Fast and accurate training and sampling of Restricted Boltzmann Machines

Anonymous Author(s)
Affiliation
Address
email

Abstract

Thanks to their simple architecture, Restricted Boltzmann Machines (RBMs) are powerful tools for modeling complex systems and extracting interpretable insights from data. However, training RBMs, as other energy-based models, on highly structured data poses a major challenge, as effective training relies on mixing the Markov chain Monte Carlo simulations used to estimate the gradient. This process is often hindered by multiple second-order phase transitions and the associated critical slowdown. In this paper, we present an innovative method in which the principal directions of the dataset are integrated into a low-rank RBM through a convex optimization procedure. This approach enables efficient sampling of the equilibrium measure via a static Monte Carlo process. By starting the standard training process with a model that already accurately represents the main modes of the data, we bypass the initial phase transitions. Our results show that this strategy successfully trains RBMs to capture the full diversity of data in datasets where previous methods fail. Furthermore, we use the training trajectories to propose a new sampling method, *parallel trajectory tempering*, which allows us to sample the equilibrium measure of the trained model much faster than previous optimized MCMC approaches and a better estimation of the log-likelihood. We illustrate the success of the training method on several highly structured datasets.

1 Introduction

Energy-based models (EBMs) are a classic approach to generative modeling that has been studied for decades. They were introduced using the Restricted Boltzmann Machine formulation by Smolensky [1] and later further developed by Sejnowski et al. [2]. They provide a straightforward method for modeling effective interactions within complex data distributions and for sufficiently simple energy functions, such as the Boltzmann machine (BM) [3], it is also possible to interpret and infer the underlying constituent rules from the observed data. This inference strategy is often associated with the inverse Ising problem and pairwise interaction models [4], and it has found a great variety of applications in fields such as neuroscience [5] or computational biology [6]. A recent work has proposed replacing the use of pairwise models with the Restricted Boltzmann Machine (RBM) [7], as it allows the same direct interpretation of its energy function as an explicit many-body interaction model while greatly extending the expressive power of the model. RBMs are also very useful for grouping data into hierarchical families [8]. On the diametrically opposite side (on interpretability) are generative ConvNets [9, 10], where the energy function is formulated as a deep neural network, which are capable of synthesizing photorealistic images but are almost impossible to interpret as a physical model.

The applications of simple EBMs in science are very diverse. For example, they are often used today to encode the Hamiltonian of physical many-body systems, such as Quantum wave functions [11] or the accurate determination of ground state wave functions of strongly interacting and entangled

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

quantum spins [12] or they have proven to be suitable for the representation of the AdS/CFT correspondence in theories of quantum gravity [13, 14]. Simple EBMs are also very common to encode the evolutionary constraints in protein families [6, 15], and to predict mutations [16], or to generate realistic synthetic sequences, such as fake human genomes [17, 18]. These examples show that, despite their somewhat old-fashioned architecture, shallow EBMs are increasingly seen as useful tools for better understanding modern physics/biology, as they allow for a certain level of analytical description.

Despite the appealing modeling properties of RBMs, they are notoriously difficult to train, a challenge common to EBMs in general. The main difficulty arises from the computation of the log-likelihood gradient, which requires an ergodic exploration of a dynamically evolving and potentially complex free energy landscape using Markov Chain Monte Carlo (MCMC) processes. Recent studies have shown that models trained with non-convergent MCMC processes suffer from out-of-equilibrium dynamic memory effects [19, 20, 21]. This dynamical behavior can be explained analytically using moment-matching arguments [19, 22]. While exploiting these effects can yield fast and accurate generative models, even for highly structured data [23] or high-quality images with RBMs [24], this approach results in a sharp separation between the model's Gibbs-Boltzmann distribution and the dataset distribution, thereby undermining the interpretability of the model parameters [22, 7]. Thus, to extract meaningful information from datasets using RBMs, it is essential to ensure proper mixing of the chains during training, in short, one needs *equilibrium models*.

Both the ability to train an RBM in equilibrium and to generate convincing new samples from its equilibrium measure strongly depend on the dataset in question. For typical image datasets such as MNIST or CIFAR-10, good RBMs can be obtained by increasing the number of MCMC steps. However, this approach is no longer feasible for highly structured datasets [25]. Datasets from which one seeks scientific insights are often highly structured, such as genomics/proteomics data or low-temperature many-body physical systems. These datasets typically exhibit distinct clusters, identifiable via principal component analysis (PCA) which form distant groups of similar entries. We show an example of the PCA of 4 clustered dataset we will be studying in this work in Fig. 1; details about these datasets are given in the caption and in the Supplemental Information (SI). During training, the model must evolve from an initial normal distribution to an increasingly multimodal distribution. Sampling from multimodal distributions is particularly challenging because the mixing times are determined by the transition times between modes. But this is not the only difficulty. These distant modes are encoded by second-order phase transitions during training [26, 27, 28], leading to diverging mixing times in these regions — a phenomenon known as *critical slowdown* —, which means that mixing times are expected to grow with a power of their system size. This sampling challenge not only hinders the training process, but also limits the model's ability to generate new samples. Obtaining new and independent configurations would require an impractically large number of sampling steps.

2 Related work

Training EBMs by maximizing log-likelihood has long been a challenge in the community [29, 10]. EBMs gained popularity with the introduction of the contrastive divergence algorithm [30], 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 [31, 32, 21]. An improvement is the persistent contrastive divergence (PCD) algorithm [33], 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 [25]. 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 [25]. 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. Other optimized MCMC methods, such as the Parallel Tempering (PT) [34] algorithm, simulate multiple models at different temperatures, facilitating mixing through temperature exchange [35, 32]. However, PT is costly and

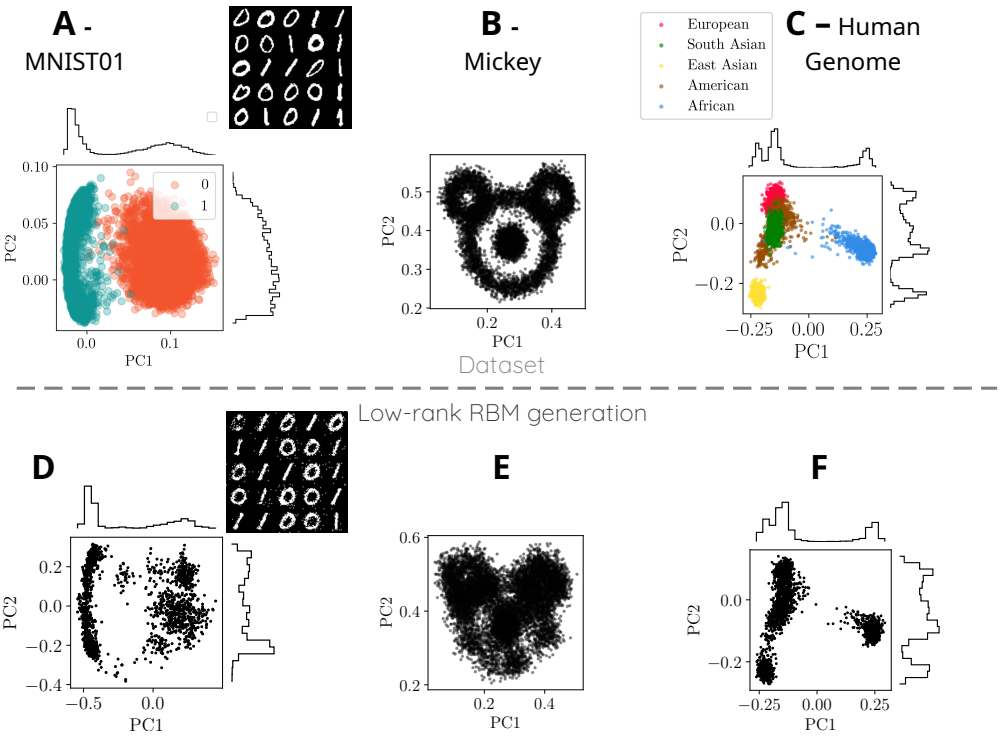

Figure 1: **Clustered datasets.** In A-C we show the 4 different clustered data sets that we will consider in this paper, projected onto their first two PCA components. In A we show the data of the MNIST 01 dataset (both projected and some instances), which contains only the 0-1 images of the complete MNIST dataset. In B, we show the Mickey dataset, an artificial dataset whose PCA forms a "Mickey"" face shape. In C, we show data from the Human Genome Dataset (HGD), which contains binary vectors each corresponding to a human individual and whose sites correspond to selected genes. A value of 1 at a particular position means that a mutation was observed there compared to an individual reference sequence. Details of these data sets can be found in the SI. In D-F we show the samples we generate with the low-rank RBMs that are used as initial point of a standard training.

often ineffective, especially because EBMs undergo first-order phase transitions at the temperature where PT typically fails because one needs too many temperatures to make the moves accepted. We will see below that a more appropriate approach exchanges the models at different training times, which only implies crossing second-order phase transitions.

The population annealing algorithm, which reweights parallel chains during learning based on their relative weight changes during parameter updates, was proposed as an alternative [36]. Similarly, reweighting chains using non-equilibrium physics concepts such as the Jarzynski equality has been proposed [37]. Both approaches struggle with highly structured data sets. To prevent the different chains to get too correlated around the training phase transitions, one must either increase the number of sampling steps or decrease the learning rate, which in practice means very long training processes to ensure a proper equilibrium training. Another strategy is to use EBMs as corrections for straightforward-to-sample flow-based models [38]. This simplifies sampling and learning, but sacrifices the interpretability of the energy function, which was our goal. An evolving flow model can be used as a fast sampling moves proposer for the EBM [39] objective. This method requires the training of two different networks in parallel and may result in the drop of the move acceptancy as the EBM becomes specialized.

For RBMs, a recent method called "stacked tempering" [40] dramatically speeds up sampling by training smaller RBMs with latent variables from previous models, allowing fast updates to be proposed using a PT like algorithm. Authors also showed that this algorithm was much faster than the standard PT. While effective, it is too cumbersome for use in training. Also for RBMs, it has

recently been shown that it is 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 [41] and the discussion below). 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.

In this paper, we will show how to drastically reduce training times by starting the RBM training process at this low-rank RBM, as this means that the first and strongest dynamic effects associated with them are directly bypassed. We also show that one can exploit the training trajectory to develop an effective sampling method, the *parallel trajectory tempering* (PTT) that outperforms the "stacked tempering" [40] and only requires saving a reduced number of models during the training. This strategy also allows to obtain reliable estimations for the log-likelihood in well-trained models, much better than those obtained with the standard Annealing Important Sampling (AIS) techniques [42]. Using both strategies, we show that we are able to train and evaluate methods that accurately represent the different modes in the dataset, where standard methods lead to mode collapse effects.

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 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

$$\frac{\partial \mathcal{L}}{\partial w_{ia}} = \langle v_i h_a\rangle_{\mathcal{D}} - \langle v_i h_a\rangle_{\rm RBM},\ \frac{\partial \mathcal{L}}{\partial \theta_i} = \langle v_i\rangle_{\mathcal{D}} - \langle v_i\rangle_{\rm RBM},\ \frac{\partial \mathcal{L}}{\partial \eta_a} = \langle h_a\rangle_{\mathcal{D}} - \langle h_a\rangle_{\rm RBM}, \quad (2)$$

where $\langle\cdot\rangle_{\mathcal{D}}$ denotes the average with respect to the entries in the dataset, and $\langle\cdot\rangle_{\rm RBM}$ with respect to $p(\boldsymbol{v},\boldsymbol{h})$. Since Z is intractable, the model averages in the gradient are typically estimated using $N_{\rm s}$ independent MCMC processes, and observable averages $\langle o(\boldsymbol{v},\boldsymbol{h})\rangle_{\rm RBM}$ 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 [21].

Typical MCMC mixing times in RBMs are very small at the beginning of the training and grow as it progresses [21], suffering with sharp increases every-time the training trajectory crosses each of the critical transitions that give birth to new modes [28]. 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 as PCD. In this scheme, the process of training can be mimicked to a slow cooling process, only that instead of varying a single parameter, the temperature, a whole set of parameters $\boldsymbol{\Theta}_t = (\boldsymbol{w}_t,\boldsymbol{\theta}_t,\boldsymbol{\eta}_t)$ are 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 Eq. (2) estimated using the configurations in $\boldsymbol{P}_t$, and γ being the learning rate.

4 The low-rank RBM pretrained

In Ref. [41], it was shown that it is possible to train exactly (i.e. by direct numerical integration instead of MCMC sampling) 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. In other words, it is possible to train a RBM with a coupling matrix of this simplified form

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

and where 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, it is possible to write $p(\boldsymbol{v})$ only as a function of d order parameters given by the *magnetizations* along each of the u_α components, $m_\alpha(\boldsymbol{v}) = \boldsymbol{u}_\alpha \cdot \boldsymbol{v}/\sqrt{N_\text{v}}$, and in particular,

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

where $\boldsymbol{m} = (m_1, \ldots, m_\alpha)$. As proposed in [41], the optimal parameters of such a model can basically be determined by solving a regression problem. We describe this method in details in the SI. This means that once the model is trained, we obtain a probability $p(\boldsymbol{m})$ defined on a much lower dimension than the original $p(\boldsymbol{v})$. Such a probability can be straightforwardly sampled using *inverse transform sampling*. Since this method requires a discretization of the $\boldsymbol{m}$-space both for training and generation, we cannot consider intrinsic space dimension $d > 4$ dimensions in practice. These low-rank RBMs are then trained to reproduce the statistics of the dataset projected in its first d principal components. Despite their simplicity, the low-rank models are already able to generate an approximate version of the dataset, as shown in Fig. 1–D-F for the 4 datasets previously presented.

In the initial stage of the standard learning process, the model encodes the strongest PCA components of the data through multiple critical transitions [26, 27, 28]. 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. Once the main directions are incorporated, training can efficiently continue with standard algorithms like PCD, as the mixing times of pre-trained machines tend to be much shorter. In particular, in the PCD-100 training with MNIST01, relaxation times for the visible variables' time correlation reach $5 \cdot 10^5$ MCMC steps at the first three transitions, coinciding with the growth of singular values in the model weight matrix W. In contrast, the pre-trained machine has a much shorter relaxation time of $\sim 10^3$, allowing us to safely restart the PCD process from a set of equilibrium samples generated by static sampling of the low-rank RBM.

Overcoming these transitions has dramatic implications for the quality of the models we can train and how accurately they reproduce the statistics of the data. In Fig. 2, we show for 3 datasets the equilibrium samples drawn from 3 RBMs trained with identical number of samples, minibatch size, $k = 100$ Gibbs steps, and learning rate $\gamma = 0.01$, but different training strategies. In particular, we consider 2 RBMs trained from scratch with the standard PCD [33] and the recently proposed Jarzynski reweighing method [23] (see SI for our specific implementation in the RBM), and a final machine trained with PCD and pre-trained with a low-rank RBM. In all cases, the quality of the generated samples is significantly better when pre-training is used. For the Mickey dataset, neither JarRBM nor normal PCD are able to generate convincing data. For the MNIST01 dataset, all 3 methods are able to generate convincing data, but only Pretrain+PCD is able to correctly balance all modes, as can be seen in Fig. 3, where we compare the histograms of the generated data projected onto the first 3 PCA directions with those of the dataset and a random selection of the generated samples. We see that the pre-training+PCD training perfectly balances the different modes (here we show the first 3 directions, but it goes much further), unlike the other 2 methods, and also generates more diverse images. We can also compare the log-likelihood of all 3 models and find that the pre-trained RBM achieves higher values. At this point, it is important to emphasize that in order to properly quantify the increase in log-likelihood, we need to use the PTT algorithm (see section 5) to correctly thermalize in these well-trained machines. For comparison, we show our PTT measure in dark and solid lines, while the standard AIS [42] estimate is shown in light dashed lines.

Already from the scatter plots we see that the pre-training has a dramatic effect in obtaining models where all modes are properly balanced, but also has important effects in the maximum test-likelihood

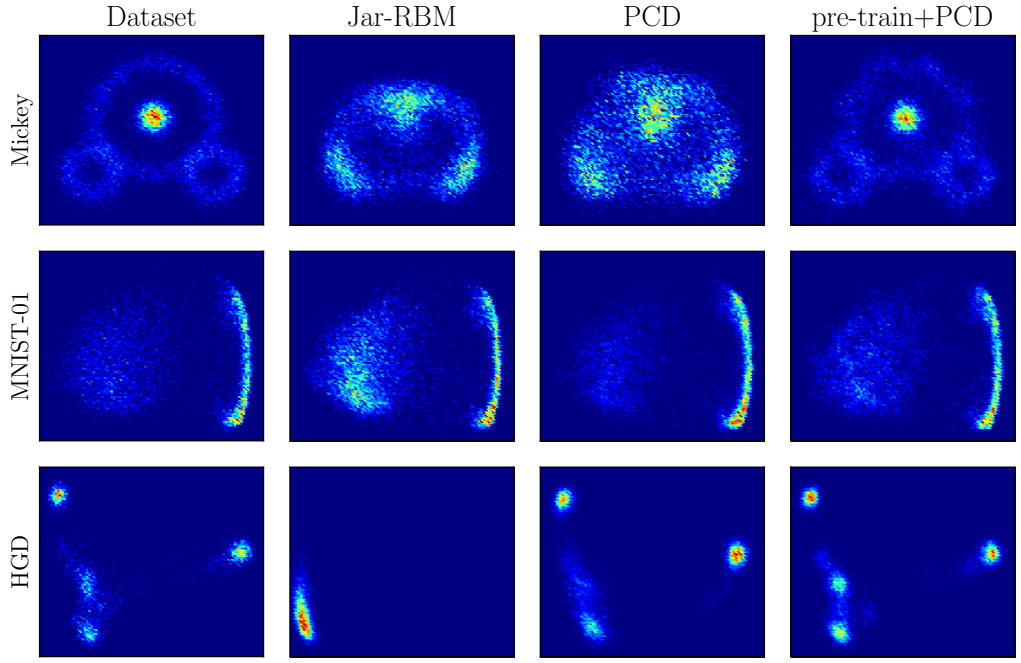

Figure 2: We compare the equilibrium samples generated by RBMs trained on the Mickey, MNIST01, and HGD datasets using three different training schemes: Jarzynski (JarRBM), PCD, and PCD initialized on low-rank RBMs (used to generate the samples in Fig. 1–D-F). To assess the fitting of the modes, we show a density plot of the projections of the data in the first two principal directions of each dataset. We compare these results with the density plot of the original datasets in the first column.

we can achieve. In all cases, these equilibrium samples are drawn using the trajectory PT algorithm that will be explained in the next section, and the log-likelihood obtained using the equilibrium configurations obtained at different epochs as a result of the trajectory PT flow.

5 Standard Gibbs sampling vs. Parallel Trajectory Tempering (PTT)

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) [42], making them inaccurate and meaningless.

To illustrate this problem, let us consider the MNIST01 and the HGD datasets. MNIST01 dataset is bimodal and the HGD highly multimodal as shown in their PCA in Figs. 1–A and C. Let us consider that we want to sample the equilibrium measure of the RBMs trained using low-rank RBM pretraining. In order to draw new samples from these models, one would typically run MCMC processes from random initialization and iterate them until convergence. The mixing time is controlled by the jumping time between clusters. To accurately estimate the relative weight between modes, the MCMC processes must be ergodic, requiring many back-and-forth jumps. However, as shown in Figs. 4–A and C for the MNIST01 and HGD datasets, Gibbs sampling dynamics are extremely slow, rarely producing jumps even after 10^4 MCMC steps. The yellow curves in Figs. 4–B and D show the mean number of jumps over 100 independent chains as a function of MCMC steps, indicating that a proper equilibrium generation would require at least $10^6 - 10^7$ MCMC steps.

One effective way to accelerate the dynamics is to exploit the training trajectory, where the model progressively specializes through second-order phase transitions. To achieve this, we save RBMs trained at various epochs and propose swaps between configurations of similarly trained models. We

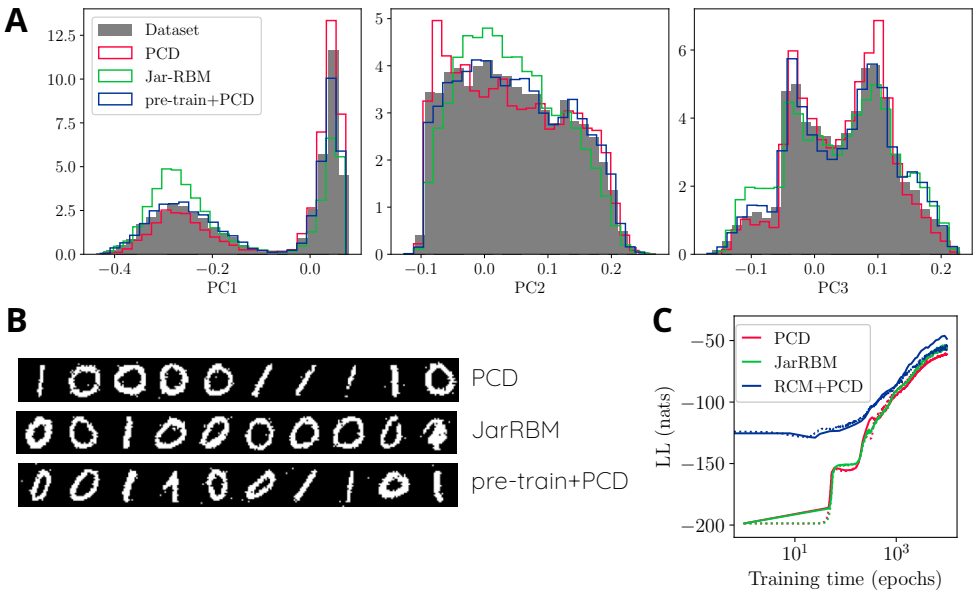

Figure 3: We compare the samples generated by the 3 RBMs (JarRBM, PCD, pretrain+PCD) trained with MNIST01 data. In A, we show the histograms of the generated data projected on the first, second and third principal directions with those of the dataset. We see that only the pretrain+PCD correctly balances the different modes. In B we show 10 images generated by each machine. In C, we compare the log-likelihood of each model's dataset as a function of training time. The dark and full curves were obtained using the PTT algorithm discussed in section 5, and the lighter and dashed curves using the AIS method [42].

call this the *Parallel Trajectory Tempering* (PTT) algorithm. Unlike the standard Parallel Tempering (PT) algorithm, which attempts swaps configurations between different temperatures, the PTT swaps between model parameters with different degrees of specialization. This approach is more natural for this problem because it involves crossing only second-order transitions, unlike the first-order transitions occurring in temperature annealing. And in fact, we show in Figs. 4–A and C, that this approach allows us to sharply accelerate the dynamics, as opposed to the standard PT algorithm (studied in detail for the MNIST dataset in [40]).

In the PTT algorithm, the configurations $\boldsymbol{x} = (\boldsymbol{v}, \boldsymbol{h})$ of neighboring machines indexed by t and $t-1$ are interchanged with the probability

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

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. As "nonspecialized" models mix very quickly, either because the distribution is essentially Gaussian at the initialisation of a standard training, or because the low-rank RBM can be sampled with a static Monte Carlo process (yielding independent configurations each time), the trajectory flow significantly accelerates convergence to equilibrium. The time interval between successive machines is selected in such a way that the probability of accepting interchanges between neighboring machines remains around 0.3. Pre-trained machines require a significant fewer number of models to be effective, because most selected models are positioned at the most prominent phase transitions. We give the number of machines used for each sampling process in the SI. We also provide there a specific and detailed description of the algorithm used.

In the red curves in Fig. (4)–B and D, we show the number of jumps between clusters as a function of the number of elementary MCMC steps, which in the PTT scheme refer to 1 Gibbs sampling step + one swap proposal. For the DNA dataset, we have two measures corresponding to jumps along the two principal component directions. We observe at 10^4 MCMC steps an increase of the number of jumps by a factor of 80 for MNIST01 and by a factor of 1350 for the HGD in this machine, although

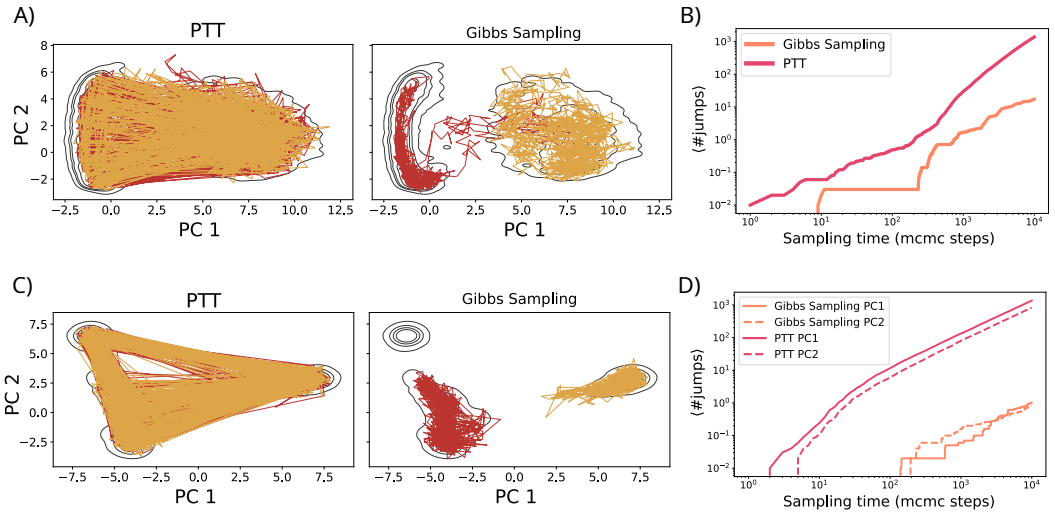

Figure 4: Comparison between PTT and classical Gibbs sampling for the MNIST01 dataset (A and B, respectively) and the human genome dataset (C and D, respectively). In A and C, we show the trajectory of two independent chains (red and orange) projected onto the PCA along the sampling process of the pretraining+PCD model for 10^4 MCMC steps. The black contour represents the density profile of the dataset and the position of the chains is plotted every 10 steps. In B and D we show the average number of jumps from one cluster to another as a function of the MCMC steps performed. The average is calculated over a population of 100 chains. In D, we show the average jump time between clusters along the first (solid line) and second (dashed line) principal components of the data.

we achieve higher factors in other machines, as we show in the SI. The sampling of RBMs training on the MNIST01 dataset was the subject of the study of the "stacked tempering" algorithm in [40]. If we compare the numbers with their work, we see that we achieve a 3-4 times higher speedup factor, where our model has the advantage that it does not need additional training, but simply uses the stored machines correctly.

Another desirable advantage of our PTT algorithm is that we can easily use it to compute an improved estimate of the AIS log-likelihood, except that in our case we consider the training trajectory instead of a cooling process and use the equilibrium samples obtained for each of the models to compute the model averages. In Figs. 3–C 5–A we compare the log-likelihood estimates obtained with our method (AIS-PTT) in full and dark lines and in light and dashed lines the AIS estimate (AIS). We see that both measures coincide for most parts of the training and that they split when the sampling becomes too long to thermalize along the temperature annealing curve in AIS. This effect is particularly evident for the JarRBM run in 5–A, where AIS takes a long time to recognize that the model suffers from a strong mode-collapse effect.

6 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 do not include the Jarzysnki method here, as this method fails to obtain a reliable model as clearly shown in the evolution of the Log-likelihood in Fig. 5. 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 [17, 18] 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. [43], which quantify the similarity and the level of "privacy" of the data generated by a model w.r.t. the training set. We have $AA_{\mathrm{TS}} = \frac{1}{2}\left(AA_{\mathrm{True}} + AA_{\mathrm{Synth}}\right)$ where AA_{True} [resp. AA_{Synth}] are two

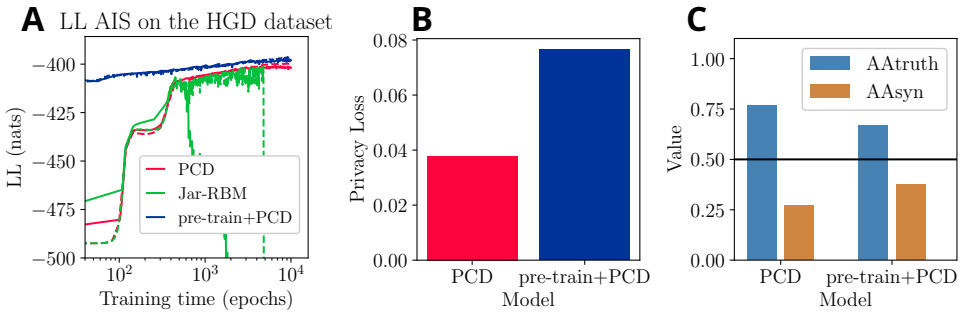

Figure 5: We compare the quality of the RBMs trained with the human genome data (HGD). In A, we show the log-likelihood as a function of the training epochs for the 3 training procedures. Solid lines correspond to AIS-PTT and dashed lines to AIS. The JarRBM falls down because the training breaks eventually. In B and C we compare privacy and overfitting based on the $AA_{\rm TS}$ indicator.

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 $AA_{\rm True}$ and $AA_{\rm Synth}$ should converge to 0.5 at large $N_{\rm s}$. $AA_{\rm TS}$ can be evaluated both with train or test samples and the privacy loss indicator is defined as $\text{Privacy loss} = AA^{\rm test}_{\rm TS} - AA^{\rm train}_{\rm TS}$ and is expected to be strictly positive. Fig. 5 shows the comparison of $AA_{\rm TS}$ and privacy loss values obtained with our two models, demonstrating that the pre-trained RBM clearly outperforms the other model, and even achieves better results ($AA_{\rm TS}$ values much closer to 0.5) than those discussed in [17, 18].

7 Conclusions

We have shown that the strategy of initiating the training on a pre-trained low-rank RBM is an extremely effective strategy to obtain high quality models for structured datasets that accurately represent all the modes in the datasets and with significantly higher log-likelihoods. We have also shown that the models obtained in that way are: (i) better generative models than those obtained with standard trainings, both, in the sense that they over-fit less at the same time they are more indistinguishable from the test samples, (ii) they display faster relaxational dynamics.

We have also proposed a new fast sampling method that exploits the progressive learning of features in the training of RBMs to design an efficient trajectory PT strategy that allows accelerating the parallel Gibbs sampling dynamics by many orders of magnitude and overcome the performance of recent efficient sampling methods without adding any extra cost than saving models during the training.

Both strategies for training and sampling are very general, and could be generalized to more complex EBMs. In this sense, the low-rank RBM model could be used as a more efficient pre-initialisation in deeper structures, and the trajectory PT algorithm is suitable to be directly used in any EBM no matter how complex it is.

8 Code availability

The code and datasets are available at `https://github.com/nbereux/fast-RBM`.

References

[1] 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.

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

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

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Yingnian Wu. A theory of generative convnet. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2635–2644, New York, New York, USA, 20–22 Jun 2016. PMLR.

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

[11] Giuseppe Carleo and Matthias Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606, 2017.

[12] Roger G Melko, Giuseppe Carleo, Juan Carrasquilla, and J Ignacio Cirac. Restricted boltzmann machines in quantum physics. *Nature Physics*, 15(9):887–892, 2019.

[13] Koji Hashimoto, Sotaro Sugishita, Akinori Tanaka, and Akio Tomiya. Deep learning and the ads/cft correspondence. *Physical Review D*, 98(4):046019, 2018.

[14] Koji Hashimoto. Ads/cft correspondence as a deep boltzmann machine. *Physical Review D*, 99(10):106017, 2019.

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

[16] Juan Rodriguez-Rivas, Giancarlo Croce, Maureen Muscat, and Martin Weigt. Epistatic models predict mutable sites in sars-cov-2 proteins and epitopes. *Proceedings of the National Academy of Sciences*, 119(4):e2113118119, 2022.

[17] 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.

[18] 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.

[19] 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.

[20] 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, pages 5272–5280, 2020.

[21] 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.

[22] 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.

[23] 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.

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

[25] 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.

[26] A. Decelle, G. Fissore, and C. Furtlehner. Spectral dynamics of learning in restricted boltzmann machines. *Europhysics Letters*, 119(6):60001, nov 2017.

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

[28] A. Anonymous. Cascade of phase transitions in the training of energy-based models. *arXiv:*, 2024.

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

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

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

[32] 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*, pages 145–152. JMLR Workshop and Conference Proceedings, 2010.

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

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

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

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

[37] 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.

[38] 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.

[39] 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.

[40] 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.

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

[42] Oswin Krause, Asja Fischer, and Christian Igel. Algorithms for estimating the partition function of restricted boltzmann machines. *Artificial Intelligence*, 278:103195, 2020.

[43] 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.

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:

$$\boldsymbol{w} = \sum_{\alpha=1}^{d} w_\alpha \bar{\boldsymbol{u}}_\alpha \boldsymbol{u}_\alpha^\top, \qquad \text{with} \qquad (\boldsymbol{u}_\alpha, \bar{\boldsymbol{u}}_\alpha) \in \mathbb{R}^{N_v} \times \mathbb{R}^{N_h}, \tag{5}$$

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 $\boldsymbol{m} \equiv (m_1, \dots, m_d)$

$$\mathcal{H}(\boldsymbol{v}) = -\sum_a \log\cosh\left(\sqrt{N_\mathrm{v}}\bar{u}_a \sum_{\alpha=1}^{d} w_\alpha m_\alpha + \eta_a\right) = \mathcal{H}(\boldsymbol{m}(\boldsymbol{v})). \tag{6}$$

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

$$p_{\mathrm{emp}}(\boldsymbol{m}) \equiv \sum_{\boldsymbol{v}} p_{\mathrm{emp}}(\boldsymbol{v}) \prod_{\alpha=1}^{d} \delta\left(m_\alpha - \frac{1}{\sqrt{N_\mathrm{v}}} \boldsymbol{u}_\alpha^T \boldsymbol{v}\right), \tag{7}$$

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

$$p(\boldsymbol{m}) = \frac{1}{Z}\sum_{\boldsymbol{v}} e^{-\mathcal{H}(\boldsymbol{v})} \prod_{\alpha=1}^{d} \delta\left(m_\alpha - \frac{1}{\sqrt{N_\mathrm{v}}}\boldsymbol{u}_\alpha^T \boldsymbol{v}\right) \tag{8}$$

$$= \frac{1}{Z}\mathcal{N}(\boldsymbol{m}) \exp \sum_a \log\cosh\left(\bar{u}_a \sum_{\alpha=1}^{d} w_\alpha m_\alpha + \eta_a\right) \tag{9}$$

$$= \frac{1}{Z} e^{-\mathcal{H}(\boldsymbol{m}) + N_\mathrm{v} s(\boldsymbol{m})} = \frac{1}{Z} e^{-N_\mathrm{v} f(\boldsymbol{m})} \tag{10}$$

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:

$$p_{\mathrm{prior}}(\boldsymbol{m}) = \frac{e^{N_\mathrm{v} s(\boldsymbol{m})}}{2^{N_\mathrm{v}}} \approx \exp\left(-N_\mathrm{v}\mathcal{I}(\boldsymbol{m})\right), \tag{11}$$

with the rate function

$$\mathcal{I}(\boldsymbol{m}) = \sup_{\boldsymbol{\mu}} \left[\boldsymbol{m}^T\boldsymbol{\mu} - \phi(\boldsymbol{\mu})\right] = \boldsymbol{m}^T\boldsymbol{\mu}^* - \phi(\boldsymbol{\mu}^*), \tag{12}$$

and

$$\phi(\boldsymbol{\mu}) = \lim_{N_\mathrm{v}\to\infty} \frac{1}{N_\mathrm{v}} \log\left\langle e^{N_\mathrm{v}\boldsymbol{m}^T\boldsymbol{\mu}}\right\rangle = \lim_{N_\mathrm{v}\to\infty} \frac{1}{N_\mathrm{v}} \log \frac{1}{2^{N_\mathrm{v}}} \sum_{\boldsymbol{v}} e^{\sqrt{N_\mathrm{v}}\sum_{\alpha=1}^{d}\mu_\alpha \sum_i u_{\alpha,i} v_i} \tag{13}$$

$$= \lim_{N_\mathrm{v}\to\infty} \frac{1}{N_\mathrm{v}} \sum_{i=1}^{N_\mathrm{v}} \log\cosh\left(\sqrt{N_\mathrm{v}} \sum_{\alpha=1}^{d} \mu_\alpha u_{\alpha,i}\right). \tag{14}$$

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:

$$m_\alpha = \frac{1}{\sqrt{N_\mathrm{v}}} \sum_{i=1}^{N_\mathrm{v}} u_i^\alpha \tanh\left(\sqrt{N_\mathrm{v}} \sum_{\beta=1}^{d} u_i^\beta \mu_\beta^*\right). \tag{15}$$

It is then straightforward to check that spins distributed as

$$p_{\mathrm{prior}}(\boldsymbol{v}|\boldsymbol{m}) \propto e^{N_\mathrm{v}\boldsymbol{\mu}^{*T}\boldsymbol{m}(\boldsymbol{v})} \tag{16}$$

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

$$p_{\mathrm{prior}}(v_i = 1|\boldsymbol{m}) = \mathrm{sigmoid}\left(2\sqrt{N_\mathrm{v}} \sum_{\alpha=1}^{d} u_{\alpha,i}\mu_\alpha^*(\boldsymbol{m})\right) \tag{17}$$

The training can therefore be done directly in the subspace of dimension d. In Ref. [41], 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 [41], 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. We will briefly outline the main steps to train the RCM. A more detailed explanation can be found in Appendix A.2.

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

$$\mathcal{H}(\boldsymbol{v}) = -\sum_i v_i \theta_i - \sum_a \log\cosh\left(\sum_i w_{ia} v_i + \eta_a\right). \tag{18}$$

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

$$m_\alpha \coloneqq \frac{1}{\sqrt{N_v}} \sum_{i=1}^{N_v} s_i u_{i\alpha}, \quad w_{\alpha a} \coloneqq \sum_{i=1}^{N_v} w_{ia} u_{i\alpha}, \quad \theta_\alpha \coloneqq \frac{1}{\sqrt{N_v}} \sum_{i=1}^{N_v} \theta_i u_{i\alpha} \tag{19}$$

with $\alpha \in \{1, \dots, d\}$ and $\boldsymbol{v}$ the projection matrix of the PCA. The projected distribution of the model is then given by

$$p_{\text{RBM}}(\boldsymbol{m}) = \frac{\exp\left(N_v \left[\mathcal{S}(\boldsymbol{m}) + \sum_{\alpha=1}^d \theta_\alpha m_\alpha + \frac{1}{N_v} \sum_{a=1}^{N_h} \log\cosh\left(\sqrt{N_v} \sum_{\alpha=1}^d m_\alpha w_{\alpha a} + \eta_a\right)\right]\right)}{Z} \tag{20}$$

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

$$\log\cosh(x) \simeq |x| - \log 2, \tag{21}$$

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

$$p_{\text{RCM}}(\boldsymbol{m}) = \frac{\exp\left(N_v \left[\mathcal{S}(\boldsymbol{m}) + \sum_{\alpha=1}^d \theta_\alpha m_\alpha + \sum_{a=1}^{N_h} q_a \left|\sum_{\alpha=1}^d n_\alpha m_\alpha + z_a\right|\right]\right)}{Z} \tag{22}$$

where

$$q_a = \sqrt{N_v \sum_{\alpha=1}^d w_{\alpha a}^2}, \quad n_a = \frac{w_{\alpha a}}{\sqrt{\sum_{\alpha=1}^d w_{\alpha a}^2}}, \quad z_a = \frac{\eta_a}{\sqrt{N_v \sum_{\alpha=1}^d w_{\alpha a}^2}}. \tag{23}$$

This can be easily inverted as

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

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 (21) to be bad.

The gradients are given by

$$\frac{\partial J(\boldsymbol{\Theta})}{\partial q_a} = \mathbb{E}_{\boldsymbol{m}\sim p_{\mathcal{D}}(\boldsymbol{m})}\left[|\boldsymbol{n}_a^T \boldsymbol{m} + z_a|\right] - \mathbb{E}_{\boldsymbol{m}\sim p_{\text{RCM}}(\boldsymbol{m})}\left[|\boldsymbol{n}_a^T \boldsymbol{m} + z_a|\right], \tag{24}$$

$$\frac{\partial J(\boldsymbol{\Theta})}{\partial \theta_\alpha} = E_{\boldsymbol{m}\sim p_{\mathcal{D}}(\boldsymbol{m})}\left[m_\alpha\right] - \mathbb{E}_{\boldsymbol{m}\sim p_{\text{RCM}}(\boldsymbol{m})}\left[m_\alpha\right]. \tag{25}$$

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.

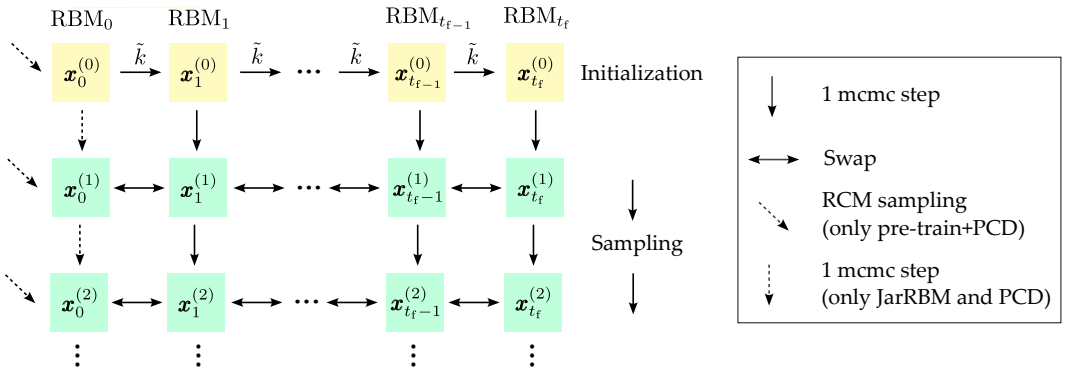

Figure 6: Scheme of PTT. We Initialize the chains of the models by starting from a configuration $\boldsymbol{x}_0^{(0)}$ and passing it through the machines along the training trajectory, each time performing $\tilde{k}$ mcmc steps. For pre-train+PCD, $\boldsymbol{x}_0^{(0)}$ is a sampling from the 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.

Model	Dataset	# of machines	Alg. # of steps	acc. factor @ 10^4 steps
pre-train+PCD	MNIST01	6 (+1)	10000	80
JarJar	MNIST01	28	10000	50
PCD	MNIST01	13	10000	30
pre-train+PCD	Human Genome	6 (+1)	10000	1350
PCD	Human Genome	13	10000	7100

Table 1: Performance comparison of different models on various datasets for the sampling using PTT versus Gibbs sampling for 10^4 mcmc steps. The acceleration factor is defined as the ratio of the average number of jumps obtained until 10^4 steps between PTT and Gibbs sampling. For pre-train+PCD, the RCM machine has not to be counted among the list of models (hence the +1) because it is very fast to sample from.

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 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. 6. 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:

$$p_\mathrm{acc}(\boldsymbol{x}_t \leftrightarrow \boldsymbol{x}_{t-1}) = \min\left(1, \exp\left(\Delta\mathcal{H}_t(\boldsymbol{x}_t) - \Delta\mathcal{H}_t(\boldsymbol{x}_{t-1})\right)\right), \tag{26}$$

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 $\{\mathcal{H}_t\}$, $t = 0, \ldots, t_\mathrm{f}$, Number of Gibbs steps k, Number of MCMC steps N_mcmc
Output: Configurations $\boldsymbol{x}_t$ for $t = 1, \ldots, t_\mathrm{f}$
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 Tab. 1.

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

Table 2: Details of the datasets used during training.

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

D Training of the RBM using the Jarzynski equation

In this section, we describe a procedure similar to the one introduced in [37] for training the RBM by leveraging the Jarzynki equation. 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

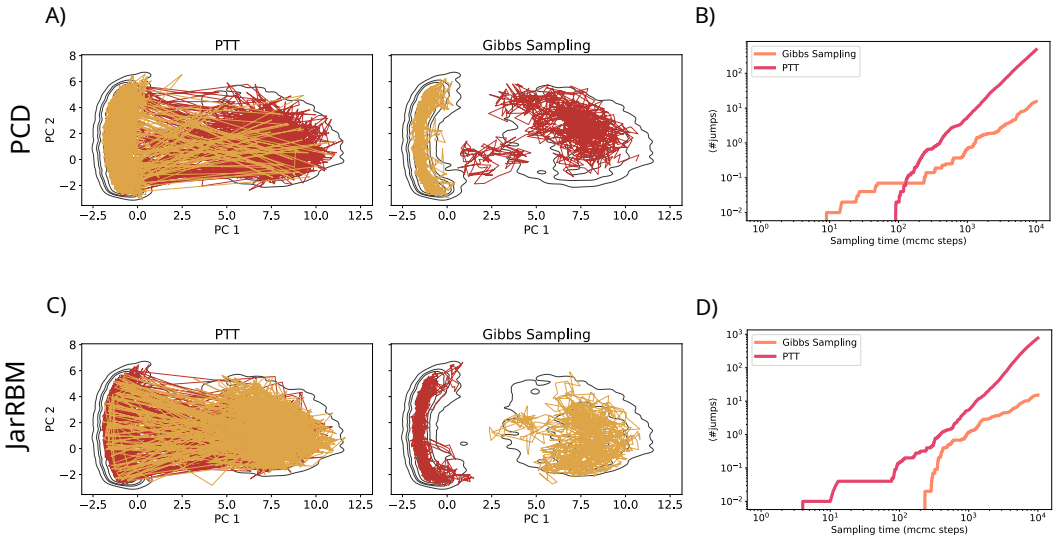

Figure 7: Comparison between PTT and standard Gibbs Sampling for RBMs trained using PCD (A and B) and JarRBM (C and D) on the MNIST01 dataset. A and C show the sampling trajectory of two chains recorded every 10 steps for a total of 10^4 mcmc steps. B and D show the average number of jumps of a population of 100 chains as a function of the sampling time.

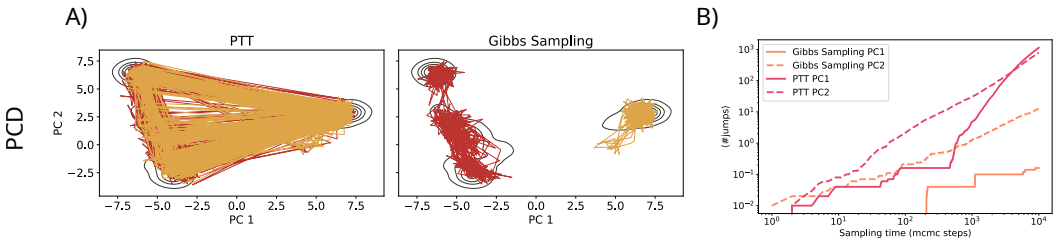

Figure 8: Comparison between PTT and standard Gibbs Sampling for RBMs trained using PCD on the Human Genome dataset. The sampling has been performed under the same conditions of fig. 7.

repetitions of an out-of-equilibrium dynamical process. 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

$$\langle \mathcal{O} \rangle_t = \frac{\left\langle \mathcal{O} e^{-W_t} \right\rangle_{\text{traj}}}{\left\langle e^{-W_t} \right\rangle_{\text{traj}}}, \tag{27}$$

where the average on the lhs is done over the last model p_t, the averages on the rhs are taken across many different trajectory realizations and W_t is a trajectory-dependent importance factor. By all practical means, under the assumption of having quasi-adiabatic parameters updates, namely $p(\Theta_{t-1} \to \Theta_t) = p(\Theta_t \to \Theta_{t-1})$, this means that we can assign to each Markov chain of the simulation $\boldsymbol{x}^{(r)}$, $r = 1, \ldots, R$, an importance weight given by:

$$W_t^{(r)} = \sum_{\tau=1}^{t} [\mathcal{H}_\tau(\boldsymbol{x}_{\tau-1}^{(r)}) - \mathcal{H}_{\tau-1}(\boldsymbol{x}_{\tau-1}^{(r)})] \tag{28}$$

and then compute the gradient of the log-likelihood by means of a weighted average over the chains:

$$\langle \mathcal{O} \rangle_t \simeq \frac{\sum_{r=1}^{R} \mathcal{O}(\boldsymbol{x}^{(r)})\, e^{-W_t^{(r)}}}{\sum_{r=1}^{R} e^{-W_t^{(r)}}}. \tag{29}$$

Table 3: Hyperparameters used for the training of RBMs.

Name	Batch size	#Chains	#Epochs	Learning rate	#MCMC steps	#Hidden nodes
HGD						
PCD	2000	2000	10 000	0.01	100	185
Jar-RBM	2000	10 000	10 000	0.01	100	185
Pre-train+PCD	2000	2000	10 000	0.01	100	185
MNIST-01						
PCD	2000	2000	10 000	0.01	100	200
Jar-RBM	2000	10 000	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	100
Jar-RBM	2000	10 000	10 000	0.01	100	100
Pre-train+PCD	2000	2000	10 000	0.01	100	100

Notice that, since Eq. (27) 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 [37]

$$\mathrm{ESS} = \frac{\left(R^{-1}\sum_{r=1}^{R} e^{-W^{(r)}}\right)^2}{R^{-1}\sum_{r=1}^{R} e^{-2W^{(r)}}} \in [0,1], \tag{30}$$

which measures the relative dispersion of the weights distribution. A way of circumventing the weight concentration on a few chains, then, 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 ($W^{(r)} = 0 \ \forall r = 1, \ldots, R$).

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and precede the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.

- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification:

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.

- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification:

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification:

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification:

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification:

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [No]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.