Mean-Field Assisted Deep Boltzmann Learning with Probabilistic Computers

Shuvro Chowdhury, Shaila Niazi, Kerem Y. Camsari

Department of Electrical and Computer Engineering
University of California Santa Barbara, Santa Barbara, CA 93106, USA
{schowdhury, sniazi, camsari} @ ucsb.edu

Abstract

Despite their appeal as physics-inspired, energy-based and generative nature, general Boltzmann Machines (BM) are considered intractable to train. This belief led to simplified models of BMs with *restricted* intralayer connections or layer-by-layer training of deep BMs. Recent developments in domain-specific hardware – specifically probabilistic computers (p-computer) with probabilistic bits (p-bit) – may change established wisdom on the tractability of deep BMs. In this paper, we show that deep and *unrestricted* BMs can be trained using p-computers generating hundreds of billions of Markov Chain Monte Carlo (MCMC) samples per second, on sparse networks developed originally for use in D-Wave's annealers. To maximize the efficiency of learning the p-computer, we introduce two families of Mean-Field Theory assisted learning algorithms, or xMFTs (x = Naive and Hierarchical). The xMFTs are used to estimate the averages and correlations during the *positive phase* of the contrastive divergence (CD) algorithm and our custom-designed p-computer is used to estimate the averages and correlations in the negative phase. A custom Field-Programmable-Gate Array (FPGA) emulation of the p-computer architecture takes up to 45 billion flips per second, allowing the implementation of CD-n where n can be of the order of millions, unlike RBMs where n is typically 1 or 2. Experiments on the full MNIST dataset with the combined algorithm show that the positive phase can be efficiently computed by xMFTs without much degradation when the negative phase is computed by the p-computer. Our algorithm can be used in other scalable Ising machines and its variants can be used to train BMs, previously thought to be intractable.

1 Introduction

Since their introduction by Hinton and colleagues [1], Boltzmann Machines (BM) have received sustained interest over the years [2]. Most recently, BMs have found renewed interest in the representation of quantum many-body wavefunctions as an alternative to quantum Monte Carlo algorithms (see, for example, [3]). Meanwhile, the nearing end of Moore's Law has been driving the development of domain-specific computers tailored for specific applications and algorithms. A notable class of such computers deals with probabilistic computers (p-computer) with probabilistic bits (p-bit) (see, for example, [4, 5]). Probabilistic computers have been implemented at various sizes and in different physical substrates. Magnetic nanodevices exploiting the ambient thermal noise to build p-bits have been implemented in small scales (10-80 p-bits, [6, 7]). Custom-made digital *emulators* using Field Programmable Gate Arrays (FPGA) has been scaled much further, up to 5,000 - 10,000

37th First Workshop on Machine Learning with New Compute Paradigms at NeurIPS 2023(ML-NPCP 2023).

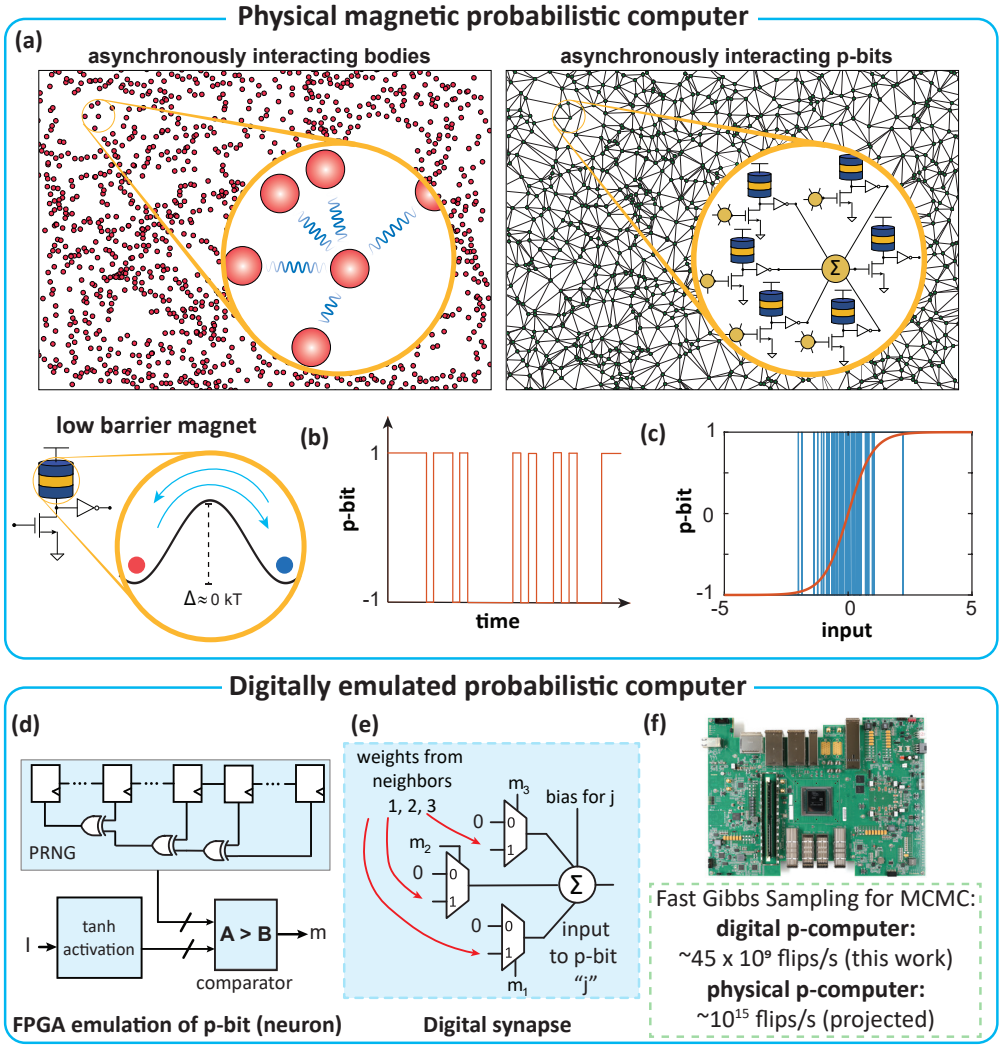

Figure 1: **p-computing overview:** (a) Analogy between interacting bodies in nature and interacting p-bit networks we build in this work. In stochastic MTJ (sMTJ) based implementations of p-bits, a low energy barrier magnet is used to generate natural noise. (b) Typical output of a p-bit against time fluctuating randomly between $+1$ and -1. (c) Input/output characteristic of a p-bit. The output (blue curve) is pinned to ± 1 at strong positive and negative inputs. The average (orange) has a tanh behavior. (d) In this work, we emulate the p-bit in a digital system (FPGA) with a pseudorandom number generator (PRNG), a lookup table for the tanh and a comparator. (e) The digital emulation of the synapse with MUXes is also shown. (f) A p-computer consisting of a network of such p-bits is then realized in an FPGA.

p-bits [8]. Despite their small sizes at table-top experiments, nanodevice-based p-computers are arguably the most scalable option to gain more performance and energy-efficiency, with projections up to million p-bit densities [9], given the success of magnetic memory technology [10]. The high costs of pseudorandom number generators required for each p-bit make it prohibitively expensive to get to such million-bit densities using existing CMOS technology [11]. Nonetheless, CMOS emulators of p-computers are useful in demonstrating the architectural and algorithmic potential of physics-inspired scalable p-computers. In this paper, we use such a custom-designed, highly efficient FPGA-based p-computer emulator (Fig. 1d-f) that can take 45 billion MCMC samples every second (or flips per second, fps) to train deep Boltzmann machines. Notably, this flips per second is about 4 - 5X faster than custom GPU/TPU implementations implemented on far simpler networks with ± 1 weights (see, for example, [12, 13, 14, 15]). Boltzmann Machines are trained by the contrastive

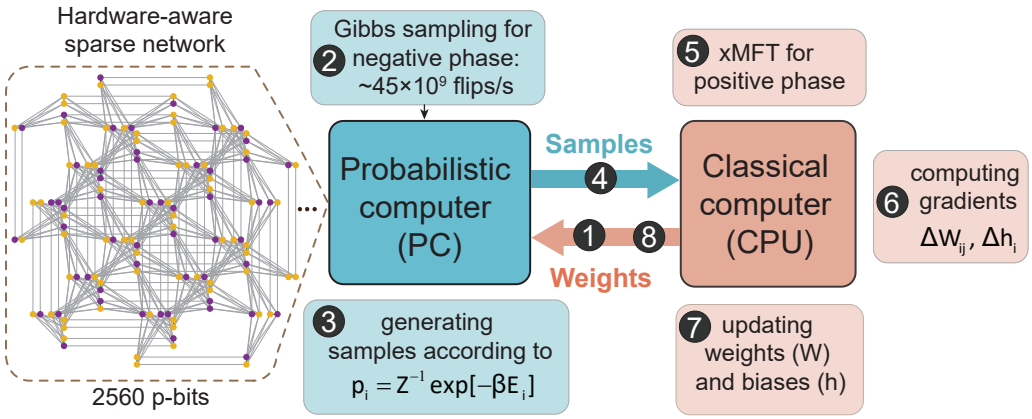

Figure 2: **Hybrid computing scheme for ML:** A hybrid computing scheme with probabilistic and classical computers is shown. Inside the classical computer, the positive phase is performed with the help of mean-field theory derivative algorithms. At the beginning of the negative phase, the classical computer sends weights and biases required to our probabilistic computer (PC) where we perform Gibbs sampling. The probabilistic computer can generate a measured 45 billion Gibbs flips in a second (FPGA). The PC returns samples to the CPU which computes the gradient. This process is repeated until convergence.

divergence algorithm, assuming a quadratic energy function [16, 17],

$$\Delta W_{ij} = \langle m_i m_j \rangle^{\text{data}} - \langle m_i m_j \rangle^{\text{model}} \qquad \text{and} \qquad \Delta h_i = \langle m_i \rangle^{\text{data}} - \langle m_i \rangle^{\text{model}} \tag{1}$$

One difficulty in training unrestricted Boltzmann machines is the need to perform explicit MCMC sampling in the positive (data) phase. To perform the visible-to-hidden layer inference in one shot, BMs typically restrict connections within a layer, removing the need for MCMC sampling. Removing these connections, however, hurts the representational ability of the network. The need for two separate phases is also detrimental to hardware development [18] where each input in a batch needs to be clamped followed by MCMC sampling, *serially*. In this paper, we propose a hybrid algorithm to circumvent the positive phase sampling of *unrestricted* Boltzmann machines. Our main contributions are as follows:

(1) We implement a fast FPGA-based digital MCMC sampler emulating physical probabilistic computers that are able to take up to 45 billion Gibbs samples per second, communicating with a classical computer in a closed-loop setup capable of training a deep *unrestricted* BM with 2560 nodes and 17984 parameters to learn the full MNIST data set entirely in hardware, which is rarely performed in direct hardware implementations of BMs.

(2) We propose a hybrid mean-field theory (MFT) assisted contrastive divergence (CD) algorithm to ease the positive phase computation of *unrestricted* and *deep* BMs. Going beyond naive MFTs (NMFT), we also propose a *hierarchical* MFT (HMFT), improving correlation estimations at the cost of making $\mathcal{O}(N^2)$ more NMFT calls.

(3) We demonstrate that the hybrid algorithm we design does not result in significant degradation compared to the MCMC method since *positive* phase correlations are much more easily handled by MFTs as opposed to *negative* phase correlations.

2 Gibbs Sampling with p-bits and Mean Field Theories

A p-bit randomly fluctuates between two states (say, in between $+1$ and -1) with a continuous-valued input. Mathematically, an interconnected network of p-bits is represented by the following two equations:

$$I_i = \sum_j W_{ij} m_j + h_i \qquad \text{and} \qquad m_i = \text{sgn}\left(\tanh\left(\beta I_i\right) - r_{[-1,1]}\right) \tag{2}$$

where $m_i \in \{-1, +1\}$ and $r_{[-1,1]}$ is a uniform random number drawn from the interval $[-1, 1]$. $\{W_{ij}\}$ are the weights, $\{h_i\}$ are the biases and β is the inverse temperature. When

solved iteratively also known as Gibbs sampling [19], Eq. (2) approximately follows the Boltzmann distribution [20]:

$$p(\{m\}) = \frac{1}{Z} \exp\left[-\beta E(\{m\})\right] \qquad \text{and} \qquad E(\{m\}) = -\sum_{i<j} W_{ij} m_i m_j - \sum_i h_i m_i \tag{3}$$

The second equation in Eq. (2) is also known as the "binary stochastic neuron" in machine learning. In the present context, the iterated evolution of these equations represents a *dynamical system* directly implemented in hardware. As long as I_i computation time is faster than m_i computation time in Eq. (2), the update order does not matter and can be random, supporting asynchronous and massively parallel designs. In general-purpose computers where Gibbs sampling is usually performed on software, it can become computationally expensive, especially without the help of any accelerator. Therefore, in many fields of physics and statistics, mean-field theory (MFT) is instead widely used where one tries to approximate the behavior of a many-body system by using an average field instead of individual interactions among the components [21],[22]. This simplification of the system description to a mere average significantly reduces the computational load. In the present context, the relevant MFT equations to be solved self-consistently are the following [23] where $\langle m \rangle \in (-1, 1)$:

$$\langle I_i \rangle = \sum_j W_{ij} \langle m_i \rangle + h_i \langle m_i \rangle \qquad \text{and} \qquad \langle m_i \rangle = \tanh\left(\beta \langle I_i \rangle\right) \tag{4}$$

It is also worthwhile to note that although MFT yields a solution of a complex system with less computational effort, the estimates from MFT are not always accurate [23].

3 Hierarchical Mean Field Assisted CD Algorithm

In the context of Boltzmann machine learning, the idea of replacing the correlations in Eq. (1) with MFTs was first introduced by Petersen et al. [24]. Improvements to this idea such as linear response correction [25, 26, 27] and its higher order extensions [28] were also made. Hinton et al. [29] proposed a deterministic variant of the contrastive divergence algorithm. Recently, a variational mean field theory has also been proposed in [30].

Different from all these approaches considered earlier, in this paper we propose a hybrid approach (Fig. 2): unlike most MFT approaches, the free running phase is performed with Gibbs sampling but on a fast p-computer and for the positive phase in the spirit of [31], we use an alternative method to compute the correlations from the vanilla MFT method which we call hierarchical mean-field theory (HMFT). In traditional MFT methods, the correlations are calculated assuming independence between interacting bodies, i.e., $\langle m_i m_j \rangle = \langle m_i \rangle \langle m_j \rangle$ [27]. In our approach, we do not use this assumption. Rather, we start from the basic definition of correlation, i.e.,

$$\langle m_i m_j \rangle = \sum_{m_i = \pm 1, m_j = \pm 1} p(m_i, m_j) m_i m_j \qquad \text{with} \qquad p(m_i, m_j) = p(m_i | m_j)\, p(m_j) \tag{5}$$

When we compute MFT, we get an estimate for $p(m_j)$. However, to use Eq. (5), we also need to know or compute $p(m_i|m_j)$. This can be done by *clamping* p-bit j to ± 1 and then performing another MFT estimating $p(m_i|m_j)$. After making $\Theta(2n)$ such MFT calls, we can estimate the second-order correlations using Eq. (5). Our HMFT approach is presented in Algorithm 1. HFMT improves correlation estimations by not baking in the independence assumption $\langle m_i m_j \rangle = \langle m_i \rangle \langle m_j \rangle$ as Naive MFT does (see Supplementary Information 1 for a discussion on how HMFT can capture correlations completely missed by MFT assuming independence, in a toy example). In fact, this Bayesian trick can be used in conjunction with other methods that approximate marginals, such as Loopy Belief Propagation [32] or with corrections to naive MFT (for example, see [25]), improving the HFMT method. Moreover, higher-order correlations of the form $\langle m_i m_j m_k \rangle$, $\langle m_i m_j m_k m_l \rangle$, ... can be *hierarchically* estimated. These can then be used to train higher-order Boltzmann machines [33], trading off parallelizable MFT computations with the fundamentally serial Gibbs sampling. In the experiments below, we investigate how the positive phase of the contrastive divergence algorithm could be performed by the MFT and the HFMT method we propose.

Algorithm 1: The Hierarchical Mean-field Algorithm

Input : weights and biases J, h, update factor λ, tolerance δ, max. iteration $T_{\max}$
Output: estimates for averages $\langle m_{\rm i} \rangle$ and correlations $\langle m_i m_j \rangle$

$\epsilon \leftarrow 1000, \quad N \leftarrow \text{length}(h), \quad T \leftarrow 1, \quad m_{\rm old} \leftarrow 0.01\,\text{rand}(-1, 1)$
for $i \leftarrow 1$ **to** $N + 1$ **do**
  for $j \leftarrow -1$ **to** $+1$ **By** 2 **do**
    if $i \neq 1$ **then**
      $m_{{\rm old},i-1} \leftarrow j$ ▷ `clamping to ±1 to get conditional probability`
    while $\epsilon \geq \delta$ **do**
      $I \leftarrow J m_{\rm old} + h$
      $m_{\rm new} \leftarrow \tanh(I), \quad m_{{\rm new},i-1} \leftarrow j$
      $\epsilon \leftarrow \left(\sum_i |m_{\rm new,i} - m_{\rm old,i}|\right) / \left(\sum_i |m_{\rm new,i} + m_{\rm old,i}|\right)$
      $m_{\rm old} \leftarrow \lambda m_{\rm new} + (1 - \lambda) m_{\rm old}$
    $m_{\rm avg} \leftarrow m_{\rm new}$ ▷ `spin averages`
    $p(m_k = \pm 1) \leftarrow 1/\left(1 + \exp\left(\mp 2 I_k\right)\right)$ ▷ `individual probabilities`
    $p(m_k = \pm 1 | m_i = j) \leftarrow 1/\left(1 + \exp\left(\mp 2 I_k\right)\right)$ ▷ `conditional probabilities`
$\langle m_i m_j \rangle \leftarrow$ Compute correlations from Eq. (5)

4 Experiments

We have used the MNIST dataset (handwritten digits, [34]) to train sparse, deep and unrestricted Boltzmann networks without any downsampling, typically performed on hardware implementations by D-Wave and others [12, 13, 14, 15]. We have used black/white images by thresholding the MNIST dataset and we choose a Pegasus graph [35] with up to 2560 p-bits (nodes) as the sparse DBM network model in this paper. The graph density of this Pegasus is 0.55% and the maximum number of neighbors is 15. The network has 834 visible p-bits (including 5 sets of labels each containing 10 p-bits) and 1726 hidden p-bits that are arranged in 2 layers as shown in the inset of FIG. 3b. The total number of connections (network parameters) in this graph is 17984. Using our fast Gibbs sampler (p-computer), we accomplish the contrastive divergence algorithm to train the MNIST dataset divided into 1200 mini-batches with 50 images in each batch. We used 10^5 sweeps in the negative phases of each epoch.

Similarly, we train the full MNIST using our hybrid MFT algorithm with naive MFT in the positive phase and Gibbs sampling (10^5 sweeps) in the negative phase. To find the classification accuracy, we perform a softmax classification over 50 label p-bits to get the 10 labels. The p-bit with the highest probability of being '1' indicates the classified digit. FIG. 3a shows that our sparse DBM with 2560 p-bits reaches around 87% accuracy with Gibbs sampling and 70% accuracy (with MFT tolerance = 10^{-2} and this accuracy may further improve with lower tolerance) with hybrid MFT technique in 100 epochs despite having a significantly lesser number of parameters than typical RBMs. For the full MNIST dataset, the computational expense of HMFT prevented us from comparing it with the results of MFT at this time but in future this could be made possible with more parallel resources like using GPUs. Moreover, sophisticated techniques like layer-by-layer learning [31] should further improve the accuracy reported in this work. Although our reported accuracy is comparable with models with less parameters (e.g., regression models), the real value of Boltzmann machines is their generative properties and has been shown recently in [18].

To be able to evaluate both the efficiency of MFT and HFMT methods in the positive phase, we performed the simpler task of training MNIST/100 (100 images randomly chosen from the MNIST dataset). Three different schemes used the same hyper-parameters where the positive phase of training is accomplished with naive MFT, HMFT (on CPU), and Gibbs sampling (on p-computer). The negative phase is performed in our probabilistic computer where we are naturally doing persistent CD algorithm (PCD) [36, 37]. This hybrid computing scheme is illustrated in Fig. 2 and the details of the experimental setup can be found in [18]. The training accuracy of 100 images reaches 100% with Gibbs sampling

and naive MFT and HMFT also perform similarly. Supplementary Table S2 indicates that despite a large difference in the training set log-likelihood, the test set shows roughly similar results, indicating how the hybrid approach does not degrade the performance significantly. The performance of this approach on larger datasets and networks remains to be seen.

It is interesting to note here that when Gibbs sampling in the negative phase is replaced by xMFTs, both training and test set accuracies degrade severely and, in fact, do not work at all. The supplementary Table S1 shows the poorer performance of xMFTs in the negative phases.

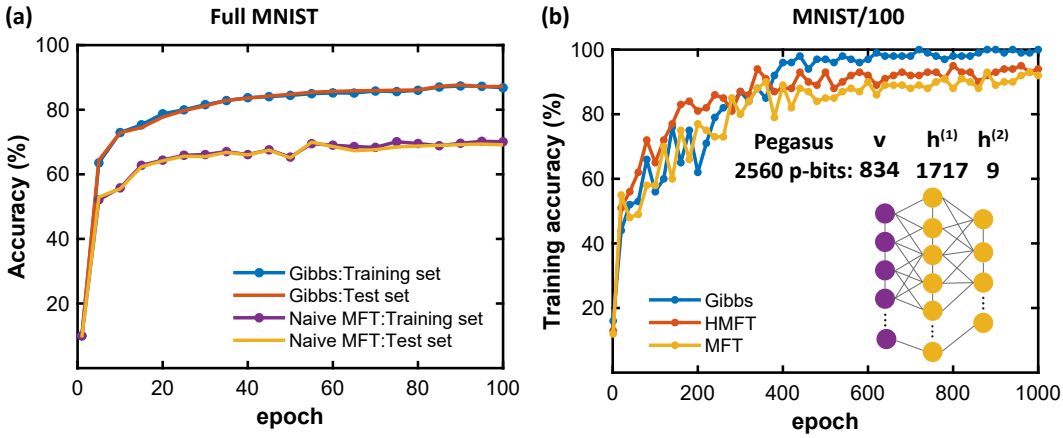

Figure 3: **MNIST accuracy with different methods:** (a) Full MNIST (60,000 images) is trained on sparse DBM (Pegasus 2560 p-bits) with Gibbs sampling (CD-10^5) and naive MFT where batch size = 50, learning rate = 0.003, momentum = 0.6. Around 87% accuracy is achieved in 100 epochs for Gibbs sampling and 70% for the naive MFT. Test accuracy represents the accuracy of all 10,000 images from the MNIST test set, while the training accuracy corresponds to the accuracy of 10,000 images randomly drawn from the training set. (b) Training accuracy of MNIST/100 with the three different schemes: naive MFT, HMFT, and Gibbs sampling where they perform similarly. Here the batch size = 10, momentum = 0.6 and learning rate varies from 0.06 to 0.006 over 1000 epochs.

5 Conclusions and Outlook

The end of Moore's law is driving the development of physics-based probabilistic computers that can accelerate MCMC algorithms by orders of magnitude. In this paper, we showed an FPGA emulation of such a computer that can take up to 45 billion Gibbs samples per second. Experimentally-validated projections indicate up to 10^{15} samples per second are possible with truly physical p-computers. Such a large increase in MCMC speeds may allow the direct training of deep and *unrestricted* BMs. As an example of this approach, we trained a 2-layer unrestricted deep BM on a sparse Pegasus graph used by D-Wave, showing promising results (near 90% classification accuracy with only ≈18k parameters). To aid with the positive phase training of unrestricted and deep BMs, we also proposed a hierarchical mean-field theory assisted learning algorithm. In accordance with common wisdom, we found that MFTs fail to estimate model correlations, especially when the weights become large. Surprisingly, however, we observed that MFTs accurately approximate data correlations, during the positive phase, greatly simplifying training in hardware. With the development of new physical computers, our results may allow the training of deep and unrestricted BMs previously thought to be intractable where hybrid MFT approaches could be used during pretraining or as a supplement to the computationally expensive Gibbs sampling. Extensions of the model by fully exploiting the parallelism offered by the MFTs in deeper networks to train harder datasets are left for future study.

Acknowledgments and Disclosure of Funding

Authors acknowledge support from the Office of Naval Research Young Investigator Program and National Science Foundation grants.

References

[1] David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for Boltzmann machines, 1985.

[2] Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In David van Dyk and Max Welling, editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 448–455, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR.

[3] Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, Maria Schuld, Naftali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborová. Machine learning and the physical sciences. *Reviews of Modern Physics*, 91(4):045002, 2019.

[4] Kerem Yunus Camsari, Rafatul Faria, Brian M Sutton, and Supriyo Datta. Stochastic p-bits for invertible logic. *Physical Review X*, 7(3):031014, 2017.

[5] Shuvro Chowdhury, Andrea Grimaldi, Navid Anjum Aadit, Shaila Niazi, Masoud Mohseni, Shun Kanai, Hideo Ohno, Shunsuke Fukami, Luke Theogarajan, Giovanni Finocchio, et al. A full-stack view of probabilistic computing with p-bits: devices, architectures and algorithms. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, 2023.

[6] William A Borders, Ahmed Z Pervaiz, Shunsuke Fukami, K. Y. Camsari, Hideo Ohno, and Supriyo Datta. Integer factorization using stochastic magnetic tunnel junctions. *Nature*, 2019.

[7] Jia Si, Shuhan Yang, Yunuo Cen, Jiaer Chen, Zhaoyang Yao, Dong-Jun Kim, Kaiming Cai, Jerald Yoo, Xuanyao Fong, and Hyunsoo Yang. Energy-efficient superparamagnetic Ising machine and its application to traveling salesman problems. *arXiv preprint arXiv:2306.11572*, 2023.

[8] Navid Anjum Aadit, Andrea Grimaldi, Mario Carpentieri, Luke Theogarajan, John M Martinis, Giovanni Finocchio, and Kerem Y Camsari. Massively parallel probabilistic computing with sparse Ising machines. *Nature Electronics*, 5(7):460–468, 2022.

[9] Brian Sutton, Rafatul Faria, Lakshmi Anirudh Ghantasala, Risi Jaiswal, Kerem Yunus Camsari, and Supriyo Datta. Autonomous probabilistic coprocessing with petaflips per second. *IEEE Access*, 8:157238–157252, 2020.

[10] CJ Lin, SH Kang, YJ Wang, K Lee, X Zhu, WC Chen, X Li, WN Hsu, YC Kao, MT Liu, et al. 45nm low power cmos logic compatible embedded stt mram utilizing a reverse-connection 1t/1mtj cell. In *Electron Devices Meeting (IEDM), 2009 IEEE International*, pages 1–4. IEEE, 2009.

[11] Keito Kobayashi, Nihal Singh, Qixuan Cao, Kemal Selcuk, Tianrui Hu, Shaila Niazi, Navid Anjum Aadit, Shun Kanai, Hideo Ohno, Shunsuke Fukami, et al. CMOS+ stochastic nanomagnets: heterogeneous computers for probabilistic inference and learning. *arXiv preprint arXiv:2304.05949*, 2023.

[12] Steven H Adachi and Maxwell P Henderson. Application of quantum annealing to training of deep neural networks. *arXiv preprint arXiv:1510.06356*, 2015.

[13] Haik Manukian, Fabio L Traversa, and Massimiliano Di Ventra. Accelerating deep learning with memcomputing. *Neural Networks*, 110:1–7, 2019.

[14] Vivek Dixit, Raja Selvarajan, Muhammad A Alam, Travis S Humble, and Sabre Kais. Training Restricted Boltzmann Machines With a D-Wave Quantum Annealer. *Frontiers in Physics*, 9:589626, 2021.

[15] Fabian Böhm, Diego Alonso-Urquijo, Guy Verschaffelt, and Guy Van der Sande. Noise-injected analog Ising machines enable ultrafast statistical sampling and machine learning. *Nature Communications*, 13(1):5847, 2022.

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

[17] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on Machine learning*, pages 473–480, 2007.

[18] Shaila Niazi, Navid Anjum Aadit, Masoud Mohseni, Shuvro Chowdhury, Yao Qin, and Kerem Y Camsari. Training Deep Boltzmann Networks with Sparse Ising Machines. *arXiv preprint arXiv:2303.10728*, 2023.

[19] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009. ISBN 0262013193.

[20] Kerem Yunus Camsari, Rafatul Faria, Brian M Sutton, and Supriyo Datta. Stochastic p-bits for invertible logic. *Physical Review X*, 7(3):031014, 2017.

[21] P. M. Chaikin and T. C. Lubensky. *Mean-field theory*, page 144–212. Cambridge University Press, 1995. doi: 10.1017/CBO9780511813467.005.

[22] Mehran Kardar. *Statistical Physics of Particles*. Cambridge University Press, 2007. doi: 10.1017/CBO9780511815898.

[23] Dalton A R Sakthivadivel. Magnetisation and Mean Field Theory in the Ising Model. *SciPost Phys. Lect. Notes*, page 35, 2022. doi: 10.21468/SciPostPhysLectNotes.35.

[24] Carsten Petersen and James R Anderson. A mean field theory learning algorithm for neural networks. *Complex Systems*, 1:995–1019, 1987.

[25] Hilbert Kappen and Francisco de Borja Rodríguez Ortiz. Boltzmann machine learning using mean field theory and linear response correction. In M. Jordan, M. Kearns, and S. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. MIT Press, 1997.

[26] H.J. Kappen and F.B. Rodríguez. Mean field approach to learning in Boltzmann Machines. *Pattern Recognition Letters*, 18(11):1317–1322, 1997. ISSN 0167-8655. doi: https://doi.org/10.1016/S0167-8655(97)00096-2.

[27] Hilbert J. Kappen and FDB Rodríguez. Efficient learning in Boltzmann machines using linear response theory. *Neural Computation*, 10(5):1137–1156, 1998.

[28] Toshiyuki Tanaka. Mean-field theory of Boltzmann machine learning. *Phys. Rev. E*, 58:2302–2310, Aug 1998. doi: 10.1103/PhysRevE.58.2302.

[29] Max Welling and Geoffrey E. Hinton. A New Learning Algorithm for Mean Field Boltzmann Machines. In José R. Dorronsoro, editor, *Artificial Neural Networks — ICANN 2002*, pages 351–357, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-46084-8.

[30] Haiping Huang. Variational mean-field theory for training restricted Boltzmann machines with binary synapses. *Phys. Rev. E*, 102:030301, Sep 2020. doi: 10.1103/PhysRevE.102.030301.

[31] Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5, pages 448–455, 2009.

[32] Marc Mezard and Andrea Montanari. *Information, physics, and computation*. Oxford University Press, 2009.

[33] Terrence J Sejnowski. Higher-order Boltzmann machines. In *AIP Conference Proceedings*, volume 151, pages 398–403. American Institute of Physics, 1986.

[34] Yann LeCun. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[35] Nike Dattani, Szilard Szalay, and Nick Chancellor. Pegasus: The second connectivity graph for large-scale quantum annealing hardware. *arXiv preprint arXiv:1901.07636*, 2019.

[36] Geoffrey E. Hinton. *A Practical Guide to Training Restricted Boltzmann Machines*, pages 599–619. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[37] 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.

Supplementary Information

1 HMFT vs NMFT in a toy 2-spin example

Consider a simple two-spin antiferromagnetic system with weights $W_{ij} = \delta_{ij} - 1$ and no biases. At any temperature, the marginals of these two spins will be identically, i.e., $\langle m_i \rangle = 0$. As such, the naive mean field method which uses conditional independence assumption $\langle m_i m_j \rangle = \langle m_i \rangle \langle m_j \rangle$ will estimate the correlation between these two spins to be identically zero, which is clearly incorrect, since at any non-zero temperature the spins will be anti-correlated. On the other hand, the hierarchical mean field method which *does not* assume the conditional independence and rather uses Eq. (4) to compute correlations, estimates a non-zero correlation. For example, at $T = 1$, the exact correlation (from Boltzmann law) between the two spins for the given weights and biases is -0.7616. The hierarchical method also estimates the same correlation value. Higher-order correlations (hierarchically obtained) can also be shown to be better estimated by the HMFT method.

2 Contrastive divergence algorithm

In this section, we briefly outline the contrastive divergence algorithm in our hybrid approach which implements Eq. (1):

Algorithm S1: Mean-field assisted training of sparse DBMs

Input : number of samples N, batch size B, number of batches N_B, epochs $N_{\rm L}$, learning rate ε, mean-field update factor λ, mean-field tolerance δ, maximum iteration for mean-field $T_{\rm max}$

Output: trained weights $J_{\rm out}$ and biases $h_{\rm out}$

$J_{\rm out} \leftarrow \mathcal{N}(0, 0.01)$, $h_{\rm out,hidden} \leftarrow 0$, $h_{\rm out,visible} \leftarrow \log\left(p_i/(1-p_i)\right)$;
for $i \leftarrow 1$ **to** N_L **do**
  for $j \leftarrow 1$ **to** N_B **do**
    /* positive phase */
    for $k \leftarrow 1$ **to** B **do**
      $h_B \leftarrow$ clamping to batch images;
      $\langle m_i \rangle^{(k)}, \langle m_i m_j \rangle^{(k)} \leftarrow$ **x**MFT_module$(J_{\rm out}, h_B, \lambda, \delta, T_{max})$;
    $\langle m_i \rangle_{\rm data} = \text{mean}\left(\{\langle m_i \rangle^{(k)}\}\right)$, $\langle m_i m_j \rangle_{\rm data} = \text{mean}\left(\{\langle m_i m_j \rangle^{(k)}\}\right)$; ▷ CPU
    /* negative phase */
    $h_{\rm Sampler} \leftarrow h_{\rm out}$, $J_{\rm Sampler} \leftarrow J_{\rm out}$;
    $\{m\} \leftarrow \text{GibbsSampler}(N)$; ▷ p-computer
    $\langle m_i \rangle_{\rm model} = \text{mean}(\{m\})$, $\langle m_i m_j \rangle_{\rm model} = \{m\}\{m\}^{\rm T}/(N)$; ▷ CPU
    /* update weights and biases */
    $J_{{\rm out},ij} \leftarrow J_{{\rm out},ij} + \epsilon\left(\langle m_i m_j \rangle_{\rm data} - \langle m_i m_j \rangle_{\rm model}\right)$; ▷ CPU
    $h_{{\rm out},i} \leftarrow h_{{\rm out},i} + \epsilon\left(\langle m_i \rangle_{\rm data} - \langle m_i \rangle_{\rm model}\right)$; ▷ CPU

3 Evolution of correlations over epochs

In this section, we discuss the evolution of correlations over epochs for naive and hierarchical MFT. Typical predictions of correlations from xMFT algorithms are shown in Supplementary Fig. S1. It can be clearly seen that in the positive phase, xMFT algorithms provide nearly accurate estimations for correlations. The clamping of many pbits during the positive phase helps xMFT algorithms to boost their performance but in the absence of such clamps, their performances degrade severely in the negative phase. In a hybrid setting, the probabilistic computer suffers from the communication delay caused by the fact that images are to be sent repeatedly whereas in the negative phase, there is no such delay. These two considerations justify the desire to replace Gibbs sampling in the positive phase with xMFT algorithms.

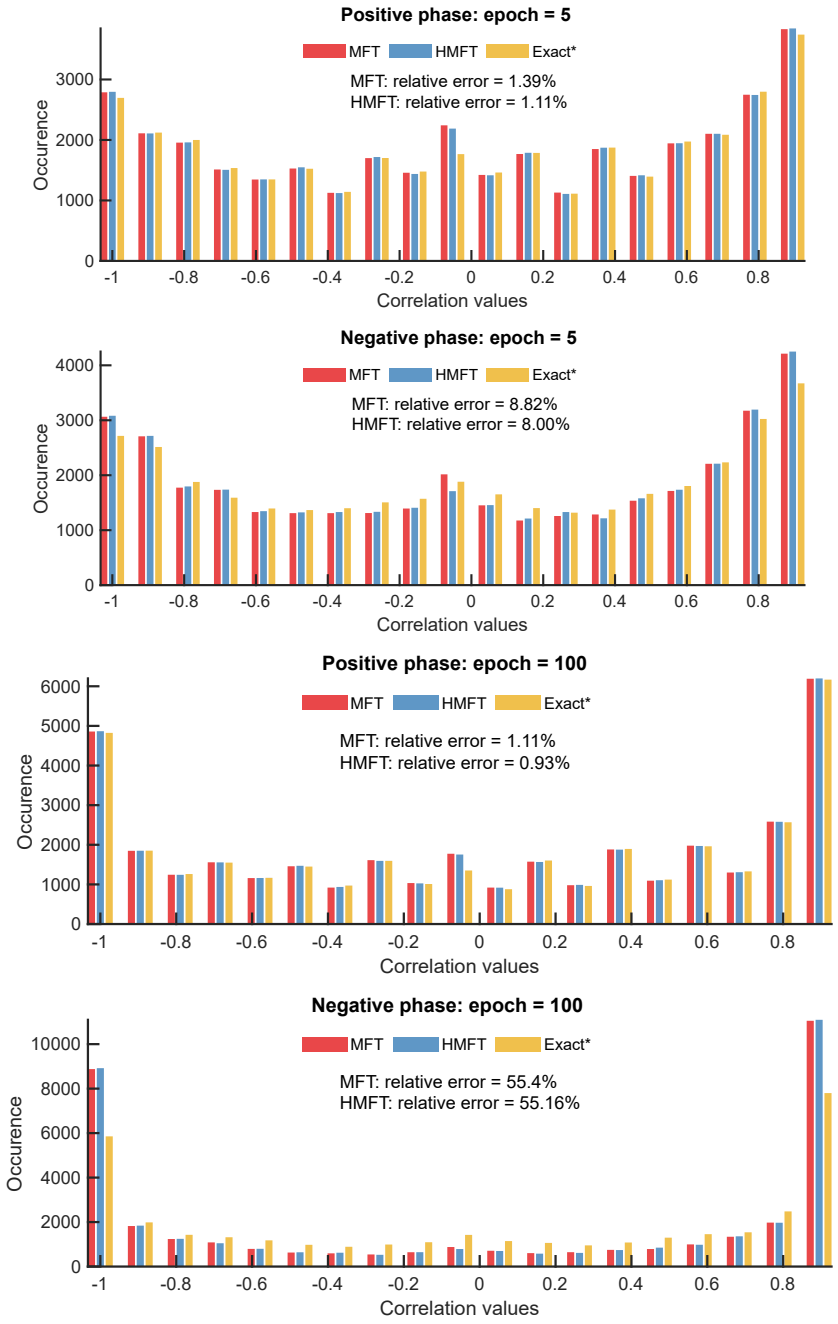

Figure S1: **Typical predictions of correlations from different methods:** The correlations predicted by the three different schemes - naive MFT, HMFT, and Gibbs sampling (the putative exact method since we cannot obtain exact Boltzmann correlations in general spin-glasses) are shown during a typical epoch in the training of a sparse DBM. We used a batch size of 10 images and for the positive phase, we obtained correlations by showing only one batch of 10 images. For Gibbs sampling, we used 10^4 sweeps in both positive and negative phases. We chose a relative error tolerance of 10^{-2} for both MFT and HMFT. 20 bins were used in both histograms. **MFT algorithms do significantly better in the positive phase than in the negative phase** allowing their use in the positive phase training of deep and unrestricted BMs, instead of the more expensive Gibbs sampling.

In order to provide a quantitative measure of the performance between the two xMFT algorithms discussed in this paper, we define the relative average error between the two correlation matrices estimated from two different approaches as $\epsilon = \sqrt{\sum_{ij}(A_{ij} - B_{ij})^2}/\sqrt{\sum_{ij} B_{ij}^2}$ where A is the correlation matrix predicted by the xMFT algorithms and B is the corresponding matrix for the Gibbs sampling. We do not include the zero correlation values in this measure. Since at the 2560 p-bits level, it is impossible to obtain correlations from the exact distribution, we use Gibbs sampling as the "putative" reference for comparison. Supplementary Table S1 lists this measure for the xMFT algorithms both in the positive and negative phases. The performance of the xMFT algorithms in both phases is consistent with Supplementary Fig. S1. As mentioned in the main text, in our HMFT implementations we do not modify the average estimations from naive MFT therefore both xMFT algorithms show the same accuracy for averages. For correlations, HMFT estimates are slightly better than naive MFT. It is also interesting to note that both models (MFT/HFMT) perform better at lower epochs when the weights are small which may suggest their use in possible pre-training supplementing the exact Gibbs sampling approach.

Table S1: Relative average error is shown in positive and negative phases for two MFT algorithms. The error is measured with respect to Gibbs sampling at each phase (we take 10^4 sweeps for both positive and negative phases). The tolerance of the MFT algorithms is set to 10^{-2}. xMFT predicted averages and correlations are significantly better in the positive phase than in the negative phase.

	Positive phase				**Negative phase**			
epoch	**averages** $\langle m_i \rangle$		**correlations** $\langle m_i m_j \rangle$		**averages** $\langle m_i \rangle$		**correlations** $\langle m_i m_j \rangle$	
	MFT	**HMFT**	**MFT**	**HMFT**	**MFT**	**HMFT**	**MFT**	**HMFT**
1	0.72%	0.72%	1.55%	1.19%	2.00%	2.00%	3.82%	2.967%
5	0.69%	0.69%	1.39%	1.11%	5.82%	5.82%	8.82%	8.00%
10	0.63%	0.63%	1.24%	1.00%	11.73%	11.73%	16.40%	16.07%
50	0.59%	0.59%	1.11%	0.92%	35.85%	35.85%	46.80%	46.61%
100	0.61%	0.61%	1.11%	0.93%	44.44%	44.44%	55.4%	55.16%

4 Log-likelihood measure for xMFT algorithms

Table S2: Train (100 images) and test set (20 images) log-likelihood for different samplers i.e., Gibbs-Gibbs, naive MFT-Gibbs, HMFT-Gibbs for sparse deep Boltzmann machines.

MNIST/100	**Gibbs-Gibbs**	**HMFT-Gibbs**	**MFT-Gibbs**
Train set	-35.08	-71.07	-89.82
Test set	-33.87	-37.71	-37.63

In this section, we report the log-likelihood ($\mathcal{L}$) measure defined as

$$\mathcal{L} = \sum_{i=1}^{N} \sum_{c=1}^{C} y_{i,c} \cdot \log(p_{i,c}) \tag{S.1}$$

for the xMFT algorithms in both the training and the test set. Here, $y_{i,c}$ is the true label of the i^{th} sample for class c (1 if the sample belongs to class c and 0 otherwise), and $p_{i,c}$ is the predicted probability that the i^{th} sample belongs to class c. We measure the performance after training MNIST/100, with three different choices of algorithms to be used in the positive phase namely, Gibbs sampling, naive MFT and HMFT. The negative phase is always performed with Gibbs sampling. Supplementary Table S2 lists this measure.

As can be seen, the HMFT-trained model performs better than the naive MFT-trained model in both the training and test set although in the test set case, this difference is minimal.