Editors' Suggestion

Generative Thermodynamic Computing

Stephen Whitelam[*]
Molecular Foundry, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, California 94720, USA

(Received 25 June 2025; accepted 14 November 2025)

We introduce a generative modeling framework for thermodynamic computing, in which structured data are synthesized from noise by the natural time evolution of a physical system governed by Langevin dynamics. While conventional diffusion models use neural networks to perform denoising, here the information needed to generate structure from noise is encoded by the dynamics of a thermodynamic system. Training proceeds by maximizing the probability with which the computer generates the reverse of a noising trajectory, which ensures that the computer generates data with minimal heat emission. We demonstrate this framework within a digital simulation of a thermodynamic computer. If realized in analog hardware, such a system would function as a generative model that produces structured samples without the need for artificially injected noise or active control of denoising.

DOI: 10.1103/kwyy-1xln

Introduction—In this Letter we describe a generative modeling framework for thermodynamic computing. Thermodynamic computing is closely related to the field of probabilistic computing [1–3]. It is based on the ideas that we can do energy-efficient computation by using small physical devices whose microscopic states change with time in response to thermal fluctuations, and that the fluctuations of a suitably designed device can encode the outcome of a desired calculation [4–8]. Here, we show that thermodynamic computers can perform generative modeling in a manner analogous to diffusion models, with key differences.

In a diffusion model, structured inputs are degraded by the gradual addition of noise. A neural network is trained to enact the reverse process, allowing the generation of structure from noise [9–12]. This process is implemented on a digital computer, where noise is introduced in the form of artificially generated pseudorandom numbers. Here, we use analytic calculations and digital simulations to suggest an alternative approach, in which the noise-driven dynamics of a thermodynamic computer—the noise arising naturally from the system's interaction with its environment—generates structure from noise. If realized in analog hardware, such a system would generate structured outputs simply by evolving with time under its natural dynamics. It would not require added pseudorandom noise, or the guidance of a digital neural network.

In more detail, we consider a model thermodynamic computer, a set of fluctuating nonlinear degrees of freedom coupled by bilinear interactions. The computer's degrees of freedom evolve according to overdamped Langevin dynamics. This design is inspired by existing hardware that can perform linear algebra [7], and by our recent work showing that a nonlinear version of such hardware can function as the thermodynamic version of a neural network [13]. We provide input to the computer to make it display images of digits from the MNIST data set [14], and allow these images to degrade by running the dynamics of the computer with its interunit couplings set to zero. Such degradation is called *noising* in the diffusion model literature. As we do so, we compute from the Langevin equation the probability that a computer with hypothetical nonzero couplings would have generated the *reverse* of this noising trajectory, and we adjust the values of these hypothetical couplings by gradient descent in order to maximize that probability. After running several such noising trajectories, we construct a denoising computer using the trained couplings, and verify that its natural dynamics, starting from noisy initial conditions, leads to the generation of structured MNIST-like digits. Independent dynamical trajectories of the same computer produce a variety of outcomes, some of which are not contained in the training set.

In this approach the denoising dynamics is encoded by the couplings of the trained thermodynamic computer, which plays the role of a denoising neural network in a diffusion model. If realized in analog hardware—for example, using networks of mechanical [15], electrical [8], or superconducting [16] oscillators—the information required for denoising would be encoded in the energy landscape of the computer, rather than in a digital neural network. As a result, denoising would not be simulated, but physically enacted.

In this mode of operation the thermodynamic computer resembles a nonequilibrium, continuous-spin analog of a Boltzmann machine, a statistical mechanical model that represents probability distributions over binary variables [17,18]. The key difference is that a Boltzmann machine

[*] Contact author: swhitelam@lbl.gov

encodes information in its equilibrium Boltzmann distribution, whereas our device runs on a physical clock: the computation is carried out by the dynamics of the system at a designated time, with no requirement to attain equilibrium. We therefore refer to the device considered here, which uses Langevin trajectories to perform a calculation, as a *Langevin computer*.

We also show that the training process has a direct physical interpretation: it adjusts the computer's couplings in order to minimize the thermodynamic irreversibility of the generative process. By finding the computer most likely to have generated the reverse of a noising trajectory, we minimize the expected heat emission and entropy production of the denoising computer. Our results therefore link the design of generative thermodynamic models to fundamental physical principles, and broaden our understanding of the capabilities of thermodynamic computers.

Training a generative Langevin computer—Consider a model of a thermodynamic computer. The computer is composed of N classical, real-valued fluctuating degrees of freedom $\boldsymbol{x} = \{x_i\}$, which could represent voltage states in electrical circuits [8], oscillator positions in a mechanical system [15], or phases in Josephson junction devices [16,19]. The computer's units x_i evolve in time according to the overdamped Langevin dynamics

$$\dot{x}_i = -\mu \partial_i V_{\boldsymbol{\theta}}(\boldsymbol{x}) + \sqrt{2\mu k_{\mathrm{B}} T}\eta_i(t). \tag{1}$$

Here, μ is the mobility parameter, which sets the basic time constant of the computer. For the thermodynamic computers of Refs. [7,8], μ^{-1} is of order a microsecond. For damped oscillators made from mechanical elements [15] or Josephson junctions [16,19], μ^{-1} is of order a millisecond or a nanosecond, respectively. The first term on the right-hand side of Eq. (1) is the force arising from the computer's potential energy $V_{\boldsymbol{\theta}}(\boldsymbol{x})$, given a set of parameters (couplings and biases) $\boldsymbol{\theta} = (\{J_{ij}\}, \{b_i\})$; note that $\partial_i \equiv \partial/\partial x_i$. The second term on the right-hand side of Eq. (1) models temporally uncorrelated thermal fluctuations: $k_{\mathrm{B}}T$ is the thermal energy scale, and the Gaussian white noise terms satisfy $\langle \eta_i(t) \rangle = 0$ and $\langle \eta_i(t)\eta_j(t')\rangle = \delta_{ij}\delta(t-t')$.

The potential energy $V_{\boldsymbol{\theta}}(\boldsymbol{x})$ of the computer is

$$V_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{i=1}^{N} (J_2 x_i^2 + J_4 x_i^4) + \sum_{i=1}^{N} b_i x_i + \sum_{(ij)} J_{ij} x_i x_j. \tag{2}$$

The first sum in Eq. (2), which runs over the N units, sets the intrinsic couplings of the computer. For $J_4 = 0$ we have a linear model [7], whose unit activations are linear functions of their inputs, while for $J_4 > 0$, the case we consider, we have a nonlinear model that can act as the thermodynamic analog of a neural network [13] (positive J_4 also ensures the thermodynamic stability of the computer as the J_{ij} are adjusted). We consider the case $J_2 > 0$, which creates units with one stable state, analogous to the s-units of thermodynamic computing [8]. The alternative choice, $J_2 < 0$, creates bistable units, analogous to the p-spins in the field of probabilistic computing [1].

The remaining terms in (2) contain the trainable parameters of the computer. The parameters b_i are input signals or biases applied to each unit. The parameters J_{ij} are pairwise couplings between units, inspired by the bilinear interactions of the thermodynamic computers of Refs. [7,8], with the sum running over all interunit connections. Equation (2) describes a thermodynamic computer of arbitrary connectivity, and the following discussion applies to the same.

Imagine that we observe a dynamical trajectory of the computer at a series of discrete times, $\omega = \{\boldsymbol{x}(t_k)\}_{k=0}^{K}$, where $t_k = k\Delta t$. The probability that any step of this trajectory was generated by a thermodynamic computer with parameters $\boldsymbol{\theta}$ can be calculated from the Onsager-Machlup action associated with the Langevin equation [20,21]. A time-discretized version of this action can be derived by first considering a standard Euler integration scheme for Eq. (1),

$$x_i(t+\Delta t) = x_i(t) - \mu \partial_i V_{\boldsymbol{\theta}}(\boldsymbol{x})\Delta t + \sqrt{2\mu k_{\mathrm{B}} T \Delta t}\eta_i. \tag{3}$$

Here, Δt is the integration time step, and η_i is a Gaussian random variable with zero mean and unit variance. Writing $\Delta x_i \equiv x_i(t+\Delta t) - x_i(t)$, we can rearrange (3) to read

$$\eta_i = \frac{\Delta x_i + \mu \partial_i V_{\boldsymbol{\theta}}(\boldsymbol{x})\Delta t}{\sqrt{2\mu k_{\mathrm{B}} T \Delta t}}. \tag{4}$$

Next, note that the probability of generating the step $\boldsymbol{x} \to \boldsymbol{x} + \Delta\boldsymbol{x}$ is that of drawing N noise values η_i,

$$P_{\boldsymbol{\theta}}^{\mathrm{step}}(\Delta\boldsymbol{x}) = (2\pi)^{-N/2} \prod_{i=1}^{N} \exp(-\eta_i^2/2), \tag{5}$$

with the η_i given by Eq. (4). Hence, the negative log-probability that a computer with parameters $\boldsymbol{\theta}$ generated the step $\boldsymbol{x} \to \boldsymbol{x} + \Delta\boldsymbol{x}$ is

$$-\ln P_{\boldsymbol{\theta}}^{\mathrm{step}}(\Delta\boldsymbol{x}) = \sum_{i=1}^{N} \frac{[\Delta x_i + \mu \partial_i V_{\boldsymbol{\theta}}(\boldsymbol{x})\Delta t]^2}{4\mu k_{\mathrm{B}} T \Delta t}, \tag{6}$$

up to an unimportant constant term. The negative log-probability that a computer with parameters $\boldsymbol{\theta}$ would generate the *reverse* step, $\boldsymbol{x} + \Delta\boldsymbol{x} \equiv \boldsymbol{x}' \to \boldsymbol{x}$, is

$$-\ln \tilde{P}_{\boldsymbol{\theta}}^{\mathrm{step}}(\Delta\boldsymbol{x}) = \sum_{i=1}^{N} \frac{[-\Delta x_i + \mu \partial_i V_{\boldsymbol{\theta}}(\boldsymbol{x}')\Delta t]^2}{4\mu k_{\mathrm{B}} T \Delta t}. \tag{7}$$

To increase the probability with which the computer would have generated the entire reverse trajectory

$\tilde{\omega} = \{\boldsymbol{x}(t_{K-k})\}_{k=0}^{K}$, we can sum Eq. (7) over all steps of the trajectory, differentiate that expression with respect to each parameter of the computer, and update the parameters as

$$J_{ij} \to J_{ij} + \alpha \sum_{k=1}^{K} \frac{\partial}{\partial J_{ij}} \ln \tilde{P}_{\boldsymbol{\theta}}^{\text{step}}[\Delta \boldsymbol{x}(t_k)], \tag{8}$$

$$b_i \to b_i + \alpha \sum_{k=1}^{K} \frac{\partial}{\partial b_i} \ln \tilde{P}_{\boldsymbol{\theta}}^{\text{step}}[\Delta \boldsymbol{x}(t_k)], \tag{9}$$

where α is a learning rate. Recall that $\Delta\boldsymbol{x}(t_k)$ is the displacement generated at time step k in the observed (forward) trajectory. The gradient terms in (8) and (9) can be calculated analytically, and are

$$-\frac{\partial}{\partial J_{ij}} \ln \tilde{P}_{\boldsymbol{\theta}}^{\text{step}}(\Delta \boldsymbol{x}) = \frac{-\Delta x_i + \mu \partial_i V_{\boldsymbol{\theta}}(\boldsymbol{x}')\Delta t}{2k_{\text{B}}T} x_j + \frac{-\Delta x_j + \mu \partial_j V_{\boldsymbol{\theta}}(\boldsymbol{x}')\Delta t}{2k_{\text{B}}T} x_i, \tag{10}$$

and

$$-\frac{\partial}{\partial b_i} \ln \tilde{P}_{\boldsymbol{\theta}}^{\text{step}}(\Delta \boldsymbol{x}) = \frac{-\Delta x_i + \mu \partial_i V_{\boldsymbol{\theta}}(\boldsymbol{x}')\Delta t}{2k_{\text{B}}T}, \tag{11}$$

where

$$\partial_i V_{\boldsymbol{\theta}}(\boldsymbol{x}) = 2J_2 x_i + 4J_4 x_i^3 + b_i + \sum_{j \in \mathcal{N}(i)} J_{ij} x_j. \tag{12}$$

Here, $\mathcal{N}(i)$ denotes the set of units connected to unit i.

When the forward trajectories depict noising processes, training over many such trajectories identifies couplings that allow a thermodynamic computer to transform noise into structured data.

Numerical illustration of generative thermodynamic computing—To illustrate this result we carried out a digital simulation of a thermodynamic computer. We set $J_2 = J_4 = 10k_{\text{B}}T$, and consider a computer with $N_{\text{v}} = 28^2$ visible units and $N_{\text{h}} = 512$ hidden units. The visible units will be used as a display, and the hidden units used to do computation. The computer has $N_{\text{v}}N_{\text{h}}$ trainable couplings J_{ij} between visible and hidden units, and $N_{\text{h}}(N_{\text{h}}-1)/2$ trainable couplings between hidden units. Hidden units have trainable biases b_i. Visible units have no trainable biases, but during training we impose visible-unit biases $b_i \propto P_i$, where P_i denotes the i^{th} pixel of an MNIST digit (each digit's pixels were adjusted to have zero mean and unit variance). We display the visible units in a 28×28 grid, matching the presentation of an MNIST digit.

To construct noising trajectories we project an MNIST image onto the visible units, via their nontrainable biases. We project part of the same image onto the hidden-unit biases, in order to provide them with some signal, and set all couplings J_{ij} to zero. We then let the computer come to equilibrium, by simulating Eq. (1) for a sufficiently long time. We then run a dynamical trajectory of time $t_{\text{f}} = 2.5$, slowly diminishing the intensity of the imposed digit. The result is an image that becomes increasingly noisy, as shown in Fig. 1(a). Positive values of the unit activations x_i are shown blue, while negative values are shown white.

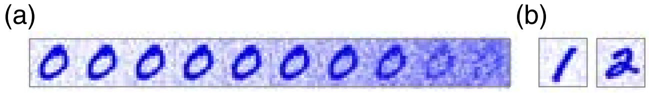

FIG. 1. (a) Example noising trajectory. (b) The remaining digits used in the training set.

As we run each noising trajectory, we update Eqs. (8) and (9). This process, repeated over many trajectories, identifies the parameters $\boldsymbol{\theta}$ of the computer that would, with maximum likelihood, generate the reverse of the trajectory, and so convert noise into signal. In this small-scale example we trained the computer using only three **Q1** digits, shown in Figs. 1(a) and 1(b).

In Fig. 2(a) we show three independent trajectories of the thermodynamic computer trained in this way. Trajectories begin from a noisy initial state prepared by bringing the coupling-free thermodynamic computer to equilibrium. In each case, the trained thermodynamic computer gradually transforms noise into structure, illustrating its ability to perform noise-to-structure generation. These results indicate that the computer has internalized representations of the digits and can reproduce them via physical evolution. In the case of the digit "1", the generated image appears

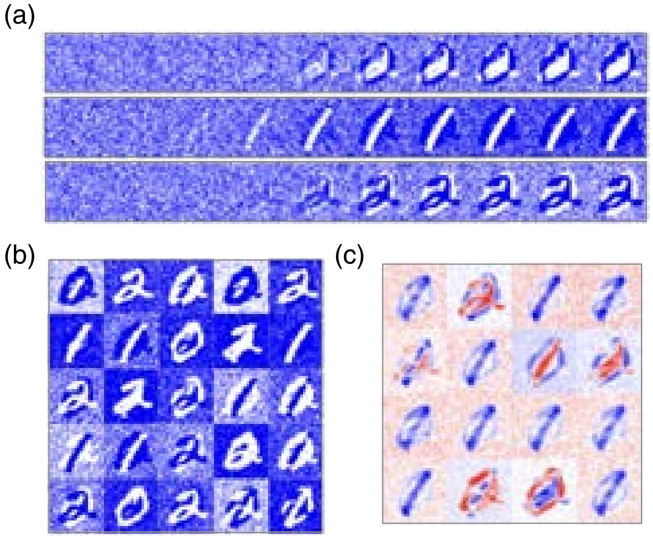

FIG. 2. (a) Three independent dynamical trajectories of the trained denoising thermodynamic computer. (b) The outcome at time $t = t_{\text{f}}$ of 25 independent trajectories of the trained computer. (c) Coupling patterns between 16 representative hidden units and the visible layer.

inverted. This behavior reflects a breaking of the visible layer's approximate symmetry under the transformation $x_i \to -x_i$ (the digits presented to the computer have pixel values that are roughly symmetric about zero).

In Fig. 2(b) we show the output at time $t = t_{\rm f}$ of 25 independent trajectories generated by the trained thermodynamic computer. The outputs exhibit diversity in style and form, behavior that is typical of diffusion models trained to approximate, rather than exactly replicate, the data distribution. Not all outputs are necessarily desirable or clearly interpretable—some display "mode mixing," in the language of diffusion models—but this simple example is intended as a proof of principle, a demonstration that structure can be generated from noise by physical dynamics alone.

Some insight into the behavior of the computer is provided by Fig. 2(c), which shows 16 of the computer's hidden units' learned couplings to the visible layer. Each panel represents the pattern of couplings J_{ij} between one hidden unit and all visible units: blue indicates positive coupling, red negative (either can be excitatory or inhibitory, via the interaction $J_{ij}x_i x_j$, depending on the values of the unit activations x_i). These patterns act as receptive fields that determine how each hidden unit responds to features in the visible layer. Several of the units have learned localized, digitlike structures, suggesting that the computer decomposes input patterns into interpretable visual components. Some hidden units exhibit complementary structure—activating for certain strokes while suppressing others—enabling the system to encode multiple features. Together, these receptive fields determine the energy landscape that guides the computer's dynamics, allowing the system to transform noise into structured outputs.

Physical interpretation of the learning process—The generative model we have considered is a thermodynamic system, and we can interpret the learning algorithm used to train it in physical terms. To do so, consider one step of a noising trajectory $\omega = \{\boldsymbol{x}(t_k)\}_{k=0}^{K}$, generated by any means. Consider the ratio of two probabilities (a standard device in nonequilibrium statistical mechanics [22]): the probability that the step was generated by a computer with a set of reference couplings (call them $\boldsymbol{\theta} = \mathbf{0}$), and the probability that our denoising computer (with parameters $\boldsymbol{\theta}$) generated the reverse of that step. This ratio is

$$\begin{aligned}\ln\frac{P_{\mathbf{0}}^{\rm step}(\Delta\boldsymbol{x})}{\tilde{P}_{\boldsymbol{\theta}}^{\rm step}(\Delta\boldsymbol{x})} &= \sum_{i=1}^{N}\frac{(-\Delta x_i + \mu\partial_i V_{\boldsymbol{\theta}}(\boldsymbol{x}')\Delta t)^2}{4\mu k_{\rm B}T\Delta t} \\ &\quad - \sum_{i=1}^{N}\frac{(\Delta x_i + \mu\partial_i V_{\mathbf{0}}(\boldsymbol{x})\Delta t)^2}{4\mu k_{\rm B}T\Delta t} \\ &\approx -\sum_{i=1}^{N}\frac{\Delta x_i\partial_i V_{\mathbf{0}}(\boldsymbol{x}) + \Delta x_i\partial_i V_{\boldsymbol{\theta}}(\boldsymbol{x})}{2k_{\rm B}T} \\ &\approx -\frac{\Delta Q_{\mathbf{0}} + \Delta Q_{\boldsymbol{\theta}}}{2k_{\rm B}T}, \qquad (13)\end{aligned}$$

to leading order in Δt. Here, $\Delta Q_{\mathbf{0}}$ and $\Delta Q_{\boldsymbol{\theta}}$ denote the incremental heat dissipated during the forward step by the reference and trained computers, respectively. Over the entire trajectory, in the limit $\Delta t \to 0$, we have

$$\ln\frac{P_{\mathbf{0}}[\omega]}{P_{\boldsymbol{\theta}}[\tilde{\omega}]} = -\frac{1}{2}[\beta Q_{\mathbf{0}}(\omega) + \beta Q_{\boldsymbol{\theta}}(\omega)]. \qquad (14)$$

Here, $\beta \equiv 1/(k_{\rm B}T)$, and $P_{\mathbf{0}}[\omega]$ and $P_{\boldsymbol{\theta}}[\tilde{\omega}]$ denote the probabilities of observing the forward trajectory with the reference computer, and the reverse trajectory with the denoising computer, respectively. The terms $Q_{\mathbf{0}}(\omega)$ and $Q_{\boldsymbol{\theta}}(\omega)$ represent the total heat dissipated along the forward trajectory by the two computers.

Training proceeds by minimizing the negative log-probability $-\ln P_{\boldsymbol{\theta}}[\tilde{\omega}]$. According to the fluctuation relation (14), and given that the reference process is fixed, this is equivalent to minimizing $-Q_{\boldsymbol{\theta}}(\omega)$, the negative total heat dissipated by the denoising computer when generating the noising trajectory ω. Given that heat changes sign upon time reversal, the learning process therefore minimizes the heat $Q_{\boldsymbol{\theta}}(\tilde{\omega}) = -Q_{\boldsymbol{\theta}}(\omega)$ emitted by the trained computer along the trajectory $\tilde{\omega}$, i.e., as it generates structure from noise. In this sense, the trained dynamics is thermodynamically optimal: it is the dynamics that reconstructs the imposed data with the least heat emitted or entropy produced.

Thermodynamic advantage—We can estimate the thermodynamic advantage, the ratio of the energy costs of digital to thermodynamic computation, by considering the energy scales of denoising using a digital neural network and a hardware version of our simulated thermodynamic computer. The basic energy scale of a digital neural network is set by a multiply accumulate (MAC) operation, which differs by hardware implementation but is typically about 1 pJ [23], or $2.4 \times 10^8 k_{\rm B}T$ at room temperature. For a modest multilayer perceptron denoiser (784 → 128 → 128 → 784), a single denoising step requires $\sim 2.2 \times 10^5$ MACs. If we make the very conservative assumption that the denoiser is used only 10 times within a denoising trajectory, the order-of-magnitude energy budget of denoising using a neural network is not less than $5 \times 10^{14} k_{\rm B}T$.

The energy cost of the thermodynamic computer is much smaller. We can calculate the heat emitted by the computer from the difference of the potential energy (2) between the start and end of a trajectory, $Q = V_{\boldsymbol{\theta}}[\boldsymbol{x}(0)] - V_{\boldsymbol{\theta}}[\boldsymbol{x}(t_{\rm f})]$ (no work is done on the computer after the hidden-unit biases are established). Over 1000 independent denoising trajectories of the trained computer we calculate a mean heat emission of $\langle Q \rangle = 2.9 \times 10^3 k_{\rm B}T$, with standard deviation $3.5 \times 10^2 k_{\rm B}T$. Comparing this value to the digital estimate gives a ratio of more than 10^{11}. That is, if implemented in hardware, the thermodynamic computer would be more than 10 orders of magnitude more energy efficient than a digital neural network. The example shown here is rudimentary by the standards of state-of-the-art diffusion models, but shows the potential, and potential energy savings, of thermodynamic computation.

Conclusions—We have proposed a generative modeling framework in which structure is produced from noise by the physical dynamics of a thermodynamic system. This approach follows the logic of a diffusion model, but instead of using a digital neural network and artificially injected noise, the information required for generation is encoded in the system's energy landscape and emerges from a physical dynamics. Used in this way, the thermodynamic computer is a Langevin computer, a nonequilibrium, continuous-spin analog of a Boltzmann machine [17,18].

The learning process, maximizing the likelihood that a trained system could have produced the reverse of a noising trajectory, admits a natural interpretation in terms of entropy production: the model learns to reverse the forward dynamics in the most thermodynamically reversible way. Thermodynamic learning therefore links generative modeling to fundamental physical principles.

Reference [24] proposed the idea of making a generative model by controlling analog physical dynamics with a digital neural network. Here, we have shown that analog hardware on its own can be generative. However, having a neural network control the couplings of the thermodynamic computer would indeed make it more expressive: for instance, the neural network could adjust the couplings of the computer as a function of time, or set the computer's couplings so as to produce conditioned outputs.

We have used digital simulation to demonstrate that nonlinear, nonequilibrium analog hardware can learn to generate structured outputs from noise. This Letter provides another example of a thermodynamic computer trained to operate under nonequilibrium conditions [13]. Realized physically—trained digitally, with the learned couplings implemented in hardware—such systems could perform autonomous generative computation without external control or artificial randomness, opening new avenues of exploration for physically grounded, energy-efficient machine learning.

Acknowledgments—I thank Isaac Tamblyn and Adrianne Zhong for discussions. This work was done at the Molecular Foundry, supported by the Office of Science, Office of Basic Energy Sciences, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231, and partly supported by U.S. DOE Office of Science Scientific User Facilities AI/ML project "A digital twin for spatiotemporally resolved experiments."

Data availability—The data that support the findings of this article are openly available [25].

[1] J. Kaiser and S. Datta, Appl. Phys. Lett. **119** (2021).

[2] N. A. Aadit, A. Grimaldi, M. Carpentieri, L. Theogarajan, J. M. Martinis, G. Finocchio, and K. Y. Camsari, Natl. Electron. Rev. **5**, 460 (2022).

[3] S. Misra, L. C. Bland, S. G. Cardwell, J. A. C. Incorvia, C. D. James, A. D. Kent, C. D. Schuman, J. D. Smith, and J. B. Aimone, Adv. Mater. **35**, 2204569 (2023).

[4] T. Conte, E. DeBenedictis, N. Ganesh, T. Hylton, G. E. Crooks, and P. J. Coles, arXiv:1911.01968.

[5] T. Hylton, Entropy **22**, 256 (2020).

[6] G. Wimsatt, Olli P. Saira, Alexander B. Boyd, Matthew H. Matheny, S. Han, Michael L. Roukes, and James P. Crutchfield, Phys. Rev. Res. **3**, 033115 (2021).

[7] M. Aifer, K. Donatella, M. H. Gordon, S. Duffield, T. Ahle, D. Simpson, G. Crooks, and P. J. Coles, npj Unconv. Comput. **1**, 13 (2024).

[8] D. Melanson, M. Abu Khater, M. Aifer, K. Donatella, M. Hunter Gordon, T. Ahle, G. Crooks, A. J. Martinez, F. Sbahi, and P. J. Coles, Nat. Commun. **16**, 3757 (2025).

[9] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, in *International Conference on Machine Learning* (PMLR, 2015), pp. 2256–2265.

[10] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, and M.-H. Yang, ACM Comput. Surv. **56**, 1 (2023).

[11] G. Biroli, T. Bonnaire, V. De Bortoli, and M. Mézard, Nat. Commun. **15**, 9957 (2024).

[12] Z. Yu and H. Huang, Phys. Rev. E **111**, 014111 (2025).

[13] S. Whitelam and C. Casert, arXiv:2412.17183.

[14] Y. LeCun, C. Cortes, and C. J. Burges, MNIST leaderboard, http://yann.lecun.com/exdb/mnist/ (1998).

[15] S. Dago, J. Pereda, N. Barros, S. Ciliberto, and L. Bellon, Phys. Rev. Lett. **126**, 170601 (2021).

[16] Kyle J. Ray and James P. Crutchfield, Phys. Rev. Appl. **19**, 014049 (2023).

[17] G. Hinton, in *Encyclopedia of Machine Learning and Data Mining* (Springer, New York, 2017), pp. 164–168.

[18] R. Salakhutdinov and G. Hinton, in *Artificial Intelligence and Statistics* (PMLR, 2009), pp. 448–455.

[19] C. Z. Pratt, K. J. Ray, and J. P. Crutchfield, Chaos **35**, 043112 (2025).

[20] H. Risken and H. Risken, *Fokker-Planck Equation* (Springer, New York, 1996).

[21] L. F. Cugliandolo and V. Lecomte, J. Phys. A **50**, 345001 (2017).

[22] U. Seifert, Rep. Prog. Phys. **75**, 126001 (2012).

[23] M. Horowitz, in *Proceedings of the 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)* (IEEE, New York, 2014), pp. 10–14.

[24] P. J. Coles, C. Szczepanski, D. Melanson, K. Donatella, A. J. Martinez, and F. Sbahi, in *Proceedings of the 2023 IEEE International Conference on Rebooting Computing (ICRC)* (IEEE, New York, 2023), pp. 1–10.

[25] S. Whitelam, The code used in this Letter can be found at the GitHub repository $\boldsymbol{\theta}$, 2025, https://github.com/swhitelam/generative_thermodynamic_computing. **Q3**