https://doi.org/10.1038/s44335-026-00075-3

An efficient probabilistic hardware architecture for diffusion-like models

Andraž Jelinčič[1,3], Owen Lockwood[1], Akhil Garlapati[1], Peter Schillinger[1], Isaac L. Chuang[2], Guillaume Verdon[1] & Trevor McCourt[1,3] ✉

The proliferation of probabilistic AI has prompted proposals for specialized stochastic computers. Despite promising efficiency gains, these proposals have failed to gain traction because they rely on fundamentally limited modeling techniques and exotic, unscalable hardware. In this work, we address these shortcomings by proposing an all-transistor probabilistic computer that implements powerful denoising models at the hardware level. A system-level analysis indicates that devices based on our architecture could achieve performance parity with GPUs on a simple image benchmark using ~10,000 times less energy.

The unprecedented recent investment in large-scale AI systems will soon put a strain on the world's energy infrastructure. Every year, U.S. firms are spending an amount larger than the inflation-adjusted cost of the Apollo program on AI-focused data centers[1,2]. By 2030, these data centers could consume around 10% of all of the energy produced in the U.S.[3].

By no measure is this scaling frivolous: existing AI systems based on autoregressive large language models (LLMs) are extremely valuable tools[4–9], and are being adopted by consumers faster than the internet[10]. However, LLMs were architected specifically for GPUs[11], hardware originally intended for graphics, whose suitability for machine learning was discovered accidentally decades later[12,13].

Had a different style of hardware been popular in the last few decades, AI algorithms would likely have evolved in a different, and possibly more energy-efficient, direction. This interplay between algorithm research and hardware availability is known as the "hardware lottery"[14], and it entrenches hardware-algorithm pairings that may be far from optimal.

Therefore, prudent planning calls for systematic exploration of other types of AI systems in search of energy-efficient alternatives. Active efforts include mixed-signal compute-in-memory accelerators[15], photonic neural networks[16], and neuromorphic processors that emulate biological spiking[17,18].

Among these alternatives, probabilistic computing sticks out as an attractive approach because it can connect directly to AI at the system level via Energy-Based Models (EBMs). EBMs are a well-established model class in contemporary deep learning and have been competitive with the state of the art in tasks like image generation and robotic path planning[19,20]. Using probabilistic hardware to accelerate EBMs falls under the broad (and poorly defined) umbrella of *thermodynamic computing*[21].

Hardware implementations of EBMs work with special model families that adhere to physical constraints such as locality, sparsity, and limited connection density. Thanks to these constraints, probabilistic computers can utilize specialized stochastic circuitry to efficiently and quickly produce samples from a Boltzmann distribution[22]. Depending on the precise kind of hardware being used, this sampling may occur as part of the natural dynamics of the device[23–27] or may be orchestrated using an algorithm like Gibbs sampling[22,28–30].

Past attempts at EBM accelerators have suffered from issues at both the architectural and hardware levels. All previous proposals used EBMs as monolithic models of data distributions, which is known to be challenging to scale[31]. Additionally, existing devices have relied on exotic components such as magnetic tunnel junctions as sources of intense thermal noise for random-number generation (RNG)[22,32,33]. These exotic components have not yet been tightly integrated with transistors in commercial CMOS processes and do not currently constitute a scalable solution[34–36].

In this work, we address these issues and propose a commercially viable probabilistic computing system.

First, we introduce a new machine learning model, the Denoising Thermodynamic Model (DTM), which can be efficiently mapped to probabilistic computer hardware via the Denoising Thermodynamic Computer Architecture (DTCA). DTMs repurpose hardware EBMs as denoising steps rather than monolithic models of the data distribution, thereby mitigating the mixing-expressivity tradeoff that plagues monolithic EBMs. The DTCA tightly integrates DTMs into probabilistic hardware built from sparse, locally connected Boltzmann machines.

Then, by combining trained DTMs with a realistic physical model of a probabilistic computer, we show that a future device could match the performance of GPU-based generative models on binarized Fashion-MNIST while using ~10,000 times less energy per generated sample. This result is summarized in Fig. 1, and is based on measurements from a real probabilistic chip featuring an ultra-efficient, all-transistor source of random bits, along with circuit models and simulations.

[1]Extropic Corp., Boston, MA, USA. [2]Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA. [3]These authors contributed equally: Andraž Jelinčič, Trevor McCourt. ✉e-mail: trevor@extropic.ai

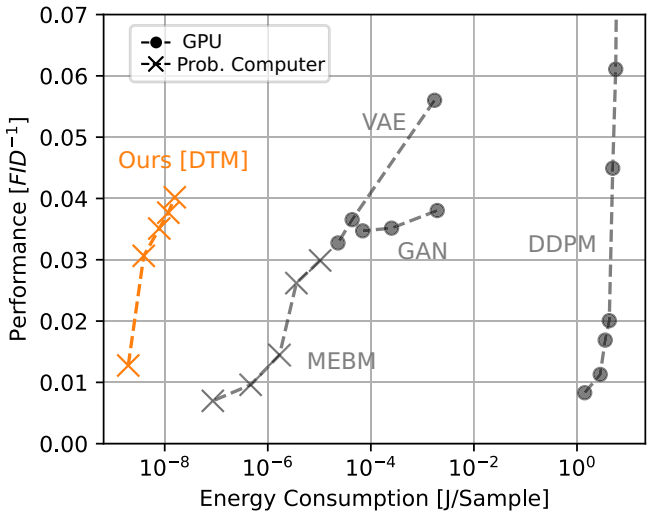

Fig. 1 | Leveraging CMOS probabilistic hardware in ultra-efficient AI systems. The central result of this article: an all-transistor probabilistic computer running a denoising thermodynamic model (DTM) could match GPU performance on a simple modeling benchmark while using about 10,000× less energy. All models are trained on binarized Fashion-MNIST[69] and evaluated with Fréchet Inception Distance (FID)[70]. DTM variants are of increasing depth, chaining 2–8 sequential Energy-Based Models (EBMs). GPU baselines cover single-step VAE[71] and GAN[72], plus DDPM[45] at varying numbers of steps. We also compare DTM to a monolithic EBM across multiple mixing-time limits. The horizontal axis shows the energy needed for generating a single new image using the trained model (inference).

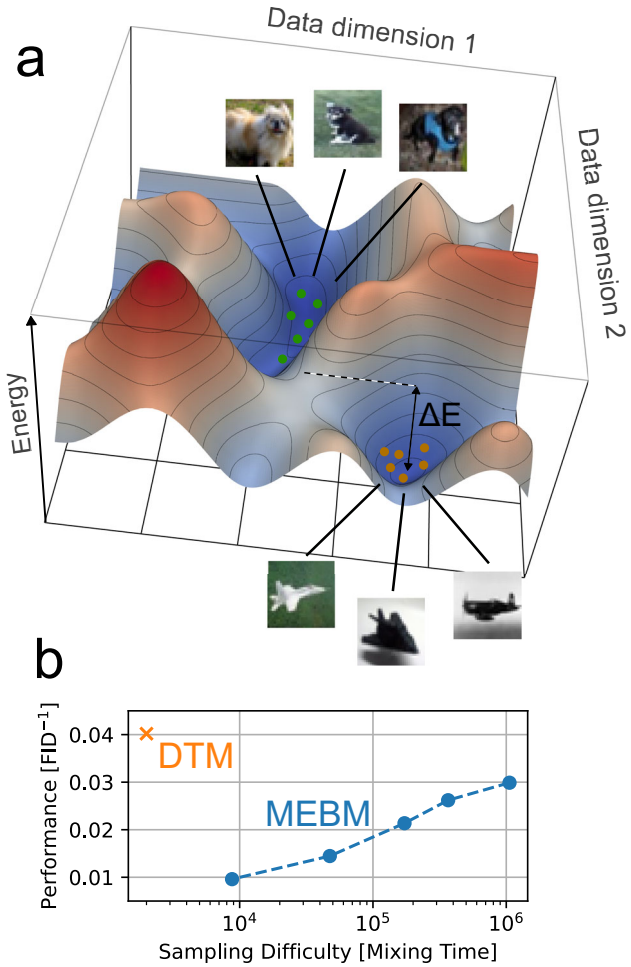

Fig. 2 | The mixing–expressivity tradeoff. a A cartoon illustrating the mixing–expressivity tradeoff in EBMs. It shows a projection of an energy landscape fit to a simple dataset. The "airplane" mode is well separated from the "dog" mode, with very little data in between. Progressively better fits of the EBM to the data tend to feature larger energy barriers ΔE between the modes, making the EBM increasingly difficult to sample from. **b** An example of the effect of the mixing–expressivity tradeoff on model performance as measured using the Fashion-MNIST dataset. The blue curve in the plot shows the results of experiments on MEBMs with limited allowed mixing time. Performance and mixing time are strongly correlated. Mixing times were computed by fitting an exponential function to the large-lag behavior of the autocorrelation function; see Supplement L. In contrast, a DTM (orange cross) has higher performance despite substantially lower sampling requirements.

Results

The challenge with EBMs

The fundamental problem of machine learning is inferring the probability distribution that underlies some data[37,38]. An early approach[39] to this was to use a monolithic EBM (MEBM) to fit a data distribution directly by shaping a parameterized energy function $\mathcal{E}$:

$$P(x;\theta) \propto e^{-\mathcal{E}(x,\theta)}, \tag{1}$$

where x is a random variable representing the data and θ represents the parameters of the EBM.

Fitting an MEBM corresponds to assigning low energies to values of x where data are abundant and high energies to values of x that are far from data. Real-world data are often clustered into distinct modes[40,41], meaning that an MEBM that fits data well will have a complex, rugged energy landscape with many deep valleys surrounded by tall mountains. This complexity is illustrated by the cartoon in Fig. 2a.

Unlike the systems we propose, existing probabilistic computers based on MEBMs struggle with the multimodality of real-world data, which hinders their efficiency. Namely, the amount of energy the computer must expend to draw a sample from the MEBM's distribution can be tremendous if its energy landscape is very rough.

Specifically, sampling algorithms that operate in high dimensions (such as Gibbs sampling[42]) are locally informed iterative procedures, meaning that they sample a landscape by randomly making small movements in the space based on low-dimensional information. When using such a procedure to sample from Eq. (1), the probability that the iteration will move up in energy to some state $X[k+1]$ is exponentially small in the energy increase compared to the current state $X[k]$, i.e.,

$$\mathbb{P}(X[k+1] = x' | X[k] = x) \propto e^{-(\mathcal{E}(x')-\mathcal{E}(x))}. \tag{2}$$

For large differences in energy, like those encountered when trying to move between two valleys separated by a significant barrier, this probability can be very close to zero. In standard Markov-chain analyses, such barriers lead to exponentially large expected transition times between modes, which is reflected in a rapidly growing mixing time.

The mixing-expressivity tradeoff (MET) summarizes this issue with existing probabilistic computer architectures, reflecting the fact that modeling performance and sampling hardness are coupled for MEBMs. Specifically, as the expressivity (modeling performance) of an MEBM increases, its *mixing time* (the amount of computational effort needed to draw independent samples from the MEBM's distribution) becomes progressively longer, resulting in expensive inference and unstable training[43,44]; see Supplement G for a definition of mixing time.

The empirical effect of the MET on the efficiency of MEBM-based probabilistic computing systems is illustrated in Fig. 2b. Mixing time increases very rapidly with performance, inflating the amount of energy required to sample from the model. The effect of this increased mixing time is reflected in Fig. 1: despite the MEBM-based solution using the same EBMs and underlying hardware as the DTM-based solution, its energy consumption is several orders of magnitude larger due to the glacially slow mixing.

Denoising thermodynamic models

The MET makes it clear that MEBMs have a flaw that makes them challenging and energetically costly to scale. However, this flaw is avoidable, and many types of probabilistic machine learning models have been developed to solve the distribution modeling problem while circumventing the MET.

Denoising diffusion models were explicitly designed to sidestep the MET by gradually building complexity through a series of simple, easy-to-sample probabilistic transformations[45]. By doing so, they allowed for much more complex distributions to be expressed given a fixed compute budget and substantially expanded the capabilities of generative models[46–48].

DTMs merge EBMs with diffusion models, offering an alternative path for probabilistic computing that mitigates the MET. DTMs are a slight generalization of recent work from deep learning practitioners that has pushed the frontier of EBM performance[49–52].

Instead of trying to use a single EBM to model the data, DTMs chain many EBMs to gradually build up to the complexity of the data distribution. This gradual buildup of complexity allows the landscape of each EBM in the chain to remain relatively simple (and easy to sample) without limiting the complexity of the distribution modeled by the chain as a whole; see Fig. 2b.

Denoising models attempt to reverse a process that gradually transforms the data distribution $Q(x^0)$ into simple noise. This forward process is given by the Markov chain

$$Q(x^0, \ldots, x^T) = Q(x^0) \prod_{t=1}^{T} Q(x^t | x^{t-1}). \tag{3}$$

The forward process is typically chosen such that it has a unique stationary distribution $Q(x^T)$, which takes a simple form (e.g., Gaussian or uniform).

Reversal of the forward process is achieved by learning a set of distributions $P_\theta(x^{t-1}|x^t)$ that approximate the reversal of each conditional in Eq. (3). In doing so, we learn a map from simple noise to the data distribution, which can then be used to generate new data.

In traditional diffusion models, the forward process is made to be sufficiently fine-grained (using a large number of steps T) such that the conditional distribution of each step in the reverse process takes some simple form (such as Gaussian or categorical). This simple distribution is parameterized by a neural network, which is then trained to minimize the Kullback-Leibler (KL) divergence between the joint distributions Q and P_θ,

$$\mathcal{L}_{DN}(\theta) = D(Q(x^0, \ldots, x^T) \parallel P_\theta(x^0, \ldots, x^T)), \tag{4}$$

where the joint distribution of the model is the product of the learned conditionals:

$$P_\theta(x^0, \ldots, x^T) = Q(x^T) \prod_{t=1}^{T} P_\theta(x^{t-1}|x^t). \tag{5}$$

See Supplement B.2 for more details.

EBM-based denoising models approach the problem from a different angle[49]. In many cases, it is straightforward to re-cast the forward process in an exponential form,

$$Q(x^t|x^{t-1}) \propto e^{-\mathcal{E}^f_{t-1}(x^{t-1}, x^t)}, \tag{6}$$

where $\mathcal{E}^f_{t-1}$ is the energy function associated with the forward process step that adds noise to x^{t-1}. We then use an EBM with a particular energy function to model the conditional, i.e.,

$$P_\theta(x^{t-1}|x^t) \propto e^{-(\mathcal{E}^f_{t-1}(x^{t-1}, x^t) + \mathcal{E}^\theta_{t-1}(x^{t-1}, \theta))}. \tag{7}$$

In this parameterization, the dependence on x^t enters entirely through the forward energy $\mathcal{E}^f_{t-1}(x^{t-1}, x^t)$, which constrains x^{t-1} to stay close to the noisy input x^t, while $\mathcal{E}^\theta_{t-1}$ shapes the local structure of the distribution over x^{t-1}.

Equation (7) allows for a compromise between the number of steps in the approximation to the reverse process and the difficulty of sampling at each step. As the number of steps in the forward process is increased, the effect of each noising step becomes smaller, meaning that $\mathcal{E}^f_{t-1}$ more tightly binds x^t to x^{t-1}. This binding can simplify the distribution given in Eq. (7) by imposing an energy penalty that prevents it from being strongly multimodal; see Supplement B.4 for further discussion.

As illustrated in Fig. 2a, models of the form given in Eq. (7) reshape simple noise into an approximation of the data distribution. Increasing T while holding the EBM architecture constant simultaneously increases the expressive power of the chain and makes each step easier to sample from, entirely bypassing the MET.

To maximally leverage probabilistic hardware for EBM sampling, DTMs generalize Eq. (7) by introducing latent variables $\{z^t\}$:

$$P_\theta(x^{t-1}|x^t) \propto \sum_{z^{t-1}} e^{-(\mathcal{E}^f_{t-1}(x^{t-1}, x^t) + \mathcal{E}^\theta_{t-1}(x^{t-1}, z^{t-1}, \theta))}. \tag{8}$$

Introducing latent variables allows the size and complexity of the probabilistic model to be increased independently of the data dimension. In this generalized form, $\mathcal{E}^f_{t-1}$ still enforces proximity between x^{t-1} and x^t, while $\mathcal{E}^\theta_{t-1}$ uses the latent variables z^{t-1} to enrich the local conditional structure without directly depending on x^t.

A convenient property of DTMs is that if the approximation to the reverse-process conditional is exact ($P_\theta(x^{t-1}|x^t) \to Q(x^{t-1}|x^t)$), one also learns the marginal distribution at $t-1$,

$$Q(x^{t-1}) \propto \sum_{z^{t-1}} e^{-\mathcal{E}^\theta_{t-1}(x^{t-1}, z^{t-1}, \theta)}. \tag{9}$$

See Supplement B.6 and ref. 49 for further details and discussion of this property. Note that it relies on the normalizing constant associated with the distribution in Eq. (6) being independent of x^{t-1}.

Denoising thermodynamic computers

The Denoising Thermodynamic Computer Architecture (DTCA) tightly integrates DTMs into probabilistic hardware, allowing for the highly efficient implementation of EBM-aided diffusion models.

Practical implementations of the DTCA utilize natural-to-implement EBMs that exhibit sparse and local connectivity, as is typical in the literature[28]. This constraint allows sampling of the EBM to be performed by massively parallel arrays of primitive circuitry that implement Gibbs sampling. Refer to Supplements C, D for further theoretical discussion of the hardware architecture.

A key feature of the DTCA is that $\mathcal{E}^f_{t-1}$ can be implemented efficiently using our constrained EBMs. Specifically, for both continuous and discrete diffusion, $\mathcal{E}^f_{t-1}$ can be implemented using a single pairwise interaction between corresponding variables in x^t and x^{t-1}; see Supplements B.1, D.1 for details. This structure can be reflected in how the chip is laid out to implement these interactions without violating locality constraints.

Critically, Eq. (8) places no constraints on the form of $\mathcal{E}^\theta_{t-1}$. Therefore, we are free to use EBMs that our hardware implements especially efficiently. At the lowest level, this corresponds to high-dimensional, regularly structured latent-variable EBMs. If more powerful models are desired, these hardware latent-variable EBMs can be arbitrarily scaled by combining them into software-defined graphical models.

The modular nature of DTMs enables various hardware implementations. For example, each EBM in the chain can be implemented using distinct physical circuitry on the same chip, as shown in Fig. 2b. Alternatively, the various EBMs may be split across several communicating chips or implemented by the same hardware, reprogrammed with distinct sets of weights at different times. For any given EBM in the chain, both the data variables x^t, x^{t-1} and the latent variables z^{t-1} are physically embodied in sampling circuits that are connected in a simple way that reflects the structure of Eq. (7). This variable structure is shown schematically in Fig. 2c.

To understand the performance of a future hardware device, we developed a GPU simulator of the DTCA and used it to train a DTM on the Fashion-MNIST dataset. We measure the performance of the DTM using FID and utilize a physical model to estimate the energy required to generate new images (see Methods). These numbers can be compared to conventional algorithm/hardware pairings, such as a VAE running on a GPU; these results are shown in Fig. 1.

The DTM that produced the results shown in Fig. 1 used Boltzmann machine EBMs. Boltzmann machines, also known as Ising models in physics, use binary random variables and are the simplest type of discrete-variable EBM.

Boltzmann machines are hardware efficient because the Gibbs sampling update rule required to sample from them is simple. Boltzmann machines implement energy functions of the form

$$\mathcal{E}(x) = -\beta\left(\sum_{i\neq j} x_i J_{ij} x_j + \sum_{i=1} h_i x_i\right), \tag{10}$$

where each $x_i \in \{-1, 1\}$. The Gibbs sampling update rule for sampling from the corresponding EBM is

$$\mathbb{P}(X_i[k+1] = +1 | X[k] = x) = \sigma\left(2\beta\left(\sum_{j\neq i} J_{ij}\, x_j + h_i\right)\right), \tag{11}$$

where σ is the sigmoid function. This can be evaluated simply using an appropriately biased source of random bits.

Implementing our proposed hardware architecture using Boltzmann machines is particularly simple. A device will consist of nodes connected into a regular grid. Each node represents a single Bernoulli random variable x_i, implemented as a transistor-based sampling circuit (later referred to as RNG). The bias of a sampling circuit (the probability that it produces 1 as opposed to −1) is constrained to be a sigmoidal function of an input voltage, allowing its update probability to be conditioned by the values of its neighbors as given in Eq. (11). This can be implemented using a simple circuit that adds currents such as a resistor network (see Fig. 2d and it's caption for further details).

Specifically, the EBMs employed in this work were sparse, deep Boltzmann machines comprising $L \times L$ grids of binary variables, where $L = 70$ was used in most cases. Each variable was connected to several (in most cases, 12) of its neighbors following a simple pattern. At random, some of the variables were selected to represent the data x_{t-1}, and the rest were assigned to the latent variables z_{t-1}. Then, an extra node was connected to each data node to implement the coupling to x_t. See Supplement D for further details on the Boltzmann machine architecture.

Due to our chosen connectivity patterns, our Boltzmann machines are bipartite (two-colorable), meaning that nodes can be separated into two blocks, such that no two nodes within the same block are connected to each other. Since each color block can be sampled in parallel, a single iteration of Gibbs sampling corresponds to first sampling one color block conditioned on the other and then swapping their roles. Therefore, the total duration of one full Gibbs iteration (so updating all nodes) takes roughly $2\tau_0$ in wall-clock time, where τ_0 is the decorrelation time of an individual RNG, that is the time it takes for a sampling circuit inside a single node of the Boltzmann machine to equilibrate. It should be noted that τ_0 is set by the intrinsic stochastic dynamics of the subthreshold circuit and its operating point. It is not a freely tunable algorithmic parameter, but can only be modified indirectly through circuit redesign.

Starting from some random initialization, this block-sampling procedure can be repeated for K iterations (where K is longer than the mixing time of the sampler, typically $K \approx 1000$) to draw samples from Eq. (7) for each step in the approximation to the reverse process. For a DTM running on a DTCA chip, the total time required to draw a single sample is therefore proportional to $TK\,\tau_0$, where T is the number of denoising steps (i.e., EBMs that the DTM is comprised of). It is important to distinguish τ_0 from K_{mix}. The former is an entirely hardware-related parameter, while the latter is the number of Gibbs iterations that the abstract Boltzmann machine (irrespective of hardware implementation) needs to approximately converge to its equilibrium distribution.

To enable a near-term, large-scale realization of the DTCA, we leveraged the shot-noise dynamics of subthreshold transistors[53] to build an RNG that is fast, energy-efficient, and small. Our all-transistor RNG is programmable and has the desired sigmoidal response to a control voltage, as shown by experimental measurements in Fig. 3a. The stochastic voltage signal output from the RNG has an approximately exponential autocorrelation function that decays in ~100 ns, as illustrated in Fig. 3b. As shown in ref. 53, this time constant is much larger than the lower limit imposed by the correlation time of the noise in our transistors. The RNG could, therefore, be made much faster via an improved design. Supplement K provides further details about our RNG.

In practice, semiconductor manufacturing is never perfectly precise. Each fabrication step introduces small deviations in transistor characteristics, producing global shifts across wafers as well as random differences between neighboring devices on the same chip. Because these variations can change how fast or efficiently a circuit operates, it is important to verify that our approach remains reliable under realistic fabrication conditions. A practical advantage of our all-transistor RNG is that detailed and proven foundry-provided models can be used to study the effect of manufacturing variations on our circuit design. In Fig. 3c, we use this process development kit (PDK) to study the speed and energy consumption of our RNG as a function of both systematic inter-wafer skews to the transistor parameters (process corners) and the expected variation within a single chip. We find that the RNG works reliably despite these non-idealities, meaning it can be readily scaled to the massive grids required by the DTCA.

Training DTMs

A well-trained denoising model generates new examples that resemble the training data by incrementally pulling them out of noise; the outputs of an 8-step denoising model trained on the Fashion-MNIST dataset are shown in Fig. 4 (a). At the final time T, the images are random bits. Structure begins to emerge as the chain progresses, ultimately resulting in clean images at time $t = 0$.

DTMs alleviate the training instability that is fundamental to MEBMs. The parameters of MEBMs are usually initialized using a strategy that results in an easy-to-sample-from energy landscape[54]. For this reason, in the early stages of training, sampling is efficient and gradient estimates are unbiased. However, as these gradients are followed, the MEBM is reshaped according to the data distribution and begins to become complex and multimodal. This induced multimodality greatly increases the sampling complexity of the distribution, causing samples to deviate from equilibrium. Gradients computed using non-equilibrium samples do not necessarily point in a meaningful direction, which can halt or, in some cases, even reverse the training process.

This instability in MEBMs leads to unpredictable training dynamics that can be sensitive to implementation details. An example of the training dynamics for several different types of models is shown in Fig. 4b. The top plot displays the quality of images generated during training, while the bottom plot shows a measure of the sampler's mixing. Image quality is measured using the FID metric, and mixing quality is measured using normalized autocorrelation r_{yy} (see Supplement G). The lower plot in Fig. 4b shows the autocorrelation at a delay equal to the total number of sampling iterations used to estimate the gradients during training. Generally, if r_{yy} is close to 1, gradients were estimated using far-from-equilibrium samples and were likely of low quality. If it is close to zero, the samples should be close to equilibrium and produce high-quality gradient estimates. See Supplement G for further discussion.

The destabilizing effect of non-equilibrium sampling is apparent from the blue curves in Fig. 4b. At the beginning of training, both quality and r_{yy} increase, indicating that the multimodality of the data is being imprinted on

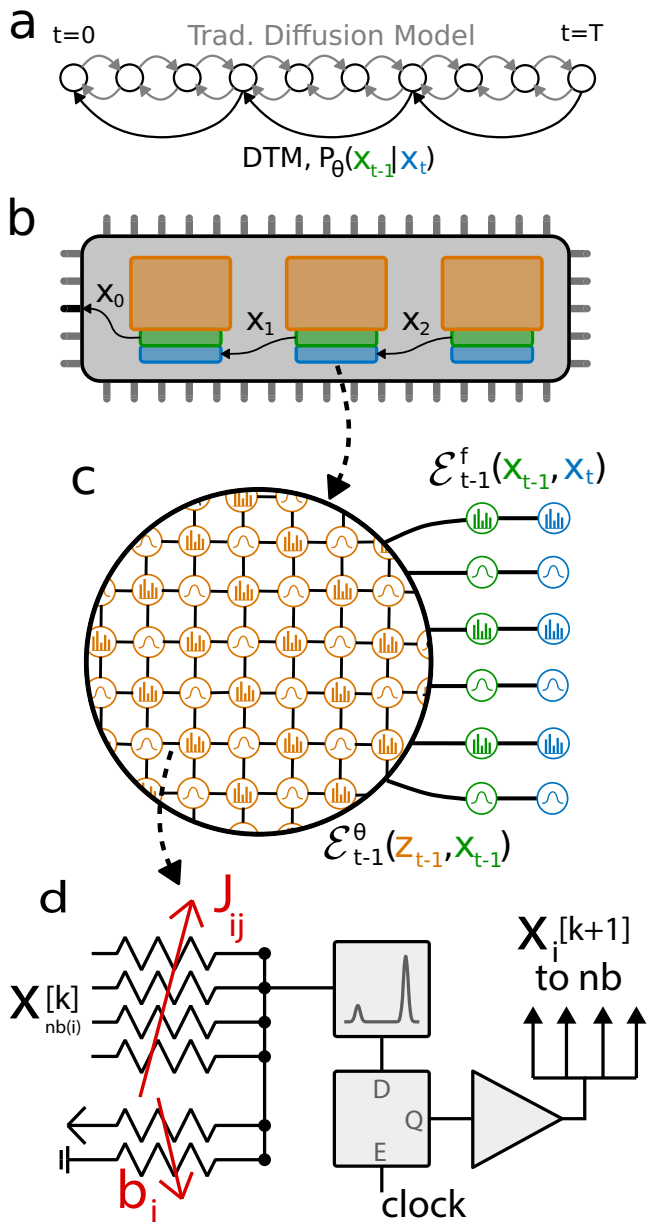

Fig. 3 | The denoising thermodynamic computer architecture. a Traditional diffusion models have simple conditionals and must take small steps when approximating the reverse process. Since EBMs can express more complex distributions, DTMs can take potentially much larger steps. **b** A sketch of how a chip based on the DTCA chains hardware EBMs to approximate the reverse process. Each EBM is implemented by distinct circuitry, parts of which are dedicated to receiving the inputs and conditionally sampling the outputs and latents. **c** An abstract diagram of a hardware EBM. The state variables x^t and x^{t-1} map onto distinct physical degrees of freedom represented by the blue and green nodes, respectively. The coupling between these two sets of nodes implements the forward process energy function $\mathcal{E}^f_t(x^{t-1}, x^t)$. The set of orange nodes represents a set of latent variables z^{t-1}. The couplings between these nodes and to the x^{t-1} nodes implements $\mathcal{E}^\theta_{t-1}(z^{t-1}, x^{t-1})$. **d** A mixed-signal cell that implements the Boltzmann machine update given by Eq. (11). The current states of the neighboring cells of node i ($x_{\mathrm{nb}(i)}[k]$) are received as voltages over wires and are used to compute a bias voltage using a tunable resistor network (see Supplement E.1). This bias voltage sigmoidally tunes the output distribution of an RNG (see Fig. 3). After waiting out the RNG's correlation time, a clock signal enables a d-latch that memorizes the RNG's current output voltage, producing $x_i[k+1]$. A driver amplifier is then used to broadcast this updated state out to the neighboring cells so they may be updated in the same way.

the model. As training progresses and the energy landscape becomes more rugged, r_{yy} becomes so large that the quality of the gradient starts to decline, resulting in a plateau and, ultimately, a degradation of the model's quality.

Denoising alone significantly stabilizes training. Because the transformation carried out by each layer is simpler, the distribution that the model must learn is less complex and, therefore, easier to sample from. The orange curve in Fig. 4b shows the training dynamics for a typical denoising model. The autocorrelation and performance remain good for much longer than the MEBM.

As training progresses, the DTM eventually becomes unstable, which can be attributed to the development of a complex energy landscape among the latent variables. To combat this, we employed an Adaptive Correlation Penalty (ACP) training procedure. This method penalizes models that mix poorly and dynamically adjusts the penalty strength to ensure sampling remains tractable for each layer. The green curves in Fig. 4b show an example of training dynamics under this closed-loop control policy. Model quality increases monotonically, and the autocorrelation stays small throughout training. This closed-loop control of the correlation penalty was employed during the training of most models used to produce the results in this article, including those shown in Fig. 1. See the Methods section for the mathematical formulation of the ACP.

Generally, the performance of DTMs improves as their size increases. As shown in Fig. 1, increasing the depth of the DTM from 2 to 8 substantially improves the quality of generated images. As shown in Fig. 4c, increasing the width, degree, and allowed mixing time of the EBMs in the chain also generally improves performance.

However, some subtleties prevent this specific EBM topology from being scaled indefinitely. The top plot in Fig. 4c shows that scaling the number of latent variables (with fixed allowed mixing time) only improves performance if the connectivity of the graph is also scaled; otherwise, performance can decrease. This dependence makes sense, as increasing the number of latent variables in this way increases the depth of the Boltzmann machine, which is known to make sampling more difficult. Beyond a certain point, increasing the model's ability to express complex energy landscapes may render it unable to learn, given the allowed mixing time of $K \approx 1000$. This same effect is shown in the bottom plot of Fig. 4c, which demonstrates that larger values of K are required to support wider models while holding connectivity constant. Further study of how performance scales with sampling time K and the number of denoising steps T is provided in Supplement M.

In general, it would be naive to expect that a hardware-efficient EBM topology can be scaled in isolation to model arbitrarily complex datasets. There is no good reason why a connectivity pattern that is convenient from a wire-routing perspective would also be well suited to represent the correlation structure of a complex real-world dataset.

Hybrid thermodynamic-deterministic machine learning

The core doctrine of modern machine learning is the relentless scaling of models as a means of solving ever-harder problems. Models that utilize probabilistic computers may be similarly scaled to enhance their capabilities beyond the relatively simple dataset considered in this work so far.

However, we hypothesize that the correct way to scale probabilistic machine learning hardware systems is not in isolation but rather as a component in a larger *hybrid thermodynamic-deterministic machine learning* (HTDML) system. Such a hybrid system integrates probabilistic hardware with more traditional machine learning accelerators. This hybrid approach is particularly important for moving beyond binarized grayscale datasets such as Fashion-MNIST toward richer, higher-dimensional data.

A hybrid approach is sensible because there is no a priori reason to believe that a probabilistic computer should handle every part of a machine learning problem, and sometimes a deterministic processor is likely a better tool for the job.

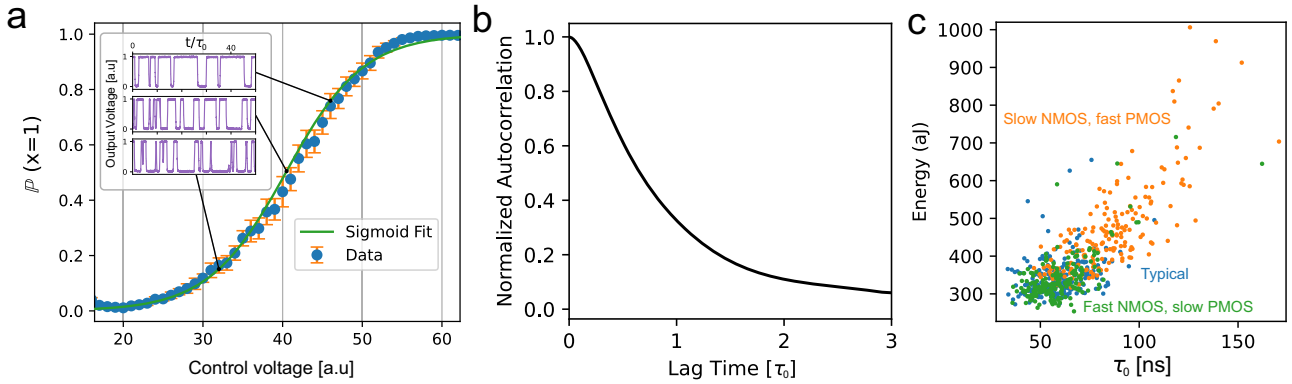

Fig. 4 | A programmable source of random bits. a A laboratory measurement of the operating characteristic of our RNG. The probability of the output voltage signal being in the high state ($x = 1$) can be programmed by varying an input voltage. The relationship between $\mathbb{P}(x = 1)$ and the input voltage is well-approximated by a sigmoid function. The inset shows the output voltage signal as a function of time for different input voltages. **b** The autocorrelation function of the RNG at the unbiased point ($\mathbb{P}(x = 1) = 0.5$). The decay is approximately exponential with the rate $\tau_0 \approx 100$ ns. **c** Estimating the effect of manufacturing variation on RNG performance. Each point in the plot represents the results of a simulation of an RNG circuit with transistor parameters sampled according to a procedure defined by the manufacturer's PDK. Each color represents a different process corner, each for which ~200 realizations of the RNG were simulated. The "typical" corner represents a balanced case, whereas the other two are asymmetric corners where the two types of transistors (NMOS and PMOS) are skewed in opposite directions. The slow NMOS and fast PMOS case is worst performing for us due to an asymmetry in our design.

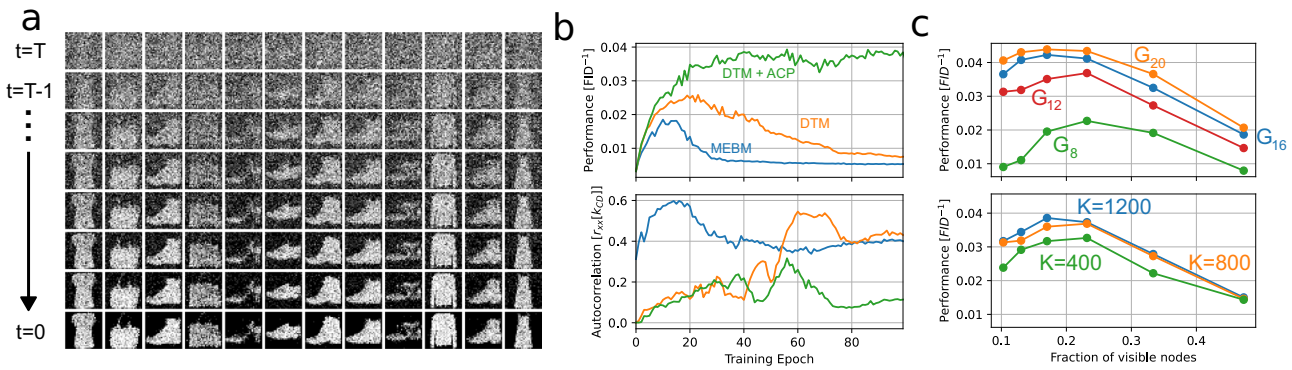

Fig. 5 | Detailed results on the Fashion-MNIST dataset. a Images generated by a denoising model. Here, to achieve better-looking images, several binary variables were combined to represent a single grayscale pixel. The noisiness of the grayscale levels is an artifact of our embedding method; see Supplement I. **b** An experiment showing how DTMs are more stable to train than MEBMs. Complementing DTMs with the ACP completely stabilizes training. For the DTMs, the maximum $r_{yy}[K]$ value over all the layers is shown. **c** The effect of scaling EBM complexity on DTM performance. The grid size L was modified to change the number of latent variables compared to the (fixed) number of data variables. Generally, EBM layers with more connectivity and longer allowed mixing times can utilize more latent variables and, therefore, achieve higher performance.

The goal of HTDML is to design practical machine learning systems that minimize the energy used to achieve desired modeling fidelity on a particular task. This efficiency will be achieved through a cross-disciplinary effort that eschews the software/hardware abstraction barrier to design computers that respect physical constraints and leverage each type of hardware where it is most effective.

For example, more rigorous methods of embedding data into hardware EBMs will need to be developed to go beyond the relatively simple datasets considered here. Indeed, binarization is not viable in general, and embedding into richer types of variables (such as categorical) at the probabilistic hardware level is not particularly efficient or principled. Instead, one may try a hybrid (HTDML) approach where a small neural network (the deterministic part) learns to embed the data into a binary latent space and a DTM (the thermodynamic part) can then be trained inside this binary space. A similar approach has been very beneficial in the context of diffusion models[50], to the point that now many state-of-the-art diffusion models operate in the latent space of a pre-trained VAE.

We show the results of a quick initial attempt at such hybrid models in Fig. 5. Here, we train a small neural network to embed the CIFAR-10 dataset[55] into a binary DTM. The embedding network was trained using an autoencoder loss to binarize the data, which was then used to train a DTM. The decoder of the embedding network was then trained further using a GAN objective to increase the quality of the generated images. This training procedure is described in further detail in Supplement J. Example CIFAR-10 images generated by our model can be found in Supplement N.

Despite the overhead of the embedding neural network, this primitive hybrid model is efficient. As shown in the figure, to achieve equivalent performance with a purely NN-based GAN, the deterministic part of that GAN would need to have roughly ten times more parameters than the deterministic part of our hybrid model. Informally, one might say that therefore the DTM holds about 90% of the expressivity of our model while the neural network accounts for only about 10% of the parameters.

We should again stress that this is just a naive first attempt at a hybrid model, and we believe that further research into hybrid thermodynamic-deterministic models can lead to even more significant efficiency gains and scale this approach to larger tasks. For example, one major flaw of the embedding procedure employed in Fig. 6 is that the autoencoder and DTM are not jointly trained, which means that the embedding learned by the autoencoder may not be well-suited to the way information can flow in the

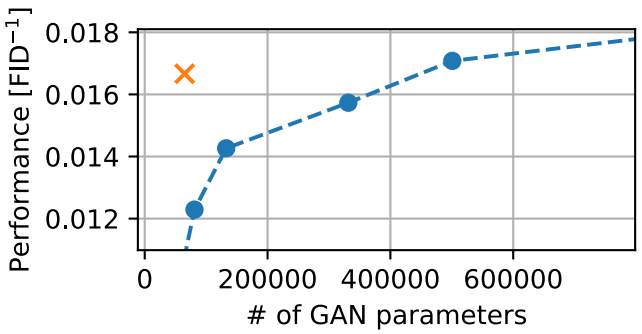

Fig. 6 | Embedding data into a DTM using a neural network. Here, we show the results of using a simple embedding model in combination with a DTM. The DTM is trained to generate CIFAR-10 images and achieves performance parity with a traditional GAN using a ~10× smaller deterministic neural network in terms of parameter count.

DTM, given its limited connectivity. Finding a good way to jointly train them seems like a promising future research direction.

Discussion

The broad analysis presented in this manuscript, which spans from high-level algorithmic ideas to laboratory measurements of novel analog circuits, establishes, for the first time, that a probabilistic computing system could substantially outperform traditional AI hardware. Taken as a whole, this work presents a compelling case for significant investment in the continued development of low-energy probabilistic computing systems.

However, the presented system should be interpreted as a first step towards a new type of AI system, rather than a finished and polished product in itself. While our benchmark results show a several orders-of-magnitude improvement in efficiency on a small problem, they also highlight several issues with our approach that must be addressed to scale it further. Moreover, our analysis is approximate and omits many practically important details that will require further refinement of our hardware and algorithms in the future.

While the data shown in Figs. 1 and 5 cleanly illustrates the potential efficiency advantage of probabilistic hardware *at a given level of performance*, it also unambiguously shows that our proposed system is very far from the state-of-the-art in terms of raw modeling power. Cutting-edge neural-network-based architectures are able to achieve substantially lower FID scores than our probabilistic model (at the expense of drastically more energy use) on the Fashion MNIST and CIFAR-10 benchmarks[56,57]. Moreover, the datasets considered in this paper are far less challenging than those that are used to evaluate cutting-edge LLM systems.

This elucidates a critical gap in our research that should be the primary focus of future study: unlike traditional neural networks, we don't yet know how to build progressively larger probabilistic AI systems capable of representing arbitrarily complex data. A concrete example of this is shown in Fig. 4: simply scaling the size or connectivity of a hardware EBM used in a DTM does not improve performance beyond a certain threshold.

Solving this capabilities-scaling issue will involve both figuring out how to most efficiently leverage probabilistic hardware and how to best combine probabilistic hardware with traditional neural networks (as in the HTDML paradigm discussed in the previous section).

The space of possibilities for effectively scaling probabilistic computers is almost entirely unexplored, and significant performance gains are likely being left on the table. This work (and also related prior art[28]) only considered simple EBM architectures that fit naively into the hardware (sparse, local, regular connectivity), which could severely limit the class of distributions that the hardware can represent efficiently. Future work could examine architectures that fuse multiple EBMs to implement each step in the reverse process, for example, using software-defined graphical models that enable non-local information routing.

It's important to stress that the results presented in this paper were based on a moderately complex physical model of the computing system and leave out several effects that may hinder the efficiency and efficacy of a real probabilistic computer. Indeed, there are so many such effects that it would be nearly impossible to consider them all in a theoretical study; the most effective way to prove that this concept works in real life is to demonstrate it on a real chip.

The most critical and risky part of our proposal is the all-transistor RNG, which is why we grounded its contribution to the modeling in real laboratory measurements and detailed simulations that leveraged the chip manufacturer's PDK. Figure 3c shows that the speed and energy consumption of our RNG is robust to the extreme process variation that is typical in subthreshold circuits, and Fig. 3a shows that the measured bias curve of a real RNG is sigmoidal (within error bars) despite all of the non-idealities associated with a real circuit.

However, there are many other effects that we ignored. For example, a real transistor implementation of the biasing circuit shown in Fig. 2d will not be completely linear, which will lead to deviations in the conditional update away from Eq. (11). Additionally, all real circuits experience drift[58], which can degrade the performance of an initially well-calibrated and accurate analog computer over time. Moreover,even though the bias curve shown in Fig. 3a is *almost* sigmoidal, it is of course not exact, and could deviate further from ideal in the case of more extreme process variation.

Compensating for the myriad complications associated with real circuits constitutes the main challenge that we face in moving from this proposal to building a real chip that implements the DTCA. Interestingly, our preliminary results indicate that the effects of many of these non-idealities are less than the error introduced by quantizing the model weights (e.g., at 5–8 bits of precision). This suggests that the DCTA strikes a good balance between analog and digital computation[59], keeping the analog part local and simple and relying on robust digital signaling for longer-range communication. Only time (and further engineering work) will tell how far these complications will degrade the performance of a real system away from the idealized one studied in this manuscript.

The experiments presented in this paper were done using simulations of thermodynamic hardware, running on a traditional computer, which substantially limited the maximum size of the thermodynamic model we could represent. Specifically, the largest DTM shown in Fig. 1 uses only around 50,000 nodes in total, which is much smaller than what we expect the capacity of early DTCA chips will be. In fact, based on the size of our RNG, it can be estimated that ~10^6 sampling cells could be fit into a 6 × 6 μm chip (see Supplement K). Furthermore, measurements of our RNG suggest that early DTCA chips will be about two to three orders of magnitude faster than the simulations used in this article.

Realizing large-scale probabilistic computers using advanced transistor processes[60–62] will substantially speed up future research into other thermodynamic algorithms.

Methods

Energy estimation model

The energy estimates given in Fig. 1 for the probabilistic computer were constructed using a physical model of an all-transistor Boltzmann machine Gibbs sampler. The dominant contributions to this model are captured by the formula

$$E = TK_{\text{mix}}L^2E_{\text{cell}}, \tag{12}$$

K_{mix} is the number of sampling iterations required to satisfactorily mix the chain for inference, which is generally less than the number of iterations used during training. $K_{\text{mix}} = 250$ was used for the DTM (see Supplement E.4), while the mixing time measured in Fig. 2 was used for the MEBM.

$E_{cell} \approx 2$fJ models the energy consumed by the sampling cell shown in Fig. 2d per conditional update,

$$E_{cell} = E_{rng} + E_{bias} + E_{comm} + E_{clock}, \tag{13}$$

E_{rng} models the energy consumption of the RNG, and is taken from the data shown in Fig. 3c. E_{bias} takes into account the energy consumption of the biasing circuit, which depends on the exact configuration of the resistances and input voltages but has a simple conservative bound, see Supplement E.1. E_{comm} models the cost of communicating the state of a cell to all of it's neighbors which is dominated by the energy required to charge up the parasitic capacitance associated with long wires, see Supplement E.2. E_{clock} models the cost of sending a clock signal to every cell in the chip.

An efficient chip design will tend to balance each of the terms in Eq. (13) such that one does not strongly dominate over any of the others. This is possible for the sparse, locally connected Boltzmann machines considered in this work; see Supplement E.4 for a breakdown of how the energy is distributed among the terms. Designs with more or longer-range connections between each cell and its neighbors will have energy consumption that is increasingly dominated by E_{comm}. For our chosen connectivity patterns, E_{comm} accounts for about half of the energy budget.

Equation (13) is a high-level approximate model of the energy consumption of a real chip, but should be reasonably accurate for inference-dominated workloads, see Supplement E.5 for a longer discussion on its limitations.

Broadly, the most important contribution ignored by Eq. (13) is communication between the probabilistic chip and external hardware. The dominant considerations here are the flashing of the model's weights to the probabilistic chip and the transfer of samples from the model back to a host computer. In an inference scenario, the weights are flashed once, and then many samples are taken from the model; the energy consumption of the flash will be relatively small as long as sufficiently many samples are taken with a given set of weights. Similarly, since many internal sampling iterations are completed per sample that is seen by external hardware, the relative cost of data transfer off of the chip can be very small. The fact that the cost of external communication can be asymptotically driven to zero by increasing the complexity of the program run on the probabilistic chip is a nice property that makes a general case for hardware sampling accelerators.

We use a simple model for GPU energy consumption that underestimates the actual values. We compute the total number of floating-point operations (FLOPs) required to generate a sample from the trained model and divide it by the manufacturer's FLOP/joule specification. See Supplement F for further discussion.

Gradient estimation

The EBMs used in the experiments presented in Fig. 1 were trained by applying the standard Monte Carlo estimator for the gradients of EBMs[63] to Eq. (4), which yields

$$\nabla_\theta \mathcal{L}_{DN}(\theta) = \sum_{t=1}^{T} \mathbb{E}_{Q(x_{t-1},x_t)}[\mathbb{E}_{P_\theta(z_{t-1}|x_{t-1},x_t)}[\nabla_\theta \mathcal{E}^m_{t-1}] - \mathbb{E}_{P_\theta(x_{t-1},z_{t-1}|x_t)}[\nabla_\theta \mathcal{E}^m_{t-1}]]. \tag{14}$$

Notably, each term in the sum over t can be computed independently. To estimate either term in Eq. (14), first, sample tuples (x_{t-1}, x_t) from the forward process $Q(x_{t-1}, x_t)$. Then, for each of these tuples, clamp the reverse process EBM to the sampled values appropriately and use a time average over K iterations of Gibbs sampling to estimate the inner expectation value. Averaging the result over the tuples yields the desired gradient estimate.

It should be noted that the DTCA allows our EBMs to have finite and short mixing times, which enables sufficient sampling iterations to be used to achieve nearly unbiased estimates of the gradient. Unbiased gradient estimates are not possible for MEBMs in most cases due to their long mixing times[64].

Adaptive correlation penalty

We add a term to the loss function that nudges the optimization towards a distribution that is easy to sample from, i.e.,

$$\mathcal{L}^{TC}_t = \mathbb{E}_{Q(x^t)}\left[D\left(\prod_{i=1}^{M} P_\theta(s_i^{t-1}|x^t) \parallel P_\theta(s^{t-1}|x^t)\right)\right], \tag{15}$$

where $s^{t-1} = (x^{t-1}, z^{t-1})$ and x_i^{t-1} indicates the i^{th} of the M variables in x^{t-1}. This term penalizes the distance between the learned conditional distribution and a factorized distribution with identical marginals and is a form of total correlation penalty[65]. See Supplements H.2 and M for further discussion of this penalty and its relationship to mixing time and sampling effort.

The total loss function is the sum of Eq. (4) and this total correlation penalty:

$$\mathcal{L} = \mathcal{L}_{DN} + \sum_{t=1}^{T} \lambda_t \mathcal{L}^{TC}_t. \tag{16}$$

The parameters λ_t control the relative strength of the total correlation penalty for each step in the reverse process.

We use an Adaptive Correlation Penalty (ACP) to set the λ_t as large as necessary to keep sampling tractable for each layer. During training, we periodically measure the autocorrelations of each learned conditional at a delay equal to the number of sampling iterations used during gradient estimation. If the autocorrelation for the j^{th} layer is close to zero, λ_j is decreased, and vice versa.

Circuit measurements

The measurements presented in Fig. 3 were taken using an integrated circuit chip dedicated to the purpose of testing probabilistic circuits. The chip used for this paper was another instance of the same design that we used for the experiments in ref. 53. Note that the purpose of this chip was the scientific study of individual circuits, and it does not itself implement an ising machine or any other computational architecture.

Simulations

To address the current unavailability of probabilistic hardware, we have open-sourced a software library[66] that enables XLA-accelerated[67] simulation of hardware EBMs. This library is written in JAX[68] and automates the complex slicing operations that enable hardware EBM sampling. We also provide additional code that wraps this library to implement the specific experiments presented in this article github.com/pschilliOrange/dtm-replication. By making these tools available, we aim to lower the barrier for others to explore alternative thermodynamic algorithms, hardware topologies, and hybrid architectures, and to stress-test and refine the DTCA design in a broader range of applications.

Data availability

The data used to train the models in this study are the Fashion-MNIST dataset[22] and the CIFAR-10 dataset[60], both of which are publicly available.

Code availability

The code which reproduces our results and data is available at github.com/pschilliOrange/dtm-replication. The underlying simulation library is available at ref. 66.

Received: 6 October 2025; Accepted: 28 May 2026;
Published online: 02 July 2026

References

1. Chien, A. A. GenAI: Giga$$, TeraWatt-Hours, and GigaTons of CO_2. *Commun. ACM* **66**, 5 (2023).

2. Stine, D. D. *The Manhattan Project, the Apollo Program, and Federal Energy Technology r&d Programs: A Comparative Analysis*. Report RL34645, Congressional Research Service (Washington, D.C., 2009).
3. Aljbour, J., Wilson, T. & Patel, P. Powering Intelligence: Analyzing Artificial Intelligence and Data Center Energy Consumption. EPRI White Paper no. 3002028905, https://www.epri.com/research/products/000000003002028905 (2024).
4. Li, Y. et al. Competition-level code generation with AlphaCode. *Science* **378**, 1092–1097 (2022).
5. Katz, D. M., Bommarito, M. J., Gao, S. & Arredondo, P. GPT-4 passes the bar exam. *Philos. Trans. R. Soc. A* **382**, 20230254 (2024).
6. Nori, H., King, N., McKinney, S. M., Carignan, D. & Horvitz, E. Capabilities of gpt-4 on medical challenge problems. *arXiv* https://doi.org/10.48550/arXiv.2303.13375 (2023).
7. Noy, S. & Zhang, W. Experimental evidence on the productivity effects of generative artificial intelligence. *Science* **381**, 187–192 (2023).
8. Brynjolfsson, E., Li, D. & Raymond, L. Generative AI at work. *Q. J. Econ.* **140**, 889–942 (2025).
9. Peng, S., Kalliamvakou, E., Cihon, P. & Demirer, M. The impact of ai on developer productivity: evidence from github copilot. *arXiv* https://doi.org/10.48550/arXiv.2302.06590 (2023).
10. Bick, A., Blandin, A. & Deming, D. J. *The Rapid Adoption of Generative AI*. Tech. Rep. (National Bureau of Economic Research, 2024).
11. Vaswani, A. et al. Attention is all you need. In *Advances in Neural Information Processing Systems*, vol. 30 (eds Guyon, I. et al.) (Curran Associates, Inc., 2017).
12. Coates, A. et al. Deep learning with cots hpc systems. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, (JMLR.org, 2013).
13. Chellapilla, K., Puri, S. & Simard, P. High performance convolutional neural networks for document processing. In *Tenth International Workshop on Frontiers in Handwriting Recognition*. Université de Rennes 1 https://inria.hal.science/inria-00112631 (Suvisoft, 2006).
14. Hooker, S. The hardware lottery. *Commun. ACM* **64**, 58–65 (2021).
15. Ambrogio, S. et al. An analog-AI chip for energy-efficient speech recognition and transcription. *Nature* **620**, 768–775 (2023).
16. Bandyopadhyay, S. et al. Single-chip photonic deep neural network with forward-only training. *Nat. Photon.* **18**, 1335–1343 (2024).
17. Gonzalez, H. A. et al. SpiNNaker2: A large-scale neuromorphic system for event-based and asynchronous machine learning. *arXiv* https://doi.org/10.48550/arXiv.2401.04491 (2024).
18. Shrestha, S. B., Timcheck, J., Frady, P., Campos-Macias, L. & Davies, M. Efficient Video and Audio Processing with Loihi 2. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 13481–13485 (IEEE, 2024).
19. Song, Y. & Ermon, S. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, vol. 32 (eds Wallach, H. et al.) (Curran Associates, Inc., 2019).
20. Janner, M., Du, Y., Tenenbaum, J. & Levine, S. Planning with Diffusion for Flexible Behavior Synthesis. In *International Conference on Machine Learning*, 9902–9915 (PMLR, 2022).
21. Conte, T. et al. Thermodynamic computing. *arXiv* https://doi.org/10.48550/arXiv.1911.01968 (2019).
22. Singh, N. S. et al. CMOS plus stochastic nanomagnets enabling heterogeneous computers for probabilistic inference and learning. *Nat. Commun.* **15**, 2685 (2024).
23. Pratt, C., Ray, K. & Crutchfield, J. Dynamical computing on the nanoscale: superconducting circuits for thermodynamically-efficient classical information processing, https://doi.org/10.48550/arXiv.2307.01926 (2023).
24. Wimsatt, G. et al. Harnessing fluctuations in thermodynamic computing via time-reversal symmetries. *Phys. Rev. Res.* **3**, 033115 (2021).
25. Adachi, S. H. & Henderson, M. P. *Application of Quantum Annealing to Training of Deep Neural Networks*, https://doi.org/10.48550/arXiv.1510.06356 (2015).
26. Sutton, B., Camsari, K. Y., Behin-Aein, B. & Datta, S. Intrinsic optimization using stochastic nanomagnets. *Sci. Rep.* **7**, 44370 (2017).
27. Faria, R., Camsari, K. Y. & Datta, S. Low-barrier nanomagnets as p-bits for spin logic. *IEEE Magn. Lett.* **8**, 1–5 (2017).
28. Niazi, S. et al. Training deep Boltzmann networks with sparse Ising machines. *Nat. Electron.* **7**, 610–619 (2024).
29. Borders, W. A. et al. Integer factorization using stochastic magnetic tunnel junctions. *Nature* **573**, 390–393 (2019).
30. Sajeeb, M. M. H. et al. Scalable connectivity for ising machines: dense to sparse. *Phys. Rev. Appl.* **24**, 014005 (2025).
31. Du, Y. & Mordatch, I. Implicit generation and modeling with energy based models. In *Advances in Neural Information Processing Systems*, vol. 32 (eds Wallach, H. et al.) (Curran Associates, Inc., 2019).
32. Lee, W. et al. Correlation free large-scale probabilistic computing using a true-random chaotic oscillator p-bit. *Sci. Rep.* **15**, 8018 (2025).
33. Horodynski, M. et al. Stochastic logic in biased coupled photonic probabilistic bits. *Commun. Phys.* **8**, 31 (2025).
34. Abeed, M. A. & Bandyopadhyay, S. Low energy barrier nanomagnet design for binary stochastic neurons: design challenges for real nanomagnets with fabrication defects. *IEEE Magn. Lett.* **10**, 1–5 (2019).
35. Abeed, M. A. & Bandyopadhyay, S. Sensitivity of the power spectra of thermal magnetization fluctuations in low barrier nanomagnets proposed for stochastic computing to in-plane barrier height variations and structural defects. *SPIN* **10**, 2050001 (2020).
36. Drobitch, J. L. & Bandyopadhyay, S. Reliability and scalability of p-bits implemented with low energy barrier nanomagnets. *IEEE Magn. Lett.* **10**, 1–4 (2019).
37. Jordan, M. I. & Mitchell, T. M. Machine learning: trends, perspectives, and prospects. *Science* **349**, 255–260 (2015).
38. Ghahramani, Z. Probabilistic machine learning and artificial intelligence. *Nature* **521**, 452–459 (2015).
39. Hinton, G. *Boltzmann Machines: Constraint Satisfaction Networks that Learn* (Carnegie-Mellon University, Department of Computer Science, 1984).
40. Dempster, A. P., Laird, N. M. & Rubin, D. B. *Maximum likelihood from incomplete data via the EM algorithm*. *J. R. Stat. Soc. Ser. B Methodol.* **39**, 1–22 (1977).
41. Bishop, C. M. Mixture density networks https://api.semanticscholar.org/CorpusID:118227751 (1994).
42. Murphy, K. P. *Probabilistic Machine Learning: Advanced Topics* http://probml.github.io/book2 (MIT Press, 2023).
43. Carbone, D., Hua, M., Coste, S. & Vanden-Eijnden, E. Efficient training of energy-based models using Jarzynski equality. *Adv. Neural Inf. Process. Syst.* **36**, 52583–52614 (2023).
44. Desjardins, G. et al. Parallel tempering for training of restricted Boltzmann machines. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 145–152 (MIT Press, 2010).
45. Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N. & Ganguli, S. *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*. In *Proceedings of the 32nd International Conference on Machine Learning*, vol. 37 of *Proceedings of Machine Learning Research* (eds Bach, F. & Blei, D.) 2256–2265 (PMLR, 2015).
46. Ho, J. et al. Cascaded diffusion models for high fidelity image generation. *J. Mach. Learn. Res.* **23**, 1–33 (2022).
47. Rombach, R., Blattmann, A., Lorenz, D., Esser, P. & Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10684–10695 (IEEE, 2022).

48. Saharia, C. et al. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems*, vol. 35, (eds, Koyejo, S. et al.) 36479–36494 (Curran Associates, Inc., 2022).
49. Gao, R., Song, Y., Poole, B., Wu, Y. N. & Kingma, D. P. Learning energy-based models by diffusion recovery likelihood. *arXiv* https://arxiv.org/abs/2012.08125 (2021).
50. Yu, P. et al. Latent diffusion energy-based model for interpretable text modelling. In *International Conference on Machine Learning*, 25702–25720 (PMLR, 2022).
51. Xu, M. et al. Energy-based diffusion language models for text generation. *arXiv* https://doi.org/10.48550/arXiv.2410.21357 (2024).
52. Zhu, Y., Xie, J., Wu, Y. N. & Gao, R. Learning energy-based models by cooperative diffusion recovery likelihood. In *The Twelfth International Conference on Learning Representations* https://doi.org/10.48550/arXiv.2212.00168 (2021).
53. Freitas, N. et al. Taming non-equilibrium thermal fluctuations in subthreshold CMOS circuits. *Phys. Rev. App*. **25**, 034061 (2026).
54. Hinton, G. E. A practical guide to training restricted Boltzmann machines. In *Neural Networks: Tricks of the Trade: Second Edition*, 599–619 (Springer, 2012).
55. Krizhevsky, A. & Hinton, G. *Learning Multiple Layers of Features from Tiny Images*. Tech. Rep. 0 (University of Toronto, Toronto, Ontario, 2009).
56. Kim, D., Kim, Y., Kwon, S. J., Kang, W. & Moon, I.-C. Refining generative process with discriminator guidance in score-based diffusion models. Preprint at https://arxiv.org/abs/2211.17091 (2022).
57. Silvestri, G., Ambrogioni, L., Lai, C.-H., Takida, Y. & Mitsufuji, Y. Training consistency models with variational noise coupling. In *ICLR 2025 Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficacy* (ICLR, 2025).
58. Keshner, M. S. 1/f noise. *Proc. IEEE* **70**, 212–218 (1982).
59. Sarpeshkar, R. Analog versus digital: extrapolating from electronics to neurobiology. *Neural Comput.* **10**, 1601–1638 (1998).
60. Sekigawa, T. & Hayashi, Y. Calculated threshold-voltage characteristics of an XMOS transistor having an additional bottom gate. *Solid State Electron.* **27**, 827–828 (1984).
61. Wu, S.-Y. et al. A 3nm CMOS FinFlex™ platform technology with enhanced power efficiency and performance for mobile SoC and high performance computing applications. In *2022 International Electron Devices Meeting (IEDM)*, 27-5 (IEEE, 2022).
62. Liu, J. et al. A Reliability Enhanced 5nm CMOS Technology Featuring 5th Generation FinFET with Fully-Developed EUV and High Mobility Channel for Mobile SoC and High Performance Computing Application. In *2020 IEEE International Electron Devices Meeting (IEDM)*, 9.2.1–9.2.4 (IEEE, 2020).
63. Song, Y. & Kingma, D. P. How to train your energy-based models. *arXiv* https://doi.org/10.48550/arXiv.2101.03288 (2021).
64. Carreira-Perpinan, M. A. & Hinton, G. On contrastive divergence learning. In *International Workshop on Artificial Intelligence and Statistics*, 33–40 (PMLR, 2005).
65. Chen, R. T., Li, X., Grosse, R. B. & Duvenaud, D. K. Isolating sources of disentanglement in variational autoencoders. *Adv. Neural Inf. Process. Syst*. **31** https://doi.org/10.48550/arXiv.1802.04942 (2018).
66. Extropic. thrml: Thermodynamic Hypergraphical Model Library https://github.com/extropic-ai/thrml (2025).
67. Sabne, A. XLA : Compiling Machine Learning for Peak Performance. 50530 (2020).
68. Bradbury, J. et al. *JAX: composable transformations of Python +NumPy programs* http://github.com/jax-ml/jax (2018).
69. Xiao, H., Rasul, K. & Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv* https://arxiv.org/abs/1708.07747 (2017).
70. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B. & Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, vol. 30 (eds, Guyon, I. et al.) (Curran Associates, Inc., 2017).
71. Kingma, D. P. & Welling, M. Auto-Encoding Variational Bayes. *arXiv* https://arxiv.org/abs/1312.6114 (2022).
72. Goodfellow, I. J. et al. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, vol. 27 (eds. Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. & Weinberger, K.) (Curran Associates, Inc., 2014).

Acknowledgements

This research was funded by Extropic Corp.

Author contributions

A.J. and T.M. contributed equally, with T.M. doing most of the writing and energy estimation work, whereas A.J. contributed most to the algorithms for training and evaluating the models presented. O.L. evaluated the performance of competitive models that we were comparing against. A.G. helped design some of the mentioned integrated circuits. P.S. helped with some experiments, and formatted the research code (which was published on GitHub) to be easy to understand and use. I.C. advised T.M. at the outset of this project. G.V. provided suggestions at the idea stage. All authors approve of the submission of this manuscript.

Competing interests

Some of the authors were part of Extropic corp., which produces and aims to sell products using some of the technologies described in the paper. All other authors declare no competing interests. Extropic Corp. has a patent pending for the entirety of this manuscript. The details are as follows: Applicant: Extropic Corp. Inventors: Guillaume Verdon-Akzam, Trevor McCourt, Andraž Jelinčič. Application number: US63/831,061. Current status: Provisional patent application.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s44335-026-00075-3.

Correspondence and requests for materials should be addressed to Trevor McCourt.