Demon in the Machine: Learning to Extract Work and Absorb Entropy from Fluctuating Nanosystems

Stephen Whitelam*

Molecular Foundry, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, California 94720, USA

(Received 30 November 2022; revised 26 January 2023; accepted 13 March 2023; published 10 April 2023)

We use Monte Carlo and genetic algorithms to train neural-network feedback-control protocols for simulated fluctuating nanosystems. These protocols convert the information obtained by the feedback process into heat or work, allowing the extraction of work from a colloidal particle pulled by an optical trap and the absorption of entropy by an Ising model undergoing magnetization reversal. The learning framework requires no prior knowledge of the system, depends only upon measurements that are accessible experimentally, and scales to systems of considerable complexity. It could be used in the laboratory to learn protocols for fluctuating nanosystems that convert measurement information into stored work or heat.

DOI: 10.1103/PhysRevX.13.021005

Subject Areas: Complex Systems, Computational Physics, Statistical Physics

I. INTRODUCTION

Driving a fluctuating molecular machine or nanoscale system out of equilibrium costs energy, on average, according to the second law of thermodynamics [1,2]. For example, when averaged over many experiments, an optical trap requires an input of work to drag a colloidal particle through water, and a nanomagnetic system whose state is flipped by a magnetic field produces entropy. Individual realizations of these processes can generate work or absorb entropy, and the fluctuation relations, which generalize the second law, make statements about the distributions of work and entropy that result from various kinds of nonequilibrium processes [3–6]. However, if the nonequilibrium protocol involves feedback, i.e., has knowledge of the state of the system, then the mean of the work- and entropy-production distributions can be negative: On average, work can be extracted from the system, or entropy can be absorbed by it [7–13]. Such behavior does not violate the second law because these changes are paid for by the acquisition and destruction of information, which increases the entropy of the universe. Thus, an agent or "demon" that measures a system in order to control it can convert measurement information into work or heat. Such "information engines" have been demonstrated experimentally, using ratchetlike mechanisms to extract work from colloidal particles in electric or gravitational fields [14,15].

Here, we demonstrate a technically simple and generally applicable procedure for learning feedback-control protocols for fluctuating nanosystems. We consider two model computational systems, a particle pulled by an optical trap through a viscous medium [16] and an Ising model undergoing magnetization reversal [17,18]. We introduce a demon, a deep neural network whose inputs are the elapsed time of the experiment and any information we provide to it from the system, and whose outputs indicate the new values of the control parameters, the trap position or the Ising model temperature and magnetic field. We train the demon using Monte Carlo [19] and genetic algorithms [20,21] to minimize the work done by the trap or the entropy produced by the Ising model. We allow the demon no prior knowledge of what constitutes an efficient protocol. When the input to the demon is time alone, the protocols it learns reproduce the optimal-control protocols known analytically or from numerical studies [16,22,23]. When the demon is also provided the force on the particle or the magnetization of the Ising model, it learns protocols that *extract* work from the trap and *absorb* entropy from the Ising model's thermal bath, thus converting measurement information into alternative forms of energy.

The learning procedure described here can be applied to experiments the same way it is applied to simulations. The order parameter that the procedure minimizes is the dissipation or work produced over the *entire* trajectory, averaged over many trajectories, quantities that are accessible experimentally. It does not require time-resolved

*swhitelam@lbl.gov

information about the order parameter or gradients of the order parameter along a trajectory. The number of trajectories required for meaningful learning is not prohibitively large, in the context of prior experiments. Furthermore, it is possible to apply the learning framework to protocols of considerable complexity. The neural-network demon can accommodate an arbitrary number of input neurons (information from the system) and output neurons (control parameters of the experiment), and the learning algorithms we use have been shown empirically to work with neural networks having of order 1000 inputs and of order tens of millions of parameters [19,24].

II. MODEL OF A PARTICLE PULLED BY AN OPTICAL TRAP

We consider the first problem of Ref. [16], a particle at position x in a potential

$$V(x,\lambda) = \frac{1}{2}(x-\lambda)^2, \tag{1}$$

in units such that $k_B T = 1$; see Fig. 1(a). The trap center is λ. The particle undergoes the Langevin dynamics

$$\dot{x} = -\partial_x V(x,\lambda) + \xi(t), \tag{2}$$

where the noise ξ has zero mean and correlation function $\langle \xi(t)\xi(t')\rangle = 2\delta(t-t')$. The aim is to move the trap center from an initial position $\lambda_i = 0$ to a final position $\lambda_f = 5$, in finite time t_f, minimizing the work averaged over many realizations of the process. If λ is a function of time alone, then the optimal (work-minimizing) protocol is the linear form $\lambda^\star(t) = \lambda_f(t+1)/(t_f+2)$, for $0 < t < t_f$, with jump discontinuities at the start ($t = 0$) and end ($t = t_f$). This protocol produces mean work $\lambda_f^2/(t_f+2)$ [16].

To develop a feedback-control protocol for this system, we introduce a demon, the deep neural network shown in Fig. 1(b). The neural network is fully connected, with five hidden layers (each of width four apart from the last, which is of width ten) and hyperbolic tangent activations. We apply layer norm preactivation [25]. Deep neural

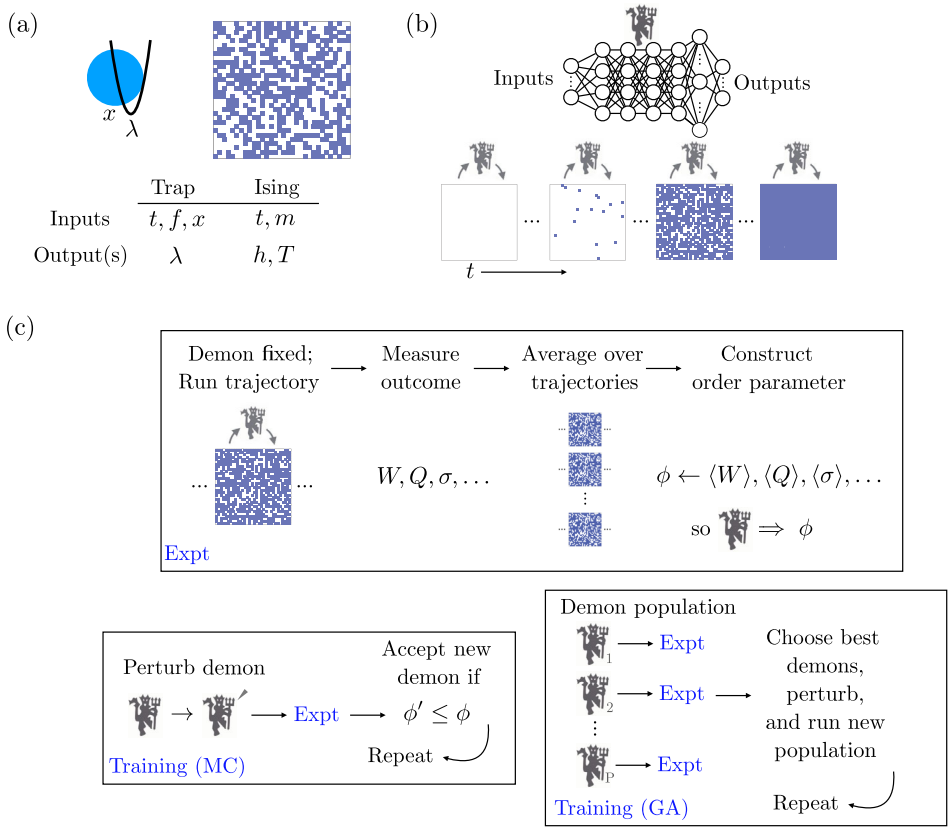

FIG. 1. (a) We develop feedback-control protocols for an overdamped particle pulled by a harmonic potential and the Ising model undergoing magnetization reversal. (b) Feedback is enacted by a demon, a deep neural network, which controls the protocol to which the system is subjected by periodically taking in information from the system and outputting new values of the system's control parameters. (c) The demon is trained to extremize a desired physical observable, such as heat or work. Dynamical trajectories of the system, generated using the demon-mandated protocol, result in an order parameter ϕ that is composed of ensemble averages of work, heat, entropy production, or other measurable quantities (box labeled "expt"). The demon is trained iteratively, by Monte Carlo (box "MC") or genetic algorithms (box "GA"), to extremize ϕ. In this paper, the trajectories are generated by computer simulations of model systems, but the same learning procedure could be applied to trajectories generated in laboratory experiments.

networks are convenient ways of expressing potentially high-dimensional functions, and this particular architecture can be trained faster than single-layer nets to express rapidly varying functions [19]. The network has as many input neurons as degrees of freedom it is provided, and as many output neurons as there are control parameters of the system. If $\boldsymbol{s}$ is the vector of state information provided to the demon, and $\boldsymbol{\lambda}$ the vector of experimental control parameters, then the demon, when queried, sets these control parameters to the values

$$\boldsymbol{\lambda} = \boldsymbol{g}_{\boldsymbol{\theta}}(t/t_f, \boldsymbol{s}), \tag{3}$$

where t/t_f is scaled time (information that is always available), $\boldsymbol{g}$ is the vector function expressed by the neural network, and $\boldsymbol{\theta}$ is the vector of neural-network parameters (weights and biases). Equation (3) is a parametrization of the control parameters of the experiment as a function of time and (potentially) state-dependent information. The aim is to adjust the numbers $\boldsymbol{\theta}$ by training so that the protocol enacted by the demon achieves the desired objective.

For the trap system, there is one control parameter, the trap center λ, so the demon needs only one output neuron. In the main text, we consider three different sets of inputs: time alone; time and force; or time, particle position, and trap position. The neural network needs one, two, or three input neurons in each case.

The trap position is initially $\lambda_0 = \lambda_i$, and the particle position is $x_0 \sim \mathcal{N}(0,1)$, reflecting thermal equilibrium in the potential (1). At each step $k = 1, \ldots, \lfloor t_f/\Delta t \rfloor$ of the simulation, we choose a new trap position by consulting the demon, $\lambda_k = g_{\boldsymbol{\theta}}(t_k/t_f)$, where $t_k = k\Delta t$; we calculate the work done by this change,

$$\Delta W_k = V(x_k, \lambda_k) - V(x_k, \lambda_{k-1}); \tag{4}$$

and we update the position of the particle according to the forward Euler discretization of Eq. (2) with time step $\Delta t = 10^{-3}$. At the end of the trajectory, at time t_f, the trap center is set to position λ_f, and the work is updated accordingly. The total work done along the trajectory, W, is the sum of all changes (4) plus the final-time work update. The order parameter we wish to minimize is $\phi = \langle W \rangle$, where $\langle \cdot \rangle$ denotes the average over M independent trajectories of the procedure just described. The demon starts with all parameters set to zero, $\boldsymbol{\theta} = \boldsymbol{0}$, and so has no prior knowledge of what constitutes a good protocol. The protocol enacted by this untrained demon is a jump at time t_f from $\lambda = \lambda_i$ to $\lambda = \lambda_f$, which gives mean work $\lambda_f^2/2$.

To train the demon, we use the adaptive Monte Carlo (aMC) algorithm of Ref. [19], or a genetic algorithm (GA) [20,21,26,27] using mutations only. Training is illustrated in Fig. 1(c). aMC is an adaptive version of the Metropolis algorithm [28], and it proceeds as follows (see box labeled "MC"). We evaluate the order parameter ϕ by generating trajectories using the demon-mandated protocol (see box labeled "expt"). We add independent Gaussian random numbers ("mutations") to each demon parameter and evaluate the objective ϕ' under the new demon. If $\phi' \leq \phi$, then we accept the new demon; otherwise, we return to the current one. This part of the procedure is zero-temperature Metropolis Monte Carlo applied to the parameters of a neural network; in addition, aMC adjusts the mean and variance of the mutations in order to propose moves that are more likely to be accepted. We use the aMC hyperparameters $\sigma_0 = 10^{-1}, \epsilon = 10^{-2}$, and $n_{\text{scale}} = 10^3$, with signal norm off (we use layer norm instead) [19]. Training using a genetic algorithm (see box labeled "GA") proceeds by evaluating the order parameter ϕ under 50 randomly initialized demons [for which each parameter is an independent Gaussian random number $\theta \sim \mathcal{N}(0, \sigma^2)$, with $\sigma = 0.1$] and picking the five with the smallest values of ϕ. The next generation of 50 demons is created by randomly picking 49 times with replacement from this set of five parents, and adding independent Gaussian random numbers of zero mean and variance $\sigma^2 = 10^{-2}$ to each parameter of each demon. The final member of the population is the unmutated best demon from the previous generation, a procedure called elitism. The order parameter is evaluated for this new set of 50 neural networks, the best five are chosen to be the parents of the subsequent generation, and so on. The two methods, GA and MC, are closely related: Zero-temperature Metropolis MC is also a genetic algorithm with a population of size two with one parent, using elitism and mutations only. Both aMC and GA are simple to implement and have similar learning capacity to gradient-based methods [20,29–32] (though they do not necessarily converge as quickly, step for step [19]). They do not require gradient information from the trajectory, making them ideally suited to laboratory experiments. GA is convenient if experiments can be run in parallel, while aMC usually requires fewer function evaluations (in this case, trajectories or experiments) to reach a prescribed value of the order parameter.

To make contact with prior results, we first consider the case in which the demon knows only the elapsed time of the trajectory, so the trap position is $\lambda = g_{\boldsymbol{\theta}}(t/t_f)$. In panels (a) and (b) of Fig. 2, we consider the hypothetical limit in which the order parameter $\phi = \langle W \rangle$ is evaluated using a large number $M \to \infty$ of trajectories [when λ depends only on time, the mean work is a function of $\langle x \rangle$, and this can be calculated in the limit $M \to \infty$ using a single trajectory of the noise-free version of Eq. (2)]. In panels (c) and (d), we consider the experimentally relevant case in which a finite number ($M = 10^4$) of trajectories is used. In both cases, the order parameter converges to the optimal value. For $M \to \infty$, the demon learns the optimal protocol (black dashed line), while for finite M it learns an approximation of it. However, the outcome of the optimal protocol cannot be distinguished from that of the learned protocol:

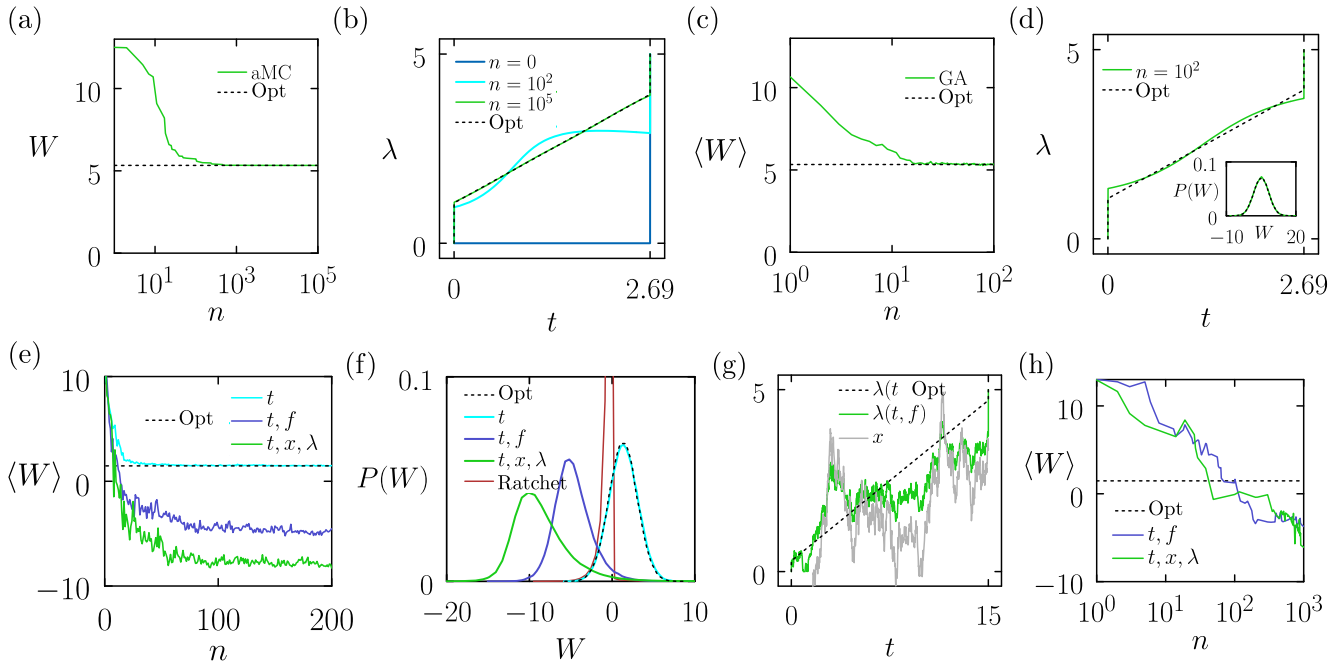

FIG. 2. Particle pulled by a trap. (a) Work W as a function of the aMC steps n for a single trajectory of the noise-free version of Eq. (2) (mimicking a large number of trajectories) under the neural-network demon protocol $\lambda = g_{\boldsymbol{\theta}}(t/t_f)$ (green line). The black dashed line is the analytic result of Ref. [16]. (b) Time-dependent protocol $\lambda = g_{\boldsymbol{\theta}}(t/t_f)$ learned after $n = 0, 10^2$, and 10^5 aMC steps (colors), compared to the optimal protocol (black dashed line). (c,d) Same as panels (a,b), but now the order parameter $\langle W \rangle$ is calculated as an average over 10^4 noisy trajectories. GA is used as a method of training; n is the number of generations (here and elsewhere, the GA result indicates the performance of the best demon in each generation). The learned protocol is a near-optimal one that, in practical terms, cannot be distinguished from the optimal protocol: The inset to panel (d) shows the work distributions produced by each. (e) Mean work $\langle W \rangle$ (averaged over 10^4 trajectories) versus GA generation n for demon protocols expressed as a function of t (cyan line), t and f (blue line), and t, x, and λ (green line). The black dashed line is the analytic result of Ref. [16]. (f) Work distributions for demon protocols learned after 200 generations (colors), together with that of a ratchet (red line). (g) Single Monte Carlo trajectory under the demon (t, f)-protocol learned after 200 generations. (h) Similar to panel (e), but averaging $\langle W \rangle$ over only 10^2 trajectories and using aMC as a training method. Trajectory lengths: $t_f = 2.69$ (a)–(d) and 15 (e)–(h).

As shown in the inset of panel (d), the work distributions produced by optimal and learned protocols are essentially identical. As n increases, the learned protocol fluctuates about the optimal one, adopting a large number of different shapes that have similar outcomes; see Fig. 4.

The analytic optimal protocol of Ref. [16] has been reproduced by other authors using gradient descent [22] and numerical methods of optimal control [23]. These methods are powerful numerically but require information that is not accessible experimentally, namely, gradients of the objective along the dynamical trajectory or knowledge of the deterministic Fokker-Planck equation that governs the evolution of the probability density, respectively. Here, we show that the optimal time-dependent protocol can be learned if we know only the total work performed during a stochastic trajectory, without time-resolved input from the system, and with no prior knowledge of what constitutes an efficient protocol.

In Figs. 2(e)–2(h), we show that providing feedback to the demon allows it to learn protocols whose mean values of work are negative. We compare cases in which the demon is a function of time alone; a function of time and the force $f = x - \lambda$ acting on the particle, $\lambda = g_{\boldsymbol{\theta}}(t/t_f, f)$; and a function of time, particle position x, and trap position, $\lambda = g_{\boldsymbol{\theta}}(t/t_f, x/\lambda_f, \lambda/\lambda_f)$. In panel (e), we show that feedback allows the mean work to be negative: The demon has learned to extract work from the system. (The demon can also extract work from the system if it is fed other information, such as the current value of the time-integrated work, but the instantaneous system coordinates are more useful in this respect; see Fig. 5.)

Panel (f) shows the distributions of work resulting from these protocols, together with that resulting from a simple ratchetlike mechanism [14] in which, at each time step, λ is set to x if $x > \lambda$ and $\lambda \leq \lambda_f$, and is otherwise unchanged. The ratchet can extract work from the thermal bath, but not as efficiently as the protocols learned by the demons. Panel (g) shows one trajectory of the time-force protocol learned after generation $n = 200$: The demon extracts work by moving the trap toward the particle but with a bias in the direction of the final-time trap position. In panel (h), we show data similar to those of panel (e) but now

taking work averages $\langle W \rangle$ over only 10^2 trajectories and training using aMC. With fewer trajectories used to calculate the mean work, fluctuations of the learning process are larger, but work-extracting protocols can still be learned. Each GA generation of panel (e) requires the evaluation of 50×10^4 trajectories, while each aMC step of panel (h) requires the evaluation of 10^2 trajectories. There exist optical traps that allow hundreds of trajectories per experiment [33], so the number of experiments required by the demon to learn work-extracting protocols (of order 100) is not prohibitive in an experimental context.

III. MAGNETIZATION REVERSAL IN THE ISING MODEL

The learning framework can be applied to more complex, many-body systems in the same way, with straightforward changes made to the neural network to accommodate the required input and control parameters. Consider magnetization reversal in the Ising model, a prototype of information erasure and copying in nanomagnetic storage devices. Reducing dissipation by finding optimal time-dependent protocols for these processes [17,18] has practical relevance for reducing computational energy demands [34]. Here, we use the learning framework to reproduce optimal time-dependent protocols that minimize dissipation upon magnetization reversal [22]. We then show that feedback control using experimentally accessible measurements allows magnetization reversal to proceed with *negative* mean dissipation, taking heat from the surroundings.

We consider the 2D Ising model [35,36] on a square lattice of $N = 32^2$ sites, with periodic boundary conditions in both directions. On each site i is a binary spin $S_i = \pm 1$. The lattice possesses an energy function

$$E = -J \sum_{\langle ij \rangle} S_i S_j - h \sum_{i=1}^{N} S_i, \tag{5}$$

in units such that $k_B = 1$. Here, J (which we set to 1) is the Ising coupling, and h is the magnetic field. The first sum in Eq. (5) runs over all nearest-neighbor bonds, while the second runs over all lattice sites. We begin with all spins down, giving magnetization $m = N^{-1} \sum_{i=1}^{N} S_i = -1$. Following Ref. [22], the aim is to change temperature T and field h from the values $\boldsymbol{\lambda}_i = (T_i, h_i) = (0.65, -1)$ to the values $\lambda_f = (t_f, h_f) = (0.65, 1)$, in finite time t_f, ensuring magnetization reversal with minimal dissipation.

We simulate the model using Glauber Monte Carlo dynamics. At each step of the algorithm, a lattice site i is chosen and a change $S_i \to -S_i$ is proposed. In Figs. 3(a)–3(c), we choose lattice sites deterministically, moving through all odd-numbered spins and then all even-numbered spins, in order to make contact with Ref. [22]. In Figs. 3(d)–3(g), we consider a random choice of lattice sites. The outcomes of these two procedures are qualitatively similar and generate numerical values of entropy production that differ by a factor of about 2. The proposed change is accepted with the Glauber probability $[1 + \exp(\beta \Delta E)]^{-1}$, where ΔE is the energy change under the proposed move, and $\beta = 1/T$ is the reciprocal temperature. If the move is rejected, the original spin state is adopted.

The entropy produced over the course of a simulation is

$$\sigma = \beta_f E_f - \beta_i E_i - \sum_k \beta_k \Delta E_k, \tag{6}$$

where E_f and E_i are the final and initial energies of the system, and ΔE_k and β_k are the energy change and reciprocal temperature at step k of the simulation. This expression can be derived by considering the path probabilities of forward and reverse trajectories or by noting that heat exchange with the bath is given by a change of energy at fixed control parameters [5,6]. The first two terms on the right-hand side of Eq. (6) cancel for any path that connects end points of equal temperature and opposite field, and for which the magnetization reverses.

The demon (3) now has two output neurons in order to specify the control-parameter vector $\boldsymbol{\lambda} = (T, h)$. The output layer contains a shear transformation in order to ensure that the protocol starts and ends at the required points: If $\tilde{\boldsymbol{g}}_{\boldsymbol{\theta}}(t/t_f, m)$ is the output of the neural network prior to the shear, then the control-parameter vector is set to

$$\begin{aligned} \boldsymbol{g}_{\boldsymbol{\theta}}(t/t_f, m) = \tilde{\boldsymbol{g}}_{\boldsymbol{\theta}}(t/t_f, m) + (1 - t/t_f)[\boldsymbol{\lambda}_{\mathrm{i}} - \tilde{\boldsymbol{g}}_{\boldsymbol{\theta}}(0, -1)] \\ + (t/t_f)[\boldsymbol{\lambda}_f - \tilde{\boldsymbol{g}}_{\boldsymbol{\theta}}(1, 1)] \end{aligned} \tag{7}$$

after the shear [37]. In addition, if T resulting from Eq. (7) is less than 10^{-3}, then it is set equal to 10^{-3}. Initially, the demon parameters $\boldsymbol{\theta}$ are set to zero, and the protocol produced by this untrained demon jumps abruptly between initial and final values. The entropy produced by a sudden change of control parameters between initial and final values, without a change of magnetization, is $2h\beta_f N \approx 3151$.

The order parameter we wish to minimize is $\phi = |\langle m_f \rangle - 1| + k_\sigma \langle \sigma \rangle$, where $\langle \cdot \rangle$ denotes the average over $M = 10^3$ independent trajectories, $k_\sigma = 10^{-4}$, σ is given by Eq. (6), and m_f is the magnetization at the end of the trajectory. This order parameter is minimized if $\langle m_f \rangle = 1$ and $\langle \sigma \rangle$ is as small as possible. The choice $k_\sigma \ll 1$ ensures that the second term in ϕ is generally smaller than the first, enforcing that the primary goal of the procedure is to reverse magnetization. Once this has been achieved, the order parameter further rewards protocols that do so with as little dissipation as possible.

We start by providing the demon with time alone as input, i.e., $(T, h) = \boldsymbol{g}_{\boldsymbol{\theta}}(t/t_f)$. The demon acts at 1000

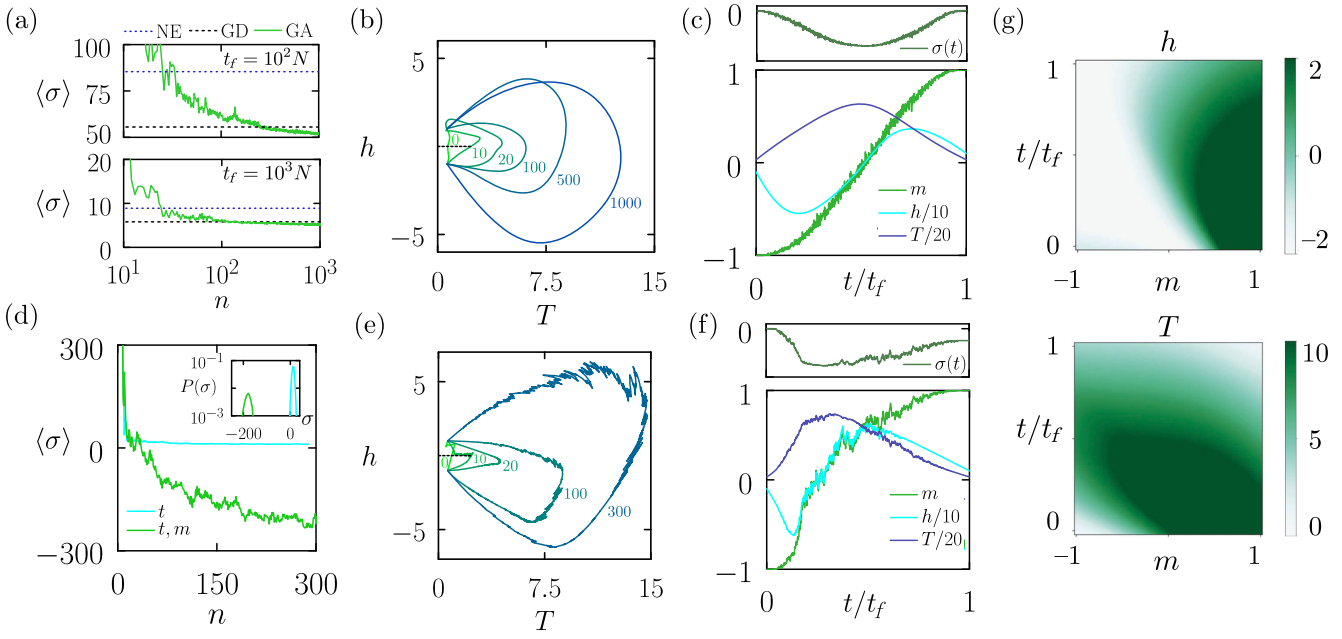

FIG. 3. Magnetization reversal in the Ising model. (a) Mean entropy production $\langle\sigma\rangle$ under protocols learned by a demon given time as input, as a function of GA generation n (green line). Trajectory lengths are 10^2 (top) or 10^3 (bottom) Monte Carlo sweeps. Also shown are values due to entropy-minimizing protocols found by near-equilibrium approximation [17] (NE) and gradient descent [22] (GD). (b) Parametric protocols $[T(t), h(t)]$ learned after n generations. The black dashed line is the Ising model first-order phase-transition line, which ends at the critical point [35]. (c) Single trajectory under the protocol learned after 10^3 generations. (d)–(f) Similar to panels (a)–(c), but now the demon receives time and global magnetization m as inputs. The top sections of panels (c) and (f) show time-resolved entropy production along each trajectory (the vertical axis is 10^3 units long): This information is not available during training nor used by the demon. In panel (d), we also show the case in which the input is time only (cyan line), and in the inset of that panel, we show entropy production distributions over 10^4 trajectories for protocols learned using time (cyan line) and time and magnetization (green line) as inputs. (g) Visualization of the demon feedback-control protocol learned after 300 generations.

evenly spaced time intervals, choosing new values of T and h each time it acts. We train the demon by GA (aMC results are shown in Fig. 7). In Fig. 3(a), we show the mean entropy produced by learned protocols as a function of GA generation n (green line). Trajectory lengths are $t_f = 10^2$ or 10^3 Monte Carlo sweeps (steps per lattice site). We also show the entropy produced by the optimal protocol obtained by the near-equilibrium approximation of Ref. [17] (blue dashed line) and that found numerically by gradient descent in Ref. [22] (black dashed line). The GA result is consistent with the latter, confirming, for this problem, that gradient-free and gradient-based methods have similar capacity for learning.

In Fig. 3(b), we show parametric protocols obtained after different evolutionary generations (here and subsequently, we consider trajectories of length $t_f = 10^3$ Monte Carlo sweeps). The demon learns to avoid the large dissipation associated with the first-order phase transition and the critical point [17]. Consistent with Ref. [18], protocols with substantially different values of T and h can have similar values of dissipation—e.g., compare the values of entropy production [panel (a), bottom] due to the protocols obtained after 500 and 1000 generations [panel (b)]. In panel (c), we show a single trajectory under a protocol learned after 10^3 generations. Spin flips against the direction of the magnetic field tend to absorb entropy while those in the direction of the field tend to produce it, and the demon controls T and h in order to produce as little excess entropy as possible.

Panels (d)–(f) of Fig. 3 are similar to panels (a)–(c); however, now the demon knows the elapsed time of the simulation and the global magnetization, so $(T, h) = \boldsymbol{g}_{\boldsymbol{\theta}}(t/t_f, m)$. Again the demon acts at 1000 evenly spaced times within a trajectory. Panel (d) shows that the demon learns a protocol for which the mean entropy dissipation is negative: The system has *absorbed* heat from the surroundings. Panels (e) and (f) show that the learned feedback-control protocols resemble those of the time-dependent protocols of panels (b) and (c), but now the demon can make small changes to control parameters in response to fluctuations of magnetization. The demon has learned to manipulate the field in response to fluctuations, so that the entropy stored as spins flip against the field exceeds the entropy produced by spins flipping with the field. [We have used entropy production as an order parameter in order to make contact with prior work; we have also verified (see Fig. 6) that the demon can learn to perform magnetization reversal with negative heat absorption, heat transfer being a more conveniently accessible quantity in experiment.]

In Fig. 3(g), we summarize the protocol learned by the demon after 300 generations, which specifies T and h when given t/t_f and m. Neural networks are convenient ways of learning smooth (though potentially rapidly varying) protocols that interpolate to values of inputs not seen during training. In this case, the deep neural net contains only 144 parameters, encoding, in an efficient way, a protocol that reverses magnetization with net negative dissipation.

IV. CONCLUSIONS

We have developed feedback-control protocols for simulated fluctuating nanosystems, using genetic algorithms and Monte Carlo algorithms applied to a deep-neural-network demon. When provided with time alone, the demon learns the optimal-control protocols known analytically or from numerical studies [16,22,23]. When also given feedback from the system, the demon learns protocols that extract work or store heat.

The problem considered here is one of process control [38,39], and in machine-learning terms, the approach is a form of deep reinforcement learning: training an agent, a deep neural network, to carry out time- and state-dependent actions in order to maximize a time-dependent order parameter. Monte Carlo and genetic algorithms applied to neural networks are forms of neuroevolution [40] and are not traditionally considered part of the canon of reinforcement-learning algorithms [41]. However, they are capable of solving reinforcement-learning problems, and for the problems considered here, maximizing a long-time return can be achieved even if no short-time reward is known (other forms of control algorithms have also been used to treat molecular-scale dynamical systems [42–45]).

There exist numerical methods for controlling fluctuating nanosystems: For instance, optimal or near-optimal time-dependent protocols can be obtained by Monte Carlo path-sampling methods [18] or by gradient-based methods used to train parametrized functions [22]. Feedback-control protocols for flashing Brownian ratchets have been developed using gradient-based methods of reinforcement learning applied to a deep neural network [46]. All of these are powerful numerical methods but are not immediately applicable to experiment: Reference [18] uses Monte Carlo moves to generate the stochastic trajectory (rather than applying Monte Carlo moves only to the protocol, with trajectories generated independently of the method), while Refs. [22,46] use gradient information not directly accessible in experiments. The significance of the present method is that it uses only experimentally accessible data, such as the total work or heat produced by a set of stochastic trajectories. From this information alone, it is possible to learn an optimal time-dependent protocol. Further, given experimentally accessible time-resolved information, such as the force on an optical trap or the global magnetization of a nanomagnetic system, the demon can learn a feedback-control protocol to extract work or store heat. The human specifies the order parameter—minimize the work done or minimize dissipation while executing magnetization reversal—but the demon learns autonomously, without human intervention.

The key step in applying the procedure outlined in Fig. 1(c) to experiment is to connect the neural-network demon to the outputs of the system and to the experimental apparatus that controls the protocol: The demon must take in information from the system and output new values for the protocol control parameters. A natural choice for the neural network is to have one input neuron for each piece of experimental information (e.g., time and magnetization) and one for each control parameter of the protocol (e.g. temperature and magnetic field). There are many possible internal neural-network structures that can express the latter as a function of the former; the fully connected deep net used here is a simple and convenient choice. (Note that if the protocol is specified as a function of time alone, then it is deterministic and can be provided to the apparatus prior to the experiment in the form of a table of control-parameter values at different times.) In this paper, we have fed the demon accurate information, but experiments contain imperfections. Learning can proceed in the presence of imperfections, such as feedback delay (see Fig. 8), and for any real system, the impact of these imperfections on learning must be assessed on a case-by-case basis.

The learning framework can be used, in principle, to control protocols of considerable complexity. The neural-network encoding of a protocol is an efficient way to cope with a large number of input and control parameters: The number of neural-net parameters scales linearly with input and output parameters, and the methods used here have been used to train neural networks containing millions of parameters [19,24].

The results presented here show that work-extracting and heat-storing protocols can be learned for fluctuating nanosystems using experimentally available measurements and no prior knowledge of protocol. In the context of interacting many-body systems such as nanomagnetic devices, these results suggest the possibility of not simply minimizing dissipation during computing but of converting part of the entropy increase of the demon as it measures the system into a form of nanoscopic cooling as computation occurs.

ACKNOWLEDGMENTS

The code for training the neural-network demon can be found in Ref. [47]. I thank Megan Engel for providing the data labeled NE and GD in Fig. 3(a), and Corneel Casert and Isaac Tamblyn for discussions. This work was performed at the Molecular Foundry at Lawrence Berkeley National Laboratory, supported by the Office of Basic Energy Sciences of the U.S. Department of Energy under Contract No. DE-AC02–05CH11231.

APPENDIX: SUPPLEMENTARY FIGURES

Figures 4–8 supplement Figs. 1–3 of the main text.

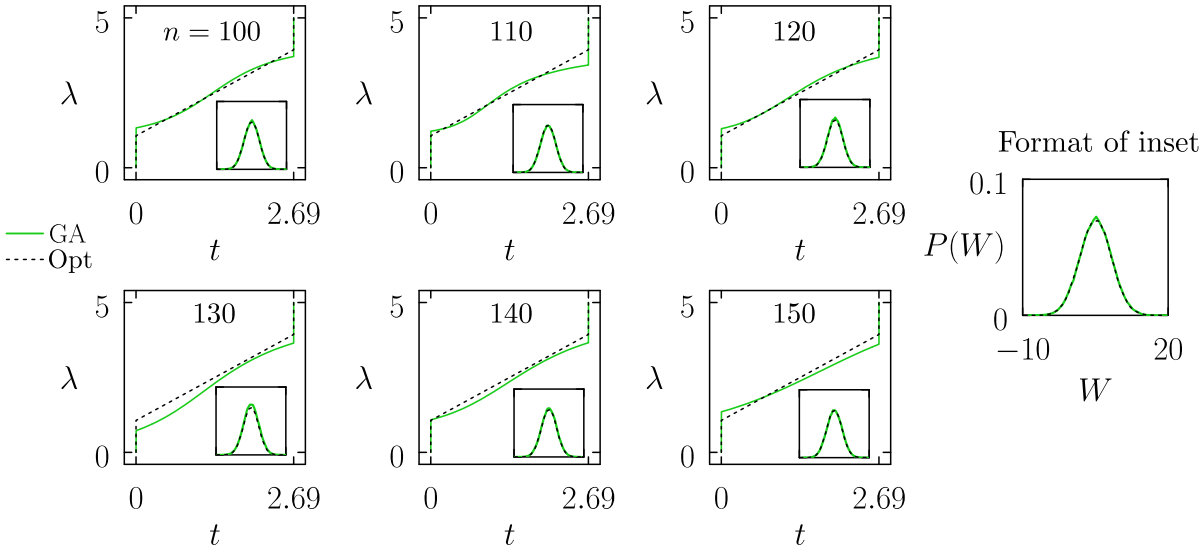

FIG. 4. Supplement to Fig. 2. (a) As in Fig. 2(d) but with results for five additional generations. The shape of the learned protocol fluctuates around the optimal one, but all are similarly efficient.

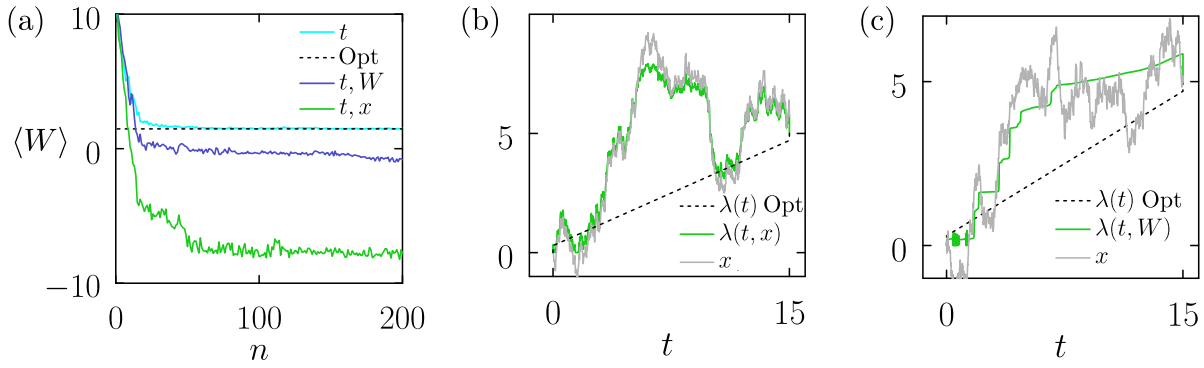

FIG. 5. Supplement to Fig. 2. (a) Similar to Fig. 2(e) but including the outcome of training when the demon inputs are time t and time-integrated work W (blue). Some work can be extracted in this case, but less than with time and bead position as input (green). Panels (b) and (c) are similar to Fig. 2(g). Panel (b) shows that, knowing the bead position, the demon can move the trap so as to extract large amounts of work from position fluctuations. (c) Time-integrated work, which contains less information about position fluctuations. This becomes increasingly true as time elapses. The demon can use this information to extract work but less efficiently than if presented with the instantaneous position.

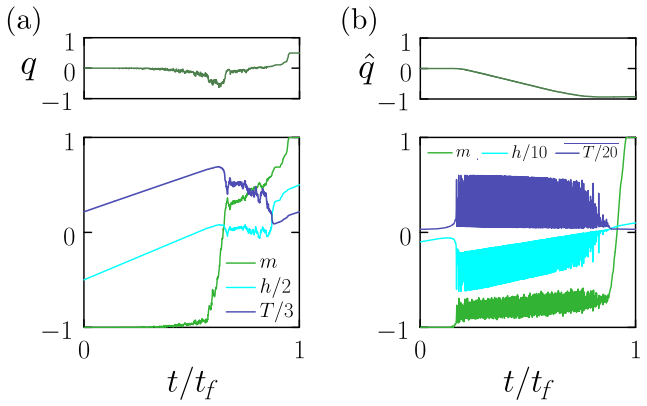

FIG. 6. Supplement to Fig. 3. Similar to Fig. 3(f) but with heat $Q = \sum_k \Delta E_k$ replacing Eq. (6) as the order parameter to be minimized. Panels (a) and (b) show magnetization-reversal cycles accompanied by heat flow out of and into the system, respectively. Here, $q \equiv Q/N$ (recall that $N = 32^2$) and $\hat{q} \equiv q/200$.

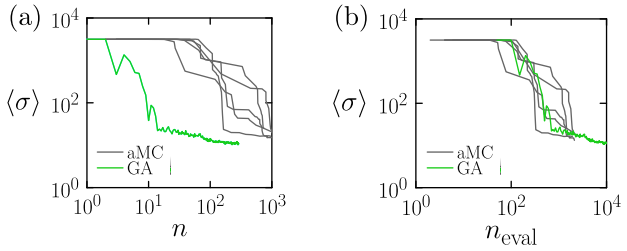

FIG. 7. Supplement to Fig. 3. Similar to Fig. 3(d) (with the demon trained using time alone), showing the GA result of that figure (green) against five results obtained using aMC as a method of training (gray). The horizontal axes of panels (a) and (b) denote the number of epochs (training cycles) and number of evaluations of the loss function (proportional to the number of trajectories run), respectively. GA is particularly convenient if trajectories can be run in batches or in parallel. In this example, aMC and GA converge similarly quickly per trajectory.

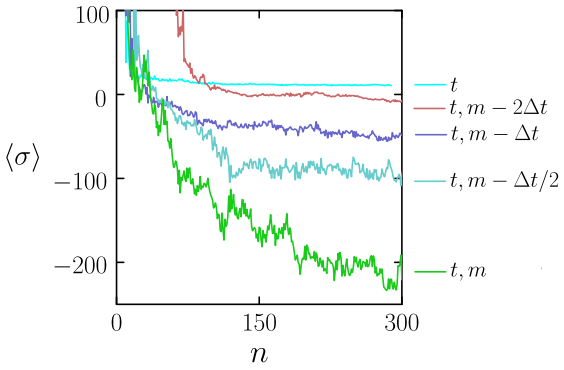

FIG. 8. Supplement to Fig. 3. Same as Fig. 3(d) but including results (red, purple, and cyan) in which the demon is fed time t and magnetization at a prior time, where $\Delta t = t_0/10^3$. This time delay is designed to mimic feedback delay in an experimental system. With delayed feedback, the demon can still arrange for entropy to be absorbed during magnetization reversal but less successfully than when it is fed magnetization without a delay. In the presence of a delay, its ability to exploit magnetization fluctuations is impaired, the more so as the delay increases.

[1] David Chandler, *Introduction to Modern Statistical Mechanics* (Oxford University Press, Oxford, England, 1987).

[2] Ralph Howard Fowler, *Statistical Thermodynamics* (Cambridge University Press Archive, Cambridge, England, 1939).

[3] Christopher Jarzynski, *Nonequilibrium Equality for Free Energy Differences*, Phys. Rev. Lett. **78**, 2690 (1997).

[4] Denis J. Evans and Debra J. Searles, *The Fluctuation Theorem*, Adv. Phys. **51**, 1529 (2002).

[5] Gavin E. Crooks, *Entropy Production Fluctuation Theorem and the Nonequilibrium Work Relation for Free Energy Differences*, Phys. Rev. E **60**, 2721 (1999).

[6] Udo Seifert, *Stochastic Thermodynamics, Fluctuation Theorems and Molecular Machines*, Rep. Prog. Phys. **75**, 126001 (2012).

[7] James Clerk Maxwell, *Theory of Heat* (Appleton, London, 1888).

[8] Leo Szilard, *On the Decrease of Entropy in a Thermodynamic System by the Intervention of Intelligent Beings*, Behav. Sci. **9**, 301 (1964).

[9] Takahiro Sagawa and Masahito Ueda, *Nonequilibrium Thermodynamics of Feedback Control*, Phys. Rev. E **85**, 021104 (2012).
[10] Massimiliano Esposito and Gernot Schaller, *Stochastic Thermodynamics for "Maxwell Demon" Feedbacks*, Europhys. Lett. **99**, 30003 (2012).
[11] Jordan M. Horowitz and Massimiliano Esposito, *Thermodynamics with Continuous Information Flow*, Phys. Rev. X **4**, 031015 (2014).
[12] Juan M. R. Parrondo, Jordan M. Horowitz, and Takahiro Sagawa, *Thermodynamics of Information*, Nat. Phys. **11**, 131 (2015).
[13] Jannik Ehrich, Susanne Still, and David A. Sivak, *Energetic Cost of Feedback Control*, arXiv:2206.10793.
[14] Shoichi Toyabe, Takahiro Sagawa, Masahito Ueda, Eiro Muneyuki, and Masaki Sano, *Experimental Demonstration of Information-to-Energy Conversion and Validation of the Generalized Jarzynski Equality*, Nat. Phys. **6**, 988 (2010).
[15] Tushar K. Saha, Joseph N. E. Lucero, Jannik Ehrich, David A. Sivak, and John Bechhoefer, *Bayesian Information Engine that Optimally Exploits Noisy Measurements*, Phys. Rev. Lett. **129**, 130601 (2022).
[16] Tim Schmiedl and Udo Seifert, *Optimal Finite-Time Processes in Stochastic Thermodynamics*, Phys. Rev. Lett. **98**, 108301 (2007).
[17] Grant M. Rotskoff and Gavin E. Crooks, *Optimal Control in Nonequilibrium Systems: Dynamic Riemannian Geometry of the Ising Model*, Phys. Rev. E **92**, 060102(R) (2015).
[18] Todd R. Gingrich, Grant M. Rotskoff, Gavin E. Crooks, and Phillip L. Geissler, *Near-Optimal Protocols in Complex Nonequilibrium Transformations*, Proc. Natl. Acad. Sci. U.S.A. **113**, 10263 (2016).
[19] Stephen Whitelam, Viktor Selin, Ian Benlolo, Corneel Casert, and Isaac Tamblyn, *Training Neural Networks Using Metropolis Monte Carlo and an Adaptive Variant*, Mach. Learn. **3**, 045026 (2022).
[20] Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O. Stanley, and Jeff Clune, *Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning*, arXiv:1712.06567.
[21] Stephen Whitelam and Isaac Tamblyn, *Learning to Grow: Control of Material Self-Assembly Using Evolutionary Reinforcement Learning*, Phys. Rev. E **101**, 052604 (2020).
[22] Megan C. Engel, Jamie A. Smith, and Michael P. Brenner, *Optimal Control of Nonequilibrium Systems through Automatic Differentiation*, arXiv:2201.00098.
[23] Adrianne Zhong and Michael R. DeWeese, *Limited-Control Optimal Protocols Arbitrarily Far from Equilibrium*, Phys. Rev. E **106**, 044135 (2022).
[24] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever, *Evolution Strategies as a Scalable Alternative to Reinforcement Learning*, arXiv:1703.03864.
[25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning* (MIT Press, Cambridge, MA, 2016).
[26] John H. Holland, *Genetic Algorithms*, Sci. Am. **267**, 66 (1992).
[27] Melanie Mitchell, *An Introduction to Genetic Algorithms* (MIT Press, Cambridge, MA, 1998).
[28] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller, *Equation of State Calculations by Fast Computing Machines*, J. Chem. Phys. **21**, 1087 (1953).
[29] Randall S. Sexton, Robert E. Dorsey, and John D. Johnson, *Beyond Backpropagation: Using Simulated Annealing for Training Neural Networks*, J. Org. End User Comput. **11**, 3 (1999).
[30] L. M. Rasdi Rere, Mohamad Ivan Fanany, and Aniati Murni Arymurthy, *Simulated Annealing Algorithm for Deep Learning*, Proc. Comput. Sci. **72**, 137 (2015).
[31] Rohun Tripathi and Bharat Singh, *RSO: A Gradient Free Sampling Based Approach for Training Deep Neural Networks*, arXiv:2005.05955.
[32] Stephen Whitelam, Viktor Selin, Sang-Won Park, and Isaac Tamblyn, *Correspondence between Neuroevolution and Gradient Descent*, Nat. Commun. **12**, 1 (2021).
[33] D. M. Carberry, James Cowie Reid, G. M. Wang, Edith M. Sevick, Debra J. Searles, and Denis J. Evans, *Fluctuations and Irreversibility: An Experimental Demonstration of a Second-Law-Like Theorem Using a Colloidal Particle Held in an Optical Trap*, Phys. Rev. Lett. **92**, 140601 (2004).
[34] Brian Lambson, David Carlton, and Jeffrey Bokor, *Exploring the Thermodynamic Limits of Computation in Integrated Systems: Magnetic Memory, Nanomagnetic Logic, and the Landauer Limit*, Phys. Rev. Lett. **107**, 010604 (2011).
[35] Lars Onsager, *Crystal Statistics. I. A Two-Dimensional Model with an Order-Disorder Transition*, Phys. Rev. **65**, 117 (1944).
[36] James J. Binney, N. J. Dowrick, A. J. Fisher, and M. Newman, *The Theory of Critical Phenomena: An Introduction to the Renormalization Group* (Oxford University Press, New York, 1992).
[37] We use an expression similar to Eq. (7) if time alone is used as input to the neural network, with $\boldsymbol{g}_{\boldsymbol{\theta}}(t/t_f, m)$ replaced by $\boldsymbol{g}_{\boldsymbol{\theta}}(t/t_f)$.
[38] Dale E. Seborg, Thomas F. Edgar, Duncan A. Mellichamp, and Francis J. Doyle III, *Process Dynamics and Control* (John Wiley & Sons, New York, 2016).
[39] Roy Langton, *Stability and Control of Aircraft Systems: Introduction to Classical Feedback Control* (John Wiley & Sons, New York, 2006).
[40] Dario Floreano, Peter Dürr, and Claudio Mattiussi, *Neuroevolution: From Architectures to Learning*, Evol. Intell. **1**, 47 (2008).
[41] Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning: An Introduction* (MIT Press, Cambridge, MA, 2018).

[42] Eurika Kaiser, J. Nathan Kutz, and Steven L. Brunton, *Data-Driven Discovery of Koopman Eigenfunctions for Control*, Mach. Learn. **2**, 035023 (2021).

[43] Marc Z. Miskin, Gurdaman Khaira, Juan J. de Pablo, and Heinrich M. Jaeger, *Turning Statistical Physics Models into Materials Design Engines*, Proc. Natl. Acad. Sci. U.S.A. **113**, 34 (2016).

[44] Carl P. Goodrich, Ella M. King, Samuel S. Schoenholz, Ekin D. Cubuk, and Michael P. Brenner, *Designing Self-Assembling Kinetics with Differentiable Statistical Physics Models*, Proc. Natl. Acad. Sci. U.S.A. **118**, e2024083118 (2021).

[45] Jonathan Y. Suen and Saket Navlakha, *A Feedback Control Principle Common to Several Biological and Engineered Systems*, J. R. Soc. Interface **19**, 20210711 (2022).

[46] Dong-Kyum Kim and Hawoong Jeong, *Deep Reinforcement Learning for Feedback Control in a Collective Flashing Ratchet*, Phys. Rev. Res. **3**, L022002 (2021).

[47] See https://github.com/swhitelam/demon.