Machine learning determination of dynamical parameters:

Machine learning determination of dynamical parameters: The Ising model case

Guido Cossu,[1, 2, 3] Luigi Del Debbio,[1] Tommaso Giani,[1] Ava Khamseh,[1, 4] and Michael Wilson[1]

[1]*Higgs Centre for Theoretical Physics, School of Physics & Astronomy, University of Edinburgh, Edinburgh EH9 3FD, United Kingdom.*
[2]*Alan Turing Institute, London NW1 2DB, United Kingdom.*
[3]*Ascent Robotics, Shibuya-ku, Tokyo 150-0012, Japan.*
[4]*MRC Human Genetics Unit, Institute of Genetics & Molecular Medicine, University of Edinburgh, Edinburgh EH4 2XU, United Kingdom.*[*]

We train a set of Restricted Boltzmann Machines (RBMs) on one- and two-dimensional Ising spin configurations at various values of temperature, generated using Monte Carlo simulations. We validate the training procedure by monitoring several estimators, including measurements of the log-likelihood, with the corresponding partition functions estimated using annealed importance sampling. The effects of various choices of hyper-parameters on training the RBM are discussed in detail, with a generic prescription provided. Finally, we present a closed form expression for extracting the values of couplings, for every n-point interaction between the visible nodes of an RBM, in a binary system such as the Ising model. We aim at using this study as the foundation for further investigations of less well-known systems.

[*] Email: ava.khamseh@igmm.ed.ac.uk

I. INTRODUCTION

A structured probabilistic model is a formalism used in machine learning, in particular deep learning, to describe the joint probability distribution of a set of random variables of interest and their interaction, via mathematical graphs. These models are often referred to as graphical models and consist of a set of vertices, connected by a set of edges. Over the years, various graphical models, together with their training and inference algorithms, have been developed by the machine learning community [1]. *Undirected* models are a popular subset of graphical models, often referred to as Markov Random Fields (MRFs), for which there is no directionality to the edges of the graph connecting the nodes [2]. A class of undirected models, with probability distribution of the form

$$p(\mathbf{v}) = \frac{e^{-E(\mathbf{v})}}{Z} \ , \tag{1}$$

are referred to as energy-based models (EBMs), or Boltzmann machines (BMs) if the models contains latent (hidden) variables [3–5]. In the above equation, $E(\mathbf{v})$ is the positive energy function and Z denotes the partition function. In this work, we focus on Restricted Boltzmann Machines (RBMs), a subset of BMs where the variables *within* the visible and hidden nodes are taken to be independent of each other [6]. RBMs are amongst the most common building blocks of deep probabilistic models. Our aim here is two-fold: the first is to be able to describe criteria that guarantee the machine is being adequately trained, as well as testing its limitations; the second is to derive predictions from the RBM that are not *directly* accessible from the data via standard statistical tools. In particular, to address the first aim, we not only measure the loss-function, as is usually done in the machine learning community, but also monitor the estimate for the log-likelihood per epoch as we train the machine. The latter allows us to identify whether or not the chosen hyperparameters result in an increase in the log-likelihood as training progresses. Furthermore we also measure the first and second moments on samples generated by the machine throughout the training, and compare these values to those obtained directly from the training dataset. These, together with other training criteria discussed in the paper, provide a thorough analysis of the training procedure. Addressing the second aim includes *e.g.* using the RBM to extract higher order n-point couplings from a dataset with unknown temperature and Hamiltonian, which would not be possible to compute directly from the data itself. Moreover, the two-point correlation that can be directly computed from the data, is a pair-wise quantity that ignores the existence of other sites in the system, apart from the two for which the correlation is being computed for. In contrast, the starting point of training an RBM is an all-to-all connection between the nodes, some of which are learned to be more significant than others as the machine is trained.

The explicit form of the RBMs, together with their properties and training algorithms, are presented in Sec. II. The RBM is trained for the 1- and 2-dimensional Ising models with volumes $L = 6$, $L^2 = 8\times 8$ and $L^2 = 16\times 16$, at various values of temperature. The Ising model and its properties are a paradigm in statistical physics literature, making it a suitable system on which to examine the training procedure and limitations of the RBMs [7–10]. Moreover, it is possible to generate a large number of Ising configurations using simple Monte Carlo simulations, avoiding the problem of small training sets.

We dedicate Sec. III and Sec. IV to the details of our training procedure and the extraction of observables from the states generated by the trained RBM, such as magnetisation, energy, susceptibility and heat capacity. In particular, in Sec. IV, we recommend a prescription to ensure the machine is being trained correctly, by monitoring certain quantities during training. These quantities include measurements of the log-likelihood and the loss function, as well as the first and second moments of the distribution generated by the RBM. Following our training procedure, these observables are then shown to agree well with the expected results, obtained directly from the training data. Finally, we present a method for extracting the couplings between the spins of the Ising system, based on an observation in Ref. [11]. Using this method, RBMs not only give us a model for describing the distribution of the data, but also provide us with the predictive power of estimating the relative strength of the couplings between the visible nodes. Notice that due to the indirect all-to-all connections between visible nodes of an RBM, there are no pair-wise coupling assumptions in measuring the strength of connection between the nodes.

II. TRAINING THE RBM & MAKING PREDICTIONS

A. Restricted Boltzmann Machines

A Restricted Boltzmann Machine (RBM) is a type of undirected Markov Random Field (MRF) with a two layer architecture. The detailed derivation of some of the main features of the RBMs is clearly presented in Ref. [12], and we follow their notation. An RBM consists of m visible nodes v_j, $j \in \{1, \cdots, m\}$, collectively denoted by $\mathbf{v}$ and representing the observed input data, and n hidden nodes h_i, $i \in \{1, \cdots, n\}$, collectively denoted by $\mathbf{h}$. In this work, we will focus on RBMs with binary variables, *i.e.* $v_j, h_i \in \{0, 1\}$. The energy of the joint state $\{\mathbf{v}, \mathbf{h}\}$ of the machine is as follows:

$$E(\mathbf{v}, \mathbf{h}; \theta) = -\sum_{i=1}^{n}\sum_{j=1}^{m} h_i w_{ij} v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i \,, \tag{2}$$

and we collectively call $\theta = \{\mathbf{w}, \mathbf{b}, \mathbf{c}\}$ the model parameters. The matrix $\mathbf{w}$ represents the undirected interaction between the visible and the hidden layers; with a slight abuse of language, in what follows a non-vanishing interaction between nodes will often be called a *connection*. The vectors $\mathbf{b}, \mathbf{c}$ are the biases of the visible and hidden layers respectively. Nodes on the same layer are not connected (making the machine *restricted*), leaving the interaction between them to be mediated by the connections to the other layer.

The RBM is used to encode the joint conditional probability distribution of a state $\{\mathbf{v}, \mathbf{h}\}$, given a set of parameters θ:

$$P(\mathbf{v}, \mathbf{h}|\theta) = \frac{e^{-E(\mathbf{v},\mathbf{h};\theta)}}{\mathcal{Z}(\theta)} \,, \tag{3}$$

where the partition function $\mathcal{Z}(\theta)$ normalises the probability distribution and contains the sums over all possible states,

$$\mathcal{Z}(\theta) = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h};\theta)} \,. \tag{4}$$

The probability distribution of the variables in the visible layer is obtained by marginalising over the binary hidden variables h_i (see e.g. [12])

$$p(\mathbf{v}|\theta) = \sum_{\mathbf{h}} P(\mathbf{v}, \mathbf{h}|\theta) = \frac{1}{\mathcal{Z}(\theta)} \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h};\theta)} = \frac{1}{\mathcal{Z}(\theta)} \prod_{j=1}^{m} \left(e^{b_j v_j}\right) \prod_{i=1}^{n} \left(1 + e^{c_i + \sum_{j=1}^{m} w_{ij} v_j}\right) \,. \tag{5}$$

The conditional probability of $\mathbf{v}$, given $\mathbf{h}$ and θ, and the one of $\mathbf{h}$, given $\mathbf{v}$ and θ, will be needed for training the RBM, and for generating configurations. Because the nodes of a given layer of an RBM are not connected these conditional probabilities factorise

$$p(\mathbf{h}|\mathbf{v}, \theta) = \prod_{i=1}^{n} p(h_i|\mathbf{v}, \theta) \quad \text{and} \quad p(\mathbf{v}|\mathbf{h}, \theta) = \prod_{j=1}^{m} p(v_j|\mathbf{h}, \theta) \,, \tag{6}$$

and can be readily computed:

$$p(h_i = 1|\mathbf{v}; \theta) = \sigma\left(\sum_{j=1}^{m} w_{ij} v_j + c_i\right) \,, \tag{7}$$

and

$$p(v_j = 1|\mathbf{h}, \theta) = \sigma\left(\sum_{i=1}^{n} w_{ij} h_i + b_j\right) \,, \tag{8}$$

where $\sigma(x) = 1/(1 + e^{-x})$ is the logistic function.

The training dataset is partitioned into batches, *i.e.* each batch S contains a subset of the training data, $S = \{\mathbf{v}_1, ..., \mathbf{v}_l\}$. The RBM is trained by maximising the log-likelihood function on a given batch S with respect to the parameters $\theta = \{w_{ij}, b_j, c_i\}$

$$\log \mathcal{L}(\theta|S) = \log \prod_{v\in S} p(\mathbf{v}|\theta) = \sum_{v\in S} \log p(\mathbf{v}|\theta) \,. \tag{9}$$

Given an input vector $\mathbf{v}$,

$$\frac{\partial \log \mathcal{L}(\theta|\mathbf{v})}{\partial\theta} = \frac{\partial \log p(\mathbf{v}|\theta)}{\partial\theta} = -\sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{v};\theta)\frac{\partial E(\mathbf{v},\mathbf{h};\theta)}{\partial\theta} + \sum_{\mathbf{v}',\mathbf{h}} P(\mathbf{v}',\mathbf{h}|\theta)\frac{\partial E(\mathbf{v}',\mathbf{h};\theta)}{\partial\theta} \,. \tag{10}$$

More explicitly, the gradient with respect to the weights w_{ij} takes the form:

$$\frac{\partial \log \mathcal{L}(\theta|\mathbf{v})}{\partial w_{ij}} = p(h_i = 1|\mathbf{v};\theta)v_j - \sum_{\mathbf{v}'} p(\mathbf{v}'|\theta)p(h_i = 1|\mathbf{v}';\theta)v'_j \tag{11}$$

$$= p(h_i = 1|\mathbf{v};\theta)v_j - \langle p(h_i = 1|\mathbf{v}';\theta)v_j\rangle_{p(\mathbf{v}')} \,. \tag{12}$$

We can then average the gradients over a given batch, so that

$$\frac{1}{|S|}\sum_{\mathbf{v}\in S}\frac{\partial \log p(\mathbf{v}|\theta)}{\partial w_{ij}} = \mathbb{E}_{\text{data}}\left[p(h_i = 1|\mathbf{v};\theta)v_j\right] - \mathbb{E}_{\text{model}}\left[p(h_i = 1|\mathbf{v}';\theta)v'_j\right] \,, \tag{13}$$

where we have introduced the average over the dataset,

$$\mathbb{E}_{\text{data}}\left[F(\mathbf{v})\right] = \frac{1}{|S|}\sum_{\mathbf{v}\in S} F(\mathbf{v}) \,, \tag{14}$$

and the average over the probability distribution $p(\mathbf{v}|\theta)$,

$$\mathbb{E}_{\text{model}}\left[F(\mathbf{v})\right] = \sum_{\mathbf{v}} p(\mathbf{v}|\theta)F(\mathbf{v}) \,. \tag{15}$$

Similarly, for the bias parameters,

$$\frac{1}{|S|}\sum_{\mathbf{v}\in S}\frac{\partial \log p(\mathbf{v}|\theta)}{\partial b_j} = \frac{1}{|S|}\sum_{\mathbf{v}\in S} v_j - \sum_{\mathbf{v}'} p(\mathbf{v}'|\theta)\mathbf{v}' \tag{16}$$

$$= \mathbb{E}_{\text{data}}\left[v_j\right] - \mathbb{E}_{\text{model}}\left[v'_j\right] \,, \tag{17}$$

$$\frac{1}{|S|}\sum_{\mathbf{v}\in S}\frac{\partial \log p(\mathbf{v};\theta)}{\partial c_i} = \frac{1}{|S|}\sum_{\mathbf{v}\in S} p(h_i = 1|\mathbf{v};\theta) - \sum_{\mathbf{v}'} p(\mathbf{v}')p(h_i = 1|\mathbf{v}';\theta) \tag{18}$$

$$= \mathbb{E}_{\text{data}}\left[p(h_i = 1|\mathbf{v};\theta)\right] - \mathbb{E}_{\text{model}}\left[p(h_i = 1|\mathbf{v}';\theta)\right] \,. \tag{19}$$

As shown above, the second terms in Eqs. 13, 16 and 18, *i.e.* terms that involve a model average $\mathbb{E}_{\text{model}}[.]$, contain sums over all possible states of the RBM; they are independent of the choice of the batch, and are computationally challenging. In practice, an approximation to the gradients is necessary, here we use the Contrastive Divergence (CD) algorithm [13]. The algorithm starts by choosing a training vector as the initial state of the visible units. Then a new state of the hidden units is generated using Eq. 7. This is done by setting a hidden unit h_i to be equal to 1 if the probability in Eq. 7 is greater than a random number uniformly distributed between 0 and 1; otherwise h_i is set to zero. Once the hidden units are chosen, a state of visible units, $\mathbf{v}^{(1)}$, is sampled according to the probability distribution Eq. 8. These steps can be iterated several times, the number of iterations is controlled by a parameter, which we denote by k in what follows. Finally, the gradient is computed using Eq. 13, where $\mathbf{v}' = \mathbf{v}^{(k)}$ is the state sampled at the end of the CD loop [14]. Given the gradient the parameters can be updated by any form of stochastic gradient ascent. In this work the parameters θ are updated by just taking a step of size α in the direction of the gradient – we refer to α as the *learning rate* of the training.

A cycle over all training batches is called a *training epoch*, resulting in a number of parameter updates equal to the number of batches. It is important for the purposes of our study to be able to quantify the quality of the training procedure. Therefore, at each epoch during the training procedure, we monitor several indicators:

1. The maximisation of the average log-likelihood. The average is computed first within each batch and then averaged over all batches. The computation of the log-likelihood requires the intractable partition function to be estimated. This estimate has been done using the annealed importance sampling procedure, discussed in detail in Sec. II B.

2. For a given state $\mathbf{v}$, $F(\mathbf{v};\theta)$ is defined as

$$F(\mathbf{v};\theta) = \log \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h};\theta)} . \tag{20}$$

   This quantity is averaged first across the states in each batch and then across all the batches, and it is expected to reach a constant once the machine is trained. To see this more clearly, notice that we can write the log-likelihood in Eq. 9 as:

$$\sum_{v\in S} \log p(\mathbf{v}|\theta) = \sum_{v\in S} \log \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h};\theta)} - \log(Z) = \sum_{v\in S} F(\mathbf{v};\theta) - \log(Z) . \tag{21}$$

   Since the log-likelihood is a quantity we are trying to maximise, once the machine is trained, it converges to a constant. Similarly for the estimation of the partition function $\log(Z)$. Therefore, during the training procedure, we can monitor how $F(\mathbf{v};\theta)$ is also progressing to a constant.

3. The loss function is defined as the average over the batches of the quantity $F\left(\mathbf{v}^{(0)};\theta\right) - F\left(\mathbf{v}^{(k)};\theta\right)$,

$$\text{Loss} = \frac{1}{N_{\text{batch}}} \sum_{\text{batches } b} \frac{1}{|S^{(b)}|} \sum_{v\in S^{(b)}} \left(F\left(\mathbf{v}^{(0)};\theta\right) - F\left(\mathbf{v}^{(k)};\theta\right)\right) \tag{22}$$

   where $\mathbf{v}^{(0)}$ is the input data from the training set, $\mathbf{v}^{(k)}$ is the state reconstructed by the machine using k Gibbs sampling steps (k is referred to as the CD parameter used for training the machine), and N_{batch} is the total number of batches.

4. The reconstruction error ϵ, is defined as

$$\epsilon = \frac{1}{N_{\text{batch}}} \sum_{\text{batches } b} \frac{1}{|S^{(b)}|} \sum_{v\in S^{(b)}} |\mathbf{v}^{(0)} - \mathbf{v}^{(1)}|^2 \tag{23}$$

   where $\mathbf{v}^{(0)}$ is as stated above and $\mathbf{v}^{(1)}$ is the state reconstructed by the machine using a single step of Gibbs sampling.

In summary, if the training procedure is correct, the log-likelihood should be an increasing function of the training steps, approaching a plateau where it stabilises at a fixed value. The quantity $F(\mathbf{v};\theta)$ is expected to reach a constant value as described above. The loss function and the reconstruction error are expected to decrease along the training, as the machine is required to reduce the difference between the generated $\mathbf{v}$'s and those used for training.

B. Annealed importance sampling

As discussed in the previous section, training the RBM corresponds to maximising the log-likelihood on the input training set, with respect to parameters of the model. Therefore, in order to monitor the training progress it is helpful to obtain an estimate of $\mathbb{E}_{\text{model}}[\log p(\theta)]$. This in turns requires the partition function for the model to be computed, which is intractable in most cases. However, there are methods to estimate the partition function. A simple method is to use a proposal distribution $p_0(\mathbf{v}) = \frac{1}{Z_0} p_0^*(\mathbf{v})$ whose partition function is tractable and can be measured exactly. The asterisk here indicates that the probability is unnormalised, with its corresponding partition function being the normalisation factor. Suppose we wish to estimate an intractable partition function Z_1 for $p_1(\mathbf{v}) = p_1^*(\mathbf{v})/Z_1$. We can write

$$Z_1 = \int p_1^*(\mathbf{v}) d\mathbf{v} = Z_0 \int p_0(\mathbf{v}) \frac{p_1^*(\mathbf{v})}{p_0^*(\mathbf{v})} d\mathbf{v} \ , \tag{24}$$

and measure its estimator

$$\hat{Z}_1 = \frac{Z_0}{M} \sum_{m=1}^{M} \frac{p_1^*(\mathbf{v}^{(m)})}{p_0^*(\mathbf{v}^{(m)})} \,, \tag{25}$$

where the sample $\{\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(M)}\}$ is drawn according to $p_0(\mathbf{v})$. One can readily observe that if the two distributions p_0 and p_1 have large overlap, states that are drawn with a high probability from p_0, also have a high probability in p_1, and the sum is close to the original integral. In contrast, if the overlap is poor, the drawn states make negligible contribution to the sum, making it a biased estimator for $\hat{Z}_1$. The issue can be further quantified by computing the variance of $\hat{Z}_1$:

$$\mathrm{Var}[\hat{Z}_1] = \frac{Z_0^2}{M^2} \mathrm{Var}\left[\sum_{m=1}^{M} \frac{p_1^*(\mathbf{v}^{(m)})}{p_0^*(\mathbf{v}^{(m)})}\right] = \frac{Z_0^2}{M^2} \sum_{m=1}^{M} \mathrm{Var}\left[\frac{p_1^*(\mathbf{v}^{(m)})}{p_0^*(\mathbf{v}^{(m)})}\right] = \frac{Z_0^2}{M} \mathrm{Var}\left[\frac{p_1^*(\mathbf{v})}{p_0^*(\mathbf{v})}\right] , \tag{26}$$

where in the second equality, it is assumed that the samples are drawn independently. Expanding the variance of the ratio, we obtain

$$\mathrm{Var}\left[\frac{p_1^*(\mathbf{v})}{p_0^*(\mathbf{v})}\right] = \int p_0(\mathbf{v}) \left[\frac{p_1^*(\mathbf{v})}{p_0^*(\mathbf{v})} - \frac{Z_1}{Z_0}\right]^2 d\mathbf{v} = \frac{Z_1^2}{Z_0^2} \int p_0(\mathbf{v}) \left[\frac{p_1(\mathbf{v})}{p_0(\mathbf{v})} - 1\right]^2 d\mathbf{v} \,. \tag{27}$$

Hence, the estimate for the variance of $\hat{Z}_1$ becomes

$$\hat{\mathrm{Var}}[\hat{Z}_1] = \frac{\hat{Z}_1^2}{M^2} \sum_{m=1}^{M} \left[\frac{p_1(\mathbf{v}^{(m)})}{p_0(\mathbf{v}^{(m)})} - 1\right]^2 \quad \text{s.t. : } \mathbf{v}^{(m)} \sim p_0 \,. \tag{28}$$

If the two distributions p_0 and p_1 are not close to each other, in the Monte Carlo simulation, there will be many samples drawn from p_0 with a high probability which have a corresponding low probability with respect to p_1, leading to a large contribution to the sum in Eq. 28. In fact, it has been discussed in Refs. [15, 16] that the above variance can become large and even infinite, leading to very poor estimates.

In most cases, as it is with the RBM, p_1 is a multimodal distribution over a high dimensional space. As a result, it is difficult to propose a simple distribution p_0 with a tractable partition function that gives a good estimate for the original Z_1. In such situations a method known as annealed importance sampling is adopted [16–19]. This method tries to bridge the original distribution p_1 and the simple distribution p_0, by introducing intermediate closer distribution $p_{\beta_0}, p_{\beta_1}, \cdots, p_{\beta_n}$ such that $0 = \beta_0 < \beta_1 < \cdots < \beta_{n-1} < \beta_n = 1$. We therefore try to estimate Z_1/Z_0 via:

$$\frac{Z_1}{Z_0} = \frac{Z_{\beta_1}}{Z_0} \frac{Z_{\beta_2}}{Z_{\beta_1}} \cdots \frac{Z_{\beta_{n-2}}}{Z_{\beta_{n-1}}} \frac{Z_1}{Z_{\beta_{n-1}}} = \prod_{j=0}^{n-1} \frac{Z_{\beta_{j+1}}}{Z_{\beta_j}} \,, \tag{29}$$

where $\frac{Z_{\beta_{j+1}}}{Z_{\beta_j}}$ can be estimated using the simple importance sampling Eq. 25 described above. These estimates are reliable as the corresponding distributions are close. The intermediate distribution can be chosen such that they are proportional to the geometric average of p_1 and p_0, *i.e.* ,

$$p_\beta \propto p_1^*(\mathbf{v})^\beta p_0^*(\mathbf{v})^{1-\beta} \,. \tag{30}$$

To see why this is a sensible choice, notice that

$$p_1^*(\mathbf{v})^\beta p_0^*(\mathbf{v})^{1-\beta} = e^{-\beta E_1(\mathbf{v})} e^{-(1-\beta) E_0(\mathbf{v})} = e^{-E_0(\mathbf{v})} e^{-\beta[E_1(\mathbf{v}) - E_0(\mathbf{v})]} \,. \tag{31}$$

In other words, when β is very small, p_β is also very close to the simple distribution p_0. Increasing β gradually, until it is close to one, will have p_β being close to p_1, as required. Here, we make the choice $E_0 = 0$, *i.e.* , we sample from a uniform starting distribution $p_0^*(\mathbf{v})$ for $\beta_{j=0} = 0$. This implies that, for $\beta_1, \cdots, \beta_n = 1$,

$$p_{\beta_j}^*(\mathbf{v}) \propto e^{-\beta_j E_1(\mathbf{v})} \,, \tag{32}$$

where $e^{-E_1}(\mathbf{v})$ is the numerator term in Eq. 5. In the first step $\mathbf{v}$ is sampled from a uniform distribution and $p^*_{\beta_1}$ is computed according to Eq. 32, for a value of β close to zero. A set of configurations $\mathbf{v}$ is sampled according to that probability using Gibbs sampling, after having taken into account the presence of β in Eqs. 7 and 8, where it enters as a multiplicative factor in the argument of the logistic functions. Contrastive divergence is taken to be $k_{\mathrm{CD}} = 1$, as described in the previous section. We chose to increase β by 0.0001 in each step and continue until $\beta_n = 1$ is reached. The process is repeated for several states $\mathbf{v}$ and the average is taken at the end. Multiplying the result by Z_0 gives the estimate for Z_1, as required. As a final note, in order to avoid numerical overflow, it is recommended to compute the logarithm of the ratio of the partition functions, leading to a sum of logarithms of the bridging ratios.

C. 1D and 2D Ising model simulations

We train the RBM on Ising spin configurations in 1D and 2D, distributed according to the Boltzmann weight:

$$P(s|J) = \frac{e^{-H_J(s)}}{Z_{\mathrm{Ising}}} = \frac{e^{-J/(k_B T)\sum_{\langle i,j\rangle} s_i s_j}}{Z_{\mathrm{Ising}}} \,, \tag{33}$$

where $\langle i,j\rangle$ indicates that the sum is over nearest neighbour spins, T is the temperature of the system and k_B is the Boltzmann constant. The partition function is the sum over all possible configurations of the system,

$$Z(J) = \sum_s e^{-H_J(s)}. \tag{34}$$

Each Ising spin can take a binary value -1 or 1, and the external field is set to zero. The 2D Ising configurations were generated using `Magneto` [20], a fast parallel C++ code available online. Magneto uses the Swendsen-Wang Monte Carlo algorithm to generate the configurations minimising the autocorrelation. For simplicity we set the combination $J/k_B = 1$, and the temperature becomes the defining feature in differentiating the behaviour of various Ising systems in a given number of dimensions. Note that spins ± 1 are converted to $0, 1$ before setting them as input to the RBM.

The expectation value for an Ising observable at a given temperature,

$$\langle O\rangle\,|_T = \sum_{\mathbf{s}} p(\mathbf{s}|T) O(\mathbf{s}) \,, \tag{35}$$

is estimated by the average of the measured observable over a finite set of independent configurations:

$$\langle\hat{O}\rangle|_T = \frac{1}{N}\sum_{n=1}^{N} O\left(\mathbf{s}(n), T\right) \quad , \quad \text{s.t.} \quad \mathbf{s}^{(N)} \sim p(\mathbf{s}|T) \,, \tag{36}$$

where N is the total number of sampled states. The observables under considerations are the average magnetisation, susceptibility, energy and heat capacity:

$$\begin{aligned}
\langle m\rangle &= \frac{1}{L^2}\left\langle \left|\sum_{i=1}^{L^2} s_i\right| \right\rangle, \\
\langle \chi\rangle &= \frac{L^2}{T}\left\langle \langle m^2\rangle - \langle m\rangle^2 \right\rangle, \\
\langle E\rangle &= -\frac{1}{L^2}\left\langle \sum_{\langle i,j\rangle} s_i s_j \right\rangle, \\
\langle c_v\rangle &= \frac{L^2}{T^2}\left\langle \langle E^2\rangle - \langle E\rangle^2 \right\rangle .
\end{aligned} \tag{37}$$

Once the training process is completed we generate configurations according to the probability encoded by the RBM. We then compare the results with the corresponding values from Magneto. The methods for generating new configurations from the RBM distribution, as well as measurements of observables in the case of the 2-dimensional Ising model are given in detail in Sec. IV.

III. VALIDATION IN ONE DIMENSION

To validate our code, we trained a machine on a set consisting of 1-dimensional Ising states with 6 sites, sampled from a distribution with $J = T = 1$, similar to Ref. [8]. The training was divided into three steps of 2000, 1000 and 1000 epochs respectively, with a decreasing learning rate at each step, as summarised in Table I. In this case the analytical form of the probability distribution is relatively easy to compute so it was used to validate our implementation of both the training code and the annealing algorithm, similar to Ref. [8].

Visible	Hidden	N_{train}	Batch size	Epochs 1	LR α_1	CD k_1	Epochs 2	LR α_2	CD k_2	Epochs 3	LR α_3	CD k_3	T
6	6	100000	200	2000	0.01	1	1000	0.001	1	1000	0.0001	1	1

TABLE I: Parameters for training an RBM on the 1-dimensional Ising model with six spins, corresponding to the visible nodes. With this choice, there are a total of 64 possibles states, making it possible for the partition function to be measured exactly. We start the training with $\alpha = 0.01$ and gradually fine-tune it to smaller values as the training progresses. In this cases keeping $k = 1$ throughout the training is sufficient for learning the distribution of the data accurately.

The results for this test are shown in Fig. 1. For a 1D chain with 6 sites, there are 64 different possible states. The x-axis of the plot on the left hand side of Fig. 1 indicates these states, with their corresponding probabilities indicated on the y-axis. The true probability of these states, measured exactly, has been plotted with a black dashed line. The prediction for the underlying distribution improves as the number of training epochs increases, with the colour blue corresponding to the learned distribution at 20 epochs and the colour red indicating the learned distribution at epoch 4000, which is very close to the true distribution. The plot on the right hand side of Fig. 1 shows, in blue, how the true log-likelihood increases vs epochs, while the annealing algorithm prediction, plotted in orange, is able to reproduce the exact log-likelihood with approximately 10% error, keeping the same overall functional form. We will assume that this is the typical order of magnitude for accuracy of the log-likelihood resulting from the annealing algorithm; and will use the log-likelihood computed with the annealing procedure as our measure for the effectiveness of a training procedure, particularly for more complicated systems where measurement of the true partition function is intractable. Notice that the annealing underestimates Z, thus resulting in an upper bound for the true log-likelihood [16].

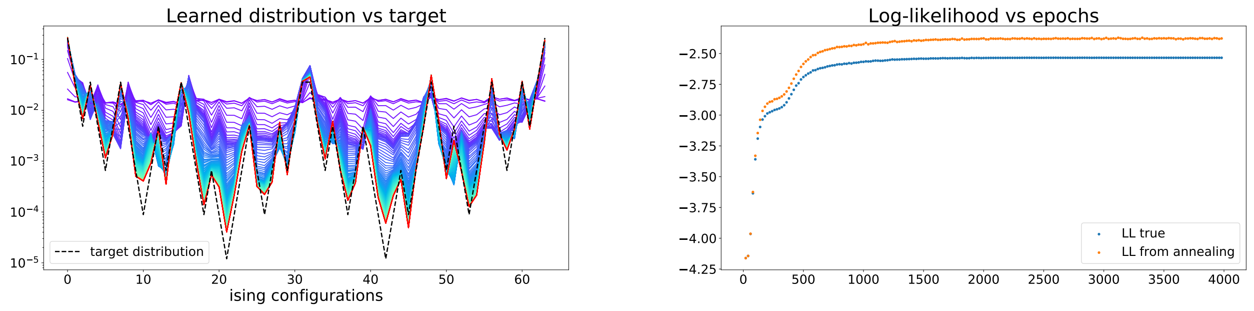

FIG. 1: The left hand plot shows the RBM distributions obtained at different number of training epochs, from 20 (blue) to 4000 (red) compared to the target distribution (dotted black line). The right hand plot presents is a comparison between the exact log-likelihood (blue) and that obtained from the annealing algorithm (orange). The uncertainty for the estimated log-likelihood is smaller than the 10%.

At the end of the training we expect

$$\frac{e^{-H(v)}}{Z^{\text{Ising}}} = \frac{e^{-H_\lambda^{\text{RBM}}(v)}}{\mathcal{Z}}, \tag{38}$$

which, for every v, implies

$$H(v) - H_\lambda^{\text{RBM}}(v) = \log \mathcal{Z} - \log Z^{\text{Ising}} = \text{constant}. \tag{39}$$

We can reformulate this statement in the following way: minimising the KL divergence only requires the difference between the Ising probability distribution and that of the RBM to go to zero

$$\begin{aligned} D_{\mathrm{KL}}\Big[q_{\mathrm{Ising}}(\mathbf{v})||p_\theta(\mathbf{v})\Big] &= \sum_{\mathbf{v}} \Big(q_{\mathrm{Ising}}(\mathbf{v}) \log\left(q_{\mathrm{Ising}}(\mathbf{v})\right) - q_{\mathrm{Ising}}(\mathbf{v}) \log\left(p_\theta(\mathbf{v})\right) \Big) \\ &= \sum_{\mathbf{v}} q_{\mathrm{Ising}}(\mathbf{v}) \Bigg[(\log \mathcal{Z} - \log Z^{\mathrm{Ising}}) - (H(\mathbf{v}) - H_\lambda^{\mathrm{RBM}}(\mathbf{v})) \Bigg], \end{aligned} \tag{40}$$

which in turn implies that the Hamiltonians and the corresponding partition functions may differ by a constant, getting cancelled in the above equation. Notice also, that this statement has to be true term-by-term once the machine has learned, *i.e.* for any state $\{\mathbf{v}\}$. This is observed numerically, where

$$H(\mathbf{v}^{(1)}) - H^{\mathrm{RBM}}(\mathbf{v}^{(1)}) = \cdots = H(\mathbf{v}^{(64)}) - H^{\mathrm{RBM}}(\mathbf{v}^{(64)}) = \log \mathcal{Z} - \log Z^{\mathrm{Ising}} = \mathrm{constant}, \tag{41}$$

up to numerical errors, for all the 64 possible states. In order to verify if this is in indeed the case in the training the machine, on the left hand side of Fig. 2, we plot the average $\langle H - H^{\mathrm{RBM}}\rangle_v$ of the 64 energy differences in Eq. 41, together with their corresponding standard deviation $\sigma(H - H^{\mathrm{RBM}})_v$, as a function of the training epochs. The plot on the right hand side of Fig. 2, shows only the values of the standard deviation per epoch which decreases until it reaches a constant. Notice that both the average and the standard deviation converge to constant values, -10.56 and 0.55 respectively, showing how the differences of Eq. 41 indeed converge to a common constant value of -10.56 up to a numerical error of 0.55. We have also checked that this numerical values are consistent with $\log \mathcal{Z} - \log Z^{\mathrm{Ising}}$, which is plotted, in orange, together with $\langle H - H^{\mathrm{RBM}}\rangle_v$ in the left hand side of Fig. 2: the former lies within 1 sigma from the latter, showing how the KL divergence in Eq. 40 indeed decreases towards zero along the training, converging to a situation where $H(v^{(i)}) - H^{\mathrm{RBM}}(v^{(i)})$ is constant.

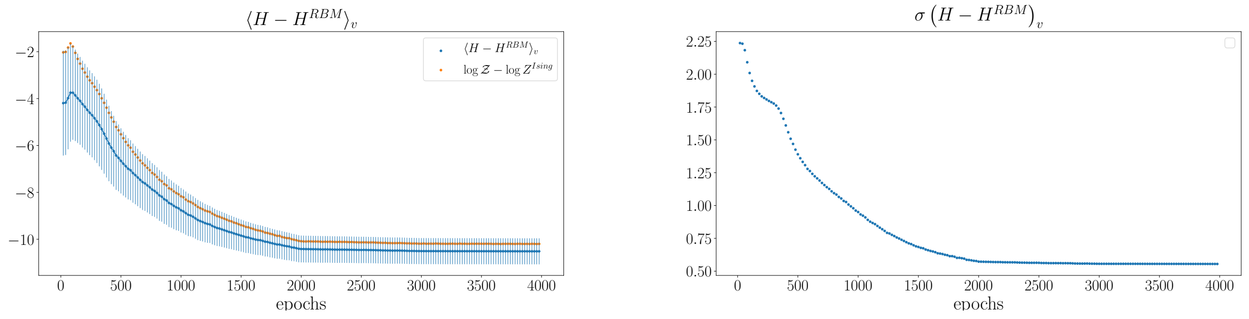

FIG. 2: The left hand plot shows $\langle H - H^{\mathrm{RBM}}\rangle_v$ (blue) and $\log \mathcal{Z} - \log Z^{\mathrm{Ising}}$ (orange) per epoch. The right hand plot shows the standard deviation on $\langle H - H^{\mathrm{RBM}}\rangle_v$ per epoch.

IV. RBM FOR THE 2D ISING MODEL

We now consider training sets consisting of spin configurations of 2-dimensional Ising systems at various temperatures. The configurations are generated using Magneto, setting $J/k_B = 1$, with temperatures above, at and below the critical temperature, starting from $T = 1.8$ to $T = 3.0$ in steps of $\Delta T = 0.1$. The training of each RBM on these configurations has been studied independently, with the aim of obtaining the set of parameters which provide the correct value of observables at each temperature, as compared with Magneto values. The observables for each trained RBM have been computed as described in Sec. II C, for different checkpoints along the training, and plotted as a function of the number of epochs. This allows us to monitor the training procedure, and to assess the number of epochs required to converge to the underlying distribution of the training data. This procedure has been performed for the case of an 8^2 lattice, as well as a 16^2 lattice. We highlight the increasing difficulties in training when the volume is increased.

L^2	N_{therm}	N_{measure}	Binning	algorithm	T	Steps ΔT
8^2	50000	100000	1	SW	1.8-3.0 & 2.27	0.1
16^2	50000	100000	1	SW	1.8-3.0 & 2.27	0.1

TABLE II: 2D Ising model parameters for data generation. Using the Swendsen-Wang (SW) algorithm, the autocorrelation drops below 1 therefore no binning was required. N_{therm} denotes the number of MC iterations used for thermalisation, while N_{measure} denotes the measurement taken after thermalisation, which are saved to be used as training examples for the RBM.

A. Ising simulation and RBM training parameters

The simulation parameters for generating Ising spins using Magneto [20] and training the RBM are presented in Tables II and III respectively. In the following we present detailed results for three systems of 8^2 spin configurations, at temperatures $T = 1.8$, $T = 2.2$ and $T = 3.0$ respectively. The procedure for training a system of 16^2 spins is reported in Appendix. B.

In order to train the RBMs we used 10^5 samples as training examples with batch-size 200. Although the training procedure is specific to each dataset and RBM architecture, we give a general prescription for the case where the number of hidden and visible nodes are chosen to be the same. Most our trainings were performed in three steps:

1. We start the training with a large learning rate α_1, the value of which has to be tested for. The idea for such a choice is to speed up the process of maximising the log-likelihood at the beginning. The CD parameter k_1, at this stage, is set equal to 1. These settings were kept as long as they were enough to ensure an increasing log-likelihood. As the number of epoch increases, we observe either the log-likelihood stabilising at a fixed value or starting decreasing. An example of this behaviour is reported in the Appendix, where, in the case of $T = 1.8$, it is observed that between 3000 and 4000 epochs, the log-likelihood starts to decrease. This behaviour is observed elsewhere in the literature, *e.g.* see Ref. [12].

2. In order to prevent the decrease in the log-likelihood, we decrease the variance in the estimate of the gradient of the log-likelihood, namely we increase the CD parameter to a larger value k_2. Again, the value is chosen according to whether or not it results in an increase in the log-likelihood. At this stage, we also decrease the learning rate α, in order to fine-tune the training as we approach the maximum of the log-likelihood.

3. A single fine-tuning such as the one described above may not be enough to fully train the machine. In that case, we iterate step 2 with a higher value of k and lower value of α. This has been done for all our trained machines.

4. At each of the above steps, we monitor the moments of the learned distribution and compare the values to the expected ones, obtained directly from the training set. This allows us to monitor if and how the training is progressing.

Therefore, the training of each machine in our case is characterised by three different phases with specific number of epochs, learning rates, and values of k, as well as comparing the moments extracted from the machine with the expected values obtained from the training data. Our choices for the hyperparameters are summarised in Table III. Further analyses regarding changes in the number of hidden nodes, training examples and overfitting are discussed in the appendix.

The four quantities used to monitor training, *i.e.* the log-likelihood, free energy, loss function and reconstruction error are reported as functions of the training epoch in Fig. 3, for three different temperatures. They all have the expected behaviour, as described in Sec. II A. We stop the training process when there is no further increase in the log-likelihood and, after fine-tuning with a higher value of k and smaller α, the first and second moments of distributions generated by machines at different epochs are statistically equivalent. We can conclude that the training algorithm, so far, has been successful. In the next section we assess how well the machine is able to predict the observables, as compared to the measured Magneto values.

Visible	Hidden	N_{train}	Batch size	Epochs 1	LR α_1	CD k_1	Epochs 2	LR α_2	CD k_2	Epochs 3	LR α_3	CD k_3	T	Steps ΔT
8^2	8^2	10^5	200	3000	0.1	1	1000	0.01	5	1000	0.001	10	1.8-2.1	0.1
8^2	8^2	10^5	200	2000	0.01	1	1000	0.001	5	1000	0.0001	5	2.2-3.0	0.1
16^2	16^2	10^5	200	8000	0.01	10	2000	0.01	20	//	//	//	1.8	//

TABLE III: Parameters for training the RBMs. The number of visible and hidden nodes for each case is presented. N_{train} denotes the number of examples used to train the machine. T indicates the temperatures of the given systems, with intervals $\Delta T = 0.1$. The learning rate (LR), α, is set fixed for Epochs 1 for the initial training and is later fine-tuned until the observables results for the last several epochs converge within statistics. As advised in Ref. [12], whenever the increase in the log-likelihood plateaus and then starts to decrease, we reduce the value of α and increase k. The number of epochs in each phase depends on the size and temperature of each system. Our tests indicate that a higher value of k, *i.e.* $k = 10$, needs to be used to train the machine on the larger configuration $v^2 = 16 \times 16$, which is then increased to $k = 20$ for the last phase of the training.

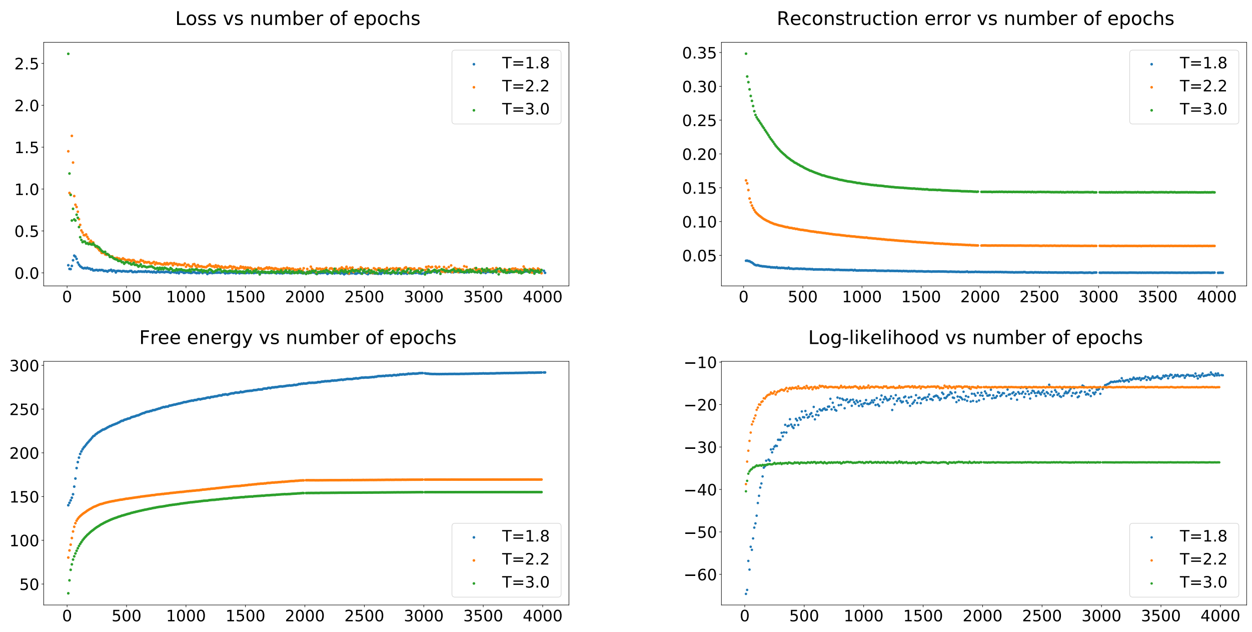

FIG. 3: From left to right, loss function and reconstruction error in the first line, free energy and log-likelihood in the second one. The results are presented for three different $T = 1.8$, $T = 2.2$ and $T = 3.0$.

B. Gibbs sampling

As anticipated in Sec. II C, for each trained machine we have computed the average magnetisation, susceptibility, energy and heat capacity, together with their corresponding uncertainties. The results have been compared to the values obtained from the training set, in order to examine if our machines are able to reproduce the first and second moments of the underlying distribution. In order to sample the states from the RBM distribution, required to implement Eq. (36), we have used Gibbs Sampling, Ref. [12]. Block Gibbs sampling is a particular Markov Chain Monte Carlo method (MCMC). The general idea behind any MCMC method is that of setting up a Markov Chain that converges to the distribution we want to sample from, in this case Eq. (3), and then running the chain long enough to reach its stationary point. In the case of Gibbs sampling, this translates into building the chain

$$\mathbf{v}^{(0)} \to \mathbf{h}^{(0)} \to \mathbf{v}^{(1)} \to \mathbf{h}^{(1)} \to \mathbf{v}^{(2)} \;\; \cdots \;,$$

where $\mathbf{h}^{(l)}$ and $\mathbf{v}^{(l+1)}$ are sampled from the conditional probabilities given in Eq. 7 and Eq. 8 respectively. It can be shown that this chain converges to $P(\mathbf{v}, \mathbf{h}; \theta)$, which implies that at a certain point in the chain, the states $\mathbf{h}^{(l)}$ and

Algorithm	N_{therm}	N_{measure}	Binning
Gibbs	20000	2×10^6	100
Metropolis	50000	1×10^6	50

TABLE IV: Gibbs and Metropolis sampling parameters, for MC simulations on an already trained RBM. The measurements are made on configurations generated after the initial thermalisation step N_{therm}. The binning factor indicates the number of successive measurement binned to ensure the remaining are indeed independent.

$\mathbf{v}^{(l)}$ will be sampled according to the joint probability distribution encoded in the RBM. The word *block* here, refers to the fact that we can simultaneously update all variables h_i given $\mathbf{v}^{(l)}$, and vice versa, as the RBM is a restricted network which means that the variables within each layer are independent of each other. Note that the Contrastive Divergence algorithm, used to obtain estimates of the log-likelihood gradient as described in Sec.II, is only a Gibbs sampling run with a single step, to get $\mathbf{v}^{(1)}$ from $\mathbf{v}^{(0)}$ through $\mathbf{v}^{(0)} \to \mathbf{h}^{(0)} \to \mathbf{v}^{(1)}$. One of the main advantages of Gibbs sampling over other MCMC methods (see the next section on Metropolis sampling for an example) is that it converges relatively fast. This allowed us to measure and plot the observables at many epochs along the training, together with their corresponding uncertainties, as an indication of progress in learning. The details concerning the Gibbs sampling parameters are reported in Table (IV). After a thermalisation step of 2×10^4 configurations, we measure magnetisation and energy on every 100th configuration, allowing the Markov chain to run for 2×10^6 configurations.

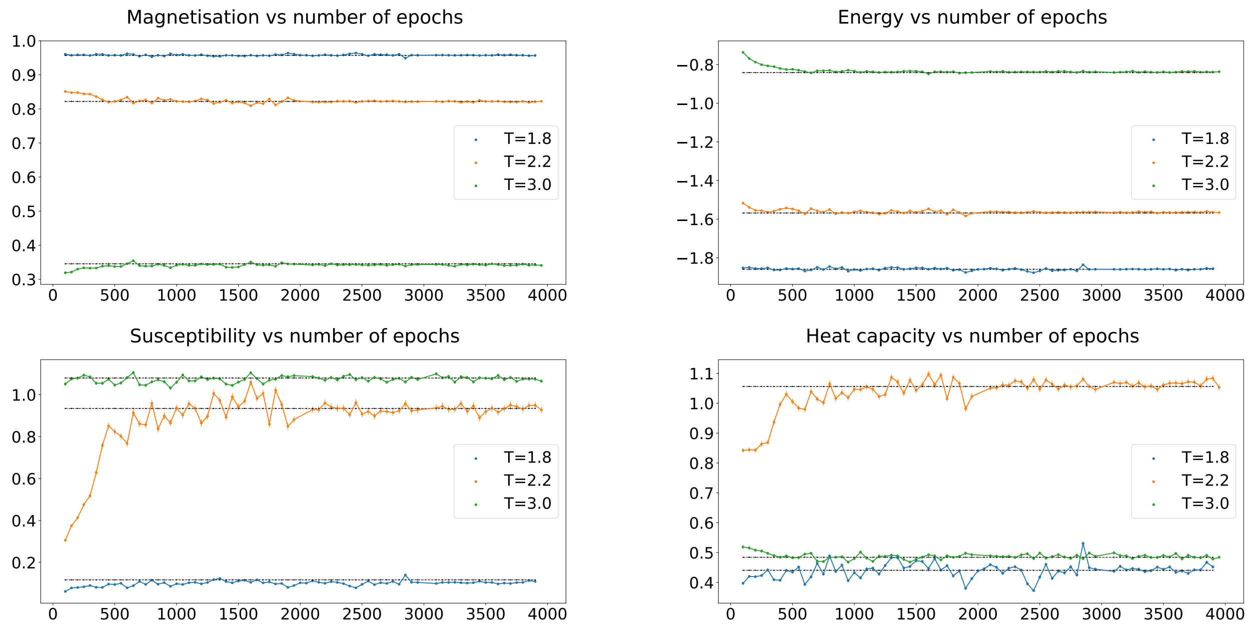

FIG. 4: From left to right, magnetization and energy in the first line, susceptibility and heat capacity in the second one. The results are presented for three different T=1.8, T=2.2 and T=3.0, as function of the training epoch.

C. Metropolis sampling

In order to have an independent measurement of the observables, we implemented the Metropolis algorithm. The algorithm takes, as input, the parameters of a trained RBM, $\theta = \{\mathbf{w}, \mathbf{b}, \mathbf{c}\}$. Starting from a random configuration, a new spin configuration is proposed by randomly flipping a spin using the transformation $v_i = 1 - v_i$. Letting u to be a random number between zero and one from the uniform distribution, if,

$$\min\left(1, \frac{p_{\mathrm{RBM}}(\mathbf{v}_{\mathrm{new}})}{p_{\mathrm{RBM}}(\mathbf{v}_{\mathrm{old}})}\right) > u \ , \tag{42}$$

we accept the proposed configuration. Otherwise we reject the new configuration and return to the original one. At each Monte Carlo sweep, this process is repeated for all spin variables. Since we take the ratio between the new and the old probabilities, the partition functions cancel and so it does not need to be estimated. The procedure is repeated until the thermalisation is reached. These can be checked by checking that the Monte Carlo history of an observable e.g. magnetisation, follows a normal distribution. We chose to discard the first 5×10^5 configurations for the thermalisation step. See *e.g.* Fig. 27 in the appendix. Note that the generated configurations in the $0, 1$ basis are mapped to those of the Ising model in the $-1, 1$ according to $v_{\mathrm{ising}} = 2v_{RBM} - 1$.

In order to take independent measurements of the observables, successive number of measurements of $|m|$ and E were binned. To determine the bin size, the error on *e.g.* $|m|$ vs bin size was plotted in Fig. 29. To confirm independence, the autocorrelation time defined as:

$$A(T) = \frac{1}{N_{\mathrm{meas}} - t}\left[\sum_{i=1}^{N_{\mathrm{meas}}-t} m^{(i)} m^{(i+t)} - (\overline{m}^{(i)})^2\right] / \left(\overline{(m^{(i)})^2} - (\overline{m}^{(i)})^2\right) \ , \tag{43}$$

where N is the total number of Monte Carlo steps, was also computed. The autocorrelation time vs MC steps is plotted in Fig. 30. Based on these two plots, choosing bin size equal to 50, seems sufficient for independent measurements to be obtained. We present the parameters of the Metropolis sampling in Table. IV. The magnetisation $\langle |m| \rangle$, energy $\langle E \rangle$, susceptibility χ and heat capacity C_V, are then measured. The error on these quantities have been computed using the statistical bootstrap procedure.

We also tried to extract the third and fourth moments, however, the corresponding uncertainties turned out to be of the same order as the moments themselves.

D. Observable predictions at all temperatures

In the following we report on the results of Ising observable measurements, as extracted from a trained RBM, for 13 different values of temperature. In Fig. 5 the values for the different observables are shown, obtained independently from Gibbs and Metropolis sampling. The expected values from magneto are also plotted. We can observe resulting curves reproduce the expected values for magnetisation, energy, susceptibility and heat capacity. However in these plots it is not possible to discern between the results obtained by different sampling methods and expected magneto values. To indicate more clearly the precision with which the RBM is able to reproduce the moments, we plot the observables normalised over their expected values, with their corresponding error bars in Fig. (6). Notice that, as expected, the first moments *i.e.* magnetisation and energy, agree better with the corresponding magneto results as compared to the second moments. Moreover, the measurements using the two different sampling methods always agree within 2σ. For the first moments, the values measured from the RBM distribution are almost always compatible with the expected values within 2σ, while in the case of the second moments some discrepancy is observed, *e.g.* see susceptibility at $T = 1.8$. We conclude that the RBM is in general able to obtain precise values for the first moments of the underlying distribution at all temperature studied, however, for the second moments the agreement is not always as precise.

V. EXTRACTING COUPLINGS FROM THE RBM

In the previous sections we described how to train an RBM for a 2D Ising model and assess the quality of the training by comparing the derived first and second moments, from the RBM, to the corresponding observables measured using the target distribution. This is a strong check that the machine has trained correctly, as first and second moments can always be measured from the training data directly. The predictive power of the machine, however, is in capturing the interactions between visible units, at every temperature. In this section we address how to extract the couplings between the visible units from the trained RBM, in the case of a binary system *e.g.* the Ising model.

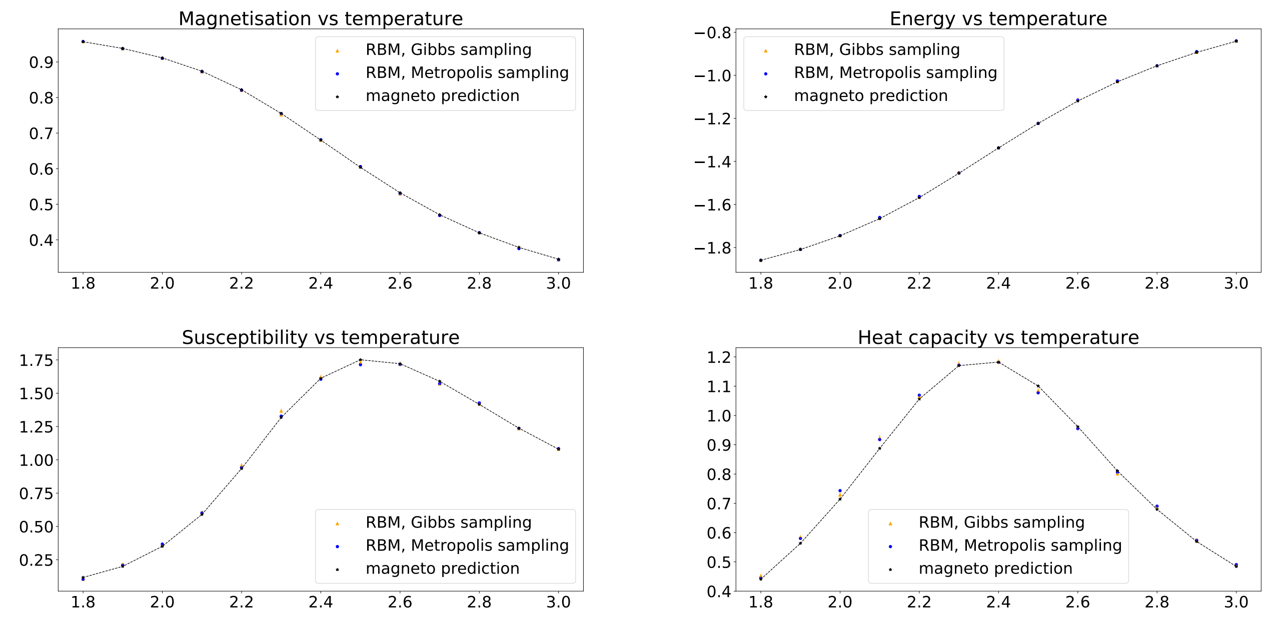

FIG. 5: Observables vs temperature. Results coming from Gibbs and Metropolis samplings are compared to the expected values from Magneto.

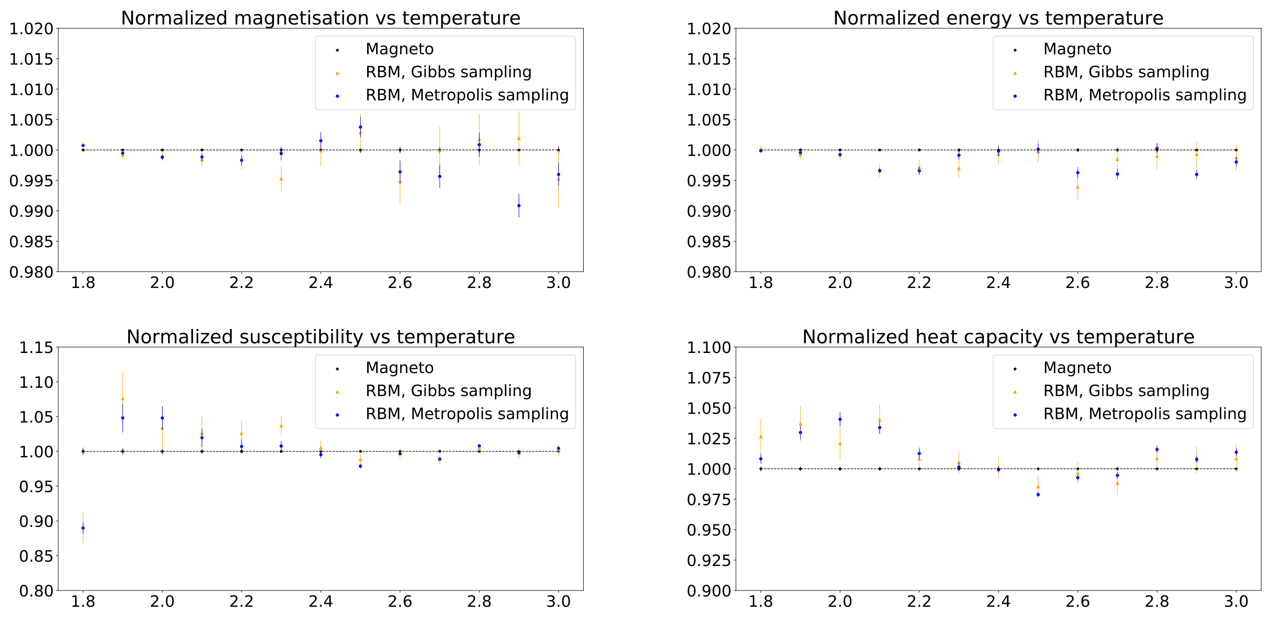

FIG. 6: Observables, normalised by the expected magneto value, vs temperature.

Starting from an observation made in Chapter 16 of Ref. [11], we derive a closed form expression for the two-point coupling between the visible units, as a function of the parameters of the trained RBM, and compare the results with the two-points interactions expected from the Ising model. We then generalise the computation to higher order interactions, presenting a closed form expression for the tensor describing the n-point coupling between visible units learned by the machine, as a function of the learned RBM parameters.

In order to extract the coupling from the weight matrix of the trained RBM, W_{ij}, we expand the generating function in powers of v_{j_1}. Marginalising over the hidden units, the energy of the RBM as a function of the visible units can be written as:

$$\begin{aligned} E(\mathbf{v}) &= \ln \sum_{\mathbf{h}} e^{E(\mathbf{v},\mathbf{h})} \\ &= \ln \sum_{h_i} e^{-\sum_j b_j v_j - \sum_i c_i h_i - \sum_{i,j} h_i W_{ij} v_j} . \end{aligned} \tag{44}$$

Since the first term in $E(\mathbf{v},\mathbf{h})$ is independent of the hidden layer, it can be brought outside the sum. This is the *visible bias* term. $E(\mathbf{v})$ then becomes,

$$\begin{aligned} E(\mathbf{v}) &= -\sum_j b_j v_j - \sum_i \ln \sum_{h_i} e^{c_i h_i} e^{\sum_j h_i W_{ij} v_j} \\ &= -\sum_j b_j v_j - \sum_i \ln \sum_{h_i} q(h_i) e^{t h_i} \ , \end{aligned} \tag{45}$$

where we have defined $t \equiv \sum_j W_{ij} v_j$ and $q(h_i) \equiv e^{c_i h_i}$. This allows us to introduce the cumulant generating function, similar to what was done in Ref. [11]:

$$K_i(t) \equiv \ln \sum_{h_i} q(h_i) e^{t h_i} = \sum_n \frac{\kappa_i^{(n)} t^n}{n!} \ , \tag{46}$$

where the nth cumulant $\kappa_i^{(n)} = \partial_t^n K_i(t)|_{t=0}$. Expanding the generating function as a power series in n gives,

$$\begin{aligned} E(\mathbf{v}) &= -\sum_j b_j v_j - \sum_i \kappa_i^{(0)} - \sum_i \kappa_i^{(1)} t - \sum_i \frac{\kappa_i^{(2)} t^2}{2!} - \dots \\ &= -\sum_i \kappa_i^{(0)} - \sum_j \left(b_j + \sum_i \kappa_i^{(1)} W_{ij} \right) v_j - \frac{1}{2!} \sum_{j_1,j_2} \left(\sum_i \kappa_i^{(2)} W_{ij_1} W_{ij_2} \right) v_{j_1} v_{j_2} - \dots \ , \end{aligned} \tag{47}$$

where in the final line we have used the definition of t to rewrite $E(\mathbf{v})$ in terms of n-point interactions between the visible units. When the RBM has learned the physics of the Ising model, we would expect that $E(\mathbf{v}) = H_{\text{ising}}(\mathbf{v})$ up to a constant. Since the two energies can differ by an overall constant without affecting the physical observables, we are not interested in $-\kappa_i^{(0)}$. However, the two energies are equal across states and we would expect that $E(\mathbf{v})$ and $H_{\text{ising}}(\mathbf{v})$ to be equal, order-by-order in $\mathbf{v}$. For the standard Ising interactions, with no external field,

$$H_{\text{Ising}}(\tilde{\mathbf{v}}) = \sum_{j_1,j_2} H_{j_1 j_2} \tilde{v}_{j_1} \tilde{v}_{j_2} \ , \tag{48}$$

where $\tilde{\mathbf{v}}$ describes the spin states, with each spin taking the value $+1$ or -1. $H_{j_1 j_2}$ is the appropriate Ising matrix with nearest neighbour interactions. In this Ising model, the nearest neighbour coupling is set to $J = -1/T$ where T is the temperature of the model. Therefore, $H_{j_1 j_2}$ is zero except for components which correspond to nearest neighbour interactions, which are equal to $\frac{1}{2T}$. This is shown pictorially in Fig. 7 for the case of an 8×8 lattice. The x and y axes represent spins on this two-dimensional lattice, numbered from 0 to 64. Notice the structure of couplings due to periodic boundary conditions.

States in the $\{-1, 1\}$ basis can be related to those in the $\{0, 1\}$ basis by the simple transformation $\tilde{\mathbf{v}} = 2\mathbf{v} - 1$. This allows us to rewrite the Ising Hamiltonian in the $\{0, 1\}$ basis as:

$$H_{\text{ising}}(\mathbf{v}) = 4 \sum_{j_1,j_2} H_{j_1 j_2} v_{j_1} v_{j_2} - 4 \sum_{j_1} \left(\sum_{j_2} H_{j_1 j_2} \right) v_{j_1} + \left(\sum_{j_1,j_2} H_{j_1 j_2} \right) , \tag{49}$$

where the symmetric form of $H_{j_1 j_2}$ has been exploited to simplify the expression. The 2-point interaction in Eq. 49 and Eq. 47 can now be compared to extract the coupling, $\frac{1}{2T}$, from the trained RBM.

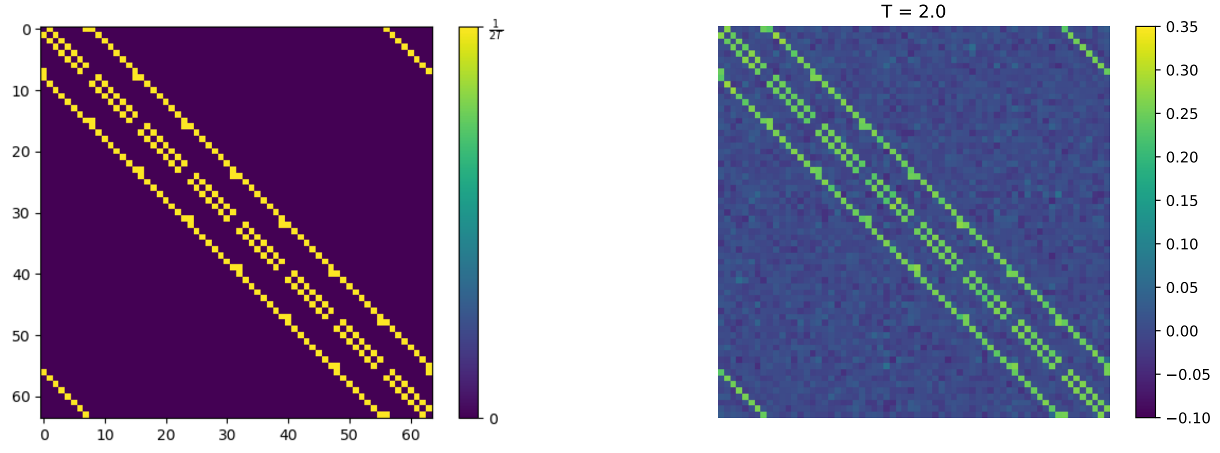

FIG. 7: The matrix of interactions, $H_{j_1 j_2}$, for an 8×8 Ising lattice with periodic boundary conditions as used in the generation of the training set (left) and the one learnt by the RBM at the end of the training (right). In this example T=2.0. The spins are labelled from 0 to 64 and the nearest neighbour structure is evident.

There is however a subtlety which arises in the expansion of the cumulant generating function. We can see that in $\{0,1\}$ basis:

$$v_j^n = v_j \quad , \quad n \in \mathbb{Z}^+ \; . \tag{50}$$

This implies that higher order terms in n also contribute to the N_v-point function, where N_v is the number of *distinct* vertices. To make this statement more clear, let us write Eq. 47 in a more general form,

$$E(\mathbf{v}) = -\kappa_i^{(0)} - \sum_j \left(b_j + \sum_i \kappa_i^{(1)} W_{ij} \right) v_j - \sum_{n>1} \frac{1}{n!} \sum_{j_1 \ldots j_n} \left(\sum_i \kappa_i^{(n)} W_{ij_1} \ldots W_{ij_n} \right) v_{j_1} \ldots v_{j_n} \; . \tag{51}$$

In other words, n can be thought of as counting the powers of W_{ij}. For each n, the contributions from the sum over the visible vertices, $\sum_{j_1 \ldots j_n}$, can be grouped by the number N_v of distinct vertices being multiplied, *i.e.* the N_v-point functions. For example, using Eq. 50, we see that the numerical contributions to the 2-point interactions are terms proportional to $v_{j_1}^k v_{j_2}^{n-k} = v_{j_1} v_{j_2}$ where the vertices are distinct, *i.e.* , $j_1 \neq j_2$. The number of combinations of powers of v_{j_1} and v_{j_2} for a given n are simply the binomial coefficients. Therefore, 2-point contributions from all orders in n can be written as:

$$\sum_{n>1} \frac{1}{2(n!)} \sum_{0<k<n} \sum_{j_1 \neq j_2} \left(\sum_i \kappa_i^{(n)} \binom{n}{k} W_{ij_1}^k W_{ij_2}^{n-k} \right) v_{j_1} v_{j_2} \; , \tag{52}$$

where the factor of two has been included to account for the double counting arising from the symmetry under the exchange of j_1 and j_2. Performing the sum over k gives,

$$\frac{1}{2} \sum_{n>1} \frac{1}{n!} \sum_{j_1 \neq j_2} \left(\sum_i \kappa_i^{(n)} \left[(W_{ij_1} + W_{ij_2})^n - (W_{ij_1})^n - (W_{ij_2})^n \right] \right) v_{j_1} v_{j_2} \; . \tag{53}$$

Comparing this result to the Ising 2-point interaction in Eq. 49, provided that the RBM is properly trained, the Ising pair-wise coupling can be written as:

$$H_{j_1 j_2} = \frac{1}{8} \sum_{n>1} \frac{1}{n!} \sum_i \kappa_i^{(n)} \left[(W_{ij_1} + W_{ij_2})^n - (W_{ij_1})^n - (W_{ij_2})^n \right] \cdot \tag{54}$$

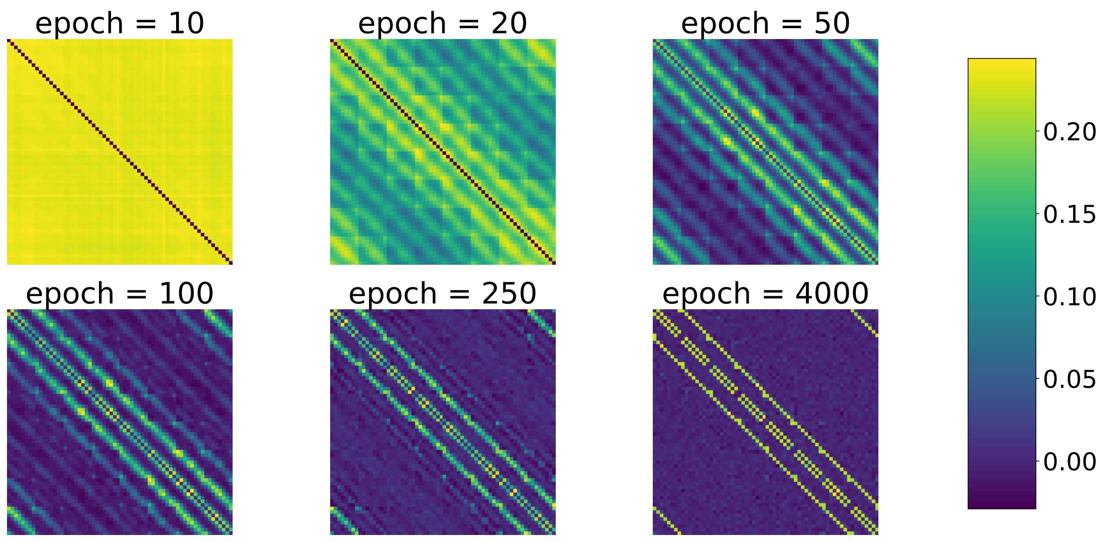

FIG. 8: $H_{j_1j_2}$ extracted from RBMs at different stages of the training (10, 20, 50, 100, 250 and 4000 epochs), for $L^2 = 8\times 8$, $h^2 = 8\times 8$ and temperature $T = 2.2$. As the machine approaches the end of the training, the expected structure of Fig.7 becomes more and more evident.

On the other hand, $\kappa_i^{(n)} = \partial_t^n K_i(t)|_{t=0}$, giving

$$\begin{aligned} H_{j_1j_2} =& \frac{1}{8}\sum_n \left(\frac{1}{n!}\sum_i \left[(W_{ij_1}+W_{ij_2})^n - (W_{ij_1})^n - (W_{ij_2})^n\right]\partial_t^n K_i(t)|_{t=0}\right) + \frac{1}{8}\sum_i K_i(0) \\ =& \frac{1}{8}\sum_i \left(e^{(W_{ij_1}+W_{ij_2})\partial_t} - e^{W_{ij_1}\partial_t} - e^{W_{ij_2}\partial_t} + 1\right) K_i(t)|_{t=0} \;. \end{aligned} \tag{55}$$

We recognise the shift operators $e^{a\partial_x} f(x) = f(a+x)$ in the second line of the above equation. Therefore, the expression can be simplified further as:

$$H_{j_1j_2} = \frac{1}{8}\sum_i \Big(K_i(W_{ij_1}+W_{ij_2}) - K_i(W_{ij_1}) - K_i(W_{ij_2}) + K_i(0)\Big), \tag{56}$$

which is the closed form expression for $H_{j_1j_2}$, including all order contributions in n. More explicitly, using Eq.(46) we have

$$K_i(t) = \log\left(1+e^{c_i+t}\right) \tag{57}$$

so that

$$H_{j_1j_2} = \frac{1}{8}\sum_i \ln\frac{(1+e^{c_i+W_{ij_1}+W_{ij_2}})(1+e^{c_i})}{(1+e^{c_i+W_{ij_1}})(1+e^{c_i+W_{ij_2}})} \tag{58}$$

Using the trained RBMs, the coupling given by Eq. 56 was evaluated for temperatures ranging from $T = 1.8$ to $T = 3.0$. The results, presented in Fig. 9, have the same nearest neighbour structure as the Ising model used to train the machines, in Fig. 7 with generic temperature T. In Fig. 10, we have plotted the corresponding histograms of the values $H_{j_1j_2}$ at various temperatures. The expected bimodal structure of coupling matrices is observed, with most of the entries, *i.e.* those associated with the non-nearest neighbour spins interactions, being centred around zero, while the nearest neighbour interactions introduce a distinct second peak, around the value of the coupling. By taking the

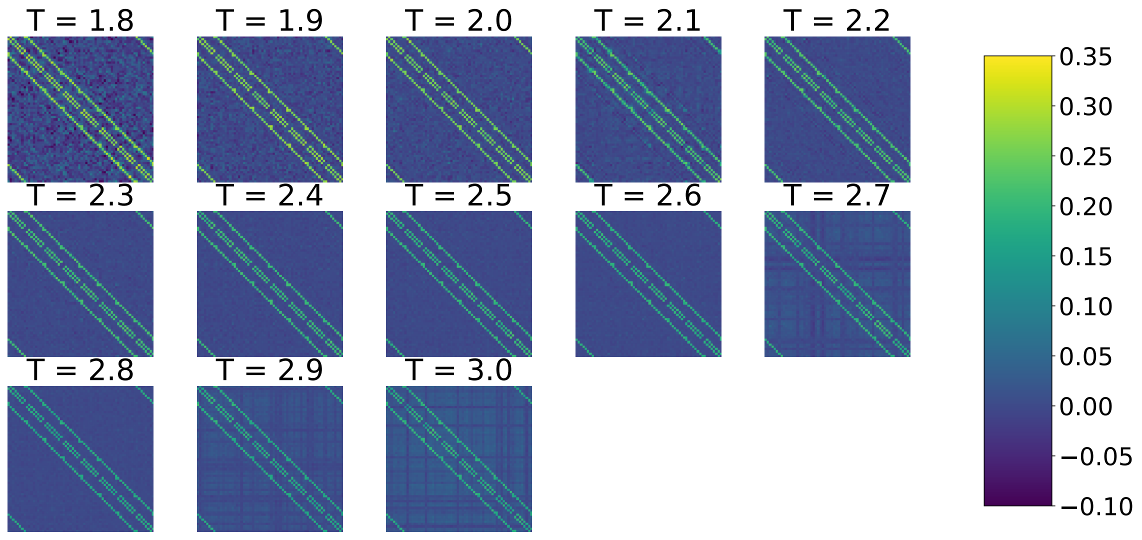

FIG. 9: The interaction matrix $H_{j_1 j_2}$ extracted from RBMs, with $L^2 = 8 \times 8$ and $h^2 = 8 \times 8$, trained at a temperature indicated above each subplot. Again, the spins are labelled from 0 to 64 and show the same structure as for the generic Ising training set in Fig. 7.

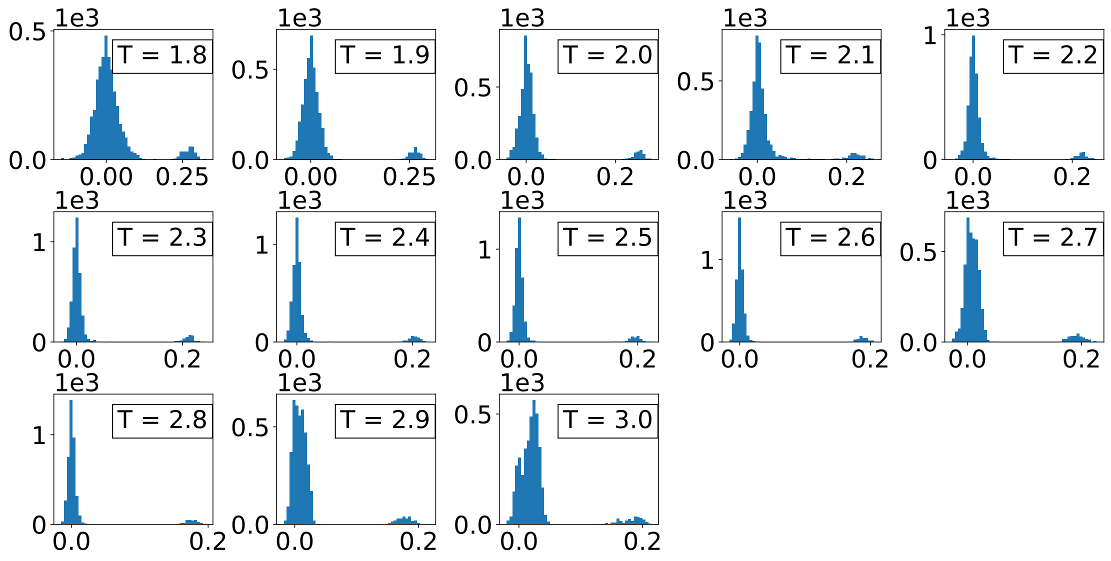

FIG. 10: Histograms of the entries of $H_{j_1 j_2}$ extracted from RBMs trained at a temperature indicated above each subplot. As it can be observed, there are always two peaks: The smaller peak, represents the number of nearest neighbour on the y-axis, with the value of the coupling indicated on the x-axis; the larger peak represent all other sites that are not nearest neighbours and are not expected to couple to each other, hence it being centred around zero on the x-axis.

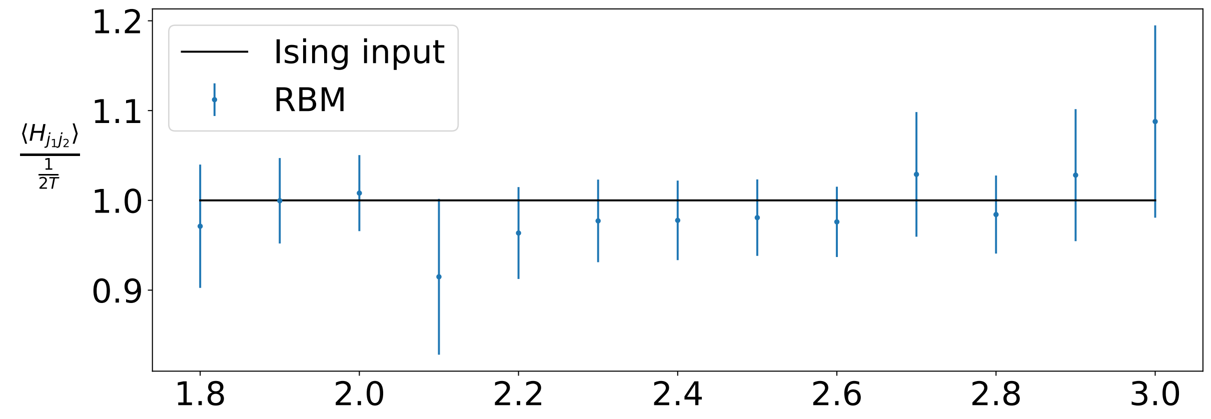

FIG. 11: The predicted 2-point interaction coupling, for $L^2 = 8 \times 8$ at different values of temperature, normalised by the corresponding true value $1/2T$. The coupling is extracted from the nearest neighbour diagonals observed in the interaction matrices, with its error bar computed by taking the standard deviation of the diagonal components. The predicted values agree with the expected ones within statistics.

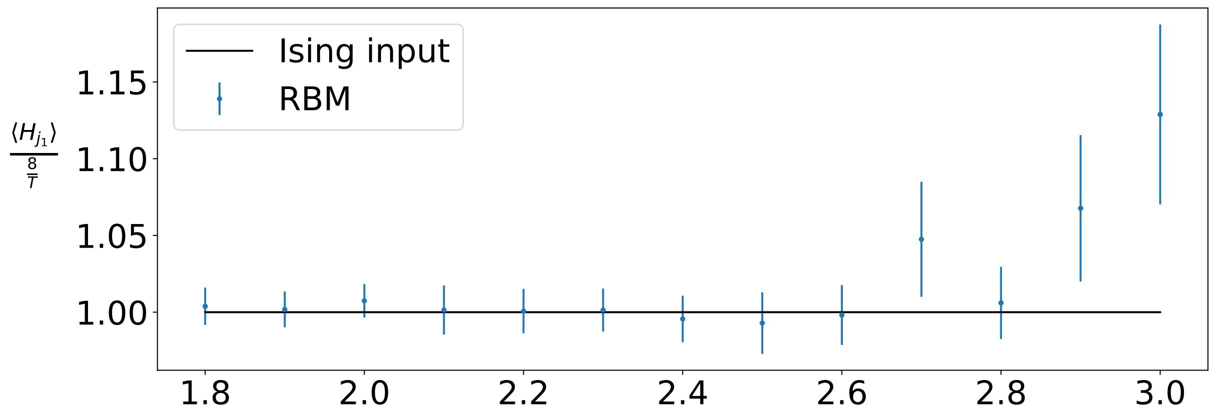

FIG. 12: The linear terms extracted by the trained machine, normalised by the corresponding true value $8/T$. All predicted values are compatible with the true ones within 2σ.

average and the standard deviation across the nearest neighbour diagonals we can compare the couplings extracted from the RBM against the exact values from the Ising model, as shown in Fig. 11. We see that the RBM predictions agrees with the analytical results within statistics.

It is also interesting to look at the change in the couplings along the training: this is shown in Fig. 8, where the 2-points interaction matrix for $T = 2.2$ is plotted at different stages of the training. The training algorithm starts by initialising the RBM visible-hidden interaction terms W_{ij} from a normal distribution centred around zero. The bias terms are also initialised to zero. In the early epochs, we observe no particular structure in the H_{j_1,j_2} interaction matrix. As learning progresses, more of the nearest neighbour structure appears, together with further non nearest neighbour correlations. Towards the final phases of the training, only the nearest neighbour structure remains, with the rest of the interactions being almost zero.

Let us now return to Eq. 51 and collect terms linear in v_j. The terms contributing are those with $j_1 = j_2 = \ldots = j_n$, giving,

$$\sum_j J_j v_j = -\sum_j \left(b_j + \sum_i \kappa_i^{(1)} W_{ij} \right) v_j - \sum_{n>1} \frac{1}{n!} \sum_j \left(\sum_i \kappa_i^{(n)} \left(W_{ij}\right)^n \right) v_j . \tag{59}$$

The vector J_i can be read from the linear term in Eq. 49,

$$J_i = 4 \sum_j H_{ij} = \frac{8}{T} \ , \tag{60}$$

where we have used $\sum_j H_{ij} = 2/T$, for every component i in the Ising 2-point interaction matrix H_{ij}. The second term on the right hand side of the Eq. 59 can be treated in the same way as the 2-point interaction described above above,*i.e.* ,

$$\sum_{n>1} \frac{1}{n!} \sum_j \left(\sum_i \kappa_i^{(n)} \left(W_{ij}\right)^n \right) v_j = \sum_j \sum_i \left[K_i \left(W_{ij}\right) - K_i \left(0\right) \right] v_j - \sum_j \sum_i \kappa_i^{(1)} W_{ij} v_j \ . \tag{61}$$

It then follows that,

$$J_j = -\left(b_j + \sum_i K_i \left(W_{ij}\right) - \sum_i K_i \left(0\right) \right) = -\left(b_j + \sum_i \log \left(\frac{1 + e^{c_i + W_{ij}}}{1 + e^{c_i}} \right) \right) . \tag{62}$$

This allows us to extract the value of an external magnetic field, if present, from the parameters of the trained RBM. The original training set was generated without any external magnetic fields, therefore, if working in the $\{-1, 1\}$ basis, we do not expect to observe any linear term from the trained machine. However, we train the machine in the $\{0, 1\}$ basis, which implies the existence of a linear term $\vec{J}$ such that $J_i = 8/T$ for every component i. To extract this value from the trained machine we have taken the average across all the entries of the vector J_j defined in Eq. 62. The linear term extracted from the trained machine are plotted in Fig. 12, where the error bar is obtained by computing the standard deviation of the entries J_j. Again, we see how these values are compatible with the expected ones within 2σ.

The 3- and 4-point interactions can also be calculated and plotted. This is mainly used as a sanity check to confirm that the RBM has not incorrectly learned unexpected additional interactions. For the 3 point-interactions, we want to include all combinations which leave three distinct indices, similar to Eq. 52 we get,

$$\sum_{n>2} \frac{1}{6(n!)} \sum_{0<l<(n-k)} \sum_{0<k<(n-1)} \sum_{j_1 \neq j_2 \neq j_3} \left(\sum_i \kappa_i^{(n)} \binom{n}{k} \binom{n-k}{l} W_{ij_1}^k W_{ij_2}^l W_{ij_3}^{n-k-l} \right) v_{j_1} v_{j_2} v_{j_3} \ . \tag{63}$$

Summing over k and l gives

$$\begin{aligned} \sum_{n>2}^{\infty} \frac{1}{6(n!)} \sum_{j_1 \neq j_2 \neq j_3} \Big(\sum_i \kappa_i^{(n)} \big[& (W_{ij_1} + W_{ij_2} + W_{ij_3})^n - (W_{ij_1} + W_{ij_2})^n - (W_{ij_1} + W_{ij_3})^n - (W_{ij_2} + W_{ij_3})^n \\ & + (W_{ij_1})^n + (W_{ij_2})^n + (W_{ij_3})^n \big] \Big) v_{j_1} v_{j_2} v_{j_3} \ . \end{aligned} \tag{64}$$

If we add an subtract terms corresponding to $n = 0, 1, 2$ then the sum over n can be evaluated as before, giving

$$\begin{aligned} \frac{1}{6} \sum_{j_1 \neq j_2 \neq j_3} \sum_i \Big(& K_i(W_{ij_1} + W_{ij_2} + W_{ij_3}) - K_i(W_{ij_1} + W_{ij_2}) - K_i(W_{ij_1} + W_{ij_3}) - K_i(W_{ij_2} + W_{ij_3}) \\ & + K_i(W_{ij_1}) + K_i(W_{ij_2}) + K_i(W_{ij_3}) + K_i(0) \Big) v_{j_1} v_{j_2} v_{j_3} \ . \end{aligned} \tag{65}$$

We can then write the interaction tensor explicitly as

$$\frac{1}{6}\sum_i \ln \frac{(1+e^{c_i+W_{ij_1}+W_{ij_2}+W_{ij_3}})(1+e^{c_i+W_{ij_1}})(1+e^{c_i+W_{ij_2}})(1+e^{c_i+W_{ij_3}})}{(1+e^{c_i+W_{ij_1}+W_{ij_2}+})(1+e^{c_i+W_{ij_1}+W_{ij_3}})(1+e^{c_i+W_{ij_2}+W_{ij_3}})(1+e^{c_i})} \,. \tag{66}$$

One can observed the pattern emerging from (58) and (66) and write down a general form for the N-point interaction tensor

$$\frac{1}{N!}\sum_{l=0}^{N}(-1)^l \sum_{\alpha_1<\ldots<\alpha_{N-l}} K_i(W_{i,j_{\alpha_1}}+\ldots+W_{i,j_{\alpha_{N-l}}}) \tag{67}$$

where $\{\alpha_i\} = [1..N]$ and the product simply multiplies the $\binom{N}{N-l}$ combinations of choosing $N-l$ unique indices from $\{\alpha_i\}$. Using Eq. 67, histograms for the tensors for the 3- and 4-point interaction can then be plotted for the trained RBMs, to check if any higher order interactions were learned. It is expected that the histograms have a single peak at zero. These histograms are presented in Fig. 25 and Fig. 26 of App. E, respectively.

VI. CONCLUSIONS

We have trained several RBMs on 1- and 2-dimensional Ising models at various values of temperature. The training procedure in each case has been discussed in detail. We have used five different criteria to test whether the learning process has been successful. The first four are measurements of the loss function, reconstruction error, free energy and log-likelihood throughout the training procedure. The measurement of the latter, which is the quantity that is being maximised in the algorithm, involves the estimation of the partition function of the model, which is computed using annealed importance sampling. The fifth and last criterion, is the measurements of the first and second moments of the distribution, as given by the RBM, and comparing them to those obtained directly from the training set. These five criteria are essential to the RBM being trained correctly. Hence, we have provided a generic prescription in training an RBM on a binary model, from our experience. Moreover, we have used the RBM to predict the interaction couplings between the Ising spins by re-summing the cumulant generating function. The re-summations can be performed for any model with binary $\{0,1\}$ states. The ability to extract the couplings exactly can be useful in studying the relation between the RBM and the renormalization group (RG), as first noted by Ref. [7]. In this paper, we have also demonstrated the difficulty in training an RBM with smaller number of hidden nodes as compared to the visible. The numerical verification of RG is therefore left for future work. Another application of predicting the couplings is for the case where the interactions between the visible nodes of the RBM are unknown. In general, there couplings are more complicated than simple pair-wise interactions and as such are not extractible from the data directly with conventional methods. The RBM has no such assumption, as there are indirect all-to-all connections between the visible nodes, via the hidden layer. As the machine learns from the data, some of these connections turn-off while others increase in strength. This makes the RBMs a powerful tool for predicting complex connections between the visible nodes and their relative strength.
In this work, the RBM is limited in the input data format being binary but we explicitly chose a simple model to have the most complete theoretical understanding of the issues in training it. The RBM, given the limitation in the input data, is in fact a general solution to the problem of approximating an arbitrary distribution of boolean states. Delving further into the relation between the RBMs and RG is left for future work. Furthermore, we plan to extend this work to gain a better understanding of other generative models like VAEs and GANs.

ACKNOWLEDGMENTS

AK is a cross-disciplinary post-doctoral fellow supported by funding from the University of Edinburgh and Medical Research Council (core grant to the MRC Institute of Genetics and Molecular Medicine). AK is also most grateful to Chris Ponting, both for his academic and financial support of this project via the MRC Programme, grant number MC_UU_00007/15. GC acknowledges funding by Intel and an STFC IAA award, and is supported by STFC, grant ST/L000458/1 and ST/P002447/1. MW is supported by STFC, grant ST/R504737/1. TG is supported by The Scottish Funding Council, grant H14027. LDD is supported by an STFC Consolidated Grant, ST/P0000630/1 and a Royal Society Wolfson Research Merit Award, WM140078.

Appendix A: The training procedure in more detail: $L^2 = 8 \times 8$

Here we report further details regarding the training of the RBM on the 8×8 Ising model at $T = 1.8$. The specific parameters used for the training are presented in Table III. As mentioned before, the CD parameter k was increased during the training, while the learning rate α was decreased, splitting the training in three phases. The reason for the former prescription, is exemplified in Fig. 13, where the loss function and the log-likelihood between epoch 3000 and 4000 are plotted for different values of k. While the loss function is left unchanged with varying k, the log-likelihood displays a clear difference in shape, *i.e.* it increases as k is increased, while keeping α fixed at 0.01. If, instead, we choose to keep k fixed at $k = 1$, and decrease α from 0.01 to 0.001, we see no improvement in the log-likelihood as observed on the left hand side of Fig. 13. Therefore, it is the increase in the value of k that leads to the increase in

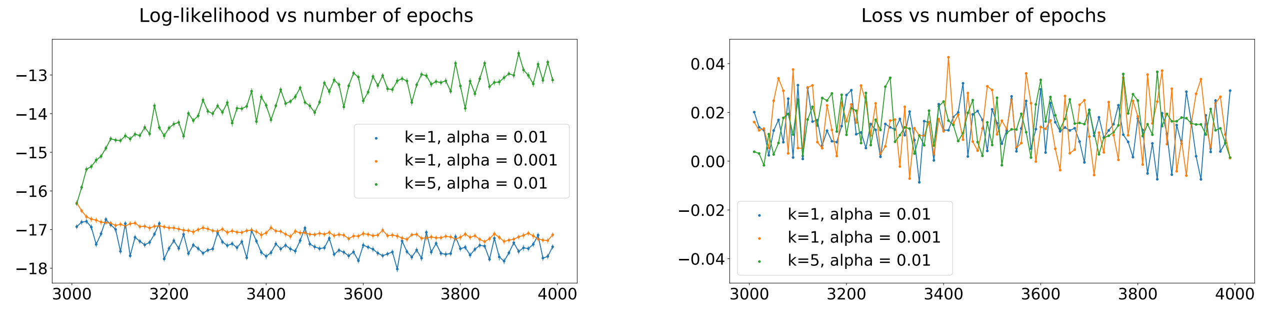

FIG. 13: From left to right, log-likelihood and loss function between 3000 and 4000 epochs for three different values of k. While the former shows different behaviors, keeping an increasing trend just for the highest k value, the latter doesn't change at all, always remaining near zero.

the log-likelihood, visible in the curves corresponding to $T = 1.8$ of Fig. 3.

The reason for reducing the learning rate along the training is visible in Fig. 14, where we have taken observables for $T = 1.8$ from Fig. 4, magnified and normalized by their expected value from Magneto. We therefore observe that decreasing the learning rate implies a reduction of the fluctuations between successive epochs, resulting in a fine tuning of our predictions. This is more evident starting from epoch 3000, where the learning rate is decreased by factor 10. As shown in the plots, at the end of the training, the predictions from the RBMs are compatible, within statistics, with the expected value from Magneto. Also, when we approach the end of the training, all the machines shown in the plots are statistically equivalent to each other, since their predictions are all compatible within 2 sigma.

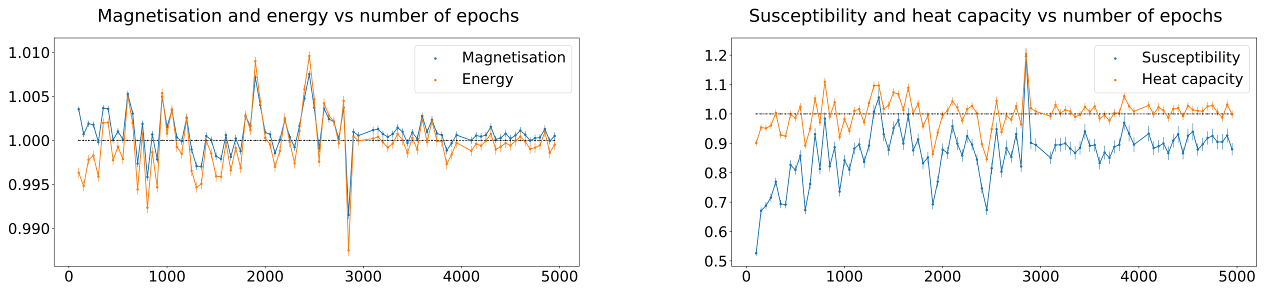

FIG. 14: Observables for $T = 1.8$ normalized by their expected values as a functions of the training epoch. Magnetization and energy are shown on the left, susceptibility and heat capacity are on the right.

Appendix B: Training on a larger system: $L^2 = 16 \times 16$

Here we report the results for a training the RBM on 16×16 Ising configurations, highlighting the main difficulties arising for larger systems. As done in the case of the 8×8 systems, we first tune the value of k in order to have an increasing log-likelihood. It turns out that the log-likelihood behaviour is sensitive to the value of k and α much more than in the case of smaller 8×8 systems. This is shown in Fig. 15, where the log-likelihood as a function of the training epoch is plotted for different values of k and α. On the left hand side we see that using $\alpha = 0.1$, the log-likelihood is not an increasing function of the epochs, even when the value of k is increased. However, decreasing the value of α to 0.01 and using $k = 5$ we manage to obtain an increasing behaviour which becomes even more evident when using $k = 10$, as shown on the right hand side. The final setting used to train this machine are reported in Table III.

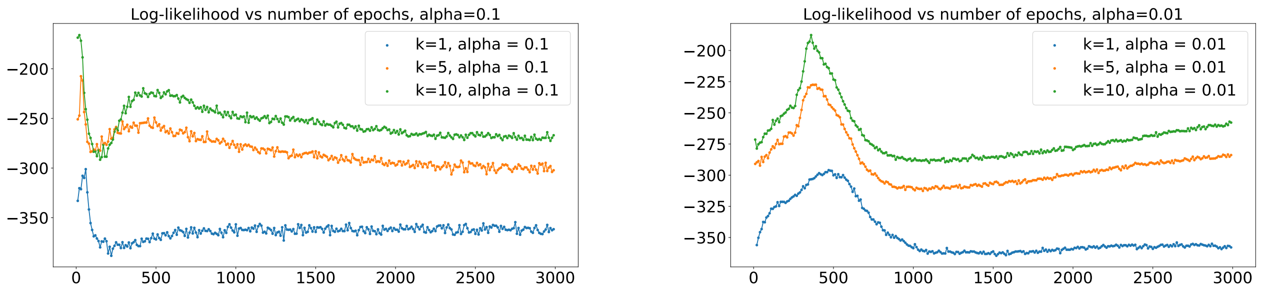

FIG. 15: Log-likelihood for different values of k and α.

The estimators for the training are presented in Fig. 16 and the observables versus epochs plots in Fig. 17: all the estimators present the correct behaviour. The initial peak visible in the log-likelihood at about 500 epochs, should not be considered as the point where the machine has been best trained, but rather as an initial fluctuation of the log-likelihood that later on dies out. This observation is also confirmed by the loss function and the reconstruction error, which also present a peak around the same number of epochs, implying that the training has not converged yet at this point. The final increase in the log-likelihood at 8000 epochs, is where we started using $k = 20$ instead of $k = 10$, as reported in Table III.

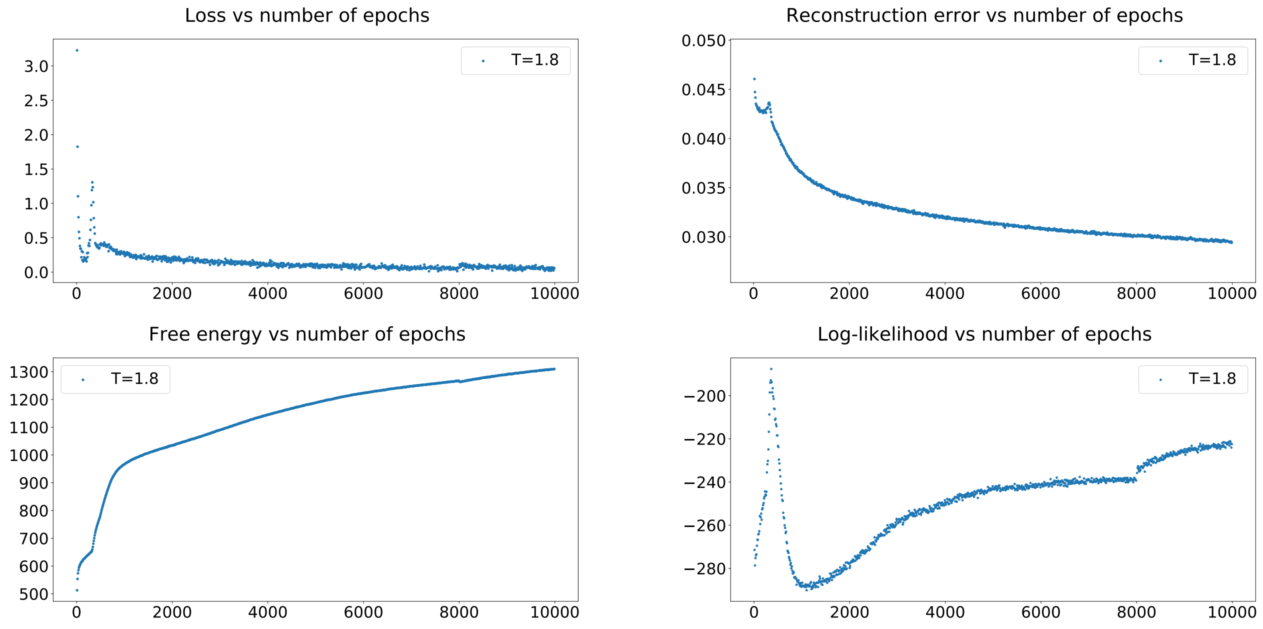

FIG. 16: Here we observed the increase in the log-likelihood behaviour for our chosen values of k and α, as given in Table III. Both the loss function and reconstruction error decrease as the training progresses.

From the plot of observables in Fig. 17, it can be seen that magnetization and heat capacity have almost converged to the expected value, and would only require a fine tuning by reducing the value of α. The energy and susceptibility are further from the expected values, therefore, longer training is required to obtain more accurate values for these two quantities. Having said that, it appears that all the moments are slowly approaching the expected values, and the fact that the log-likelihood is still increasing with the number of epochs suggests that our machine is indeed learning. We can conclude that for the case of the larger 16×16 system the convergence is much slower than the previous smaller 8×8 systems. We can convince ourselves that the machine is actually learning looking at the 2-point

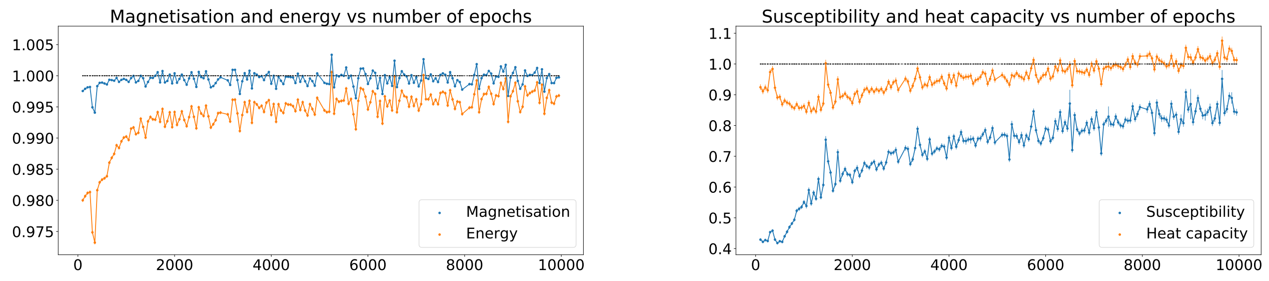

FIG. 17: Observables vs epochs for $L^2 = 16 \times 16$, $h^2 = 16 \times 16$ and batch size 200. The value of each observable, computed from the RBM, is normalised by its expected value, computed from the training set. Magnetisation (blue) and energy (orange) are plotted on the right hand side, susceptibility (blue) and head capacity (orange) are on the left.

interaction matrix for the machine at epoch 10000, plotted in Fig.(18). Clearly the expected path for the interactions can already be seen in the matrix. The coupling extracted from this matrix is $1/2T \sim 0.258 \pm 0.017$, as compared to the expected value of 0.277. This confirms that the machine agrees with the correct theoretical result within 2 sigma. A more accurate result can be obtained if the machine is trained for a longer period of time, as the log-likelihood is still increasing.

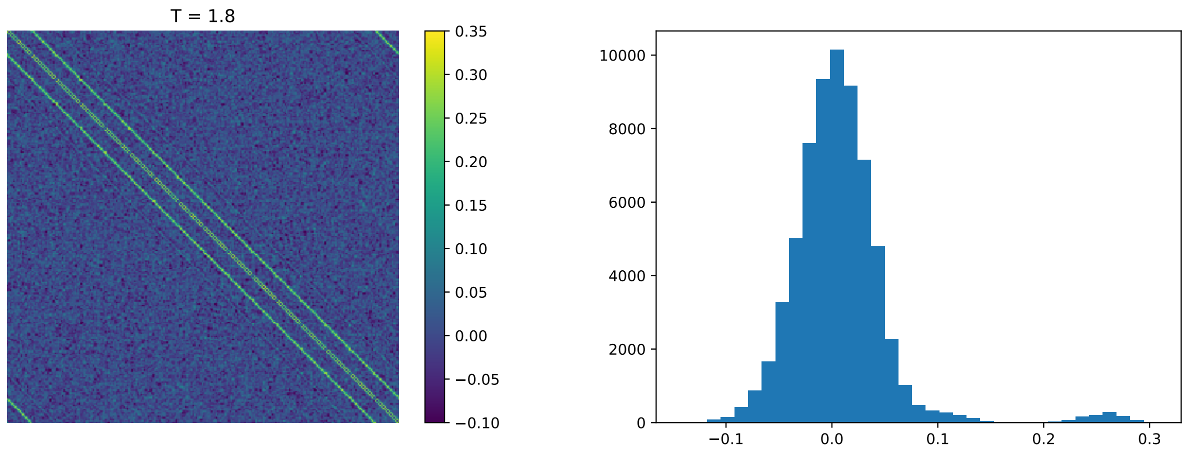

FIG. 18: The 2-point interaction matrix, H_{j_1,j_2} (left) and its corresponding histogram (right) for the machine with $L^2 = 16 \times 16$, $h^2 = 16 \times 16$ and batch size 200. Again, we observed the larger peak centred around zero, corresponding to non nearest neighbour interactions, while there is a second peak representing the coupling with the nearest neighbour spins.

Appendix C: Changing the batch size

We also examined the effect of changing the batch size for the $L^2 = 16 \times 16$ system, keeping the other hyperparameters the same as the successfully trained machine above, *i.e.* , $\alpha = 0.01$ and $k = 10$. As it can be observed from

Fig. 19, the log-likelihood has a slower increase with batch size 500, as compared to batch size 200.

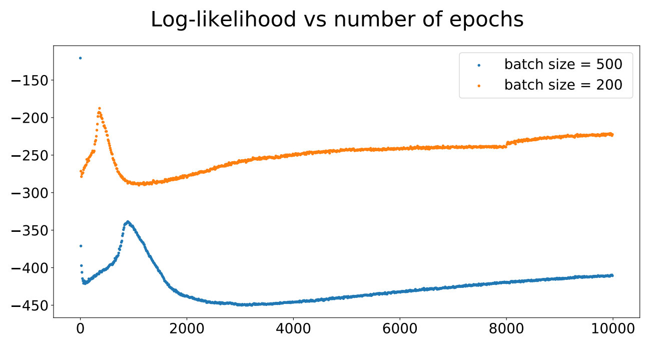

FIG. 19: The dependence of log likelihood on batch size. The curves correspond to $L^2 = 16 \times 16$ lattice with batch size 500 (blue) and batch size 200 (yellow). The choice of a smaller batch size, results in a steeper rise to the log-likelihood.

The slower training in this case can also be observed by plotting the trajectories of the observables per epoch, in Fig. 20, as compared to Fig. 17. This is also consistent with the two-point interaction matrix presented in Fig. 21,

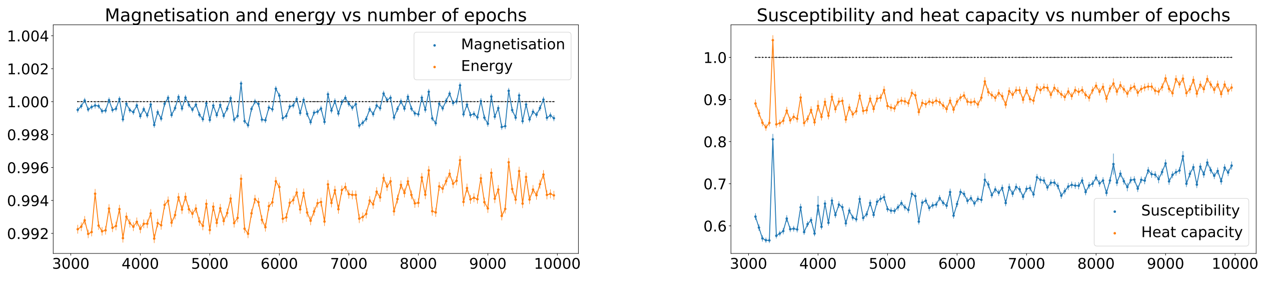

FIG. 20: Observables, normalised by their expected values, vs epochs for $L^2 = 16 \times 16$, $h^2 = 16 \times 16$ and batch size 500. We observe that it takes the observables longer to converge to the correct values, as compared to the case where a smaller batch size is used, *e.g.* , see Fig. 17.

which is more noisy in this case as compared to Fig. 18.

Appendix D: Changing the number of hidden nodes

We also attempted to train an RBM, for the $L^2 = 16 \times 16$ Ising system, where the number of hidden nodes were less than the visible nodes. More explicitly, we chose $h^2 = 12 \times 12$. The log-likelihood is plotted in Fig. 22. The first 4500 epochs were trained using $\alpha = 0.01$ and $k = 10$, as these settings were successful in training the $h^2 = 16 \times 16$ case in Sec. B. However, from the experience gained in Sec. C, the batch size was reduced to 50 in order to give an faster increasing slope to the log-likelihood.

The observables, normalised by their expected values, are plotted in Fig. 23. We can see that the values are furthest from one as compared to the other cases we have trained so far. This is consistent with the measurement of the coupling matrix at a representative epoch towards the end of this training, as presented in Fig. 24. The nearest neighbour structure is present, however, further correlations between the spins can also be observed, which are not expected. A third peak in the histogram confirms the latter observation more clearly in Fig. 24.

According to Fig. 22, the log-likelihood is still increasing, however, the run has become computationally expensive at this stage with $\alpha = 0.0001$ and $k = 40$. Recall that in Fig. 8, it is observed that there are correlations which are further than nearest neighbours predicted by the RBM, before the learning is complete. Therefore we could expect

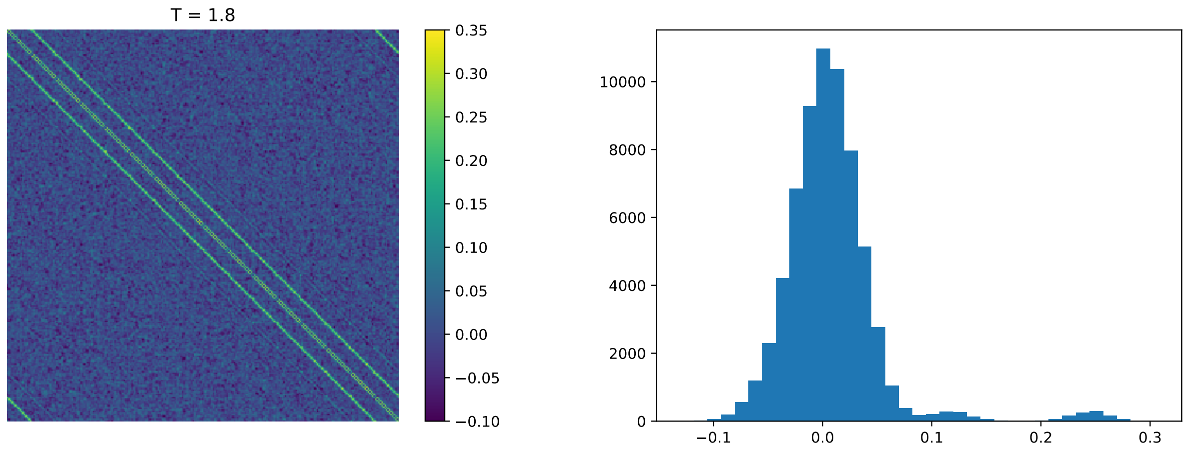

FIG. 21: The 2-point interaction matrix, H_{j_1,j_2} (left) and its corresponding histogram (right) for the machine with $L^2 = 16 \times 16$, $h^2 = 16 \times 16$ and batch size 500. There is a large peak centred around zero, corresponding to non nearest neighbour interactions, however, a second smaller peak can also be observed next to it. As already discussed, the machine with a larger batch size, *i.e.* 500, has to be trained for longer epochs as compared to the machine with batch size 200, in order to learn that non nearest neighbour interactions are zero. Finally, the distinct peak on the right hand side of the plot represents the expected coupling with the nearest neighbour spins, compare with Fig.18

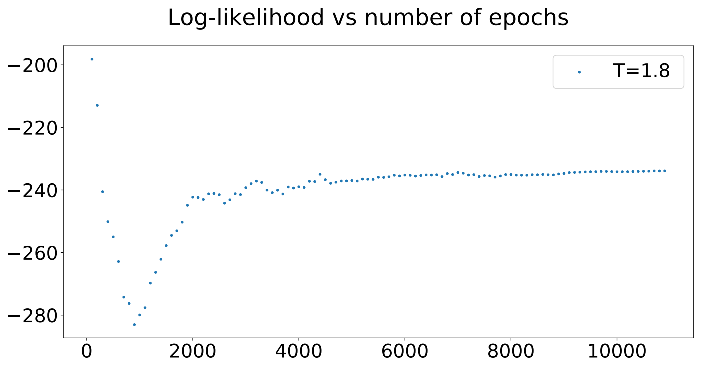

FIG. 22: Log-likelihood for an RBM with less hidden nodes than visible nodes, $L^2 = 16 \times 16$ and $h^2 = 12 \times 12$. The first 4500 epochs were trained using $\alpha = 0.01$, $k = 10$ and batch size 50. According the prescription, we then reduced the value of α and increased k, *i.e.* , From 4500 to 8000, we set $\alpha = 0.001$ and $k = 20$. From 8000 to 8700 epochs α and k were kept fixed at their previous value, while the batch size was increase to 200, in order to reduce the fluctuations in the estimate of the log-likelihood. In the last steps we chose $\alpha = 0.0001$ and $k = 30$, and $\alpha = 0.00001$ and $k = 40$.

that with more training, the RBM may eventually learn the correct nearest neighbour structure, setting the rest of the correlations to zero. Also notice that a $L^2 = 16 \times 16$ with hidden nodes of the size $h^2 = 12 \times 12$ has $\mathcal{O}(3 \times 10^5)$ less parameters than the case with $h^2 = 16 \times 16$. Hence, the difficulty in training the machine is not perhaps too surprising. Having said that, further works need to be done to gain a better understanding of the effect of reducing hidden nodes on training an RBM.

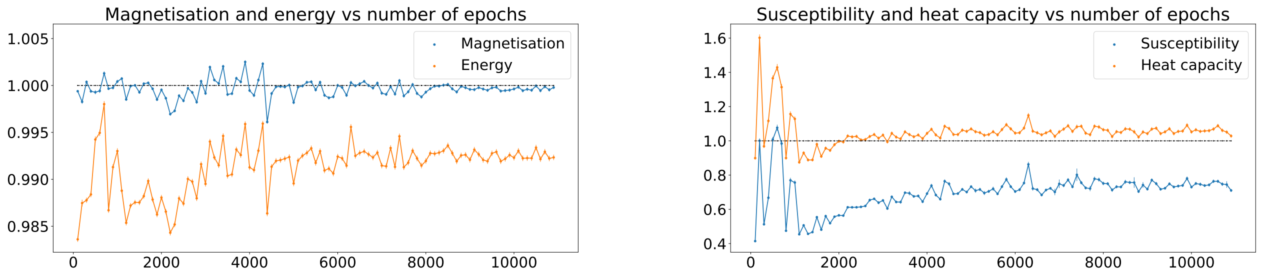

FIG. 23: Observables vs epochs for $L^2 = 16 \times 16$, $h^2 = 12 \times 12$. It can be observed that the machine has to run for more epochs for it to learn the observables and hence the correct structure.

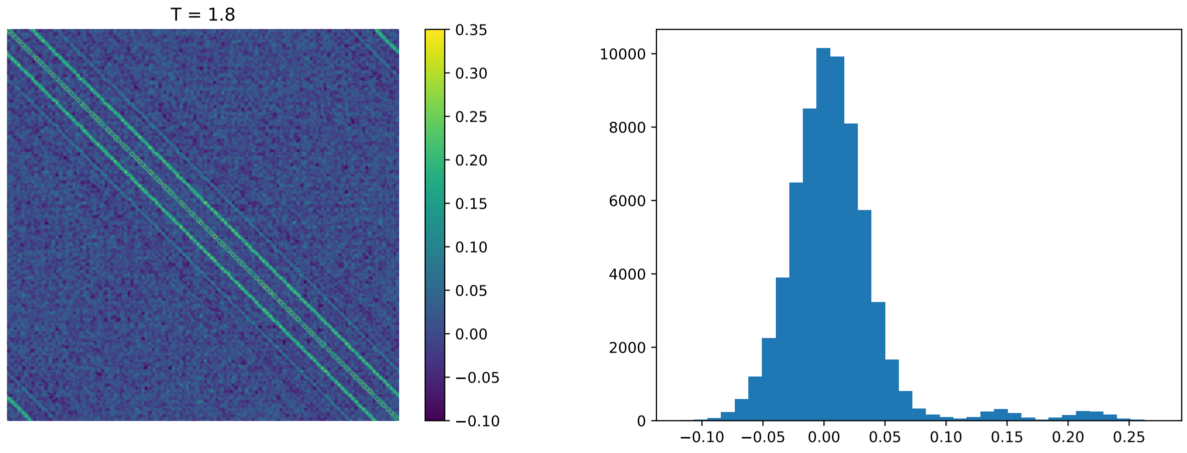

FIG. 24: The two-point interaction matrix for the $L^2 = 16 \times 16$ system (left) and the corresponding histogram (right) with $h^2 = 12 \times 12$. The first peak is centred around zero, corresponding the the non nearest neighbour interactions. The second peak around 0.15 indicates other non nearest correlations that the machine has to learned to set to zero and are expected to vanish as it trains further. The final peak on the right hand side, corresponds to the correct nearest neighbour coupling.

Appendix E: 3- and 4-point interaction histograms

We present the histograms of the 3- and 4-point couplings between the spins, in Fig. 25 and 26 respectively. Generally, the single peak expected behaviour is observed. However for some of the higher temperatures, a second mode seems to appear in the histogram of the three-point interaction. More work needs to be done to understand why these machines appear to be learning a higher order interaction than we would not expect.

Appendix F: Metropolis history plots

The absolute magnetisation and energy histograms of Ising configurations, over which the final measurements for the observables are made, are presented for two different temperatures in Fig. 27 and Fig. 28. The histograms compare well to their corresponding normal distribution, whose parameters are measured by measuring the mean and the standard deviation from the data producing the histogram. Running the Metropolis chain longer than 1×10^6 steps did not change the from of the distributions.

Note that in order for the measurements to be independent of each other, we have binned every 50 measurements of $|m|$ and E on each configuration. To decide on the bin size, we plotted the error on the measurement of magnetisation

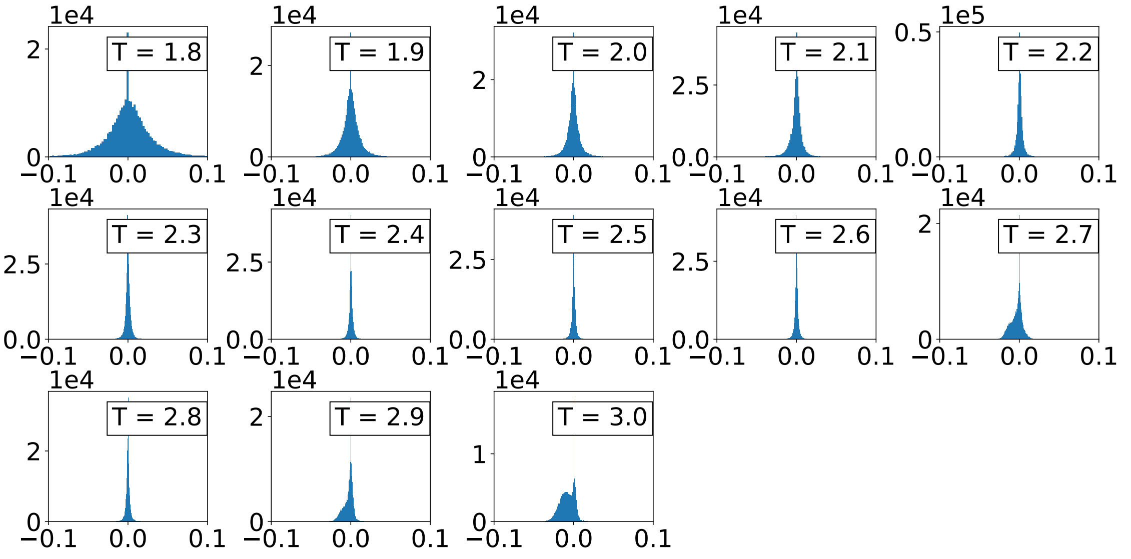

FIG. 25: The histograms of the entries of the 3-point interaction tensor extracted from RBMs trained at a temperature indicated above each subplot.

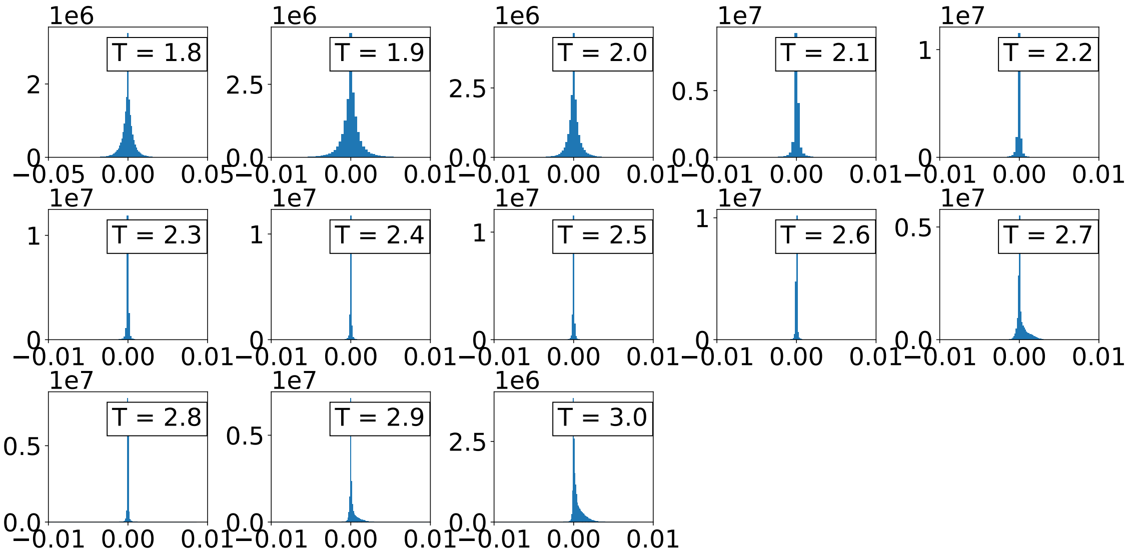

FIG. 26: The histograms of the entries of the 4-point interaction tensor extracted from RBMs trained at a temperature indicated above each subplot.

vs choice of bin size. This is shown in Fig. 29, while the autocorrelation time, defined in Eq, 43, is plotted in Fig. 30

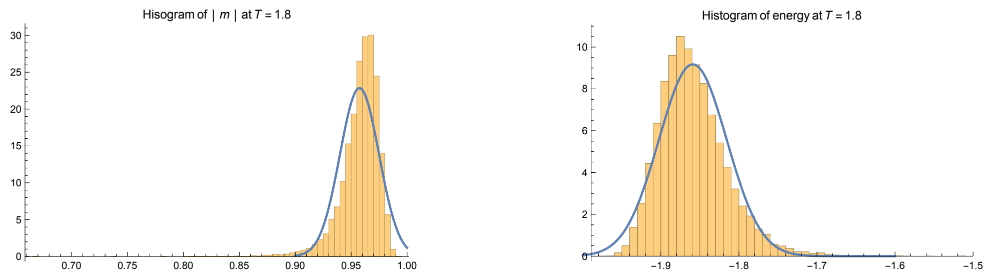

FIG. 27: Histogram of $|m|$ (left) and energy (right) for the Metropolis algorithm at $T = 1.8$. The blue line represent the normal distribution with values of its mean and standard deviation obtained from the data producing the histogram.

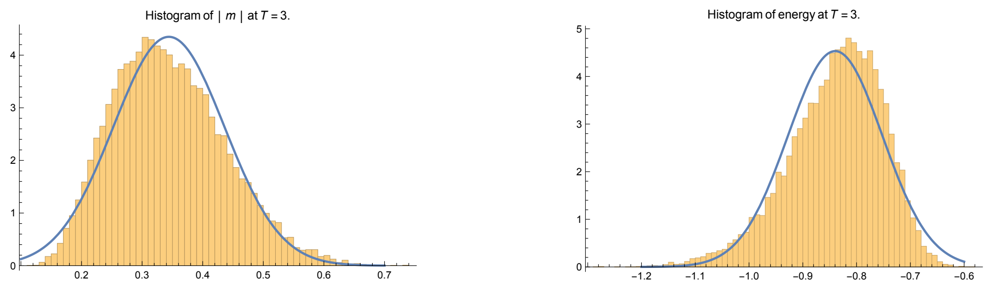

FIG. 28: Histogram of $|m|$ (left) and energy (right) for the Metropolis algorithm at $T = 3.0$. The blue line represent the normal distribution with values of its mean and standard deviation obtained from the data producing the histogram. Plot are generated using Mathematica [21].

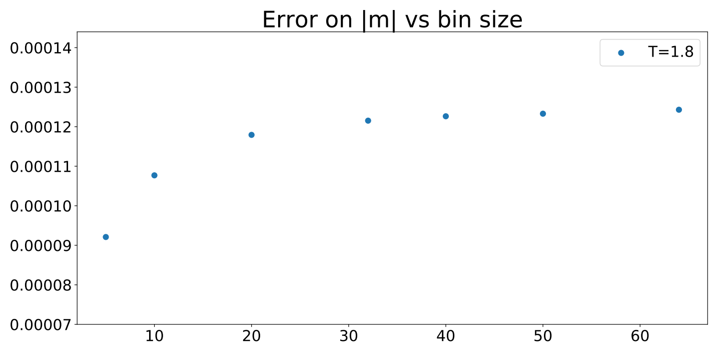

FIG. 29: Error on $|m|$ plotted for choice of bin size. As the measurements become more independent, the correlation between them decrease and hence the error increases. When the measurements are no longer dependent, the error remains constant.

Appendix G: Number of training examples

We also explore the limitation of the RBM, having set aside the issue of small training sets, as large amounts of datasets are not always accessible. To explore this issue explicitly, we have trained three different RBMs with 200, 2000 and 10,000 training examples and computed the coupling matrix in each case to demonstrates the problems that may arise in training the RBM with a small number of training examples. The machines are trained for 2000 epoch for $T = 2.2$. The resulting coupling matrices are presented in Fig. 31. With 200 training examples only, no Ising structure is captured in the output coupling matrix, with only noise being present. Using 2000, or even 10000 training examples, is not sufficient to remove the spin interactions other than the nearest neighbour couplings that

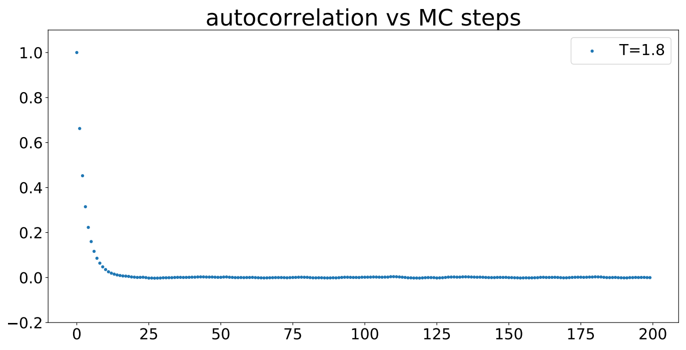

FIG. 30: Autocorrelation time as a function of MC steps.

arise early on during the training. We notice that at 2000 epochs, the machine trained on 10000 training examples is still performing worse than a machine trained for 250 epochs on 10^5 training examples, see Fig. 8. These imply that the RBM seems to require a large amount of training examples to be able to capture the correct values for the couplings.

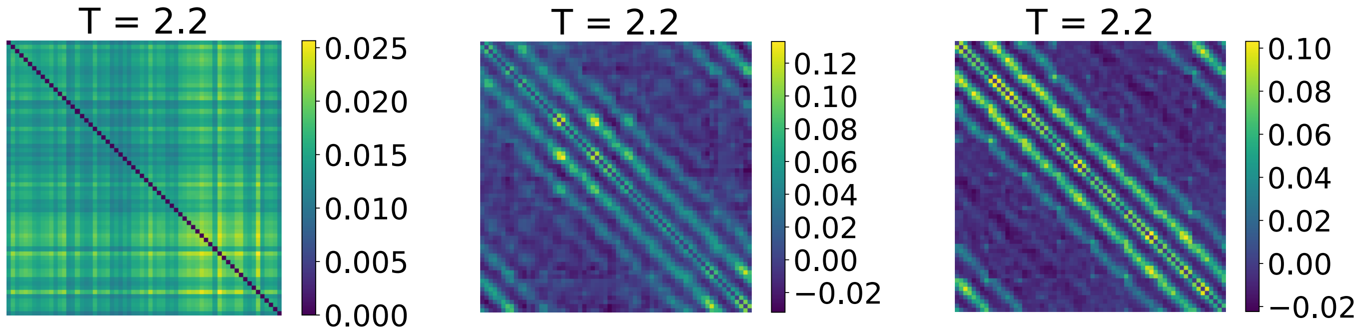

FIG. 31: RBMs trained on 200, 2000 and 10000 training respectively, for $T = 2.2$. All three RBMs are trained for 2000 epochs. With 200 training examples, none of the structure in the data is captured and the output coupling matrix only represents noise. With 2000 epochs some of the Ising structure is captured, however the coupling is still far from the correct value and interactions other than nearest neighbours are present. With 10000 training examples, the issue still persists, however, there is a slight improvement in removing some of the extra interactions.

Appendix H: Overfitting

To address the issue of overfitting, we generate a new set of Ising configurations using Magneto and feed these into the joint probability distribution determined by the RBM at every epoch. We compare the log-likelihood function between the training data and the new test set as a function of training epoch, as presented on the top plot in Fig. 32. We observe that as the machine is trained, the difference between the two values decreases, fluctuating near zero when the machine is trained, as shown on the bottom plot in Fig. 32.

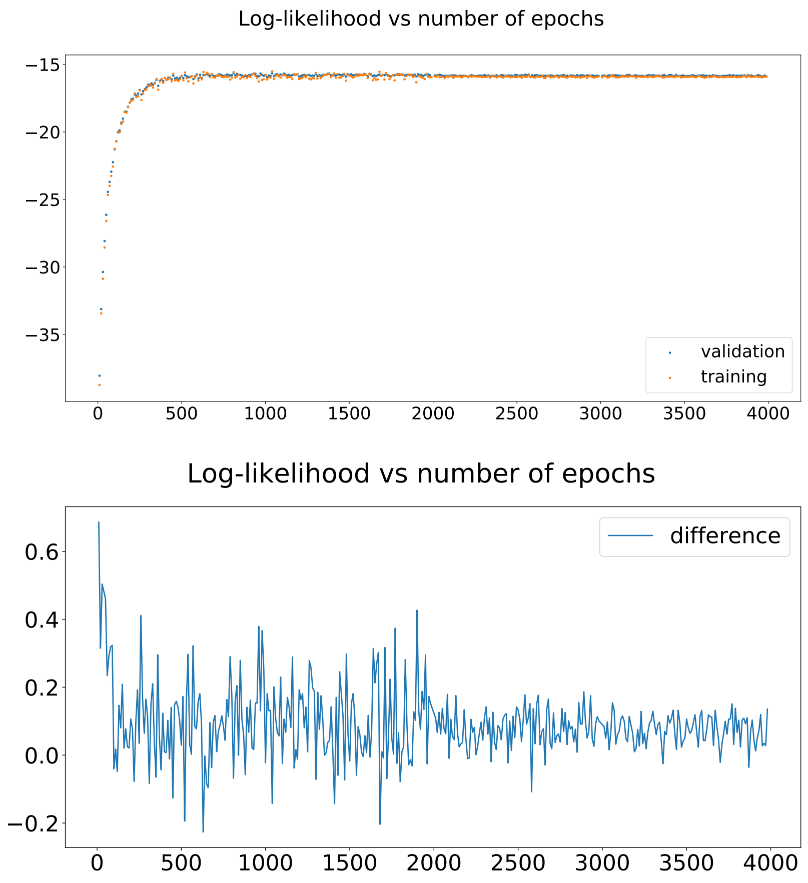

FIG. 32: The top plot shows the log-likelihood obtained at different number of training epochs, for the training (blue) and the validation (yellow) sets. The bottom plot presents the difference between these two values per epoch, *i.e.* $\mathrm{LL}_{\mathrm{validation}} - \mathrm{LL}_{\mathrm{training}}$. As the machine is trained, this difference decrease and towards the end of training reduces to fluctuations near zero. Therefore, the two datasets correspond to the same value of log-likelihood implying that there has been no over-fitting.

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, 2016) `http://www.deeplearningbook.org`.

[2] R. Kindermann and L. Snell, *Markov random fields and their applications*, Vol. 1 (American Mathematical Society, 1980).

[3] S. E. Fahlman, G. E. Hinton, and T. J. Sejnowski, "Massively parallel architectures for ai: NETL, Thistle, and Boltzmann Machines," Proceedings of the Third AAAI Conference on Artificial Intelligence AAAI'83, 109–113 (1983).

[4] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for boltzmann machines," Cognitive Science **9**, 147 – 169 (1985).

[5] G. E. Hinton, T. J. Sejnowski, and D. H. Ackley, "Boltzmann machines: Constraint satisfaction networks that learn," Carnegie-Mellon University, Dept. of Computer Science (1984).

[6] P. Smolensky, "Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1, *information processing in dynamical systems: foundations of harmony theory*," , 194–281 (1986).

[7] P. Mehta and D. J. Schwab, "An exact mapping between the variational renormalization group and deep learning," CoRR **abs/1410.3831** (2014), arXiv:1410.3831.

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

[9] A. Morningstar and R. G. Melko, "Deep learning the ising model near criticality," Journal of Machine Learning Research **18**, 163:1–163:17 (2017).

[10] S. Iso, S. Shiba, and S. Yokoo, "Scale-invariant feature extraction of neural network and renormalization group flow," Phys. Rev. E **97**, 053304 (2018).

[11] P. Mehta, M. Bukov, C.-H. Wang, A. G. R. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, "A high-bias, low-variance introduction to machine learning for physicists," CoRR **abs/1803.08823** (2018).

[12] A. Fischer and C. Igel, "An introduction to restricted boltzmann machines," CIARP (2012).

[13] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," Neural Comput. **14**, 1771–1800 (2002).

[14] G. Hinton, "A practical guide to training restricted boltzmann machines," (2010).

[15] D. J. C. MacKay, *Information Theory, Inference & Learning Algorithms* (Cambridge University Press, New York, NY, USA, 2002).

[16] R. Salakhutdinov, "Learning and evaluating boltzmann machines," Technical Report UTML TR 2008 - 002 (2008).

[17] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing." Science **220 4598**, 671–80 (1983).

[18] C. Jarzynski, "Nonequilibrium equality for free energy differences," Phys. Rev. Lett. **78**, 2690–2693 (1997).

[19] R. M. Neal, "Annealed importance sampling," Statistics and Computing **11**, 125–139 (2001).

[20] "Magneto: 2D Ising model in C++," `https://github.com/s9w/magneto` (2015), accessed: Spring 2018.

[21] W. R. Inc., "Mathematica, Version 11.3," (2014), champaign, IL, 2018.