FAST TRAINING AND SAMPLING OF RESTRICTED BOLTZMANN MACHINES

Anonymous authors
Paper under double-blind review

ABSTRACT

Restricted Boltzmann Machines (RBMs) are powerful tools for modeling complex systems and extracting insights from data, but their training is hindered by the slow mixing of Markov Chain Monte Carlo (MCMC) processes, especially with highly structured datasets. In this study, we build upon recent theoretical advancements in RBM training, focusing on the gradual encoding of data patterns into singular vectors of the coupling matrix, to significantly reduce the computational cost of training (in very clustered datasets) and evaluating and sampling in RBMs in general. The learning process is analogous to thermodynamic continuous phase transitions observed in ferromagnetic models, where new modes in the probability measure emerge in a continuous manner. Such continuous transitions are associated with the critical slowdown effect, which adversely affects the accuracy of gradient estimates, particularly during the initial stages of training with clustered data. To mitigate this issue, we propose a pre-training phase that encodes the principal components into a low-rank RBM through a convex optimization process. This approach facilitates efficient static Monte Carlo sampling and accurate computation of the partition function. Furthermore, we exploit the continuous and smooth nature of the parameter annealing trajectory to achieve reliable and computationally efficient log-likelihood estimations, enabling online assessment during the training process, and proposing a novel sampling strategy termed parallel trajectory tempering that outperforms previously optimized MCMC methods. Our results demonstrate that this innovative training strategy enables RBMs to effectively address highly structured datasets that conventional methods struggle with. Additionally, we provide evidence that our log-likelihood estimation is more accurate than traditional, more computationally intensive approaches in controlled scenarios. Moreover, the parallel trajectory tempering algorithm significantly accelerates MCMC processes compared to existing and conventional methods.

1 INTRODUCTION

Energy-based models (EBMs) are a long-established approach to generative modeling, with the Boltzmann Machine (BM) (Hinton & Sejnowski, 1983) and Restricted Boltzmann Machine (RBM) (Smolensky, 1986; Ackley et al., 1985) being among the earliest examples. They offer a clear framework for capturing complex data interactions and, with simple energy functions, can reveal underlying patterns, making them especially valuable for scientific applications. Pairwise-interacting models (Nguyen et al., 2017) have been widely used for decades for inference applications in fields like neuroscience (Hertz et al., 2011) and computational biology (Cocco et al., 2018). More recent research has shown that RBMs can also be reinterpreted as many-body physical models (Decelle et al., 2024; Bulso & Roudi, 2021), analyzed to identify hierarchical relationships in the data (Decelle et al., 2023), or used to uncover biologically relevant constituent features (Tubiana et al., 2019; Fernandez-de Cossio-Diaz, 2025). In essence, RBMs present a great compromise between expressibility and interpretability: they are universal approximators capable of modeling complex datasets and simple enough to allow a direct extraction of insights from data. This interpretative task is particularly challenging when dealing with more powerful models such as generative convolutional networks EBMs (Xie et al., 2016; Song & Kingma, 2021) and other state-of-the-art models in general. At this point, it is also important to stress that RBM are not only interpretable, but they perform particularly well when dealing with tabular datasets with discrete variables, and where data

is not abundant, which is often the case in genomics/proteomics or neural recording datasets. For instance, RBMs have shown to be among the most reliable approaches to generate synthetic human genomes (Yelmen et al., 2021; 2023) or to model inmunological response (Bravi et al., 2021a;b; 2023) or neural activity (van der Plas et al., 2023; Quiroz Monnens et al., 2024).

RBMs, like other EBMs, are challenging to train due to the difficulty of computing the log-likelihood gradient, which requires ergodic exploration of a complex free energy landscape via Markov Chain Monte Carlo (MCMC). Training with non-convergent MCMC introduces out-of-equilibrium memory effects, as shown in recent studies (Nijkamp et al., 2019; 2020; Decelle et al., 2021), which can be analytically explained using moment-matching arguments (Nijkamp et al., 2019; Agoritsas et al., 2023). While these effects enable fast and accurate generative models for structured data (Carbone et al., 2023) and high-quality images with RBMs (Liao et al., 2022), they create a sharp disconnect between the model's Gibbs-Boltzmann and dataset distributions, undermining parameter interpretability (Agoritsas et al., 2023; Decelle et al., 2024). For meaningful insights, RBMs require proper chain mixing during training, emphasizing the need for *equilibrium models*.

Working with equilibrium RBMs poses three primary challenges: **training, evaluating model quality, and sampling from the trained model**. Both training and sampling rely on MCMC methods to draw equilibrium samples, but their efficiency depends heavily on the dataset's structure. While RBMs can be effectively trained on image datasets like MNIST or CIFAR-10 with sufficient MCMC steps, this approach falters for highly structured datasets (Béreux et al., 2023), such as genomics, neural recordings, or low-temperature physical systems. These datasets often exhibit multimodal distributions with distinct clusters, as revealed by PCA (Fig. 1), which hinder mixing. Clustering is prominent in the first four datasets studied here, in contrast to the more compact PCA pattern seen in CelebA or full MNIST. Dataset details are provided in Fig. 1. Multimodal distributions exacerbate sampling challenges since mixing times depend on the MCMC chains' ability to transition between clusters. In RBMs, these distant modes emerge during training through continuous phase transitions (e.g., second-order or critical transitions) that encode patterns (Decelle et al., 2017; 2018; Bachtis et al., 2024). These transitions lead to *critical slowdown* (Hohenberg & Halperin, 1977), where mixing times diverge with system size as the transition point nears. Critical slowdown along the training trajectory was recently analyzed in (Bachtis et al., 2024). For a brief introduction to phase transitions and associated dynamical effects, see Supplemental Information (SI) G. All this hampers both the training process and the model's ability to generate independent samples, as achieving adequate mixing requires prohibitively long simulations. Model quality evaluation adds to the challenge, particularly in unsupervised learning. A common qualitative approach compares equilibrium samples to data via PCA projections. Quantitatively, moment comparisons between data and generated distributions offer insights but fail to detect mode collapse. Log-likelihood computation provides a single-score metric and tracks overfitting during training, but exact computation is infeasible due to the intractable partition function. Instead, methods like Annealed Importance Sampling (AIS) (Neal, 2001) are used for estimation but become unreliable for highly multimodal distributions and are computationally impractical for online evaluations (see Section 5.1).

In this paper, we build on recent advances in understanding the evolution of the free energy landscape during the training of RBM models to propose three innovative approaches for tackling the challenges of training, evaluating, and generating samples with RBMs trained on highly structured datasets:

1. We propose a novel pre-training strategy based on the theoretical low-rank RBM framework of (Decelle & Furtlehner, 2021a). We adapt it to real-world datasets by extending constrained the number of directions and adding a trainable bias, essential for image datasets. We also correct the static Monte Carlo sampling method to efficiently generate equilibrium samples. For details, see section 4 and SI A.
2. We propose a novel method for estimating log-likelihood (LL) by leveraging the softness of the training trajectory. This approach enables efficient online LL computation during training at minimal cost or offline reconstruction using trajectory annealing or parallel tempering. Moreover, we demonstrate that this method is not only less computationally expensive than existing techniques but also more accurate in controlled experiments where the exact LL is known. The method is introduced in Sect. 5.1.
3. We propose a variation of the standard parallel tempering algorithm, where parameters from different stages of the learning process are exchanged instead of temperatures. This new sampling approach significantly outperforms previous optimized methods and is detailed in Section 5.2.

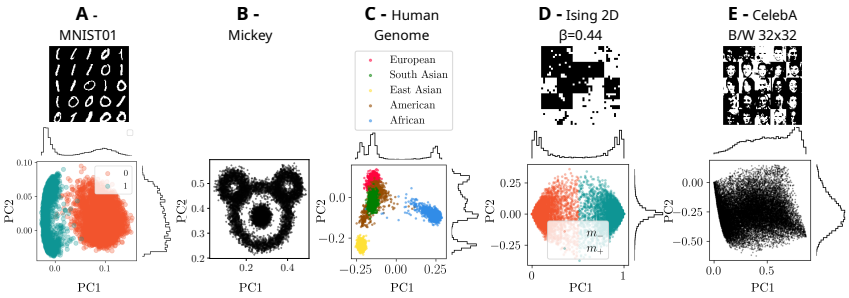

Figure 1: **Datasets.** Panels A-E display 5 distinct datasets projected onto their first two PCA components. In some instances, the dots are color-coded to indicate different labels. In A, the MNIST 01 dataset, featuring images of the digits 0 and 1 from the complete MNIST collection, along with a few sample images. In B, the "Mickey" dataset, an artificial dataset whose PCA representation forms the shape of Mickey Mouse's face. In C, the Human Genome Dataset (HGD), which consists of binary vectors representing mutations or non-mutations for individuals compared to a reference genome across selected genes. In D, the Ising dataset, showcasing equilibrium configurations of the 2D ferromagnetic Ising model at an inverse temperature of $\beta = 0.44$. In E, the CelebA dataset in black and white, resized to 32x32 pixels. For more details on these datasets, please refer to the SI.

We also provide an in-depth discussion in Sect. 2 on the physical reasons behind the failure of previous algorithms when handling clustered datasets.

2 Related work

Training EBMs by maximizing log-likelihood has long been a challenge in the community (LeCun et al., 2006; Song & Kingma, 2021). EBMs gained popularity with the introduction of the contrastive divergence algorithm (Hinton, 2002), in which a set of parallel chains is initialized on independent examples in the minibatch and the MCMC process iterates for a few steps. Despite its widespread use, this algorithm yields models with poor equilibrium properties that are ineffective as generative models (Salakhutdinov & Murray, 2008; Desjardins et al., 2010; Decelle et al., 2021). An improvement is the persistent contrastive divergence (PCD) algorithm (Tieleman, 2008), which maintains a permanent chain in which the last configurations used to estimate the previous gradient update are reused. PCD acts like a slow annealing process improving gradient estimation quality. However, it often fails on clustered data as the statistical properties of the permanent chain quickly move away from the equilibrium measure and degrade the model (Béreux et al., 2023). This problem, which is primarily related to phase coexistence, can be addressed with constrained MCMC methods if appropriate order parameters are identified. For RBMs, these order parameters are related to the singular value decomposition of the model coupling matrix, which enables efficient reconstruction of multimodal distributions (Béreux et al., 2023). Although this method is effective for evaluating model quality, it is too computationally intensive to be used in training, even if it leads to models with good equilibrium properties.

The population annealing algorithm, which adjusts the statistical weights of parallel chains during learning each time the parameters are updated, has been proposed as an alternative approach (Krause et al., 2018). Reweighting chains using non-equilibrium physics concepts like the Jarzynski equation has also been explored (Carbone et al., 2024). However, both methods struggle with highly structured data, as frequent reweighting (which involves duplicating only a few samples) biases the equilibrium measure due to an insufficient number of effective independent chains. To prevent chains from becoming too correlated around transition regions during training, either more sampling steps or a lower learning rate is required, which leads to very long training times to ensure equilibrium. In our case, we tried to used both methods with our datasets but the training becomes too unstable or two slow if learning rate and number of Gibbs steps are auto adjusted. Another approach uses EBMs as corrections for flow-based models that are easier to sample (Nijkamp et al., 2022), simplifying learning but sacrificing the interpretability of the energy function, a key goal of this work. Alternatively, evolving flow models can serve as fast sampling proposers for the EBM (Grenioux et al., 2023), but

this requires training two separate networks and the acceptance of the proposed moves is significantly reduced as the EBM becomes more specialized.

The Parallel Tempering (PT) algorithm (Hukushima & Nemoto, 1996), in which the model is simulated in parallel at different temperatures which are exchanged once in a while using the Metropolis rule, has been proposed to both speed up the sampling and to improve the training process (Salakhutdinov, 2009; Desjardins et al., 2010) showing important improvements in certain datasets (Krause et al., 2020).

However, PT is expensive because it requires sampling at multiple temperatures while using samples from only one model. It's also often ineffective with highly clustered data due to first-order phase transitions in EBMs, where modes disappear abruptly at certain temperatures, as discussed by (Decelle & Furtlehner, 2021a). This leads to sharp drops in acceptance rates for exchanges between temperatures near the transition. For this reason, first-order transitions typically hinder PT, as many intermediate temperatures are needed to allow sufficient acceptance rates in the exchanges. In contrast, continuous transitions, as seen during the training trajectory, cause modes to gradually separate as the model parameters are changed.

A recent method for RBMs, called "Stacked Tempering" (Roussel et al., 2023), significantly accelerates sampling by training smaller RBMs using latent variables from previous models. This allows for fast updates through a PT-like algorithm, and the authors demonstrated that it is much faster than standard PT. However, while this method is effective for sampling a trained model, it is not suitable for learning, as it requires sequentially training multiple models stacked on top of each other. Also for RBMs, it has recently been shown in a theoretical work that it should be possible to train a low-rank RBM that accurately reproduces the statistics of the data projected along the d first data principal directions through a convex and very fast optimization process (see (Decelle & Furtlehner, 2021a) and the discussion below) using a mapping to another model, the so-called Restricted Coulomb Machine. This low-rank model can be seen as a good approximation to the correct RBM needed to describe the data, and has the nice property that it can be efficiently sampled via a static Monte Carlo process. While this work showed the efficiency of this training process to describe simple synthetic datasets embedded in only one or two dimensions, the use of this algorithm to obtain reliable low-rank RBMs with real data requires further developments that will be discussed in this paper, in particular: (i) the training of a bias, which is crucial to gain almost for free an additional dimension and to deal with image datasets, and (ii) a tuning of the RBM fast sampling process, which was not properly investigated in the original paper.

In this paper, we show how starting the RBM training with a low-rank RBM in very clustered datasets can drastically improve the quality of the models by circumventing the initial dynamical arrest effects that occur during the training process, at a very similar computational cost. We also show that the training trajectory can be leveraged to estimate the log-likelihood at a very little computational effort, so that both the training and test set log-likelihood can be easily and reliably computed online during training. Furthermore, we propose an efficient sampling method, which we call Parallel Trajectory Tempering (PTT). This method outperforms optimized MCMC methods such as the traditional Parallel Tempering (PT) or the "Stacked Tempering" (Roussel et al., 2023) only requiring storing the parameters of the model at a few stages during training. By combining these strategies, we are able to train equilibrium models that accurately capture different modes in datasets prone to mode collapse using standard methods, easily evaluate RBMs during the training process, and easily sample from multimodal distribution.

3 THE RESTRICTED BOLTZMANN MACHINE

The RBM is composed by $N_{\rm v}$ visible nodes and $N_{\rm h}$ hidden nodes. In our study, we primarily use binary variables $\{0,1\}$ or ± 1 for both layers. The two layers (visible and hidden) interact via a weight matrix $\boldsymbol{W}$, with no direct couplings within a given layer. Variables are also adjusted by visible and hidden local biases, $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$, respectively. The Gibbs-Boltzmann distribution for this model is expressed as

$$p(\boldsymbol{v},\boldsymbol{h}) = \frac{1}{Z}\exp\left[-\mathcal{H}(\boldsymbol{v},\boldsymbol{h})\right] \text{ where } \mathcal{H}(\boldsymbol{v},\boldsymbol{h}) = -\sum_{ia} v_i W_{ia} h_a - \sum_i \theta_i v_i - \sum_a \eta_a h_a, \quad (1)$$

where Z is the partition function of the system. As with other models containing hidden variables, the training objective is to minimize the distance between the empirical distribution of the data, $p_{\mathcal{D}}(\boldsymbol{v})$, and the model's marginal distribution over the visible variables, $p(\boldsymbol{v}) = \sum_{\boldsymbol{h}} \exp\left[-\mathcal{H}(\boldsymbol{v},\boldsymbol{h})\right]/Z = \exp\left[-H(\boldsymbol{v})\right]/Z$. Minimizing the Kullback-Leibler divergence is equivalent to maximizing the likelihood of observing the training dataset in the model. Thus, the log-likelihood $\mathcal{L} = \langle -H(\boldsymbol{v})\rangle_{\mathcal{D}} - \log Z$ can be maximized using the classical stochastic gradient ascent. For a training dataset $\mathcal{D} = \{\boldsymbol{v}^{(m)}\}_{m=1,\dots,M}$, the log-likelihood gradient is given by

$$\tfrac{\partial \mathcal{L}}{\partial W_{ia}} = \langle v_i h_a\rangle_{\mathcal{D}} - \langle v_i h_a\rangle_{\mathrm{RBM}}, \quad \frac{\partial \mathcal{L}}{\partial \theta_i} = \langle v_i\rangle_{\mathcal{D}} - \langle v_i\rangle_{\mathrm{RBM}}, \quad \frac{\partial \mathcal{L}}{\partial \eta_a} = \langle h_a\rangle_{\mathcal{D}} - \langle h_a\rangle_{\mathrm{RBM}}, \tag{2}$$

where $\langle f(\boldsymbol{v},\boldsymbol{h})\rangle_{\mathcal{D}} = M^{-1}\sum_m \sum_{\{\boldsymbol{h}\}} f(\boldsymbol{v}^{(m)},\boldsymbol{h}) p(\boldsymbol{h}|\boldsymbol{v}^{(m)})$ denotes the average with respect to the entries in the dataset, and $\langle f(\boldsymbol{v},\boldsymbol{h})\rangle_{\mathrm{RBM}}$ with respect to $p(\boldsymbol{v},\boldsymbol{h})$.

Since Z is intractable, the model averages in the gradient are typically estimated using N_{s} independent MCMC processes, and observable averages $\langle o(\boldsymbol{v},\boldsymbol{h})\rangle_{\mathrm{RBM}}$ are replaced by $\sum_{r=1}^{R} o(\boldsymbol{v}^{(r)},\boldsymbol{h}^{(r)})/R$, with $(\boldsymbol{v}^{(r)},\boldsymbol{h}^{(r)})$ being the last configurations reached with each of the $r = 1,\dots,R$ parallel chains. To obtain reliable estimates, it should be ensured that each of the Markov chains mix well before each parameter update. However, ensuring equilibrium at each update is impractical, slow and tedious. The common use of non-convergent MCMC processes is the cause of most difficulties and weird dynamical behaviors encountered in training RBMs as discussed in (Decelle et al., 2021). In order to minimize out-of-equilibrium effects, it is often useful to keep R *permanent (or persistent)* chains, which means that the last configurations reached with the MCMC process used to estimate the gradient at a given parameter update t, $\boldsymbol{P}_t \equiv \{(\boldsymbol{v}_t^{(r)},\boldsymbol{h}_t^{(r)})\}_{r=1}^{R}$, are used to initialize the chains of the subsequent update $t+1$. This algorithm is typically referred to as PCD. In this scheme, the process of training can be mimicked as a slow cooling process, only that instead of varying a single parameter, e.g. the temperature, a whole set of parameters $\boldsymbol{\Theta}_t = (\boldsymbol{w}_t,\boldsymbol{\theta}_t,\boldsymbol{\eta}_t)$ is updated at every step to $\boldsymbol{\Theta}_{t+1} = \boldsymbol{\Theta}_t + \gamma\boldsymbol{\nabla}\mathcal{L}_t$ with $\boldsymbol{\nabla}\mathcal{L}_t$ being the gradient in equation 2 estimated using the configurations in $\boldsymbol{P}_t$, and γ being the learning rate.

Typical MCMC mixing times in RBMs are initially short but increase as training progresses (Decelle et al., 2021). Studies reveal sharp mixing time peaks when the model encodes new modes (Bachtis et al., 2024), a phenomenon known as *critical slowing down* Hohenberg & Halperin (1977), associated with critical phase transitions. In RBMs, these transitions during maximum likelihood training correspond to encoding the dataset's principal components into the singular vectors of the weight matrix $\boldsymbol{W}$ (Decelle et al., 2017; 2018; Decelle & Furtlehner, 2021b; Bachtis et al., 2024). Since these directions can be identified via PCA before training, pre-training the RBM to encode them avoids early phase transitions and gradient degradation associated to those transitions.

4 THE LOW-RANK RBM PRETRAINED

In (Decelle & Furtlehner, 2021a), it was shown that it is possible to train exactly an RBM containing a reduced number of modes in the weight matrix $\boldsymbol{W}$ by exploiting a mapping between the RBM and a Restricted Coulomb Machine and solving a convex optimization problem, see the SI A.2. In other words, it is possible to train an RBM whose coupling matrix has this simplified form

$$\boldsymbol{W} = \textstyle\sum_{\alpha=1}^{d} w_\alpha \bar{\boldsymbol{u}}_\alpha \boldsymbol{u}_\alpha^\top, \qquad \text{with} \qquad (\boldsymbol{u}_\alpha,\bar{\boldsymbol{u}}_\alpha) \in \mathbb{R}^{N_{\mathrm{v}}} \times \mathbb{R}^{N_{\mathrm{h}}}, \tag{3}$$

where $\{w_\alpha\}$ are the singular values of the coupling matrix and the right singular vectors $\{\boldsymbol{u}_\alpha\}_{\alpha=1}^{d}$ correspond exactly to the first d principal directions of the data set. Under this assumption, $p(\boldsymbol{v})$ a function of only d order parameters, given by the *magnetizations* along each of the $\boldsymbol{u}_\alpha$ components, i.e. $m_\alpha(\boldsymbol{v}) = \boldsymbol{u}_\alpha \cdot \boldsymbol{v}/\sqrt{N_{\mathrm{v}}}$. In particular,

$$H(\boldsymbol{v}) = -\textstyle\sum_{\alpha=0}^{d} \theta_\alpha m_\alpha - \sum_a \log\cosh\left(\sqrt{N_{\mathrm{v}}}\sum_{\alpha=1}^{d} \bar{u}_{\alpha a} w_\alpha m_\alpha + \eta_a\right) = \mathcal{H}(\boldsymbol{m}(\boldsymbol{v})), \tag{4}$$

where $\boldsymbol{m} = (m_1,\dots,m_\alpha)$ and θ_α is the projection of the visible bias onto the extended $\boldsymbol{u}$ matrix. To initialize the visible bias as proposed in (Montavon & Müller, 2012), a direction $\boldsymbol{u}_0$ is introduced for the magnetization m_0, which only exists in the bias term. This is an improvement over (Decelle & Furtlehner, 2021a) because it effectively adds an additional direction at minimal cost. We also found that the addition of the bias term was crucial to obtain reliable low-rank RBMs for image data.

The optimal parameters can be determined by solving a convex optimization problem, detailed in SI A.2. Once the model is trained, we obtain a probability distribution $p(\boldsymbol{m})$ in a much lower-dimensional space than the original $p(\boldsymbol{v})$, which can be easily sampled globally using the inverse transform sampling method. However, since this approach requires discretizing the $\boldsymbol{m}$-space for both training and generation, it becomes impractical for intrinsic space dimensions $d > 4$ due to memory constraints. These low-rank RBMs are trained to replicate the statistics of the dataset projected onto its first d principal components. Despite their simplicity, the low-rank models can still generate an approximate version of the dataset, as shown in Supplemental Figure 19 for the five datasets previously presented.

In the initial stage of the standard RBM learning process, the model encodes the strongest PCA components of the data through multiple critical transitions (Decelle et al., 2017; 2018; Bachtis et al., 2024). Pre-training with the low-rank construction allows us to bypass these transitions and avoid out-of-equilibrium effects caused by critical slowing down associated to these transitions, which becomes prohibitive in the case of very clustered datasets. Once the main directions are incorporated, mixing times stabilize at much lower values than those observed during transitions, as barriers between clusters become finite once the modes are properly aligned, as shown in Béreux et al. (2023). This makes it a much safer starting point for initializing standard PCD training. However, mixing times tend to increase as training progresses, meaning that pretraining can only mitigate issues caused by poor initialization of the primary directions. Challenges related to insufficient MCMC steps are likely to persist later in the training process. Overcoming these transitions greatly improves model quality and the accuracy with which they reproduce data statistics, particularly for short training times. In Fig. 2, we compare equilibrium samples from two RBMs trained on five datasets using identical settings: the same number of samples, minibatch size, $k = 100$ Gibbs steps, and learning rate $\gamma = 0.01$, but different strategies. One RBM is trained from scratch with standard PCD (Tieleman, 2008), while the other uses PCD following a low-rank RBM pre-training phase. We also tested the Jarzynski reweighing method (Carbone et al., 2023) (see SI E for details), but results were unstable on most datasets and are excluded here. Across all datasets, pre-training significantly improves sample quality and test log-likelihood. The pre-trained+PCD RBM balances the modes effectively, as shown by PCA projections (extending to other principal directions), and consistently achieves higher log-likelihood values measured using annealed importance sampling (Tr-AIS). Samples are drawn with the trajectory PT algorithm, both described in Section 5.1. Scatter plots highlight that pre-training is critical to obtain properly balanced models on clustered datasets and improves test log-likelihood. However, for less clustered datasets like CelebA and full MNIST, pre-training offers limited benefits, as splitting distributions into isolated modes is unnecessary for such data (see Fig. 20 in SI J).

5 Trajectory Annealing Paradigm

One major challenge with structured datasets is quantifying the model's quality, since sampling the equilibrium measure of a well-trained model is often too time-consuming. This affects the reliability of generated samples and indirect measures as log-likelihood's estimation through Annealing Importance Sampling (AIS) (Krause et al., 2020), making them inaccurate and meaningless. The evolution of parameters during training can be seen as a smoother annealing process compared to the standard temperature annealing used in algorithms like annealing importance sampling (AIS) for log-likelihood estimation or in sampling algorithms like simulated annealing or parallel tempering. This is because it avoids the first-order transitions that typically occur with temperature in clustered datasets, see SI H for more explanations about this phenomenon. However, these methods can still be applied to trajectory annealing by using the parameters from different stages of training, rather than scaling the model parameters by a common factor β_i, as is done in temperature-based protocols.

5.1 Trajectory Annealing Importance Sampling

The standard method for computing the log-likelihood (LL) in the literature is Annealed Importance Sampling (AIS) (Neal, 2001). For a detailed review of its application in RBMs, see (Krause et al., 2020). In this approach, the partition function of a model with parameters $\boldsymbol{\Theta} = \{\boldsymbol{w}, \boldsymbol{\theta}, \boldsymbol{\eta}\}$ is estimated by gradually increasing the inverse temperature β from 0 to 1 over N_β steps. At $\beta = 0$ (where $p_0 = p_{\text{ref}}$), the partition function Z_0 is tractable, enabling the estimation of Z_i at neighboring β_i

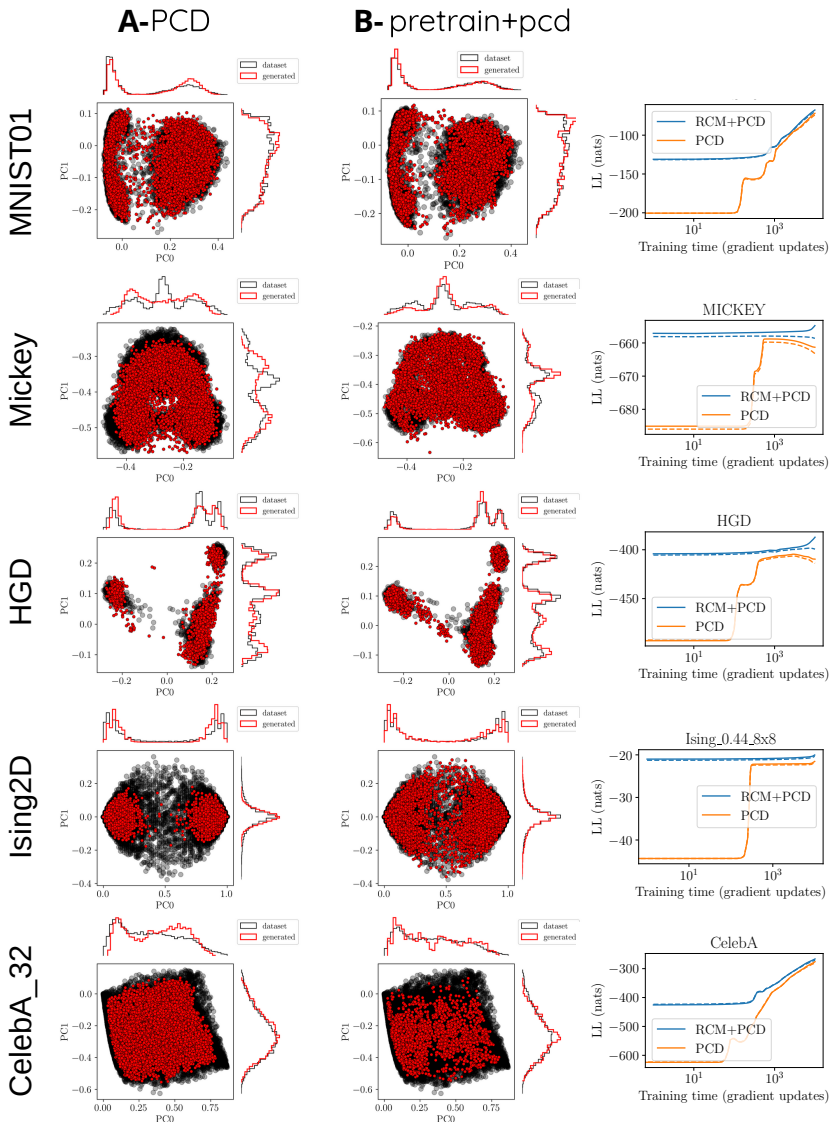

Figure 2: We compare the equilibrium samples generated by RBMs trained on the MNIST01, Mickey, HGD, Ising2D and CelebA datasets using two different training schemes: PCD, and PCD initialized on low-rank RBMs. To assess the fitting of the modes, we show a scatter plot of the projections of the samples in the first two principal directions of each dataset, together with the histograms along each direction. Black dots and histograms refer to the original data distribution, while the red color is used to represent the generated data. In D we show the evolution of the log-likelihood in time for the two training methods computed with the online AIS trajectory. In continuous lines we show the train and in dashed, the test. Results are obtained by averaging with respect to 10 trainings initialized at the same low-rank RBM, and the standard deviation between all is shown in a shade around the line, which is hardly seen due to the small variation from run to run.

values through path sampling methods. The reference configuration p_{ref} is commonly taken uniform, but improvements have been reported in RBM when fixing it to the independent site distribution that describes the empirical center of the dataset. For a detailed description of the algorithm, see SI D.

AIS is quite costly because it requires a large number of annealing steps N_β for reliable estimations, making it impractical to compute the log-likelihood online during training. As a result, it is often

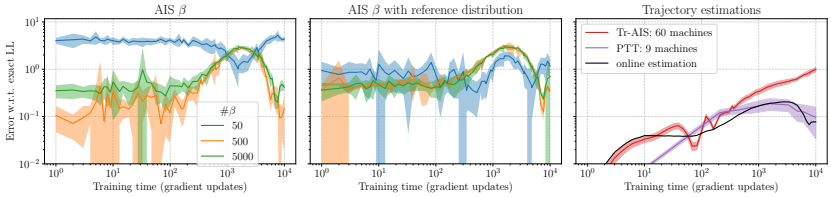

Figure 3: Comparison of the error in (LL) estimation (with respect to the exact value) using different methods: AIS (left), AIS with a reference distribution (middle), and Tr-AIS (ours, right). For Tr-AIS, we present three approaches to computing it: online during training, offline using the models saved during training, and with PTT. The RBM was pretrained and trained with a total of 20 hidden nodes (allowing for an exact computation of the LL) for 10,000 gradient steps. PTT uses a subset of saved models such that the acceptance rate between two consecutive models is 0.25.

calculated offline using a set of models at different temperatures. However, since the training process itself follows an annealing trajectory, it makes more sense to use the evolution of parameters during training as annealing steps rather than relying on temperature. Additionally, at the start of training, parameters are set to zero, making Z_0 tractable, which also holds when initializing with a low-rank RBM. Since updating the partition function only involves calculating the energy difference between the new and old parameters, this can be efficiently computed online during each parameter update, allowing for very small integration steps. We refer to this method as *online* Tr-AIS. Alternatively, similar to standard AIS, multiple models can be saved during training for an *offline* Tr-AIS estimation.

To evaluate the reliability of our new approach, we trained several RBMs using datasets with a maximum of 20 hidden nodes, allowing for the exact computation of the partition function through direct enumeration. We then calculated the difference between the true partition function and the estimations obtained from various methods. The results for the HGD dataset are presented in Fig. 3, while similar qualitative results for the other datasets can be found in the SI, see Figs. 15, 16, 17 and 18. In Fig. 3, we display the estimation errors in the LL using three different methods: standard temperature Annealed Importance Sampling (AIS) (panel A), AIS with a reference distribution (panel B), and Tr-AIS (panel C). For the temperature AIS estimates, we show results for different values of N_β in distinct colors. In panel C, we compare the online estimation in black with the offline Tr-AIS estimation, using the same models analyzed in panels A and B. Additionally, we present the error in LL estimation using a few machines and the trajectory parallel tempering (PTT), which will be detailed in the next section. Overall, the online Tr-AIS estimation consistently outperforms the other methods, and only the PTT estimation yields similar results in offline analyses.

5.2 Exploiting the training trajectory for sampling

We observe that exploiting the trajectory annealing paradigm significantly accelerates the sampling of equilibrium configurations, as shown in Fig. 4. Before presenting our parallel trajectory annealing (PTT) algorithm, we first discuss the limitations of standard methods like Alternative Gibbs Sampling (AGS) and introduce the standard parallel tempering (PT) algorithm.

Problems with the standard Alternate Gibbs sampling method– To illustrate this issue, consider the MNIST01, HGD, and Ising 2D datasets. MNIST01 and Ising 2D exhibit strong bimodality in their PCA representations, while HGD is trimodal, as shown in Fig. 1. Suppose we aim to sample the equilibrium measure of RBMs trained using low-rank RBM pretraining. Typically, this involves running MCMC processes from random initializations and iterating them until convergence. The mixing time in such cases is determined by the jumping time between clusters. Accurate estimation of the relative weights between modes requires the MCMC processes to be ergodic, necessitating frequent back-and-forth jumps. However, as shown in the chains in Figs. 4–B1-3, Alternate Gibbs Sampling (AGS) dynamics are exceedingly slow, rarely producing jumps even after 10^4 MCMC steps. The red curves in Figs. 4–C1-3 illustrate the mean number of jumps over 1000 independent chains as a function of MCMC steps, indicating that achieving proper equilibrium generation would require at least 10^6-10^7 MCMC steps. To accelerate sampling, we explore optimized Monte Carlo methods inspired by Parallel Tempering.

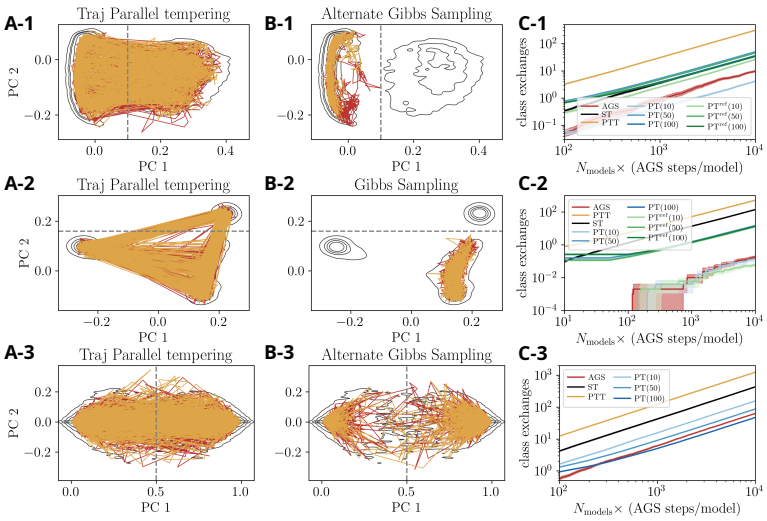

Figure 4: Comparison of the performance of different MCMC sampling methods on RBMs trained with the pretraining+PCD procedure on the MNIST01 (row-1), the HGD (row-2) and Ising 2D datasets (row-3). In A and B columns, we show the trajectory of two independent Markov chains (red and orange) iterated for 10^4 MCMC steps using either PTT or alternate Gibbs sampling (AGS), projected onto the first two principal components of the dataset. The position of the chains is plotted every 10 MCMC steps. The black contour represents the density profile of the dataset. In column C, we show the averages number of jumps between the two regions separated by the dashed grew line in the PCA plots using different MCMC methods: Alternative Gibbs Sampling (AGS), Parallel Tempering (PT) (Hukushima & Nemoto, 1996), without or with reference configuration (PTref) (Krause et al., 2020), the Stacked Tempering (Roussel et al., 2023), and the Trajectory Parallel Tempering (PTT) proposed in this work. The average is calculated over a population of 1000 chains and the shadow around the lines indicates the error of the mean.

Trajectory Parallel Tempering– Parallel tempering (PT) is commonly used to simulate multiple systems in parallel at different inverse temperatures β (Hukushima & Nemoto, 1996; Marinari & Parisi, 1992), with occasional temperature swaps between models based on the Metropolis rule. Since this move satisfies detailed balance, the dynamics will converge to the Boltzmann distribution $p_\beta(\boldsymbol{x}) \propto \exp(-\beta\mathcal{H}(\boldsymbol{x}))$ at each β. However, a more general approach involves simulating multiple systems, each with its own set of parameters. We can define a system as $p_{\boldsymbol{\Theta}^t}(\boldsymbol{x})$, where $\boldsymbol{\Theta}^t$ denotes the parameters of the model indexed by t. This includes the classic PT case, where temperature scaling is applied, $\boldsymbol{\Theta}^t_{\text{PT}} = \beta(t)\boldsymbol{\Theta}$, with $\boldsymbol{\Theta}$ being the parameters of the model to be sampled. The temperature ladder is typically defined as $\beta(t) = t/N_T$, where $t = 0, \dots, N_T - 1$, and N_T represents the number of systems simulated in parallel. Alternatively, recent work (Roussel et al., 2023) introduced a variant of this algorithm called Stacked Tempering. In this approach, a serie of nested RBMs are used, where the hidden configurations of one machine are exchanged with the visible configurations of the next in the stack, analogous to the mechanism in deep belief networks.

In this work, we propose a novel variant in which the model parameters stored in different phases of training are exchanged. We use the learning trajectory to define the parameter set $\boldsymbol{\Theta}^t$, with t a specific training step. We call this method **Parallel Trajectory Tempering (PTT)**. The rule for exchanging configurations $\boldsymbol{x} = (\boldsymbol{v}, \boldsymbol{h})$ between neighboring machines indexed by t and $t-1$ is defined as:

$$p_{\text{acc}}(\boldsymbol{x}^t \leftrightarrow \boldsymbol{x}^{t-1}) = \min\left(1, \exp\left(\Delta\mathcal{H}^t(\boldsymbol{x}^t) - \Delta\mathcal{H}^t(\boldsymbol{x}^{t-1})\right)\right), \tag{5}$$

where again the index t indicates that the Hamiltonian is evaluated using the parameters $\boldsymbol{\Theta}^t$. This move satisfies detailed balance with our target equilibrium distribution $p(\boldsymbol{x}) = \exp(-\mathcal{H}(\boldsymbol{x}))/Z$, ensuring that the moves lead to the same equilibrium measure. The reason for using the trajectory is that PT-like algorithms work very well when a system undergoes a continuous phase transition. As discussed in the appendix G, adjusting the inverse temperature can lead to discontinuous transitions for which PT is not known to perform very well, while following the trajectory, the system undergoes

several continuous transitions as discussed in (Decelle et al., 2018; Bachtis et al., 2024). The pseudocodes for the standard PT and our new PTT algorithms are provided in SI B.1.

Figs. 4–A1-3 show the evolution of two independent Markov chains on the most trained model using the PTT sampling, showing a significant increase in the number of jumps compared to Figs. 4–B1-3. We also compare the number of mode jumps observed with PTT with those of other algorithms, as a function of the total number of AGS steps performed. For the PT algorithm, we test different values of N_T and include the variant with a reference configuration (Krause et al., 2020). For Stacked Tempering, we replicate the method of (Roussel et al., 2023). The total AGS steps are taken as the product of the number of parallel models and the AGS steps per model, as shown in Fig. 4–C1-3. PTT consistently outperforms all other algorithms in all datasets.

Our PTT approach leverages the ease of sampling configurations from an initial RBM. During classical PCD training, early "non-specialized" models thermalize quickly, behaving like Gaussian distributions. For RCM+PCD, independent configurations are efficiently sampled from the low-rank RBM using a static Monte Carlo algorithm (see SI A.1). The trajectory flow greatly accelerates equilibrium convergence, particularly in multimodal structures. The time interval between successive models is set to maintain an acceptance probability for exchanges (Eq. 5) around 0.25. Notably, the total time required to thermalize a PTT chain (AGS steps × number of machines) is unaffected by acceptance values within [0.2, 0.5] (see SI B.2 for details). In practice, effective models (i.e. the parameters of the model) can be saved during training by monitoring the acceptance rate between the last saved model and the current one. Pre-trained models require far fewer saved models, as many are naturally positioned near major phase transitions. The number of models used for each sampling process, along with a detailed description of the algorithm, is provided in the SI. PTT offers the advantage of requiring no extra training (unlike Stacked Tempering) or unnecessary temperature simulations and can be fully applied to other energy-based models. Instead, the equilibrium samples obtained from different models during sampling can be used to compute the log-likelihood, similar to the standard PT approach (Krause et al., 2020). In Fig. 3–right we show that the quality of this estimation is comparable with the online trajectory AIS. Another desirable advantage of the PT algorithms is that one can easily check the thermalisation of the MCMC process by investigating the ergodicity of the random walk in models (Banos et al., 2010), as shown in SI B.2. In addition, we evaluated other indicators of sample quality and potential overfitting. As detailed analysis is in Sect. I of the SI. Our method consistently provides high-quality samples across all datasets and, thanks to precise likelihood computation, can effectively detect overfitting.

6 Conclusions

In this work, we tackle the challenges of training high-quality equilibrium RBMs on highly structured datasets. We focused on improving the sampling process and accurately evaluating test log-likelihood throughout training. Our findings reveal that for datasets that are tightly clustered in their principal component analysis representation, introducing a pre-training phase can enhance training quality. This phase involves encoding the principal directions of the data into the model using a convex optimization approach. However, this pre-training method, that is specific to RBMs, appears to have limited benefits for less structured datasets or for long trainings, as chains get out of equilibrium, and should be combined in the future with optimized MCMC methods to achieve its maximum capacity.

Furthermore, we developed a new approach that exploits the progressive learning of features during RBM training to enhance evaluation and sampling. This approach enables online and precise computation of log-likelihood and leads to an efficient sampling strategy akin to the standard parallel tempering algorithm. Our results demonstrate that both our proposed methods outperform traditional algorithms without the need for simulating models at unnecessary temperatures or training additional models to facilitate sampling. Furthermore, both the trajectory AIS method for log-likelihood estimation and the trajectory parallel tempering algorithm are highly versatile, making them applicable to more complex energy-based models. Additionally, the low-rank RBM model can serve as an effective pre-initialization step for deeper architectures, and the trajectory PT algorithm is suitable for direct use in any EBM, regardless of complexity.

7 CODE AVAILABILITY

The code and datasets are available at `https://github.com/anonymousrbm/iclr-2025`

REFERENCES

David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.

Elisabeth Agoritsas, Giovanni Catania, Aurélien Decelle, and Beatriz Seoane. Explaining the effects of non-convergent sampling in the training of energy-based models. *arXiv preprint arXiv:2301.09428*, 2023.

Daniel J Amit and Victor Martin-Mayor. *Field theory, the renormalization group, and critical phenomena: graphs to computers*. World Scientific Publishing Company, 2005.

Dimitrios Bachtis, Giulio Biroli, Aurélien Decelle, and Beatriz Seoane. Cascade of phase transitions in the training of energy-based models. *NeurIPS (2024), arXiv:2405.14689*, 2024.

R Alvarez Banos, A Cruz, LA Fernandez, JM Gil-Narvion, A Gordillo-Guerrero, Marco Guidetti, A Maiorano, Filippo Mantovani, E Marinari, V Martin-Mayor, et al. Nature of the spin-glass phase at experimental length scales. *Journal of Statistical Mechanics: Theory and Experiment*, 2010(06): P06026, 2010.

Nicolas Béreux, Aurélien Decelle, Cyril Furtlehner, and Beatriz Seoane. Learning a restricted boltzmann machine using biased monte carlo sampling. *SciPost Physics*, 14(3):032, 2023.

Barbara Bravi, Vinod P Balachandran, Benjamin D Greenbaum, Aleksandra M Walczak, Thierry Mora, Rémi Monasson, and Simona Cocco. Probing t-cell response by sequence-based probabilistic modeling. *PLOS Computational Biology*, 17(9):e1009297, 2021a.

Barbara Bravi, Jérôme Tubiana, Simona Cocco, Remi Monasson, Thierry Mora, and Aleksandra M Walczak. Rbm-mhc: a semi-supervised machine-learning method for sample-specific prediction of antigen presentation by hla-i alleles. *Cell systems*, 12(2):195–202, 2021b.

Barbara Bravi, Andrea Di Gioacchino, Jorge Fernandez-de Cossio-Diaz, Aleksandra M Walczak, Thierry Mora, Simona Cocco, and Rémi Monasson. A transfer-learning approach to predict antigen immunogenicity and t-cell receptor specificity. *ELife*, 12:e85126, 2023.

Nicola Bulso and Yasser Roudi. Restricted boltzmann machines as models of interacting variables. *Neural Computation*, 33(10):2646–2681, 2021.

Alessandra Carbone, Aurélien Decelle, Lorenzo Rosset, and Beatriz Seoane. Fast and functional structured data generators rooted in out-of-equilibrium physics. *arXiv preprint arXiv:2307.06797*, 2023.

Davide Carbone, Mengjian Hua, Simon Coste, and Eric Vanden-Eijnden. Efficient training of energy-based models using jarzynski equality. *Advances in Neural Information Processing Systems*, 36, 2024.

Simona Cocco, Christoph Feinauer, Matteo Figliuzzi, Rémi Monasson, and Martin Weigt. Inverse statistical physics of protein sequences: a key issues review. *Reports on Progress in Physics*, 81(3): 032601, 2018.

A. Decelle, G. Fissore, and C. Furtlehner. Spectral dynamics of learning in restricted boltzmann machines. *Europhysics Letters*, 119(6):60001, nov 2017. doi: 10.1209/0295-5075/119/60001. URL `https://dx.doi.org/10.1209/0295-5075/119/60001`.

Aurélien Decelle and Cyril Furtlehner. Exact training of restricted boltzmann machines on intrinsically low dimensional data. *Physical Review Letters*, 127(15):158303, 2021a.

Aurélien Decelle and Cyril Furtlehner. Restricted boltzmann machine: Recent advances and mean-field theory. *Chinese Physics B*, 30(4):040202, 2021b.

Aurélien Decelle, Giancarlo Fissore, and Cyril Furtlehner. Thermodynamics of restricted boltzmann machines and related learning dynamics. *Journal of Statistical Physics*, 172:1576–1608, 2018.

Aurélien Decelle, Cyril Furtlehner, and Beatriz Seoane. Equilibrium and non-equilibrium regimes in the learning of restricted boltzmann machines. *Advances in Neural Information Processing Systems*, 34:5345–5359, 2021.

Aurélien Decelle, Beatriz Seoane, and Lorenzo Rosset. Unsupervised hierarchical clustering using the learning dynamics of restricted boltzmann machines. *Phys. Rev. E*, 108:014110, Jul 2023. doi: 10.1103/PhysRevE.108.014110. URL https://link.aps.org/doi/10.1103/PhysRevE.108.014110.

Aurélien Decelle, Cyril Furtlehner, Alfonso de Jesús Navas Gómez, and Beatriz Seoane. Inferring effective couplings with restricted boltzmann machines. *SciPost Physics*, 16(4):095, 2024.

Guillaume Desjardins, Aaron Courville, Yoshua Bengio, Pascal Vincent, and Olivier Delalleau. Tempered markov chain monte carlo for training of restricted boltzmann machines. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 145–152. JMLR Workshop and Conference Proceedings, 2010.

Jorge Fernandez-de Cossio-Diaz. *Generative Modeling of RNA Sequence Families with Restricted Boltzmann Machines*, pp. 163–175. Springer US, New York, NY, 2025. ISBN 978-1-0716-4079-1. doi: 10.1007/978-1-0716-4079-1_11. URL https://doi.org/10.1007/978-1-0716-4079-1_11.

Louis Grenioux, Éric Moulines, and Marylou Gabrié. Balanced training of energy-based models with adaptive flow sampling. In *ICML 2023 Workshop on Structured Probabilistic Inference and Generative Modeling*, 2023.

John Hertz, Yasser Roudi, and Joanna Tyrcha. Ising models for inferring network structure from spike data. 2011.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Geoffrey E Hinton and Terrence J Sejnowski. Optimal perceptual inference. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, volume 448, pp. 448–453. Citeseer, 1983.

Pierre C Hohenberg and Bertrand I Halperin. Theory of dynamic critical phenomena. *Reviews of Modern Physics*, 49(3):435, 1977.

Koji Hukushima and Koji Nemoto. Exchange monte carlo method and application to spin glass simulations. *Journal of the Physical Society of Japan*, 65(6):1604–1608, 1996.

Martin Kochmański, Tadeusz Paszkiewicz, and Sławomir Wolski. Curie–weiss magnet—a simple model of phase transition. *European Journal of Physics*, 34(6):1555, 2013.

Oswin Krause, Asja Fischer, and Christian Igel. Population-contrastive-divergence: Does consistency help with rbm training? *Pattern Recognition Letters*, 102:1–7, 2018.

Oswin Krause, Asja Fischer, and Christian Igel. Algorithms for estimating the partition function of restricted boltzmann machines. *Artificial Intelligence*, 278:103195, 2020. ISSN 00043702. doi: 10.1016/j.artint.2019.103195. URL https://linkinghub.elsevier.com/retrieve/pii/S0004370219301948.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and Fujie Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Renjie Liao, Simon Kornblith, Mengye Ren, David J Fleet, and Geoffrey Hinton. Gaussian-bernoulli rbms without tears. *arXiv preprint arXiv:2210.10318*, 2022.

Enzo Marinari and Giorgio Parisi. Simulated tempering: a new monte carlo scheme. *Europhysics letters*, 19(6):451, 1992.

Grégoire Montavon and Klaus-Robert Müller. Deep boltzmann machines and the centering trick. *Neural Networks: Tricks of the Trade: Second Edition*, pp. 621–637, 2012.

Radford M Neal. Annealed importance sampling. *Statistics and computing*, 11:125–139, 2001.

H Chau Nguyen, Riccardo Zecchina, and Johannes Berg. Inverse statistical problems: from the inverse ising problem to data science. *Advances in Physics*, 66(3):197–261, 2017.

E Nijkamp, R Gao, P Sountsov, S Vasudevan, B Pang, S-C Zhu, and YN Wu. Mcmc should mix: learning energy-based model with neural transport latent space mcmc. In *International Conference on Learning Representations (ICLR 2022)*., 2022.

Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Learning non-convergent non-persistent short-run mcmc toward energy-based model. *Advances in Neural Information Processing Systems*, 32, 2019.

Erik Nijkamp, Mitch Hill, Tian Han, Song-Chun Zhu, and Ying Nian Wu. On the anatomy of mcmc-based maximum likelihood learning of energy-based models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5272–5280, 2020.

Sebastian Quiroz Monnens, Casper Peters, Kasper Smeets, Luuk Willem Hesselink, and Bernhard Englitz. The recurrent temporal restricted boltzmann machine captures neural assembly dynamics in whole-brain activity. June 2024. doi: 10.7554/elife.98489.1. URL `http://dx.doi.org/10.7554/eLife.98489.1`.

Clément Roussel, Jorge Fernandez-de Cossio-Diaz, Simona Cocco, and Remi Monasson. Accelerated sampling with stacked restricted boltzmann machines. In *The Twelfth International Conference on Learning Representations*, 2023.

Ruslan Salakhutdinov and Iain Murray. On the quantitative analysis of deep belief networks. In *Proceedings of the 25th international conference on Machine learning*, pp. 872–879, 2008.

Russ R Salakhutdinov. Learning in markov random fields using tempered transitions. *Advances in neural information processing systems*, 22, 2009.

Ya G Sinai. *Theory of phase transitions: rigorous results*, volume 108. Elsevier, 2014.

Paul Smolensky. *In Parallel Distributed Processing: Volume 1 by D. Rumelhart and J. McLelland*, chapter 6: Information Processing in Dynamical Systems: Foundations of Harmony Theory. 194-281. MIT Press, 1986.

Alan Sokal. Monte carlo methods in statistical mechanics: foundations and new algorithms. In *Functional integration*, pp. 131–192. Springer, 1997.

Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pp. 1064–1071, 2008.

Jérôme Tubiana, Simona Cocco, and Rémi Monasson. Learning protein constitutive motifs from sequence data. *Elife*, 8:e39397, 2019.

Thijs L van der Plas, Jérôme Tubiana, Guillaume Le Goc, Geoffrey Migault, Michael Kunst, Herwig Baier, Volker Bormuth, Bernhard Englitz, and Georges Debrégeas. Neural assemblies uncovered by generative modeling explain whole-brain activity statistics and reflect structural connectivity. *Elife*, 12:e83139, 2023.

Martin Weigel, Lev Barash, Lev Shchur, and Wolfhard Janke. Understanding population annealing monte carlo simulations. *Phys. Rev. E*, 103:053301, May 2021. doi: 10.1103/PhysRevE.103.053301. URL `https://link.aps.org/doi/10.1103/PhysRevE.103.053301`.

Jianwen Xie, Yang Lu, Song-Chun Zhu, and Yingnian Wu. A theory of generative convnet. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 2635–2644, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/xiec16.html.

Andrew Yale, Saloni Dash, Ritik Dutta, Isabelle Guyon, Adrien Pavao, and Kristin P Bennett. Generation and evaluation of privacy preserving synthetic health data. *Neurocomputing*, 416: 244–255, 2020.

Burak Yelmen, Aurélien Decelle, Linda Ongaro, Davide Marnetto, Corentin Tallec, Francesco Montinaro, Cyril Furtlehner, Luca Pagani, and Flora Jay. Creating artificial human genomes using generative neural networks. *PLoS genetics*, 17(2):e1009303, 2021.

Burak Yelmen, Aurélien Decelle, Leila Lea Boulos, Antoine Szatkownik, Cyril Furtlehner, Guillaume Charpiat, and Flora Jay. Deep convolutional and conditional neural networks for large-scale genomic data generation. *PLOS Computational Biology*, 19(10):e1011584, 2023.

Supplemental Information

A Details of the pre-training of a low rank RBM

A.1 The low-rank RBM and its sampling procedure

Our goal is to pre-train an RBM to directly encode the first d principal modes of the dataset in the model's coupling matrix. This approach avoids the standard procedure of progressively encoding these modes through a series of second-order phase transitions, which negatively impact the quality of gradient estimates during standard training. It also helps prevent critical relaxation slowdown of MCMC dynamics in the presence of many separated clusters.

Given a dataset, we want to find a good set of model parameters ($\boldsymbol{w}$, $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$) for which the statistics of the generated samples exactly match the statistics of the data projected onto the first d directions of the PCA decomposition of the training set. Let us call each of these $\alpha = 1, \dots, d$ projections $m_\alpha = \boldsymbol{u}_\alpha \cdot \boldsymbol{v}/\sqrt{N_\mathrm{v}}$ the *magnetizations* along the mode α, where $\boldsymbol{u}_\alpha$ is the α-th mode of the PCA decomposition of the dataset. A simple way to encode these d-modes is to parameterize the w-matrix as:

$$\boldsymbol{w} = \sum_{\alpha=1}^{d} w_\alpha \bar{\boldsymbol{u}}_\alpha \boldsymbol{u}_\alpha^\top, \qquad \text{with} \qquad (\boldsymbol{u}_\alpha, \bar{\boldsymbol{u}}_\alpha) \in \mathbb{R}^{N_\mathrm{v}} \times \mathbb{R}^{N_\mathrm{h}}, \tag{6}$$

where $\boldsymbol{u}$ and $\hat{\boldsymbol{u}}$ are respectively the right-hand and left-hand singular vectors of $\boldsymbol{w}$, the former being directly given by the PCA, while w_α are the singular values of $\boldsymbol{w}$. Using this decomposition, the marginal energy on the visible variables, $\mathcal{H}(\boldsymbol{v}) = \log \sum_{\boldsymbol{h}} \exp \mathcal{H}(\boldsymbol{v}, \boldsymbol{h})$ can be rewritten in terms of these magnetizations $\boldsymbol{m} \equiv (m_1, \dots, m_d)$

$$\mathcal{H}(\boldsymbol{v}) = -\sum_a \log\cosh\left(\sqrt{N_\mathrm{v}}\bar{u}_a \sum_{\alpha=1}^{d} w_\alpha m_\alpha + \eta_a\right) = \mathcal{H}(\boldsymbol{m}(\boldsymbol{v})). \tag{7}$$

Now, the goal of our pre-training is not to match the entire statistics of the data set, but only the marginal probability of these magnetizations. In other words, we want to model the marginal distribution

$$p_\mathrm{emp}(\boldsymbol{m}) \equiv \sum_{\boldsymbol{v}} p_\mathrm{emp}(\boldsymbol{v}) \prod_{\alpha=1}^{d} \delta\left(m_\alpha - \frac{1}{\sqrt{N_\mathrm{v}}}\boldsymbol{u}_\alpha^T \boldsymbol{v}\right), \tag{8}$$

where δ is the Dirac δ-distristribution. In this formulation, the distribution of the model over the magnetization $\boldsymbol{m}$ can be easily characterized

$$p(\boldsymbol{m}) = \frac{1}{Z}\sum_{\boldsymbol{v}} e^{-\mathcal{H}(\boldsymbol{v})} \prod_{\alpha=1}^{d} \delta\left(m_\alpha - \frac{1}{\sqrt{N_\mathrm{v}}}\boldsymbol{u}_\alpha^T \boldsymbol{v}\right) \tag{9}$$

$$= \frac{1}{Z}\mathcal{N}(\boldsymbol{m}) \exp \sum_a \log\cosh\left(\bar{u}_a \sum_{\alpha=1}^{d} w_\alpha m_\alpha + \eta_a\right) \tag{10}$$

$$= \frac{1}{Z} e^{-\mathcal{H}(\boldsymbol{m}) + N_\mathrm{v} s(\boldsymbol{m})} = \frac{1}{Z} e^{-N_\mathrm{v} f(\boldsymbol{m})} \tag{11}$$

where $\mathcal{N}(\boldsymbol{m}) = \sum_{\boldsymbol{v}} \prod_{\alpha=1}^{d} \delta\left(m_\alpha - \frac{1}{\sqrt{N_\mathrm{v}}}\boldsymbol{u}_\alpha^T \boldsymbol{v}\right)$ is the number of configurations with magnetizations $\boldsymbol{m}$, and thus $S(\boldsymbol{m}) = \log N(\boldsymbol{m})/N_\mathrm{v}$ is the associated entropy. Now, for large N_v the entropic term can be determined using large deviation theory, and in particular the Gärtner-Ellis theorem:

$$p_\mathrm{prior}(\boldsymbol{m}) = \frac{e^{N_\mathrm{v} s(\boldsymbol{m})}}{2^{N_\mathrm{v}}} \approx \exp\left(-N_\mathrm{v}\mathcal{I}(\boldsymbol{m})\right), \tag{12}$$

with the rate function

$$\mathcal{I}(\boldsymbol{m}) = \sup_{\boldsymbol{\mu}} \left[\boldsymbol{m}^T \boldsymbol{\mu} - \phi(\boldsymbol{\mu})\right] = \boldsymbol{m}^T \boldsymbol{\mu}^* - \phi(\boldsymbol{\mu}^*), \tag{13}$$

and

$$\phi(\boldsymbol{\mu}) = \lim_{N_\mathrm{v}\to\infty} \frac{1}{N_\mathrm{v}} \log \left\langle e^{N_\mathrm{v} \boldsymbol{m}^T \boldsymbol{\mu}} \right\rangle = \lim_{N_\mathrm{v}\to\infty} \frac{1}{N_\mathrm{v}} \log \frac{1}{2^{N_\mathrm{v}}} \sum_{\boldsymbol{v}} e^{\sqrt{N_\mathrm{v}} \sum_{\alpha=1}^{d} \mu_\alpha \sum_i u_{\alpha,i} v_i} \tag{14}$$

$$= \lim_{N_\mathrm{v}\to\infty} \frac{1}{N_\mathrm{v}} \sum_{i=1}^{N_\mathrm{v}} \log\cosh\left(\sqrt{N_\mathrm{v}} \sum_{\alpha=1}^{d} \mu_\alpha u_{\alpha,i}\right). \tag{15}$$

Then, given a magnetization $\boldsymbol{m}$, we can compute the minimizer $\boldsymbol{\mu}^*(\boldsymbol{m})$ of $\phi(\mu) - \boldsymbol{m}^T \mu$ which is convex, using e.g. Newton method which converge really fast since we are in small dimension. Note that in practice we will obviously use finite estimates of ϕ, assuming N_v is large enough. As a result we get $\boldsymbol{\mu}^*(\boldsymbol{m})$ satisfying implicit equations given by the constraints given at given N_v:

$$m_\alpha = \frac{1}{\sqrt{N_\mathrm{v}}} \sum_{i=1}^{N_\mathrm{v}} u_i^\alpha \tanh\left(\sqrt{N_\mathrm{v}} \sum_{\beta=1}^{d} u_i^\beta \mu_\beta^*\right). \tag{16}$$

It is then straightforward to check that spins distributed as

$$p_\mathrm{prior}(\boldsymbol{v}|\boldsymbol{m}) \propto e^{N_\mathrm{v} \boldsymbol{\mu}^{*T} \boldsymbol{m}(\boldsymbol{v})} \tag{17}$$

fulfill well the requirement, as $\left\langle \boldsymbol{u}_\alpha^T \boldsymbol{v}/\sqrt{N_\mathrm{v}} \right\rangle_{p_\mathrm{prior}} = m_\alpha$. In other words, we can generate samples having mean magnetization m_α just by choosing v_i as

$$p_\mathrm{prior}(v_i = 1|\boldsymbol{m}) = \mathrm{sigmoid}\left(2\sqrt{N_\mathrm{v}} \sum_{\alpha=1}^{d} u_{\alpha,i} \mu_\alpha^*(\boldsymbol{m})\right) \tag{18}$$

The training can therefore be done directly in the subspace of dimension d. In Ref. (Decelle & Furtlehner, 2021a), it has been shown that such RBM can be trained by mean of the Restricted Coulomb Machine, where the gradient is actually convex in the parameter's space. It is then possible to do a mapping from the RCM to the RBM to recover the RBM's parameters. In brief, the training of the low-dimensional RBM is performed by the RCM, and then the parameters are obtrained via a direct relation between the RCM and the RBM's parameters. The detail of the definition and of the training of the RCM is detailed in the appendix A.2.

A.2 The Restricted Coulomb Machine

As introduced in (Decelle & Furtlehner, 2021a), it is possible to exactly train a surrogate model for the RBM, called the Restricted Coulomb Machine (RCM), on a low dimensional dataset without explicitly sampling the machine allowing to learn even heavily clustered datasets.

The RCM is an approximation of the marginal distribution of the RBM with $\{-1, 1\}$ binary variables:

$$\mathcal{H}(\boldsymbol{v}) = -\sum_i v_i \theta_i - \sum_a \log\cosh\left(\sum_i W_{ia} v_i + \eta_a\right). \tag{19}$$

We then project both the parameters and variables of the RBM on the first d principal components of the dataset:

$$m_\alpha \coloneqq \frac{1}{\sqrt{N_\mathrm{v}}} \sum_{i=1}^{N_\mathrm{v}} v_i u_{i\alpha}, \quad w_{\alpha a} \coloneqq \sum_{i=1}^{N_\mathrm{v}} W_{ia} u_{i\alpha}, \quad \theta_\alpha \coloneqq \frac{1}{\sqrt{N_\mathrm{v}}} \sum_{i=1}^{N_\mathrm{v}} \theta_i u_{i\alpha} \tag{20}$$

with $\alpha \in \{0, \dots, d\}$ and $\boldsymbol{u}$ the projection matrix of the PCA. The projected distribution of the model is then given by

$$p_\mathrm{RBM}(\boldsymbol{m}) = \frac{\exp\left(N_\mathrm{v}\left[\mathcal{S}(\boldsymbol{m}) + \sum_{\alpha=1}^{d} \theta_\alpha m_\alpha + \frac{1}{N_\mathrm{v}} \sum_{a=1}^{N_\mathrm{h}} \log\cosh\left(\sqrt{N_\mathrm{v}} \sum_{\alpha=1}^{d} m_\alpha w_{\alpha a} + \eta_a\right)\right]\right)}{Z} \tag{21}$$

where we ignore the fluctuations related to the transverse directions and $\mathcal{S}[\boldsymbol{m}]$ accounts for the non-uniform prior on $\boldsymbol{m}$ due to the projection of the uniform prior on $\boldsymbol{s}$ for the way to compute it.

The RCM is then built by approximating

$$\log\cosh(x) \simeq |x| - \log 2, \tag{22}$$

which is valid for x large enough. The probability of the RCM is thus given by:

$$p_{\text{RCM}}(\boldsymbol{m}) = \frac{\exp\left(N_{\text{v}}\left[\mathcal{S}(\boldsymbol{m}) + \sum_{\alpha=1}^{d}\theta_\alpha m_\alpha + \sum_{a=1}^{N_{\text{h}}} q_a \left|\sum_{\alpha=1}^{d} n_\alpha m_\alpha + z_a\right|\right]\right)}{Z} \tag{23}$$

where

$$q_a = \sqrt{N_{\text{v}}\sum_{\alpha=1}^{d} w_{\alpha a}^2}, \quad n_a = \frac{w_{\alpha a}}{\sqrt{\sum_{\alpha=1}^{d} w_{\alpha a}^2}}, \quad z_a = \frac{\eta_a}{\sqrt{N_{\text{v}}\sum_{\alpha=1}^{d} w_{\alpha a}^2}}. \tag{24}$$

This can be easily inverted as

$$w_{\alpha a} = \frac{1}{\sqrt{N_{\text{v}}}} q_a n_a \qquad \text{and} \qquad \eta_a = q_a z_a,$$

in order to obtain the RBM from the RCM. The model is then trained through log-likelihood maximization over its parameters. However, this objective is non-convex if all the parameters are trained through gradient ascent. To relax the problem, since we're in low dimension, we can define a family of hyperplanes $(\boldsymbol{n}, \boldsymbol{z})$ covering the space and let the model only learn the weights of each to the hyperplane. We can then discard the ones with a weight low enough for the approximation equation 22 to be bad.

The gradients are given by

$$\frac{\partial J(\boldsymbol{\Theta})}{\partial q_a} = \mathbb{E}_{\boldsymbol{m}\sim p_{\mathcal{D}}(\boldsymbol{m})}\left[|\boldsymbol{n}_a^T\boldsymbol{m} + z_a|\right] - \mathbb{E}_{\boldsymbol{m}\sim p_{\text{RCM}}(\boldsymbol{m})}\left[|\boldsymbol{n}_a^T\boldsymbol{m} + z_a|\right], \tag{25}$$

$$\frac{\partial J(\boldsymbol{\Theta})}{\partial \theta_\alpha} = E_{\boldsymbol{m}\sim p_{\mathcal{D}}(\boldsymbol{m})}\left[m_\alpha\right] - \mathbb{E}_{\boldsymbol{m}\sim p_{\text{RCM}}(\boldsymbol{m})}\left[m_\alpha\right]. \tag{26}$$

The positive term is straightforward to compute. For the negative term, we rely on a discretization of the longitudinal space to estimate the probability density of the model and compute the averages.

A.3 Learning the bias

To initialize the model with its bias set to the dataset's mean, we define the 0-th direction of the longitudinal space as the direction obtained by projecting the configuration space onto the normalized mean vector of the dataset.

The RBM will only contribute to this direction through the visible bias:

$$H(\boldsymbol{m}(v)) = -\sum_{\alpha=0}^{N_{\text{v}}}\theta_\alpha m_\alpha - \frac{1}{N_{\text{v}}}\sum_a \log\cosh\left(\sqrt{N_{\text{v}}}\sum_{\alpha=1}^{d} m_\alpha w_{\alpha a} + \eta_a\right). \tag{27}$$

This implies that this direction is independent of the others within the model. Consequently, we can create an independent mesh for this direction to learn it. The rest of the training is performed using the same procedure as before.

B Sampling via Parallel Tempering using the learning trajectory

Assuming we have successfully trained a robust equilibrium model, there remains the challenge of efficiently generating equilibrium configurations from this model. Although models trained at equilibrium exhibit faster and more ergodic dynamics compared to poorly trained models, the sampling time can still be excessively long when navigating a highly rugged data landscape. Consequently, we devised a novel method for sampling equilibrium configurations that draws inspiration from the well-established parallel tempering approach. In this traditional method, multiple simulations are conducted in parallel at various temperatures, and configurations are exchanged among them using the Metropolis rule. Unlike this conventional technique, our method involves simultaneously

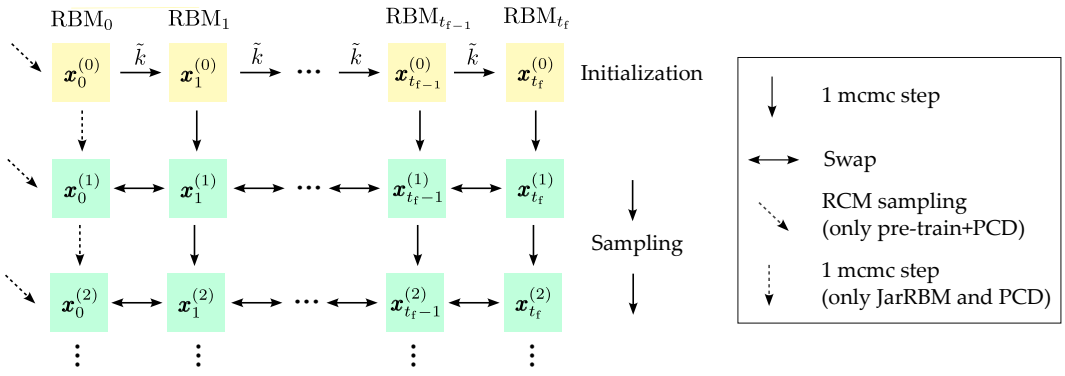

Figure 5: Scheme of PTT. We Initialize the chains of the models by starting from a configuration $\boldsymbol{x}_0^{(0)}$ and passing it through the machines along the training trajectory, each time performing $\tilde{k}$ mcmc steps. For pre-train+PCD, $\boldsymbol{x}_0^{(0)}$ is a sampling from the RCM, otherwise it is a uniform random initialization. The sampling consists of alternating one mcmc step for each model with a swap attempt between adjacent machines. For pre-train+PCD, at each step we sample a new independent configuration for RBM_0 using the RCM.

simulating different models that are selected from various points along the training trajectory. This approach is motivated by the perspective that learning represents an annealing process for the model, encountering second-order type phase transitions during training. In contrast, annealing related to temperature changes involves first-order phase transitions, making traditional parallel tempering less effective for sampling from clustered multimodal distributions.

A sketch of the Parallel Trajectory Tempering (PTT) is represented in fig. 5. Specifically, we save t_f models at checkpoints $t = 1, \dots, t_\mathrm{f}$ along the training trajectory. We denote the Hamiltonian of the model at checkpoint t as $\mathcal{H}_t$, and refer to the Hamiltonian of the RCM model as $\mathcal{H}_0$. We define $\mathrm{GibbsSampling}(\mathcal{H}, \boldsymbol{x}, k)$ as the operation of performing k Gibbs sampling updates using the model $\mathcal{H}$ starting from the state $\boldsymbol{x}$. In all our sampling simulations we used $k = 1$.

The first step is to initialize the models' configurations efficiently. This involves sampling N chains from the RCM model, $\boldsymbol{x}_0^{(0)} \sim \mathrm{RCMSampling}(\mathcal{H}^0)$, and then passing the chains through all the models from $t = 1$ to $t = t_\mathrm{f}$, performing k Gibbs steps at each stage: $\boldsymbol{x}_t^{(0)} \sim \mathrm{GibbsSampling}(\mathcal{H}_t, \boldsymbol{x}_{t-1}^{(0)}, k)$.

The sampling process proceeds in steps where we update the configuration of each model except $\mathcal{H}_0$ with k Gibbs steps, and sample a completely new configuration for the RCM model $\mathcal{H}_0$. Following this update step, we propose swapping chains between adjacent models with an acceptance probability given by:

$$p_\mathrm{acc}(\boldsymbol{x}_t \leftrightarrow \boldsymbol{x}_{t-1}) = \min\left(1, \exp\left(\Delta\mathcal{H}_t(\boldsymbol{x}_t) - \Delta\mathcal{H}_t(\boldsymbol{x}_{t-1})\right)\right), \tag{28}$$

where $\Delta\mathcal{H}_t(\boldsymbol{x}) = \mathcal{H}_t(\boldsymbol{x}) - \mathcal{H}_{t-1}(\boldsymbol{x})$.

We continue alternating between the update step and the swap step until a total of N_mcmc steps is reached. The sampling procedure is illustrated in the following pseudo-code:

Input: Set of models $\{\mathcal{H}_t\}$, $t = 0, \dots, t_\mathrm{f}$, Number of Gibbs steps k, Number of MCMC steps N_mcmc
Output: Configurations $\boldsymbol{x}_t$ for $t = 1, \dots, t_\mathrm{f}$
Initialize: Sample N chains from the RCM model $\boldsymbol{x}_0^{(0)} \sim \mathrm{RCMSampling}(\mathcal{H}_0)$
for $t = 1$ to t_f **do**
  $\boldsymbol{x}_t^{(0)} \sim \mathrm{GibbsSampling}(\mathcal{H}_t, \boldsymbol{x}_{t-1}^{(0)}, \tilde{k})$
end for
for $n = 1$ to $\lfloor N_\mathrm{mcmc}/k \rfloor$ **do**
  for $t = 1$ to t_f **do**
    $\boldsymbol{x}_t^{(n)} \sim \mathrm{GibbsSampling}(\mathcal{H}_t, \boldsymbol{x}_t^{(n-1)}, k)$

end for
Resample $\boldsymbol{x}_0^{(n)} \sim \mathrm{RCMSampling}(\mathcal{H}_0)$
for $t = 1$ to t_f **do**
Compute acceptance probability

$$p_\mathrm{acc}(\boldsymbol{x}_t^{(n)} \leftrightarrow \boldsymbol{x}_{t-1}^{(n)}) = \min\left(1, \exp\left(\Delta\mathcal{H}_t(\boldsymbol{x}_t^{(n)}) - \Delta\mathcal{H}_t(\boldsymbol{x}_{t-1}^{(n)})\right)\right)$$

Swap $\boldsymbol{x}_t^{(n)}$ and $\boldsymbol{x}_{t-1}^{(n)}$ with probability $p_\mathrm{acc}(\boldsymbol{x}_t^{(n)} \leftrightarrow \boldsymbol{x}_{t-1}^{(n)})$
end for
end for

A comparison of performances between PTT and standard Gibbs sampling is reported in Fig.4.

B.1 Pseudo-code of PTT vs PT

nico: We provide here the pseudo code for both Parallel Tempering (Alg. 1) and Parallel Trajectory Tempering (Alg. 2) sampling schemes. The difference between both algorithms lies in the way the parallel models are selected. In the case of Parallel Tempering, an annealing on the temperature is made, whereas for Parallel Trajectory Tempering several models are saved during training.

Algorithm 1 Parallel Tempering

Require: $\{\beta_j\}_{j=\{1,\ldots,\mathtt{N}_\beta\}}, \boldsymbol{\Theta} = \{\boldsymbol{w}, \boldsymbol{\theta}, \boldsymbol{\eta}\}, \mathtt{N}_\mathtt{iter}, \mathtt{N}_\mathtt{increment}, \mathtt{N}_\mathtt{sample}$
for $j = 1, \ldots, N_\beta$ **do**
$\mathtt{chains}[j] = \mathtt{RandomBinaryChains}(\mathtt{N}_\mathtt{sample})$
end for
for $i = 1, \ldots, \mathtt{N}_\mathtt{iter}$ **do**
for $j = 1, \ldots, \mathtt{N}_\beta$ **do**
$\mathtt{chains}[j] \leftarrow \mathtt{AlternateGibbsSampling}(\mathtt{N}_\mathtt{increment}, \beta_j\boldsymbol{\Theta}, \mathtt{chains}[j])$
end for
for $j = 1, \ldots, \mathtt{N}_\beta - 1$ **do**
$\mathtt{chains}[j], \mathtt{chains}[j+1] \leftarrow \mathtt{SwapConfigurations}(\mathtt{chains}[j], \beta_j\boldsymbol{\Theta}, \mathtt{chains}[j+1], \beta_{j+1}\boldsymbol{\Theta})$
end for
end for

Algorithm 2 Parallel Trajectory Tempering

Require: $\{\boldsymbol{\Theta}_j = \{\boldsymbol{w}_j, \boldsymbol{\theta}_j, \boldsymbol{\eta}_j\}\}_{j=\{1,\ldots,\mathtt{N}_\mathtt{model}\}}, \mathtt{N}_\mathtt{iter}, \mathtt{N}_\mathtt{increment}, \mathtt{N}_\mathtt{sample}$
for $j = 1, \ldots, \mathtt{N}_\mathtt{model}$ **do**
$\mathtt{chains}[j] = \mathtt{RandomBinaryChains}(\mathtt{N}_\mathtt{sample})$
end for
for $i = 1, \ldots, \mathtt{N}_\mathtt{iter}$ **do**
for $j = 1, \ldots, \mathtt{N}_\mathtt{model}$ **do**
$\mathtt{chains}[j] \leftarrow \mathtt{AlternateGibbsSampling}(\mathtt{N}_\mathtt{increment}, \boldsymbol{\Theta}_j, \mathtt{chains}[j])$
end for
for $j = 1, \ldots, \mathtt{N}_\mathtt{model} - 1$ **do**
$\mathtt{chains}[j], \mathtt{chains}[j+1] \leftarrow \mathtt{SwapConfigurations}(\mathtt{chains}[j], \boldsymbol{\Theta}_j, \mathtt{chains}[j+1], \boldsymbol{\Theta}_{j+1})$
end for
end for

B.2 Controlling equilibration times with PTT

One of the advantages of PT-like approaches is that they allow easy control over thermalization times by studying the random walk in temperatures (or models for the PTT) visited by each run over time, as proposed in Ref. (Banos et al., 2010). From now on, we will refer only to the number of models used in the PTT sampling, denoted as N_m. In an ergodic sampling process, each run should spend

dataset	$\tau_{\rm exp}$	$\tau_{\rm int}$
MNIST01	258(29)	9.0(15)
HGD	48(10)	3.8(7)
Mickey	2(0)	1(0)
Ising	18(3)	2.4(5)

Table 1: Autocorrelation times associated to the random walk dynamics between models in the PTT sampling obtained with the models trained with pre-training+PCD on different datasets. Models used in the PTT are all chosen with the $a = 0.25$ criterion.

roughly the same amount of time at each of the $N_{\rm m}$ different model indices, as all models have the same probability.

The mean temperature/model index is known to be $\langle n \rangle = (N_{\rm m} - 1)/2$ (assuming the model indices run from 0 to $N_{\rm m} - 1$). This simplifies the computation of the time-autocorrelation function of the model indices $n_i(t)$ visited by each sample i at time t:

$$C(t) = \frac{\overline{(n_i(t+t_0) - \langle n \rangle)(n_i(t_0) - \langle n \rangle)}}{\overline{(n_i(t_0) - \langle n \rangle)^2}}$$

Here, the symbol $\overline{(\cdots)}$ refers to the average over all samples i and initial times t_0 obtained from our PTT runs. This allows us to compute the exponential autocorrelation time, $\tau_{\rm exp}$, from a fit to

$$C(t) \sim A e^{-t/\tau_{\rm exp}} \quad \text{for large } t,$$

as well as the integrated autocorrelation time, $\tau_{\rm int}$, using the self-consistent equation (Sokal, 1997; Amit & Martin-Mayor, 2005):

$$\tau_{\rm int} = \frac{1}{2} + \sum_{t=0}^{6\tau_{\rm int}} C(t).$$

These two times provide different insights: $\tau_{\rm exp}$ gives insight into the time needed to thermalize the simulations (setting the length of the runs to at least $20\tau_{\rm exp}$ is generally a safe (and very conservative) choice to ensure thermalization (Sokal, 1997)), while $\tau_{\rm int}$ gives the time needed to generate independent samples. To keep track of the fluctuations, we compute a different $C^m(t)$ for each $m = 1, \ldots, N_{\rm m}$ system index simulated in parallel in the PTT, where each $C^m(t)$ is averaged over 1000 independent realizations (with all runs initialized to the same model index). We then compute the average over all parallel runs $C(t) = \overline{C^m(t)}$ and use this curve to extract $\tau_{\rm exp}$ and $\tau_{\rm int}$. The errors in C and the times are then extracted from the error of the mean of the values extracted with each of the $C^m(t)$ curves.

For example, in Fig. 6 (left), we show the averaged curves $C(t)$ obtained with the PTT sampling of the machines discussed in the main text in the MNIST01 (top) and HGD (bottom) datasets. The extracted times are $\tau_{\rm exp} = 258(29)$ and $\tau_{\rm int} = 9.0(1.5)$ for MNIST01, and $\tau_{\rm exp} = 48(10)$ and $\tau_{\rm int} = 3.8(7)$ for HGD. These numbers guarantee us that the 10^4 steps used to generate the scatterplots in Fig. 2 were long enough to ensure equilibration in this dataset, but also in the others, as shown in Table 1.

With these measures, we can try to optimize the number of models that we use for the PTT and that are automatically selected during training. The criterion here is that the acceptance of PTT exchanges between neighboring models must not go below a target acceptance a. In the figures shown so far, $a = 0.25$. Let us justify this choice. The total physical time we need to simulate $N_{\rm m}$ models grows proportionally with $N_{\rm m}$ (unless one manages to properly parallelize the sampling, which is feasible but we did not do it), while the time needed to flow between very well trained and poorly trained models controlling the mixing time scales as $\sqrt{N_{\rm m}}/a$, where a is the average acceptance of the exchanges between the models. Since $N_{\rm m}$ grows with a in practice (as shown in the inset of Fig. 6–right), it is expected that the total time can be optimized for a given value of a. In Fig. 6–right we show $\tau_{\rm exp}$ in blue (and $\tau_{\rm exp} N_{\rm m}$ in orange) as a function of a for the data sets MNIST01 (top) and HGD (bottom). These figures show that the choice of a is not particularly crucial as long as it is between 0.2 and 0.5.

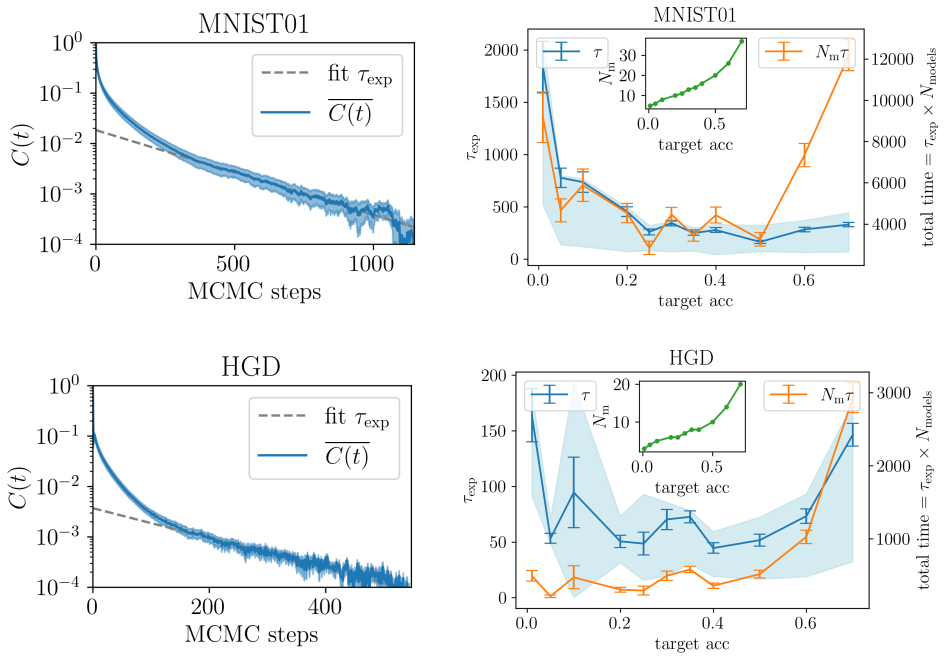

Figure 6: (Left) We show the autocorrelation function $C(t)$ obtained as in Eq. equation B.2 when we analyze the random walk in the model indices during PTT sampling with the same RBMs analyzed in the main text (trained with pretraining+PCD) for the MNIST01 (top) and HGD (bottom) datasets. Both curves were obtained with a PTT run of $5 \cdot 10^4$ MCMC steps, where the models used were automatically selected during training using the PTT moves acceptance criterion of 0.25. (Right) In blue we show the τ_{exp} as a function of the target acceptance rate for MNIST01 (top) and HGD (bottom). The error bar results from the error of the mean of the values extracted from the different $C_m(t)$ and the shading marks the range between the largest and the smallest value of the N_{m} curves. In orange, we show the total simulation costs (τ_{exp} multiplied by the number of models to be simulated in parallel) as a function of the target acceptance rate. This number of models saved, as a function of the acceptance rate, is shown in green in the inset.

C TRAINING DETAILS

We describe in Tables 2 and 3 the datasets and hyperparameters used during training. The test set was used to evaluate the metrics.

Table 2: Details of the datasets used during training.

Name	#Samples	#Dimensions	Train size	Test size
CelebA	30 000	1024	60%	40%
Human Genome Dataset (HGD)	4500	805	60%	40%
Ising	20 000	64	60%	40%
Mickey	16 000	1000	60%	40%
MNIST-01	10 610	784	60%	40%
MNIST	50 000	784	60%	40%

All experiments were run on a RTX 4090 with an AMD Ryzen 9 5950X.

Table 3: Hyperparameters used for the training of RBMs.

Name	Batch size	#Chains	#Epochs	Learning rate	#MCMC steps	#Hidden nodes
CelebA						
PCD	2000	2000	10 000	0.01	100	500
Pre-train+PCD	2000	2000	10 000	0.01	100	500
HGD						
PCD	2000	2000	10 000	0.01	100	200
Pre-train+PCD	2000	2000	10 000	0.01	100	200
Ising						
PCD	2000	2000	10 000	0.01	100	200
Pre-train+PCD	2000	2000	10 000	0.01	100	200
Mickey						
PCD	2000	2000	10 000	0.01	100	200
Pre-train+PCD	2000	2000	10 000	0.01	100	200
MNIST-01						
PCD	2000	2000	10 000	0.01	100	200
Pre-train+PCD	2000	2000	10 000	0.01	100	200
MNIST						
PCD	2000	2000	10 000	0.01	100	200
Pre-train+PCD	2000	2000	10 000	0.01	100	200

D COMPUTATION OF THE LIKELIHOOD USING AIS

To compute the likelihood of the RBMs we used the Annealed Importance Sampling method introduced by (Neal, 2001) and recently studied in the case of the RBM (Krause et al., 2020).

D.1 OFFLINE AIS

We follow the method described in (Krause et al., 2020). The AIS estimate is based on creating a sequence of machines, at one end a machine from which we can compute the partition function and at the other end the machine from which we want to compute the partition function. Let's consider that we have $i = 0, \dots, N_{\text{rbm}}$ machines described by

$$p_i(\boldsymbol{x}) = \frac{e^{-\mathcal{H}_i(\boldsymbol{x})}}{Z} = \frac{p_i^*(\boldsymbol{x})}{Z},$$

where $p_0 = p_{\text{ref}}$ denote the RBM from which we can compute the partition function (typically a RBM where the weight matrix is zero) and $p_{N_{\text{rbm}}} = p_{\text{target}}$ the machine from which we want to estimate the partition function. For each machine, we denote its set of parameters $\boldsymbol{\Theta}_i = \{\boldsymbol{w}_i, \boldsymbol{\theta}_i, \boldsymbol{\eta}_i\}$. Following (Krause et al., 2020), defining a transition rule $T_i(\boldsymbol{x}_{i-1} \to \boldsymbol{x}_i)$ in order to pass from a configuration $\boldsymbol{x}_{i-1}$ of the system $i-1$ to $\boldsymbol{x}_i$ of the system i, that we have the following identity

$$\frac{Z_{N_{\text{rbm}}}}{Z_0} = \left\langle \exp\left(-\sum_{i=1}^{N_{\text{rbm}}} \mathcal{H}_{i+1}(\boldsymbol{x}_i) - \mathcal{H}_i(\boldsymbol{x}_i) \right) \right\rangle_{\text{ForwardPath}} . \tag{29}$$

There, if one can compute Z_0, the estimate of $Z_{N_{\text{rbm}}}$ is obtained by averaging the r.h.s. of Eq. 29 over the forward trajectory starting from a configuration $\boldsymbol{x}_0$ using the operators T_i. While the method is exact, the variance of the estimation relies on on close are two successive distribution p_i, p_{i+1}.

Temperature AIS: a commonly used implementation of AIS, is to multiply the Hamiltonian of the system by a temperature, and to the different systems indexed by i, by a set of β_i where $\beta_0 = 0$,

Table 4: Training time of the RBMs

Name	Training time (s)
CelebA	
PCD	119.2 ± 0.8
Pre-train+PCD	3134.0 ± 0.6
HGD	
PCD	106.2 ± 0.3
Pre-train+PCD	266 ± 3
Ising	
PCD	62 ± 1
Pre-train+PCD	104 ± 2
Mickey	
PCD	118.3 ± 3
Pre-train+PCD	246.0 ± 0.7
MNIST-01	
PCD	100.6 ± 0.3
Pre-train+PCD	258.0 ± 0.9
MNIST	
PCD	105.7 ± 0.2
Pre-train+PCD	233.9 ± 0.2

Table 5: RCM Hyperparameters

Dataset name	# Dimensions	# Hidden nodes	Training time (s)
CelebA	4	100	3006
HGD	3	100	150
Ising	3	100	32
Mickey	2	100	118
MNIST-01	3	100	148
MNIST	3	100	119

$\beta_i < \beta_{i+1}$ and $\beta_{N_{\text{rbm}}} = 1$. Therefore, it is equivalent to choosing $\Theta_i = \beta_i\{\boldsymbol{w}, \boldsymbol{\theta}, \boldsymbol{\eta}\}$. Another version of Temperature AIS consist in choosing a slightly different reference distribution p_0 that take into account the bias of the model. In this setting, the intermediate distribution are taken as

$$p_i(\boldsymbol{x}) = \frac{p_{\text{ref}}^{(1-\beta_i)} e^{-\beta_i \mathcal{H}(\boldsymbol{x})}}{Z_i}.$$

The main difference is that a prior distribution is used as reference $p_0 = p_{\text{ref}}$ and the temperatures interpolate between this prior and the target distribution. It is quite common to take

$$p_{\text{ref}}(\boldsymbol{x}) = \frac{\exp\left(\sum_i \theta_i v_i + \sum_a \eta_a h_a\right)}{\prod_i 2\cosh(\theta_i) \prod_a 2\cosh(\eta_a)}.$$

Trajectory AIS: the general derivation of AIS allow in principle to choose the set forward probability and/or intermediate system as one sees fit. Since again, a change of temperature in the system can induce discontinuous transition for which, successive distribution will be far away. While, when considering system that are neighbors in the learning trajectory, the system should pass through

continuous transition which do not suffer this problem. Hence, we consider a set of RBMs along the learning trajectory. The RBMs are chosen following the prescription that the exchange rate between configuration (see Eq. 5) is about ~ 0.3). Then the formula Eq. 29 is used to estimate the partition function.

D.2 Online AIS

The online AIS (Annealed Importance Sampling) is the trajectory AIS evaluated at every gradient update during training. A set of permanent chains is initialized at the beginning of the training process. Every time the parameters are updated, a step is performed on these chains before using them to compute the current importance weights. This approach is computationally efficient relative to the cost of sampling (1 step vs. 100 steps) and allows for a very precise estimation.

E Training of the RBM using the Jarzynski equation

In this section, we describe the procedure introduced in (Carbone et al., 2024) for training energy-based models by leveraging the Jarzynki equation, and we adapt it to the specific case of the RBM. At variance with (Carbone et al., 2024), we introduce an additional assumption on the transition probability of the parameters that will lead us to the the reweighting implemented in population annealing methods (Weigel et al., 2021).

In one of its formulations, the Jarzynski equation states that we can relate the ensemble average of an observable $\mathcal{O}$ with the average obtained through many repetitions of an out-of-equilibrium dynamical process. Let us consider a path $\boldsymbol{y} = \{(\boldsymbol{x}^0, \boldsymbol{\theta}^0), (\boldsymbol{x}^1, \boldsymbol{\theta}^1), \dots, (\boldsymbol{x}^t, \boldsymbol{\theta}^t)\}$, where $\boldsymbol{x}^i = (\boldsymbol{v}^i, \boldsymbol{h}^t)$ refers to the model variables and $\boldsymbol{\theta}^i = (w^i, \eta^i, \theta^i)$, the model parameters, both at time i.

If we consider the training trajectory of an RBM, $p_0 \to p_1 \to \cdots \to p_{t-1} \to p_t$, we can write the average of an observable $\mathcal{O}$ over the last model of the trajectory p_t as

$$\langle \mathcal{O} \rangle_t = \frac{\left\langle \mathcal{O}\, e^{-W^t} \right\rangle_{\text{traj}}}{\langle e^{-W^t} \rangle_{\text{traj}}} = \frac{\sum_{r=1}^{R} \mathcal{O}(\boldsymbol{y}_r)\, e^{-W_r^t}}{\sum_{r=1}^{R} e^{-W_r^t}}, \tag{30}$$

where W^t is a trajectory-dependent importance factor and the averages on the rhs are taken across many different trajectory realizations. In particular, W_t is given by the recursive formula:

$$W^t(\boldsymbol{y}) = \mathcal{H}^t(\boldsymbol{x}^t) - \mathcal{H}^0(\boldsymbol{x}^0) - \sum_{i=1}^{t} \log \tilde{T}^i(\boldsymbol{y}^i \to \boldsymbol{y}^{i-1}) + \sum_{i=1}^{t} \log T^i(\boldsymbol{y}^{i-1} \to \boldsymbol{y}^i) \tag{31}$$

$$= W^{t-1}(\boldsymbol{y}) + \mathcal{H}^t(\boldsymbol{x}^t) - \mathcal{H}^{t-1}(\boldsymbol{x}^{t-1}) - \log \frac{\tilde{T}^t(\boldsymbol{y}^t \to \boldsymbol{y}^{t-1})}{T^t(\boldsymbol{y}^{t-1} \to \boldsymbol{y}^t)} \tag{32}$$

Where T and $\tilde{T}$ are, respectively, the forward and backward transition probabilities.

The transition matrix between two states $\boldsymbol{y}^{t-1} \to \boldsymbol{y}^t$ is factorized into two different subsequent movements:

1. Update the variables $\boldsymbol{x}^{t-1} \to \boldsymbol{x}^t$ on the fixed model $\boldsymbol{\theta}^{t-1}$,
2. Update the parameters $\boldsymbol{\theta}^{t-1} \to \boldsymbol{\theta}^t$ on fixed variables $\boldsymbol{x}^t$.

In other words,

$$T(\boldsymbol{y}^{t-1} \to \boldsymbol{y}^t) = w_{\boldsymbol{\theta}^{t-1}}(\boldsymbol{x}^{t-1} \to \boldsymbol{x}^t) t_{\boldsymbol{x}^t}(\boldsymbol{\theta}^{t-1} \to \boldsymbol{\theta}^t), \tag{33}$$

and

$$\tilde{T}(\boldsymbol{y}^t \to \boldsymbol{y}^{t-1}) = t_{\boldsymbol{x}^t}(\boldsymbol{\theta}^t \to \boldsymbol{\theta}^{t-1}) w_{\boldsymbol{\theta}^{t-1}}(\boldsymbol{x}^t \to \boldsymbol{x}^{t-1}), \tag{34}$$

so that

$$\log \frac{\tilde{T}^t(\boldsymbol{y}^t \to \boldsymbol{y}^{t-1})}{T^t(\boldsymbol{y}^{t-1} \to \boldsymbol{y}^t)} = \log \frac{w_{\boldsymbol{\theta}^{t-1}}(\boldsymbol{x}^t \to \boldsymbol{x}^{t-1})}{w_{\boldsymbol{\theta}^{t-1}}(\boldsymbol{x}^{t-1} \to \boldsymbol{x}^t)} + \log \frac{t_{\boldsymbol{x}^t}(\boldsymbol{\theta}^t \to \boldsymbol{\theta}^{t-1})}{t_{\boldsymbol{x}^t}(\boldsymbol{\theta}^{t-1} \to \boldsymbol{\theta}^t)}. \tag{35}$$

We know the ratio between the transition probabilities between $\boldsymbol{x}^{t-1} \to \boldsymbol{x}^t$, because evolution of the variables during the simulation is performed using MCMC simulations that satisfy detailed balance. This means that:

$$\log \frac{w_{\boldsymbol{\theta}^{t-1}}(\boldsymbol{x}^t \to \boldsymbol{x}^{t-1})}{w_{\boldsymbol{\theta}^{t-1}}(\boldsymbol{x}^{t-1} \to \boldsymbol{x}^t)} = \mathcal{H}^{t-1}(\boldsymbol{x}^t) - \mathcal{H}^{t-1}(\boldsymbol{x}^{t-1}), \tag{36}$$

and we are left with

$$W^t(\boldsymbol{y}) \quad = \quad W^{t-1}(\boldsymbol{x}) + \mathcal{H}^t(\boldsymbol{x}^t) - \mathcal{H}^{t-1}(\boldsymbol{x}^t) - \log \frac{t_{\boldsymbol{x}^t}(\boldsymbol{\theta}^t \to \boldsymbol{\theta}^{t-1})}{t_{\boldsymbol{x}^t}(\boldsymbol{\theta}^{t-1} \to \boldsymbol{\theta}^t)}. \tag{37}$$

The standard formula that we use in the paper assumes that the changes in $\boldsymbol{\theta}$ are so small that we can consider the transition probabilities to be symmetrical and the last term in the rhs cancels out. In that case, we recover the known expressions of the population annealing.

Notice that, since Eq. equation 30 is an exact result, the importance weights should, in principle, eliminate the bias brought by the non-convergent chains used for approximating the log-likelihood gradient in the classical PCD scheme. However, after many updates of the importance weights, one finds that only a few chains carry almost all the importance mass. In other words, the vast majority of the chains we are simulating are statistically irrelevant, and we expect to get large fluctuations in the estimate of the gradient because of the small effective number of chains contributing to the statistical average. A good observable for monitoring this effect is the Effective Sample Size (ESS), defined as (Carbone et al., 2024)

$$\text{ESS} = \frac{\left(R^{-1} \sum_{r=1}^{R} e^{-W^{(r)}}\right)^2}{R^{-1} \sum_{r=1}^{R} e^{-2W^{(r)}}} \in [0, 1], \tag{38}$$

which measures the relative dispersion of the weights distribution. Then, a way of circumventing the weight concentration on a few chains is to resample the chain population according to the importance weights every time the ESS drops below a certain threshold, for instance 0.5. After this resampling, all the chain weights have to be set to 1 ($W^{(r)} = 0 \ \forall r = 1, \dots, R$).

F Tuning of the hyperparameters

In order to justify our choice of hyperparameters in the main text, we illustrate on Fig. 7 the value of the log-likelihood as a function of the number of hidden nodes and of the learning of the machine. In order to be fair, the comparison is done at a constant number of gradient updates of 10000. In this regime, the machines that perform the best have a learning rate of $\gamma \sim 0.1, 0.005$ and about $N_{\text{h}} = 500, 1000$ hidden nodes. We can expect that smaller learning rates reach the same value of the log-likelihood but for a much longer training. We clearly see the benefit of the RCM, where the log-likelihood achieves much higher values in general.

G Continuous phase transition

In statistical physics, and more precisely in mean-field model, a model described by the Gibbs-Boltzmann distribution is studied in the infinite size limit, $N \to \infty$, and characterize the state of the system by investigating the free energy of the system, defined by minus the log partition function

$$F[\boldsymbol{\theta}] = -\log(Z[\boldsymbol{\theta}])$$

One of the goals is thus to determine the typical configurations that should be expected given a set of model parameters. In the case of the RBM, the question is what are the typical visible (and/or hidden) variables that one expects to observe. For example, one may ask what typical energies one expects to observe for the parameters $\boldsymbol{\theta}$. For this purpose, let us rewrite the partition function as a sum over all possible values of the energy

$$Z = \sum_E e^{-E} g(E) = \sum_E e^{-E+S(E)} = \sum_E e^{-F(E)} = \sum_E e^{-Nf(E)}$$

where $g(E)$ the number of states with a given energy E, and thus $S(E) = \log g(E)$ is the entropy. Finally, the free energy is defined as $F(E) = E - S(E)$ and $f(E) = F(E)/N$ is the free energy per variable. All Z, E and F depend on the model parameters. N refers to the number of variables.

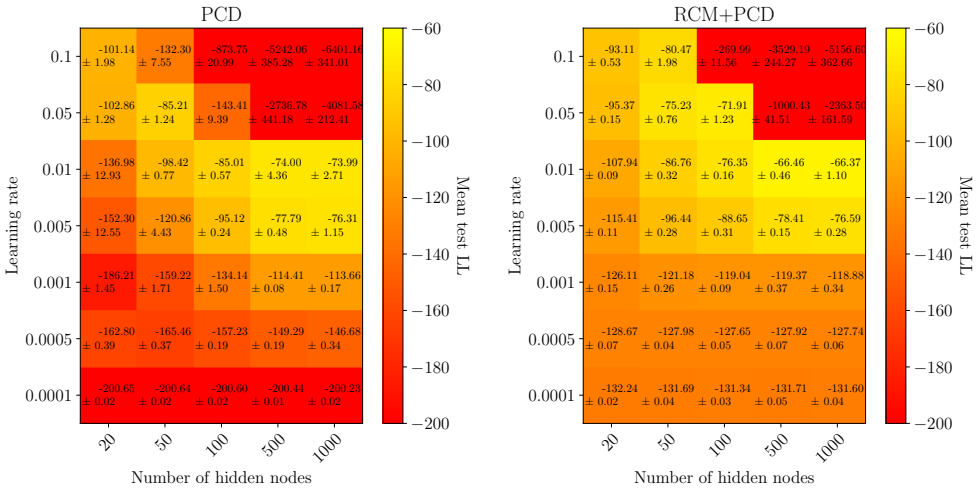

Figure 7: Comparison of the end log-likelihood reached by the RBM using different pairs of values for the learning rate and the hidden layer dimension on MNIST-01. The left panel shows these result when performing PCD training and the right one the results when pre-training the RBM before PCD.

For $N \to \infty$, the sum in Z is then dominated by the states with minimum free energy. A "first order", or discontinuous transition, occurs in the parameters if the first derivative of $f(E)$ is discontinuous with respect to the parameters (in physics, the parameter is typically the temperature). A "second order", or continuous transition, occurs when the first derivative is continuous, but the second derivative is discontinuous.

In the case of the RBM, the first learning transition can be mapped to a very simple model, the so-called Curie-Weiss model, as recently described in Ref. (Bachtis et al., 2024). The Curie-Weiss model is fully solvable, and for this reason the meaning of a continuous transition is perhaps easier to understand there. Let's give it a try. The Curie-Weiss model consists in a set of N discrete variables $s_i = \pm 1$. We define the (exponential) distribution over these variables as

$$p(s) = \frac{1}{Z} \exp(\frac{\beta}{N} \sum_{i<j} s_i s_j + \beta H \sum_i s_i)$$

where β and H are parameters of the model and Z the normalization constant also called partition function. The question is to understand the "structure" of the distribution as a function of β.

Case $H = 0$: we expect that for β small each spin behaves as an isolated Rademacher random variables (each spin being ± 1 with probability one-half), while for large value of β, the distribution of the system is dominated by configurations where the variables have (almost) all the same signs. The key mathematical aspect is to study the system in the limit where $N \to \infty$. A way to study this distribution is to investigate the moment generating function. It is possible to show (rigorously in this precise model), that the probability of $m = N^{-1} \sum_i s_i$, also called the magnetization of the system, is given by $p(m) \propto \exp(-N\Omega(m))$ (in the large N limit) where $\Omega(m)$ is a large deviation function. In the CW model, it is given by

$$\Omega(m) = \frac{\beta m^2}{2} - \log\left[2\cosh(\beta m)\right].$$

The structure of $\Omega(m)$ is such that for $\beta < \beta_c = 1$, it is a convex function with only one minimum in $m = 0$, while for $\beta > \beta_c$ it has two symmetric minima $m = \pm m(\beta) \neq 0$ with $\Omega(m(\beta)) = \Omega(-m(\beta))$ (the two minimum are equally probable). The values (in function of β) of the m are continuous: $m = 0$ for $\beta \in [0, 1]$, and then positive until it saturates to 1 as $\beta \to \infty$. In practice, it means that the distribution $p(m)$ pass from an unimodal distribution to a bimodal one as β increases. The point where it happens is at $\beta_c = 1$ and is called a second order (continuous) phase transition because de value of m changes continuously from zero to non-zero values. A particular interesting properties is that at β_c, the function $\Omega(m)$ develops a non-analyticity in its second-order derivative,

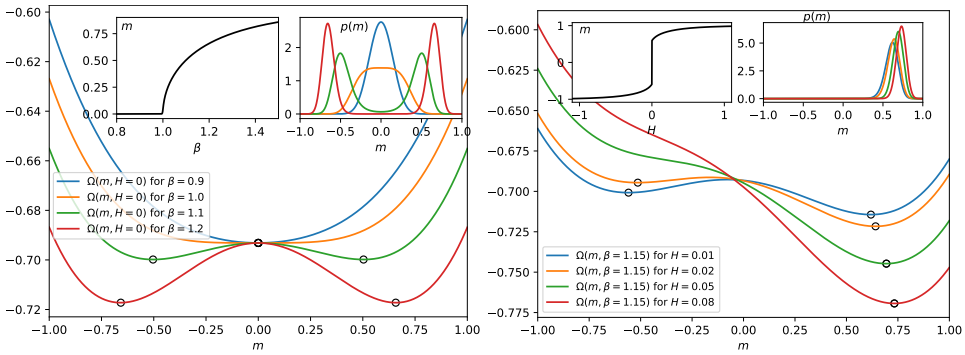

Figure 8: **Left:** The case $H = 0$ for various temperature. We illustrate the symmetric shape of the large deviation function and how the two minima emerges from the centers. The minima of m go continuously from zero to non-zero values. In the inset we also illustrate the shape of the probability distribution for moderately large size $N = 500$. For $N \to \infty$ they converge toward a δ distribution. **Right:** The case $H \neq 0$. In this example, we illustrate what happen where the bias parameter H is changed at a fix temperature $\beta = 1.15 > 1$. We see that for small values of H, Ω has two minima, yet the dominant one suppresses exponentially the other one for large system size. For larger values of the bias the metastable minimum disappears. We show how the place of the minimum jump when the bias goes from negative to positive values.

that can be linked with long-range fluctuations. For a recent review (Kochmański et al., 2013), and for more rigorous results (Sinai, 2014).

Case $H \neq 0$: in this case we expect that the system will always be biased toward configurations where the spins have the same direction(sign) as the bias (or field) H. Yet, for small value of β, the bias will be quite small, hence the magnetization will be small but not zero, $m \sim \beta H$. When β is large, the system will now have a stronger and strong magnetization toward the magnetic field $m \sim \text{sign}(H)$. Now we can again compute the large deviation function of the system

$$\Omega(m) = \frac{\beta m^2}{2} - \log\left[2\cosh(\beta m)\right] + Hm.$$

Again, it is possible to analyze the function Ω as a function of β and H. For $\beta < 1$ the function is convex and there is only one minimum located at $m > 0$. When $\beta > 1$ two situations can be distinguished. First, let's denote

$$H_{\text{sp}}(\beta) = \sqrt{\frac{\beta - 1}{\beta}} - \text{atanh}\left(\frac{\beta - 1}{\beta}\right).$$

If $H > H_{\text{sp}}(\beta)$, then the large deviation function has only one minimum for positive m. Now if $H < H_{\text{sp}}(\beta)$, then Ω will present two minima. A first one for positive m and a second one for negative m. The reason is that the pairwise interactions "manage" to create a metastable state for negative magnetization even in the presence of a positive field or bias. It is clear that this state will be unstable w.r.t. the one with positive magnetization which traduces in a minimum of higher value in the large deviation function. By changing the value of H from large positive values toward large negative ones, we then pass from a system with only one state, then two states with one being subdominant, and finally back to one state but where the magnetization has changed its sign. In the infinite size limit, the system will always dominated by the state where Ω is at its lowest value. However, if one prepares a system close to a metastable state, the system will remain trapped there until it disappears.

How is this related to the training of the RBM ? First, it has been shown in previous works (Decelle et al., 2018; Bachtis et al., 2024) that the RBM's training undergoes a set of continuous transitions, at least at the beginning of the learning. Hence, changing the parameters of the RBM following the learning trajectories, the transitions are continuous. Then in (Decelle & Furtlehner, 2021a) it is shown that the relaxed version of the RBM, namely the RCM undergoes a first-order transition when changing the temperature. Schematically to understand that, it is enough to see how

the parameters β acts on the system. In the case of the RCM, the energy term behaves in the standard way, i.e. linearly with the inverse temperature β and the free energy takes the ordinary form

$$\log(Z) = -\beta F = -\beta E + S.$$

Therefore β act as a trade-off parameters between the energy E and the entropy S. By varying β, we change the contribution of the energy while keeping the entropy fixed. Hence, when the corresponding large deviation function of the RCM has reached a point having many minima with the same value of the free energy up to $\mathcal{O}(1/N)$ corrections, all these minima realize in principle different trade-off between energy and entropy. As a result a slight change in β will offset this sensible balance and the respective free energies will acquire differences of order $\mathcal{O}(N)$ leading to favor dramatically one state against the others, i.e. the one having originally the lowest [resp. the largest] energy when β is increased. It is very likely that by changing β a discontinuous transition is encountered, where the minima of Ω vanishes far away from each other rather than a continuous one. Concerning the RBM the argument for the occurence of such a phase transition, is only slightly more involved than for Coulomb machine because for RBM the energy (obtained after summing over latent variables) has a non-linear behavior w.r.t. β. A formal argument would go schematically as follows: just writing the local minima of the free energy (function of magnetization in the reduced intrinsic space) and looking at how the equilibrium is displaced with temperature shows that their energy vary indeed in a non-linear way with temperature and non-uniformly regarding their position on the magnetization manifold, while the entropy contribution again do not change. This again necessarily implies that changing even slightly the temperature will break the highly sensitive equilibrium obtained between these state corresponding to the multimodal distribution and will again typically favor one state among all. This scenario is expected to occur as soon as the data are located on low dimensional space ($d = O(1)$) compared to a large embedding space ($d = O(N),\ N \gg 1$) which is quite common in our point of view, at least for the type of data we are interested in.

To conclude, moving the parameters along the learning trajectory, leads the system to undergo continuous transition where $p_{\boldsymbol{\Theta}(t)}(\boldsymbol{s})$ is very close to $p_{\boldsymbol{\Theta}(t+\delta t)}(\boldsymbol{s})$. Changing the temperature tends to produces discontinuous transition where $p_{\beta(t)\boldsymbol{\Theta}}(\boldsymbol{s})$ can happen to be very far from $p_{\beta(t+\delta t)\boldsymbol{\Theta}}(\boldsymbol{s})$.

H FIRST ORDER TRANSITIONS IN RBMS

In this section, we examine the appearance of first-order transitions when adjusting the temperature of a well-trained RBM on clustered datasets—a phenomenon that significantly hinders the performance of thermal sampling algorithms. We first illustrate this issue using an analytically tractable toy model, followed by an analysis of the first-order temperature transition observed in the RBM trained on the HGD dataset, as discussed in the main text.

H.1 THEORETICAL ANALYSIS IN A SIMPLE MODEL

We propose a simple model of dataset on which we will show that the learned RBM is having a first order transition in temperature. We first consider an artificial dataset using a Curie-Weiss model. We defined the following Hamiltonian on the variables $s_i = \pm 1$

$$\mathcal{H} = -\frac{1}{2N_{\mathrm{v}}}\left(\sum_i s_i\right)^2, \quad p(\boldsymbol{s}) = \frac{1}{Z}\exp(-\beta_T \mathcal{H}).$$

When the inverse temperature β_T is below one, the distribution in the $N_{\mathrm{v}} \to \infty$ limit is unimodal, with the Boltzmann average of the variables s being $\langle s \rangle = 0$. For $\beta_T > 1$, the system becomes bimodal, with $\langle s_i \rangle = m$, where m satisfies the self-consistent equation $m = \tanh(\beta_T m)$. We propose to model this dataset using a Bernoulli-Gauss RBM with one hidden node. The visible nodes, $\sigma = \{0, 1\}$, follow a Bernoulli distribution, while the hidden node, τ, follows a Gaussian distribution with zero mean and variance $1/N_{\mathrm{v}}$, combined with a local visible bias. When the inverse temperature β_T is below one, the distribution in the $N_{\mathrm{v}} \to \infty$ limit is unimodal and the average w.r.t. the Boltzmann distribution of the values of the variables s's is $\langle s \rangle = 0$. When β_T is above one, the system becomes bimodal with $\langle |s_i| \rangle = m$ when m is defined by the self-consistent equation $m = \tanh(\beta_T m)$. We now propose to learn this dataset using a Bernoulli-Gauss RBM with one hidden node, where the visible nodes are $\sigma = \{0, 1\}$, and the hidden one τ is Gaussian distribution

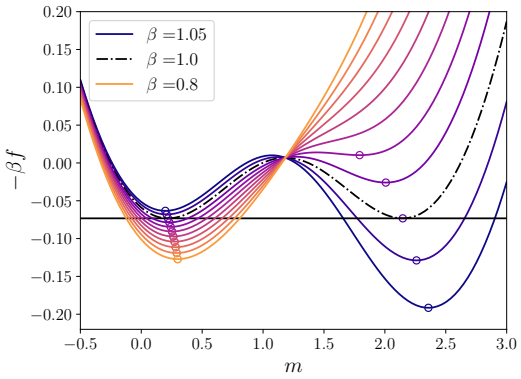

Figure 9: Free energy $-\beta f(m_\tau)$ for $\beta \in [0.8, 1.05]$ for the optimal RBM's parameters learned on a CW model with $\beta_T = 1.4$. At $\beta = 1.0$ the two minima are perfectly adjusted. As we change the temperature of the RBM, they get destabilized showing the presence of a first order transition.

of zero mean and variance $1/N_{\mathrm{v}}$, and a local visible bias. The Hamiltonian is then

$$\mathcal{H} = -\sum_i \sigma_i W_i \tau - \sum_i \eta_i \sigma_i,$$

where W_i is the weight matrix and η_i the biases. The leaning then consist in converting the binary variable of the Curie-Weiss model into $\{0, 1\}$ and to learn the coresponding dataset. By a rapid inspection (in the infinite size limit), we can easily show that the optimal learned parameters for the RBM are

$$w_i = 2\sqrt{\beta_T} \text{ and } \eta_i = -2\beta_T.$$

Now, thanks to the simplicity of the model, we can actually compute the free energy of the model where the Hamiltonian has been rescaled by an annealing temperature β, we obtain

$$-f(m_T) = \frac{m_\tau^2}{2} - N^{-1} \sum_i \log\left[1 + \exp(\beta w_i m_\tau + \beta \eta_i)\right]$$

$$\text{where } m_\tau = \frac{1}{N} \sum_i \beta w_i \mathrm{sigm}\left[1 + \exp(\beta w_i m_\tau + \beta \eta_i\right]$$

In Fig. 9, we plot $-\beta f(m_T)$ for ten different values of β within the range $[0.8, 1.05]$. It is evident that $\beta = 1$ represents the coexistence point, indicating a first-order transition in temperature. Note that the probability $p(m) \approx e^{-N\beta f(m_T)}$ for large N, implying that for $\beta > 1$, the typical configuration corresponds to $m > 1$, whereas for $\beta < 1$, it corresponds to $m < 1$. This clearly shows that even on simple case we observe such transition when doing annealing.

H.2 First order phase transition in temperature on a trained RBM on a real dataset

We now consider the RBM trained on the HGD dataset using the pretraining and 50 000 updates of PCD training. As we vary the temperature, we recover the probability $p(\boldsymbol{m})$ learned by the model using the TMC method (Béreux et al., 2023), projected along the second principal component in Fig. 11 and on both first and second principal components in Fig. 10. While $\beta < 1.0$, the probability distribution is dominated by the central mode, equilibrating between all modes $\beta = 1.0$. As β goes above 1.0, the distribution becomes dominated by the two external clusters.

The effect is clearly visible when sampling the model using the PT algorithm (Fig. 12). The sampling was performed with 200 temperatures with a minimum acceptance rate of 0.78 between two temperatures. We see a dataset cluster being ignored with only a few samples in it. However, when resampling sampling the model for 10^6 AGS steps starting from the samples obtained through PT, we see a clear change in the distribution, equilibrating with the missing cluster. In contrast, performing the same experiment with PTT instead of PT, we observe no shift in the empirical distribution of the samples after 10^6 AGS, as the upper cluster is already sampled.

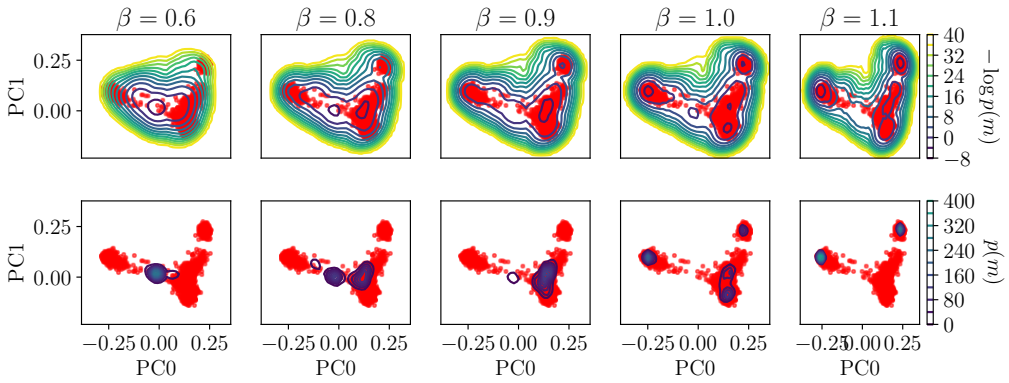

Figure 10: Projection of the negative log probability (top) and probability (bottom) learned by the model, constrained to visible magnetizations m_0 and m_1, for various values of β. The red dots represent the training data projected onto the first two principal components.

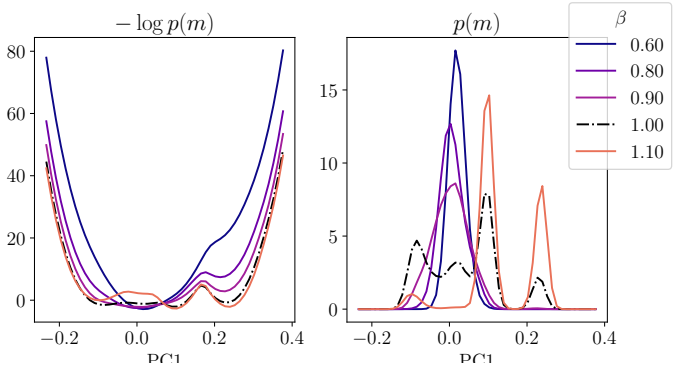

Figure 11: Negative log probability (left) and probability (right) learned by the model constrained to have a given m_1 for several values of β. The equilibrium distribution of the model is displayed as the black dotted line ($\beta = 1$).

I Overfitting and privacy loss as quality indicators

In this section, we examine the quality of the samples generated, regarding overfitting and privacy criteria which have been defined for genomic data in particular. We look at this on the models trained with PCD with and without pre-training. We focus on the human genome dataset, as shown in Fig. 1–C, to evaluate the ability of various state-of-the-art generative models to generate realistic fake genomes while minimizing privacy concerns (i.e., reducing overfitting). Recent studies (Yelmen et al., 2021; 2023) have thoroughly investigated this for a variety of generative models. Both studies concluded that the RBM was the most accurate method for generating high-quality and private synthetic genomes. The comparison between models relies primarily on the Nearest Neighbor Adversarial Accuracy (AA_{TS}) and privacy loss indicators, introduced in Ref. (Yale et al., 2020), which quantify the similarity and the level of "privacy" of the data generated by a model w.r.t. the training set. We have $AA_{\text{TS}} = \frac{1}{2}\left(AA_{\text{True}} + AA_{\text{Synth}}\right)$ where AA_{True} [resp. AA_{Synth}] are two quantities in $[0, 1]$ obtained by merging two sets of real and synthetic data of equal size N_s and measuring respectively the frequency that a real [rep. synthetic] has a synthetic [resp. real] as nearest neighbor. If the generated samples are statistically indistinguishable from real samples, both frequencies AA_{True} and AA_{Synth} should converge to 0.5 at large N_{s}. AA_{TS} can be evaluated both with train or test samples and the privacy loss indicator is defined as $\text{Privacy loss} = AA_{\text{TS}}^{\text{test}} - AA_{\text{TS}}^{\text{train}}$ and is expected to be strictly positive. Fig. 14 shows the comparison of AA_{TS} and privacy loss values obtained with our two models, demonstrating that the pre-trained RBM clearly outperforms the

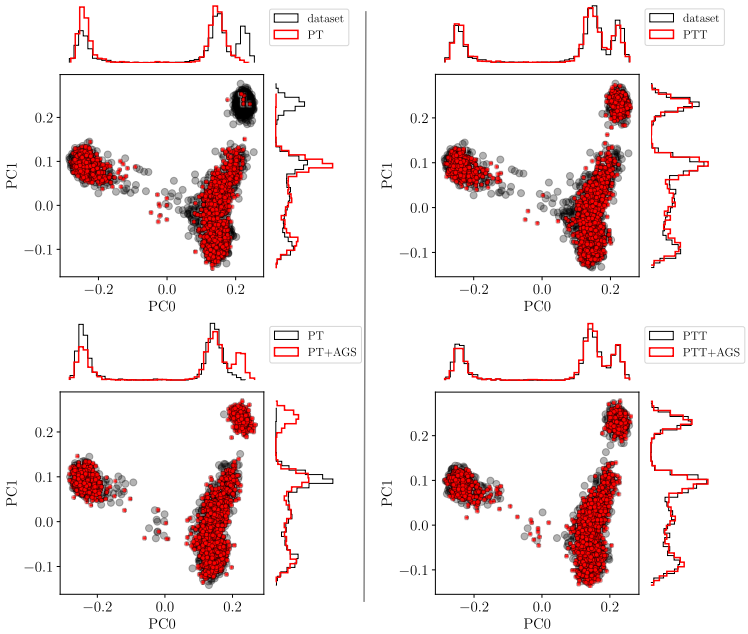

Figure 12: Comparison of the sampling results of PT (left) and PTT (right). The top row corresponds to sample generated with each methods compared with the dataset distribution. The bottom row compares samples obtained by performing 10^6 AGS starting from the PT and PTT samples with their starting position.

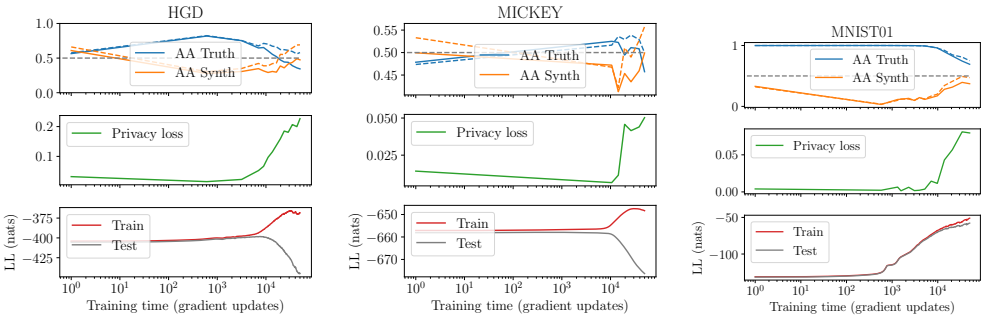

Figure 13: Evolution of the AATS (first row) and privacy loss (second row) indicator during the training of pretrained RBMs. The online estimation of the train and test LL is given at the third row. The overfitting is clearly detected in the HGD and MICKEY dataset and the divergence between the train and test LL coincides with a sharp rise in the privacy loss.

other model, and even achieves better results (AA_{TS} values much closer to 0.5) than those discussed in (Yelmen et al., 2021; 2023).

J EXTRA FIGURES

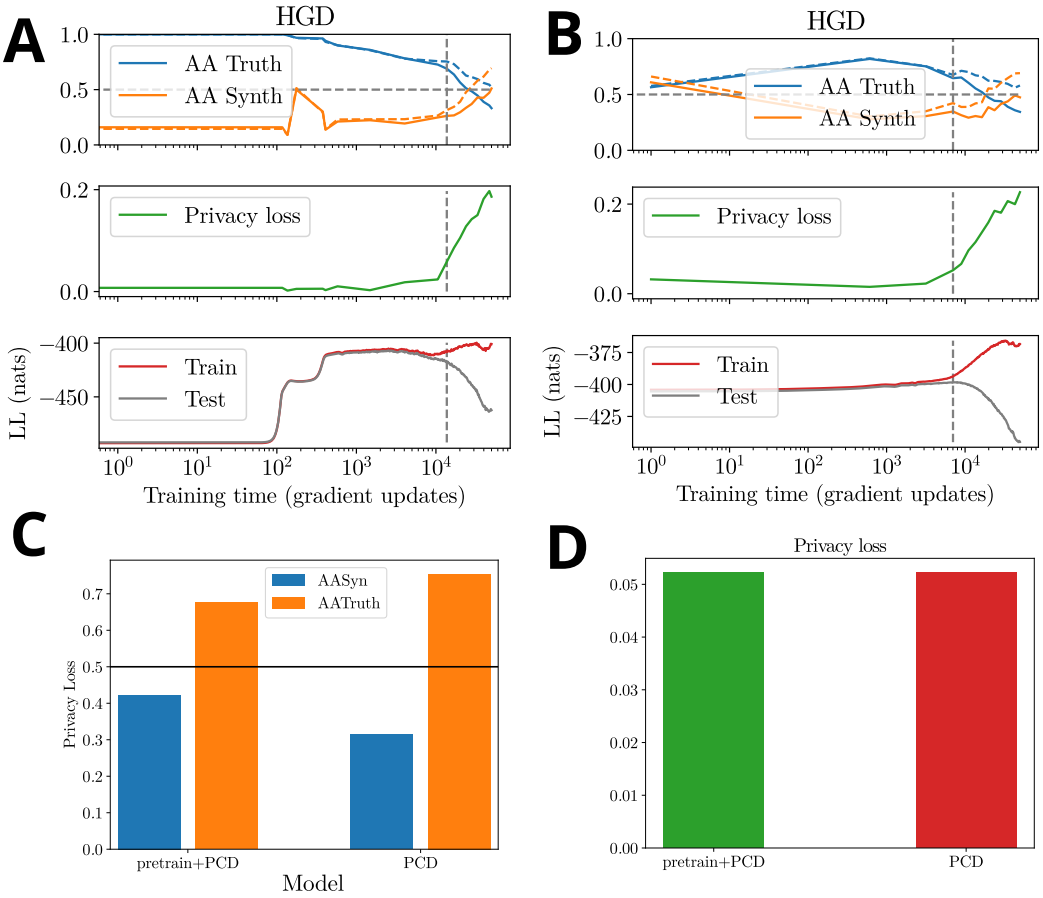

Figure 14: Comparison of the evolution of overfitting metrics along the log-likelihood between a RBM trained from scratch using PCD (**A**) and a pretrained RBM (**B**). (**C**) shows the comparison between the AASyn and AATruth between the two machines. The machines were selected in order to have a comparable privacy loss (**D**), right before starting to overfit.

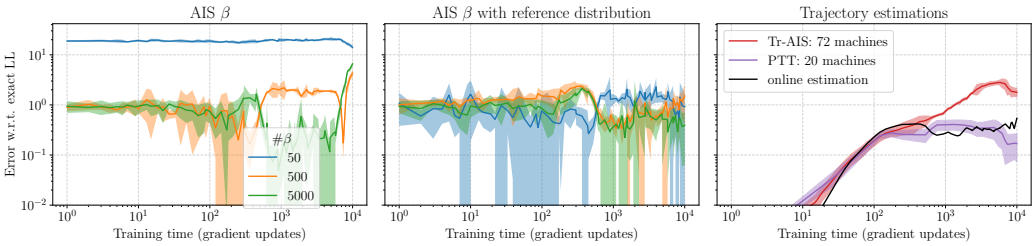

Figure 15: Comparison of log-likelihood estimation w.r.t. the exact ll on the MNIST-01 dataset.

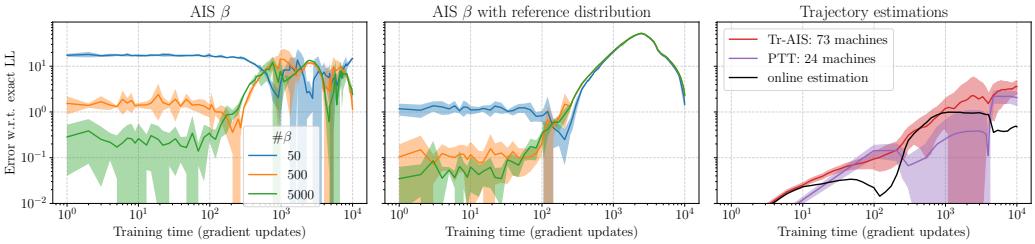

Figure 16: Comparison of log-likelihood estimation w.r.t. the exact ll on the MNIST dataset.

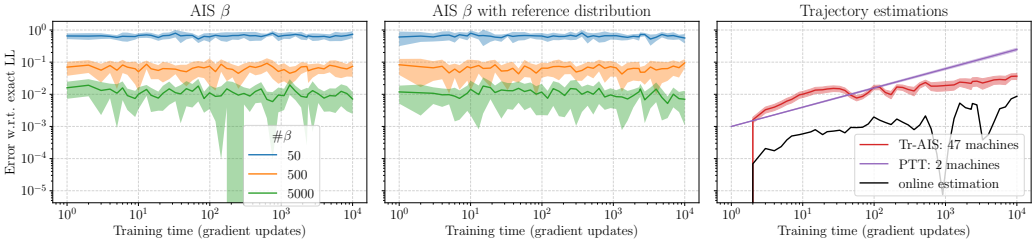

Figure 17: Comparison of log-likelihood estimation w.r.t. the exact ll on the Ising dataset.

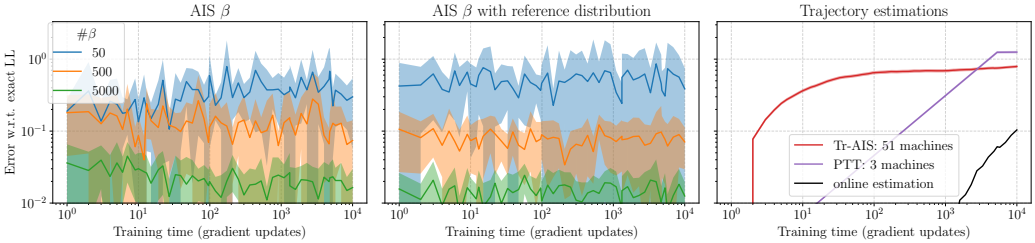

Figure 18: Comparison of log-likelihood estimation w.r.t. the exact ll on the MICKEY dataset.

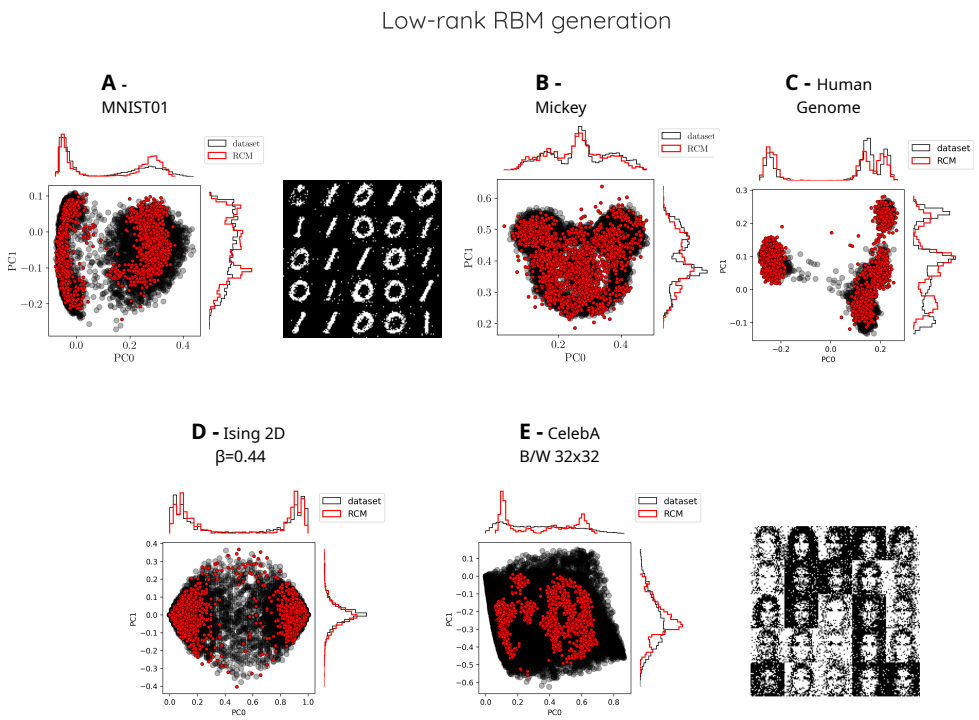

Figure 19: Samples generated with the low-rank RBM trained with 3 directions plus a bias. We show the samples generated by the pre-trained machined for the 5 datasets discussed in the paper and projected on the PCA of the dataset. For the image datasets, we also show the images generated.

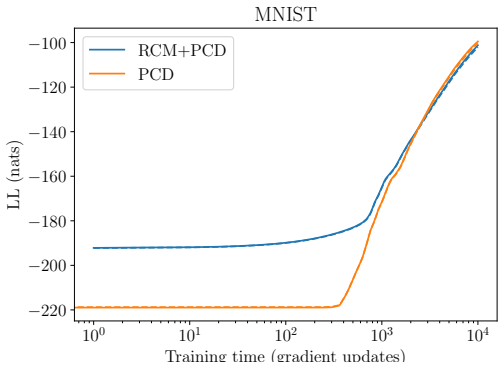

Figure 20: We show the evolution of the tr-AIS log-likelihood in the training of the entire MNIST dataset (containing digits from 0 to 9) with and without the pre-training. The pre-training does not suppose a particular advantage but neither is a disadvantage.