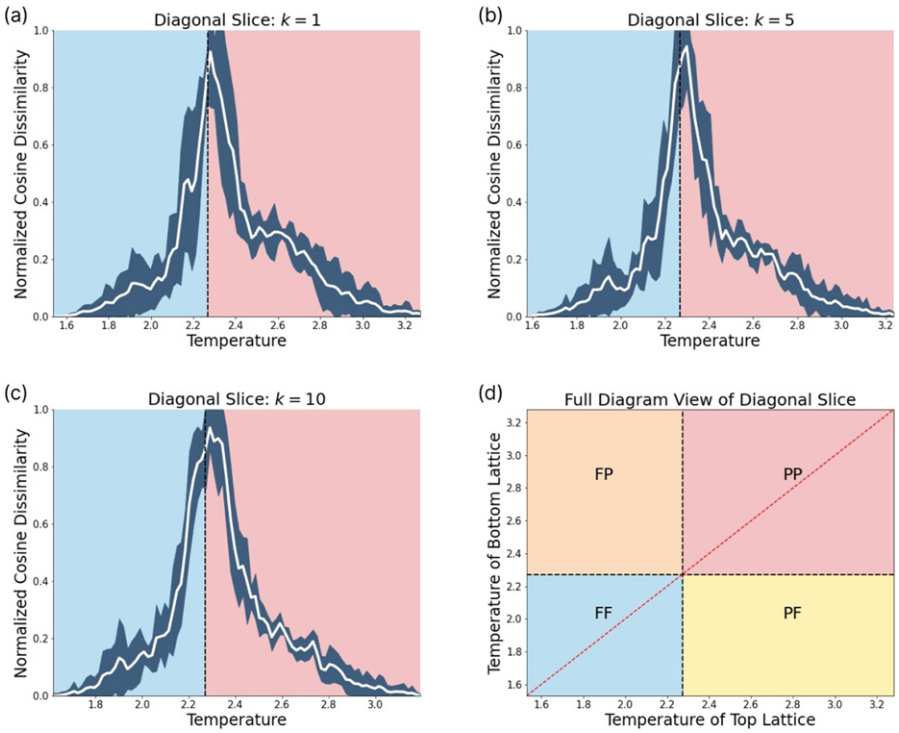

Figure D1. We scan across the configurations of the stacked Ising model belonging to the diagonal slice (d). The adjacency comparison plots (a)–(c) depict the results of this scan using three different values of k. The diagonal line in (d) only crosses the phase boundaries once in the center (FF to PP), and we correctly observe a single corresponding peak in (a)–(c). Higher values of k appear to smooth out noise. The trajectory of (a) is interrupted by a hump between $T = 2.4 \ldots 2.7$, which gradually lessens through to (c).

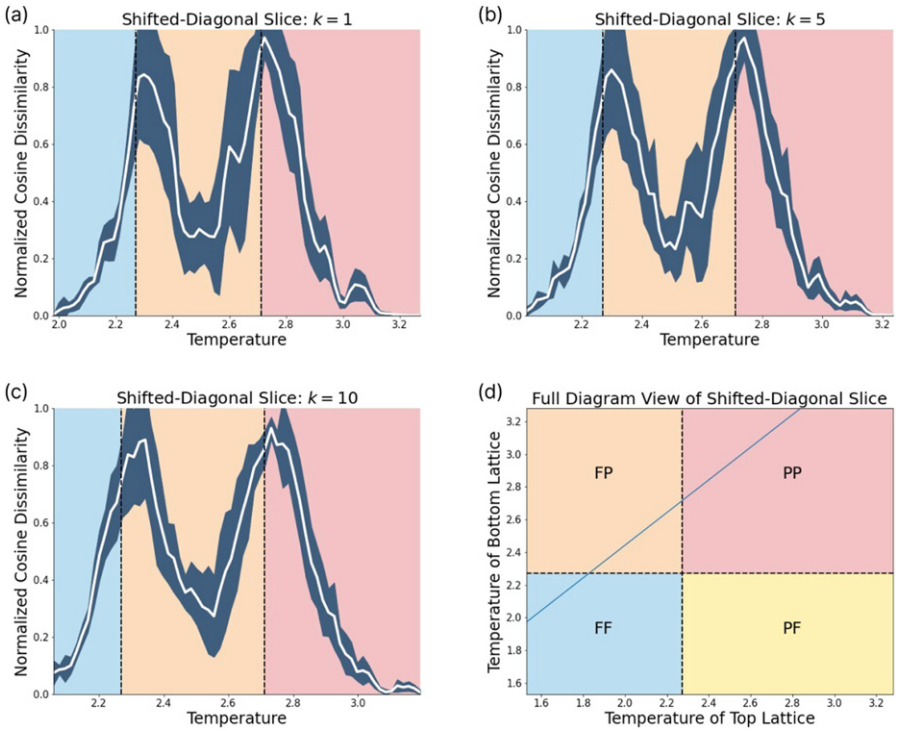

Figure D2. We scan across the configurations of the stacked Ising model belonging to the shifted diagonal slice (d). The adjacency comparison plots (a)–(c) depict the results of this scan using three different values of k. The diagonal line in (d) only crosses the phase boundaries once in the center (FF to PP), and we correctly observe a single corresponding peak in (a)–(c). Higher values of k appear to smooth out noise. For instance, the FP phase of (a) is characterized by noise, which gradually lessens through to (c).

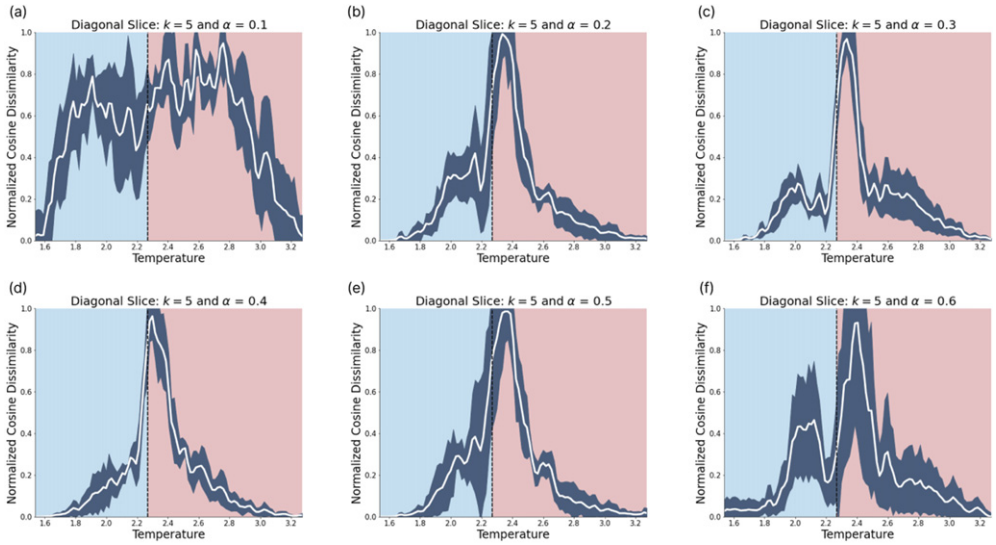

Figure D3. We scan across the configurations of the stacked Ising model belonging to the diagonal slice. The adjacency comparison plots depict the results of this scan using a fixed $k = 5$. We vary α from 7 across (a)–(f) in steps of 0.1. We observe sharp peaks as long as $\alpha \in [0.2, 0.5]$. For this reason we chose $\alpha = 0.4$ for our examinations.

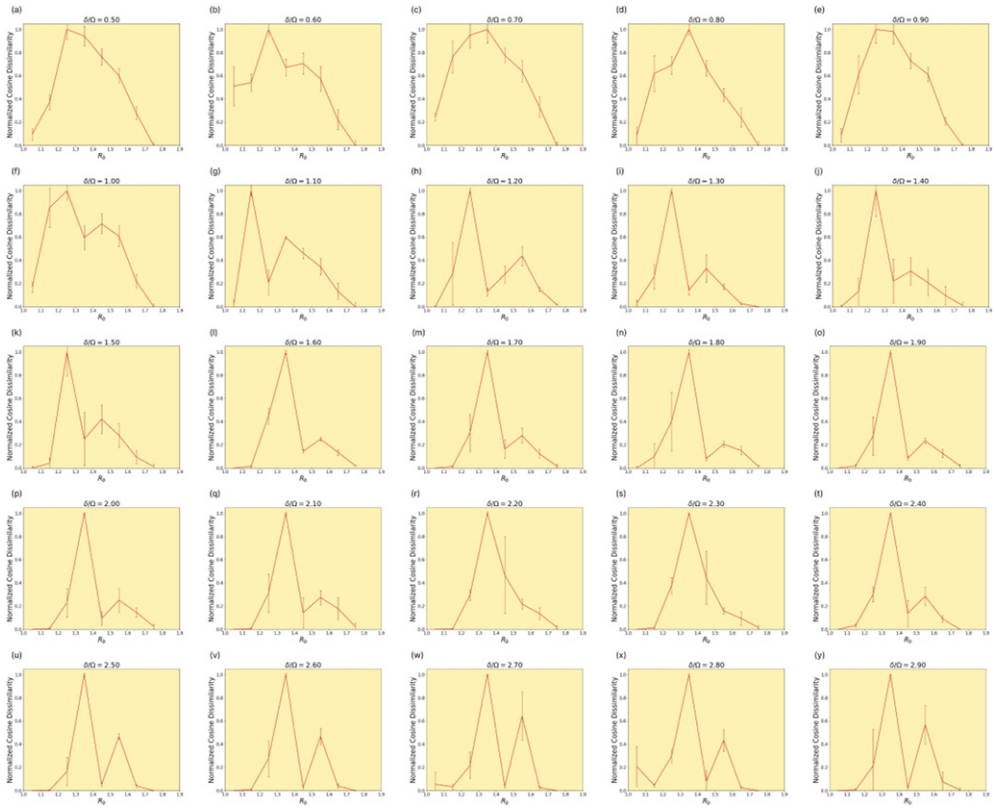

Figure E1. We apply the SNN phase recognition method to the Rydberg array in the region $R_b \in \{1.0, \ldots, 1.8\}$ and $\delta \in \{0.5, \ldots, 2.9\}$. We scan along each vertical slice by choosing a fixed δ, and traversing along all $R_b \in \{1.00, \ldots, 1.80\}$. We choose $k = 1$ to perform adjacency comparisons within these scans. The dissimilarity peaks when configurations at $R_{b,i}$ and $R_{b,i+1}$ are of different phases.

however, the SNN dissimilarity shows a wide hump (centered around points well in the disordered phase) instead of a sharp peak near the transition predicted in [89]. We believe this to be due to the SSE simulation being poorly converged.

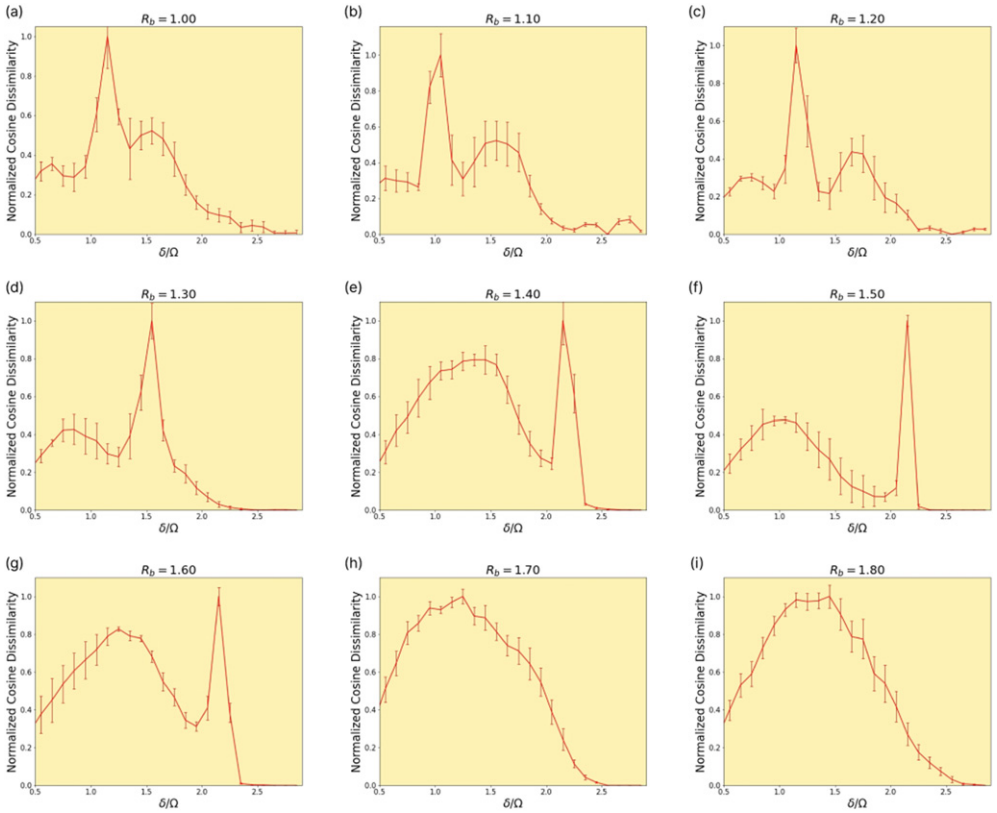

Figure E2. We apply the SNN phase recognition method to the Rydberg array in the region $R_\mathrm{b} \in \{1.0, \ldots, 1.8\}$ and $\delta \in \{0.5, \ldots, 2.9\}$ using periodic boundary conditions. We scan along each vertical slice by choosing a fixed δ, and traversing along all $R_\mathrm{b} \in \{1.00, \ldots, 1.80\}$. We choose $k = 1$ to perform adjacency comparisons within these scans. The dissimilarity peaks when configurations at $R_{\mathrm{b},i}$ and $R_{\mathrm{b},i+1}$ are of different phases. In comparison to figure 10 (which was using open boundary conditions), we see that the phase transitions in the checkerboard region are more pronounced, while exhibiting a washed-out structure beyond this region. We also observe that only a single well-defined peak is present in plot, in contrast to figure 10.

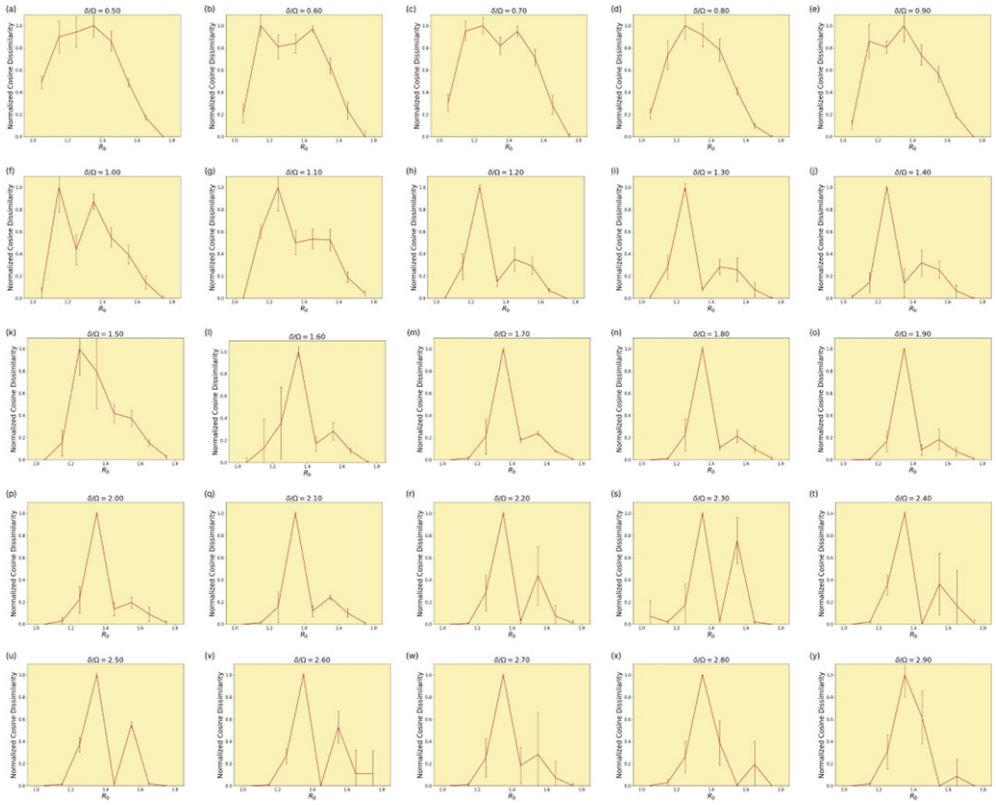

Figure E3. We apply the SNN phase recognition method to the Rydberg array in the region $R_b \in \{1.0, \ldots, 1.8\}$ and $\delta \in \{0.5, \ldots, 2.9\}$ using periodic boundary conditions. We scan along each vertical slice by choosing a fixed δ, and traversing along all $R_b \in \{1.00, \ldots, 1.80\}$. We choose $k = 1$ to perform adjacency comparisons within these scans. The dissimilarity peaks when configurations at $R_{b,i}$ and $R_{b,i+1}$ are of different phases. The Rydberg array uses periodic boundary conditions in this case. In comparison to figure E1 (using open boundary conditions), we see that the phase transitions in the checkerboard region are more pronounced, while exhibiting a washed-out structure beyond this region.

References

[1] Krizhevsky A, Sutskever I and Hinton G E 2012 *Advances in Neural Information Processing Systems* vol 25
[2] Goldberg Y 2016 *J. Artif. Intell. Res.* **57** 345–420
[3] Carrasquilla J and Melko R G 2017 *Nat. Phys.* **13** 431–4
[4] van Nieuwenburg E P L, Liu Y H and Huber S D 2017 *Nat. Phys.* **13** 435–9
[5] Wang L 2016 *Phys. Rev.* B **94** 195105
[6] Wetzel S J 2017 *Phys. Rev.* E **96** 022140
[7] Zhang Y and Kim E A 2017 *Phys. Rev. Lett.* **118** 216401
[8] Schindler F, Regnault N and Neupert T 2017 *Phys. Rev.* B **95** 245134
[9] Hu W, Singh R R P and Scalettar R T 2017 *Phys. Rev.* E **95** 062122
[10] Ohtsuki T and Ohtsuki T 2017 *J. Phys. Soc. Japan* **86** 044708
[11] Broecker P, Carrasquilla J, Melko R G and Trebst S 2017 *Sci. Rep.* **7** 8823
[12] Deng D L, Li X and Das Sarma S 2017 *Phys. Rev.* B **96** 195145
[13] Ch'ng K, Carrasquilla J, Melko R G and Khatami E 2017 *Phys. Rev.* X **7** 031038
[14] Torlai G and Melko R G 2016 *Phys. Rev.* B **94** 165134
[15] Carleo G and Troyer M 2017 *Science* **355** 602–6
[16] Inack E M, Santoro G E, Dell'Anna L and Pilati S 2018 *Phys. Rev.* B **98** 235145
[17] Hibat-Allah M, Ganahl M, Hayward L E, Melko R G and Carrasquilla J 2020 arXiv:2002.02973v2
[18] Carrasquilla J, Luo D, Pérez F, Milsted A, Clark B K, Volkovs M and Aolita L 2019 arXiv:1912.11052v1
[19] Ferrari F, Becca F and Carrasquilla J 2019 *Phys. Rev.* B **100** 125131
[20] Sharir O, Levine Y, Wies N, Carleo G and Shashua A 2020 *Phys. Rev. Lett.* **124** 020503
[21] Schmidt M and Lipson H 2009 *Science* **324** 81–5
[22] Iten R, Metger T, Wilming H, del Rio L and Renner R 2020 *Phys. Rev. Lett.* **124** 010508
[23] Wetzel S J and Scherzer M 2017 *Phys. Rev.* B **96** 184410
[24] Ponte P and Melko R G 2017 *Phys. Rev.* B **96** 205146
[25] Wetzel S J, Melko R G, Scott J, Panju M and Ganesh V 2020 *Phys. Rev. Res.* **2** 033499
[26] Greitemann J, Liu K and Pollet L 2019 *Phys. Rev.* B **99** 060404
[27] Mototake Y-i 2019 arXiv:2001.00111v2
[28] Udrescu S M and Tegmark M 2019 arXiv:1905.11481v2
[29] Krenn M, Hochrainer A, Lahiri M and Zeilinger A 2017 *Phys. Rev. Lett.* **118** 080401

[30] Krenn M et al 2022 arXiv:2204.01467
[31] Dawid A et al 2022 arXiv:2204.04198
[32] Carrasquilla J 2020 Adv. Phys.: X 5 1797528
[33] Carleo G, Cirac I, Cranmer K, Daudet L, Schuld M, Tishby N, Vogt-Maranto L and Zdeborová L 2019 Rev. Mod. Phys. 91 045002
[34] Huembeli P, Dauphin A and Wittek P 2018 Phys. Rev. B 97 134109
[35] Arnold J and Schäfer F 2022 arXiv:2203.06084
[36] van Nieuwenburg E, Bairey E and Refael G 2018 Phys. Rev. B 98 060301
[37] Hsu Y T, Li X, Deng D L and Das Sarma S 2018 Phys. Rev. Lett. 121 245701
[38] Vargas-Hernández R A, Sous J, Berciu M and Krems R V 2018 Phys. Rev. Lett. 121 255702
[39] Bachtis D, Aarts G and Lucini B 2020 Phys. Rev. E 102 033303
[40] Tanaka A and Tomiya A 2017 J. Phys. Soc. Japan 86 063001
[41] Beach M J, Golubeva A and Melko R G 2018 Phys. Rev. B 97 045207
[42] Zhang W, Liu J and Wei T C 2019 Phys. Rev. E 99 032142
[43] Suchsland P and Wessel S 2018 Phys. Rev. B 97 174435
[44] Kim D and Kim D H 2018 Phys. Rev. E 98 022138
[45] Lian W et al 2019 Phys. Rev. Lett. 122 210503
[46] Dong X Y, Pollmann F and Zhang X F 2019 Phys. Rev. B 99 121104
[47] Giannetti C, Lucini B and Vadacchino D 2019 Nucl. Phys. B 944 114639
[48] Ohtsuki T and Mano T 2020 J. Phys. Soc. Japan 89 022001
[49] Casert C, Vieijra T, Nys J and Ryckebusch J 2019 Phys. Rev. E 99 023304
[50] Zhang R, Wei B, Zhang D, Zhu J J and Chang K 2019 Phys. Rev. B 99 094427
[51] Singh V K and Han J H 2019 Phys. Rev. B 99 174426
[52] Arnold J, Schäfer F, Žonda M and Lode A U 2021 Phys. Rev. Res. 3 033052
[53] Liu Y H and van Nieuwenburg E P 2018 Phys. Rev. Lett. 120 176401
[54] Huembeli P, Dauphin A, Wittek P and Gogolin C 2019 Phys. Rev. B 99 104106
[55] Scheurer M S and Slager R J 2020 Phys. Rev. Lett. 124 226401
[56] Käming N, Dawid A, Kottmann K, Lewenstein M, Sengstock K, Dauphin A and Weitenberg C 2021 Mach. Learn.: Sci. Technol. 2 035037
[57] Alexandrou C, Athenodorou A, Chrysostomou C and Paul S 2020 Eur. Phys. J. B 93 226
[58] Yin J, Pei Z and Gao M C 2021 Nat. Comput. Sci. 1 686–93
[59] Greplova E, Valenti A, Boschung G, Schäfer F, Lörch N and Huber S D 2020 New J. Phys. 22 045003
[60] Kharkov Y A, Sotskov V, Karazeev A, Kiktenko E O and Fedorov A K 2020 Phys. Rev. B 101 064406
[61] Ch'ng K, Vazquez N and Khatami E 2018 Phys. Rev. E 97 013306
[62] Wang C and Zhai H 2017 Phys. Rev. B 96 144432
[63] Kottmann K, Huembeli P, Lewenstein M and Acín A 2020 Phys. Rev. Lett. 125 170603
[64] Jadrich R, Lindquist B and Truskett T 2018 J. Chem. Phys. 149 194109
[65] Che Y, Gneiting C, Liu T and Nori F 2020 Phys. Rev. B 102 134213
[66] Tibaldi S, Magnifico G, Vodola D and Ercolessi E 2022 arXiv:2202.09281
[67] Bromley J, Guyon I, LeCun Y, Säckinger E and Shah R 1993 Advances in Neural Information Processing Systems vol 6 pp 737–44
[68] Baldi P and Chauvin Y 1993 Neural Comput. 5 402–18
[69] Henriet L, Beguin L, Signoles A, Lahaye T, Browaeys A, Reymond G O and Jurczak C 2020 Quantum 4 327
[70] Browaeys A and Lahaye T 2020 Nat. Phys. 16 132–42
[71] Kalinowski M, Samajdar R, Melko R G, Lukin M D, Sachdev S and Choi S 2021 arXiv:2112.10790
[72] Ebadi S et al 2021 Nature 595 227–32
[73] Miles C et al 2021 arXiv:2112.10789
[74] Torlai G et al 2019 Phys. Rev. Lett. 123 230504
[75] Carrasquilla J and Torlai G 2021 PRX Quantum 2 040201
[76] Czischek S, Moss M S, Radzihovsky M, Merali E and Melko R G 2022 arXiv:2203.04988
[77] Browaeys A, Barredo D and Lahaye T 2016 J. Phys. B: At. Mol. Opt. Phys. 49 152001
[78] Samajdar R, Ho W W, Pichler H, Lukin M D and Sachdev S 2020 Phys. Rev. Lett. 124 103601
[79] Merali E, De Vlugt I J and Melko R G 2021 arXiv:2107.00766
[80] Newman M E J and Barkema G T 1999 Monte Carlo Methods in Statistical Physics (Oxford: Clarendon)
[81] Sandvik A W and Kurkijärvi J 1991 Phys. Rev. B 43 5950–61
[82] Sandvik A W 1992 J. Phys. A: Math. Gen. 25 3667–82
[83] Sandvik A W 2003 Phys. Rev. E 68 056701
[84] Sandvik A W 2005 Phys. Rev. Lett. 95 207203
[85] Sandvik A W and Evertz H G 2010 Phys. Rev. B 82 024407
[86] Schroff F, Kalenichenko D and Philbin J 2015 CoRR
[87] Miles C et al 2021 Nat. Commun. 12 3905
[88] Ewald P P 1921 Ann. Phys., Lpz. 369 253–87
[89] O'Rourke M J and Chan G K L 2022 arXiv:2201.03189
[90] Felser T, Notarnicola S and Montangero S 2021 Phys. Rev. Lett. 126 170603