2023 IEEE International Conference on Rebooting Computing (ICRC) | 979-8-3503-8204-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICRC60800.2023.10386858

Thermodynamic AI and the Fluctuation Frontier

Patrick J. Coles, Collin Szczepanski, Denis Melanson, Kaelan Donatella, Antonio J. Martinez, Faris Sbahi
Normal Computing Corporation, New York, New York, USA

***Abstract*—Many Artificial Intelligence (AI) algorithms are inspired by physics and employ stochastic fluctuations. We connect these physics-inspired AI algorithms by unifying them under a single mathematical framework that we call Thermodynamic AI, including: (1) Generative diffusion models, (2) Bayesian neural networks, (3) Monte Carlo sampling and (4) Simulated annealing. Such Thermodynamic AI algorithms are currently run on digital hardware, ultimately limiting their scalability and overall potential. Stochastic fluctuations naturally occur in physical thermodynamic systems, and such fluctuations can be viewed as a computational resource. Hence, we propose a novel computing device, called Thermodynamic AI hardware, that could accelerate such algorithms. We contrast Thermodynamic AI hardware with quantum computing where noise is a roadblock rather than a resource. Thermodynamic AI hardware can be viewed as a novel form of computing, since it uses a novel fundamental building block. We identify stochastic units (s-units) as the building blocks for Thermodynamic AI hardware. In addition to these s-units, Thermodynamic AI hardware employs a Maxwell's demon device that guides the system to produce non-trivial states. We provide a few simple physical architectures for building these devices.**

1. Introduction

In the past few years, AI has exceeded the expectations of even many optimists. The release of text-to-image models made tremendous impact in the fields of automated art and automated design while capturing people's imagination. Large language models that automate text generation have also impacted many industries. Much of this recent progress has happened through software and algorithmic advances, while largely keeping the standard hardware paradigm of digital computers with parallel processing. There may await another revolution in scaling up AI through fundamentally distinct, domain-specific hardware. This viewpoint has amassed increasing popularity, with initial strides where algorithm and hardware are considered inseparable [1].

Undoubtedly, much of the explosion in AI progress over the last two decades has been driven by the scaling up of deep neural-network architectures. Meanwhile, the fundamental algorithmic components of such architectures had been in place for multiple decades prior to their recognition of unassailable promise. Indeed, a hardware "fluke" which presented equipment to accelerate matrix-vector multiplications (MVMs) enabled this step change [2]. Namely, this has been driven by parallelized digital hardware such as graphical processing units (GPUs) and field-programmable gate arrays (FPGAs) [3]. More recently, analog hardware has been developed to reduce the power consumption in performing MVMs [4], [5], and there is a rich history of analog hardware for neural networks [6], [7], [8].

The next revolution in AI hardware may require identifying the physical basis of intelligence, possibly taking living systems as inspiration. For example, Jeremy England has connected some principles of life, such as self-organization, self-replication, and adaptation to fundamental ideas in thermodynamics such as the fluctuation-dissipation theorem [9]. To the extent that intelligence is associated with life, then intelligence in turn is associated with thermodynamics. Independent of this view, some of the most successful approaches to Artificial Intelligence (AI) are inspired by physics, and often employ stochastic fluctuations.

Let us consider an analogy to Preskill's proposed symbiosis between quantum computing and entanglement [10]. Entanglement between multiple subsystems is, in general, difficult to simulate with classical computers and hence is a main factor in what gives quantum computers their power. While entanglement is a key ingredient for quantum computers, studying it also is a key application for quantum computers, since for example topological quantum materials and strongly correlated-electronic systems such as superconductors typically exhibit many-body entanglement that quantum computers could shed light on.

By analogy, thermodynamic fluctuations could be a key ingredient for building (artificial) intelligence. Many AI primitives rely on simulating stochastic fluctuations [11]. Examples include generative diffusion models [12], time-series analysis with neural stochastic differential equations (SDEs) [13], [14] and Bayesian neural networks [15], [16]. Generative modeling necessarily involves fluctuations since one does not simply want to replicate the data but rather randomly generate new samples. Instead of simulating such fluctuations with standard digital hardware, why not design a hardware that has thermodynamic fluctuations already built in as a fundamental building block.

Conversely large-scale thermodynamic AI machines could allow us to pursue the answers to deep questions about living systems or other adapting, self-evolving systems. There is a general principle in computing that one should simulate apples with apples [17]. One should match the hardware design to the system that one would like to simulate, to maximize efficiency. There remain many interesting open questions about the origins of life, the origins of intelligent life, and the connections between life and the

laws of physics, including testing England's theories related to these questions. Perhaps one could use thermodynamic AI machines as simulators for complex fluctuating systems. In this sense, thermodynamic AI machines will allow us to push forward into the fluctuation frontier.

The first goal of this article is to stimulate discussion around a new computing paradigm. We note that Thermodynamic AI can be viewed as a sub-field of Thermodynamic Computing. The latter was discussed broadly in a workshop report [18], and was further explored by Hylton et al. [19], [20]. Additional studies along these lines include thermodynamic neuromorphic systems [21], [22], [23], connecting machine learning to work production [24], and other thermodynamic perspectives on learning [25], [26], [27], [28]. We use the term Thermodynamic AI to refer to a thermodynamic viewpoint on hardware for modern AI applications. Hence we hope to draw attention to this viewpoint.

Our second goal is to propose a mathematical and conceptual framework for Thermodynamic AI hardware. In what follows we provide a framework that consists of:

1) Proposing stochasticity as a computational resource.
2) Identifying s-units (stochastic-units) as the basic building block of Thermodynamic AI hardware.
3) Discussing possible physical architectures for building s-units.
4) Connecting the thermodynamic concept of Maxwell's demon to AI algorithmic primitives., thus necessitating that Thermodynamic AI hardware include a physical realization of Maxwell's demon.
5) Presenting noise robustness and error correction concepts for Thermodynamic AI hardware.
6) Identifying key algorithms that could be sped up by Thermodynamic AI hardware, and unifying these algorithms under a single mathematical framework called Thermodynamic AI algorithms.

2. Stochasticity as a computing resource

"Fluctuation" is an intuitive term used by physicists oftentimes to describe the tendency to deviate from the average value. For example, vibrations of atoms in a crystal cause their spatial displacements to fluctuate about their equilibrium locations. We will use the term "fluctuation" synonymously with the term "stochasticity". Stochasticity has a precise mathematical description, and therefore it will allow us to precisely characterize the building blocks of thermodynamic AI systems. The key idea will be to view stochasticity as a resource that one can use to accomplish computational tasks. At first this may seem counter-intuitive, since stochasticity has inherent randomness to it.

However, randomness is useful as a resource for various tasks. Randomness is widely recognized as a resource in cryptography [29]. Randomness is also viewed as a resource in computing, including quantum computing [30]. This provides motivation for probabilistic computing based on p-bits [31], [32], [33]. In particular, Monte Carlo algorithms, which represent one of the most widely used class of algorithms, also employ randomness as a resource.

There is a close relationship between randomness and stochasticity, since these resources can be interconverted, albeit with some overhead. We note that stochasticity as a computational resource has been promoted previously by Mansinghka [11], in the context of digital stochastic circuits, and Ref. [11] elaborates on how stochasticity can accelerate various tasks, such as sampling problems. Here, we consider several examples to illustrate how stochasticity is a resource:

(1) In generative modeling, one often adds noise to a target distribution to facilitate the learning process. This is due to the manifold hypothesis (the data resides near a low dimensional manifold), which makes score matching poorly defined in the noise-free case. This gave rise to diffusion models [34] whereby noise is digitally generated and added at discrete levels to the original data. Although effort has been made to make the noise levels more continuous, ultimately digital devices are inherently discrete. In contrast, a physical system with stochastic building blocks would naturally produce its own noise and would naturally evolve continuously. In this spirit of simulating apples with apples, the physical stochastic hardware would better match the underlying mathematics of diffusion models, and as a consequence one would expect better computational efficiency.

(2) In annealing-based optimization such as simulated annealing, the (simulated) thermal fluctuations allow the system to explore different minima in the landscape before settling down in a high-quality minimum. If one was to use a physical system at finite temperature, then these thermal fluctuations would naturally occur, allowing one to explore many different minima in the landscape. Along these lines, researchers have indeed repurposed quantum annealers [35] as physical thermal annealers to make use of these thermal fluctuations, although we remark that in principle thermal annealers do not need to be quantum devices.

(3) Suppose that one wants to integrate a stochastic differential equation that describes the price of a financial asset. Digitally, one would discretize time and mimic stochasticity with pseudo-random elements and then perform matrix-vector multiplications for several time steps to compute the integral. But if one had access to physical building blocks that were inherently stochastic, then one could appropriately set initial conditions and the parameters of their dynamics and then just check the state at a later time, without any computation at all and obtain the desired result.

3. Unification of modern AI algorithms

Our goal is to motivate a hardware paradigm that is relevant to multiple AI applications. A stepping stone towards this goal is to mathematically unify different AI algorithms under the same framework. Hence, a byproduct of our efforts is a conceptual and mathematical unification of AI algorithms that are generally considered different and unrelated. This is illustrated in Figure 1. Specifically we consider the following applications:

1) Generative diffusion models

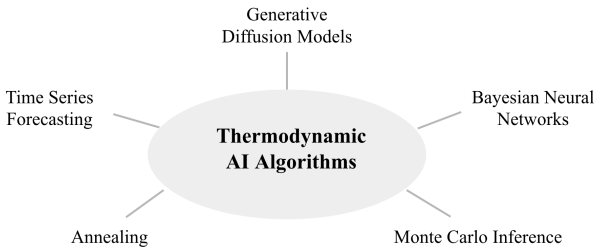

Figure 1. Illustration of various algorithms unified under the single mathematical framework of Thermodynamic AI algorithms.

2) Bayesian neural networks
3) Monte Carlo inference
4) Annealing
5) Time series forecasting

Each application is elaborated in more detail in Ref. [36], while we focus particularly on the Diffusion Model application in Sec. 8 to illustrate our framework. Through careful consideration, we manage to formulate a mathematical framework that encompasses all of the aforementioned algorithms as special cases. We say that these algorithms belong to a class called *Thermodynamic AI algorithms*.

At a conceptual level, we can define Thermodynamic AI algorithms as those consisting of at least two subroutines:

1) A subroutine in which a stochastic differential equation (SDE) is evolved over time.
2) A subroutine in which a Maxwell's demon (see Sec. 6 for elaboration) observes the state variable in the SDE and applies a drift term in response.

At the mathematical level, we propose that Thermodynamic AI algorithms are ones that simulate or implement the following set of equations (or some subset of them):

$$d\mathbf{p} = [\mathbf{f} - BM^{-1}\mathbf{p}]\,dt + D\,d\mathbf{w} \tag{1}$$

$$d\mathbf{x} = M^{-1}\mathbf{p}\,dt \tag{2}$$

$$\mathbf{f} = -\nabla_{\mathbf{x}} U_\theta \tag{3}$$

One can see that these correspond to Newton's laws of motion, with the addition of diffusion and friction. In these equations, $\mathbf{p}$, $\mathbf{x}$, and $\mathbf{f}$ respectively are the momentum, position, and force. The matrices M, D, and B are hyperparameters, with M being the mass matrix and D being the diffusion matrix. The $d\mathbf{w}$ term is a Wiener process. Finally, U_θ is a (trainable) potential energy function. Typically, much of the application-specific information, regarding the task to be solved, is encoded in the potential energy function U_θ. Note that for readability we omit the dependencies of the variables on time t and space $\mathbf{x}$ in the above equations.

As we said, this unification is crucial to developing a hardware paradigm that is broadly applicable to many AI algorithms. However, the mathematical unification can itself be useful, even outside of the development of novel hardware. All of the aforementioned algorithms are currently implemented in standard, digital hardware, but typically with different software programs. In principle, one can use the mathematical framework presented in this article to develop a unified algorithmic framework for running these applications on standard, digital hardware. For example, see Ref. [37] for early work on a unified software approach for probabilistic algorithms. In this article we aim to keep the focus on our novel hardware paradigm. Hence, we simply leave this algorithmic unification as a remark that we do not discuss further in this article.

4. Fundamental building blocks

	classical	quantum	thermodynamic
discrete	bit	qubit	s-bit
continuous	mode	qumode	s-mode

Let us now discuss the fundamental building blocks of thermodynamic AI hardware. As the name "thermodynamic" suggests, a thermodynamic system is inherently dynamic in nature. Therefore, the fundamental building blocks should also be dynamic. This is contrast to classical bits or qubits, where the state of the system ideally remains fixed unless it is actively changed by gates. The thermodynamic building block should passively and naturally evolve over time, even without the application of gates.

But what dynamical process should it follow? A reasonable proposal is a stochastic Markov process. Naturally this should be continuous in time, since no time point is more special than any other time point. Hence, the discrete building block, which we call an s-bit, would follow a continuous-time Markov chain (CTMC). Here the "s" in s-bit stands for stochastic.

For the continuous building block, which we call an s-mode, the natural analog would be a Brownian motion (also known as a Wiener process). One can impose that this process is a Martingale (which is typically assumed for Brownian motion), which means that it has no bias. We use s-unit as a generic term to encompass both s-bits and s-modes. We note that stochastic building blocks were also considered by Mansinghka et al. [11], [38] using digital logic, which is different from our (analog) approach.

5. Physical realizations of stochastic units

In what follows, we focus our attention on s-modes, since continuous variables are particularly relevant to AI applications (Ref. [36] gives some discussion of s-bits.) An s-mode represents a continuous stochastic variable whose dynamics are governed by drift, diffusion, or other physical processes (akin to a Brownian particle). At the heart of any physical implementation of such a variable will be a source of stochasticity. A natural starting point for implementing thermodynamic AI hardware is analog electrical circuits, as these circuits have inherent fluctuations that could be harnessed for computation.

The most ubiquitous source of noise in electrical circuits is thermal noise [39]. Thermal noise, also called Johnson-Nyquist noise, comes from the random thermal agitation of

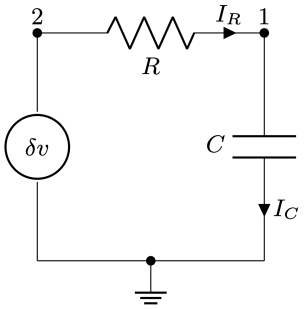

Figure 2. Circuit diagram of a possible physical realization of an s-mode, comprising of a noisy resistor and a capacitor.

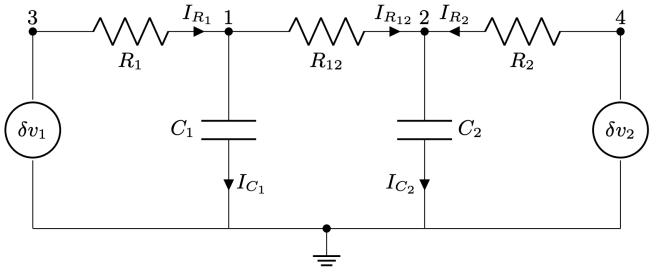

Figure 3. Circuit diagram of a possible means to couple s-modes, using a coupling resistor.

the charge carriers in a conductor, resulting in fluctuations in voltage or current inside the conductor. Thermal noise is Gaussian and has a flat frequency spectrum (white noise) with fluctuations in the voltage of standard deviation

$$v_{\mathrm{tn}} = \sqrt{4k_B T R \Delta f}, \tag{4}$$

where R is the resistance, k_B is the Boltzmann constant, T is the absolute temperature and Δf is the frequency bandwidth. The amplitude of the voltage fluctuations can be controlled by changing the temperature or the resistance. In practice, a thermal noise source can be implemented using a large resistor in series with a voltage amplifier.

Another type of electrical noise is shot noise [39]. Shot noise arises from the discrete nature of charge carriers and from the fact that the probability of a charge carrier crossing a point in a conductor at any time is random. This effect is particularly important in semiconductor junctions where the charge carriers should overcome a potential barrier to conduct a current. The probability of a charge carrier passing over the potential barrier is an independent random event. This induces fluctuations in the current through the junction. Shot noise is Gaussian with a flat frequency spectrum (white noise) and current fluctuations of standard deviation

$$I_{\mathrm{tn}} = \sqrt{2q|I|\Delta f}, \tag{5}$$

where I is the current through the junction, q is the electron charge and Δf is the frequency bandwidth. The amplitude of the current fluctuations can be controlled by changing the magnitude of the DC current passing through the junction. In practice, a shot noise source can be implemented using a pn diode (for example, a Zener diode in reverse bias configuration) in series with a controllable current source [40].

In general, any physical implementation of s-modes should have the amplitude of its stochasticity be independently controllable with respect to the other system parameters. We have seen that electrical thermal and shot noise, for example, both have tuning knobs to control the amplitude of the noise to some extent. In addition, one must ensure that the amplitude of the fluctuations be compatible, i.e. measurable, with the rest of the system. Thermal and shot noise sources typically have voltage fluctuations of the order of a few μV or less. For these fluctuations to be measurable by analog-to-digital converters measuring voltages on the order of hundreds of mV, amplification will be necessary. This amplification can be done using single- or multi-stage voltage amplifiers. Variable-gain amplifiers can also let one independently control the amplitude of the fluctuations.

The s-mode can be represented through the dynamics of any degree of freedom of an electrical circuit. If we chose the voltage on a particular node in the circuit as our degree of freedom of choice, a simple stochastic voltage noise source plays the role of the s-mode. This can be realized by using a noisy resistor at non-zero temperature. The circuit schematic in Fig. 2 shows the typical equivalent noise model for a noisy resistor composed of a stochastic voltage noise source, $\delta v(t)$, in series with an ideal (non-noisy) resistor of resistance R. The inherent terminal capacitance, C, of the resistor is also added to the equivalent resistor model [41], [42]. We chose to use the voltage on node 1 (labeled simply as $v(t)$ here) as our s-mode. The dynamics of the s-mode in this case, obeys the following SDE model:

$$-\frac{\mathrm{d}v(t)}{\mathrm{d}t} = \frac{v(t) + \delta v(t)}{RC}. \tag{6}$$

The voltage fluctuations in (6) have the usual Gaussian white noise properties $\langle \delta v(t) \rangle = 0$ and $\langle \delta v(t) \delta v(t') \rangle = 2k_B T R \delta(t - t')$ where $\langle\ \rangle$ denotes statistical averaging and $\delta(t - t')$ denotes the Dirac delta function. We notice the form of the SDE comprises a drift term proportional to $v(t)$ and a diffusion or stochastic term proportional to $\delta v(t)$.

5.0.1. Coupling s-modes. When building systems of many s-modes, one will wish to couple them to express correlations and geometric constraints. Again, the medium of analog electrical circuits presents a natural option for the coupling of s-modes. As an example, two s-modes could be coupled through a resistor, as pictured in Fig. 3. The coupled s-modes, represented by the voltage on nodes 1 and 2, are then coupled through their drift terms as (we omit the time dependencies for readability)

$$-\frac{\mathrm{d}v_1}{\mathrm{d}t} = \frac{v_1 + \delta v_1}{R_1 C_1} - \frac{v_2 - v_1}{R_{12} C_1}, \tag{7}$$

$$-\frac{\mathrm{d}v_2}{\mathrm{d}t} = \frac{v_2 + \delta v_2}{R_2 C_2} - \frac{v_1 - v_2}{R_{12} C_2}. \tag{8}$$

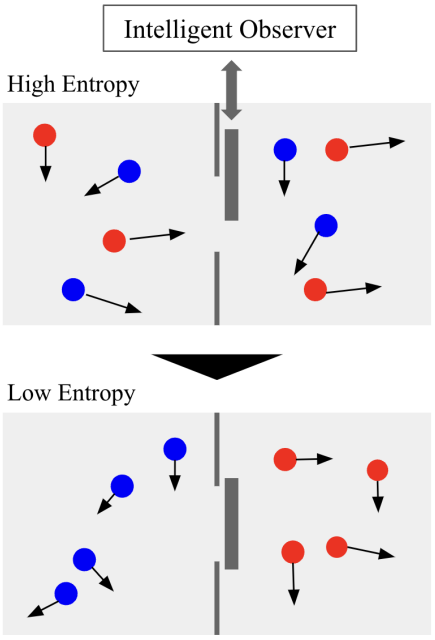

Figure 4. Illustration of Maxwell's Demon. An intelligent observer (the demon) monitors the locations of the components of a gaseous mixture, allowing it to separate the components of the gas, hence reducing the entropy of the system over time.

To simplify the notation, this system of coupled equations can be written in matrix form as follows

$$-\dot{\mathbf{v}} = \mathbf{C}^{-1}\left(\mathbf{J}\mathbf{v} + \mathbf{R}^{-1}\delta\mathbf{v}\right), \tag{9}$$

where $\dot{\mathbf{v}} \equiv \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{v}$ and

$$\mathbf{v} \equiv \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}, \ \mathbf{C} \equiv \begin{bmatrix} C_1 & 0 \\ 0 & C_2 \end{bmatrix}, \ \mathbf{R} \equiv \begin{bmatrix} R_1 & 0 \\ 0 & R_2 \end{bmatrix},$$
$$\mathbf{J} \equiv \begin{bmatrix} \frac{1}{R_1} + \frac{1}{R_{12}} & -\frac{1}{R_{12}} \\ -\frac{1}{R_{12}} & \frac{1}{R_2} + \frac{1}{R_{12}} \end{bmatrix}, \ \delta\mathbf{v} \equiv \begin{bmatrix} \delta v_1 \\ \delta v_2 \end{bmatrix}.$$

Here we introduced the self-resistance matrix $\mathbf{R}$, the capacitance matrix $\mathbf{C}$, and the conductance matrix $\mathbf{J}$. Capacitive coupling is an alternative means to couple s-modes [36].

6. Maxwell's Demon

Maxwell's Demon is a thermodynamic concept that is similar to refrigeration, as it allows a system's entropy to be reduced over time by interacting with an external system. Here, the demon acts as an intelligent observer who regularly gathers data from (i.e., measures) the system, and based on the gathered information, the demon performs some action on the system. The classic example involves a gaseous mixture and a physical barrier as illustrated in Figure 4. More generally, Maxwell's demon has been experimentally implemented in a variety of platforms [43], [44].

We argue that a Maxwell Demon is both: (1) Essential to the success of Thermodynamic AI systems due to the complex entropy dynamics required for AI applications, and (2) Quite straightforward to implement in practice for several different hardware architectures.

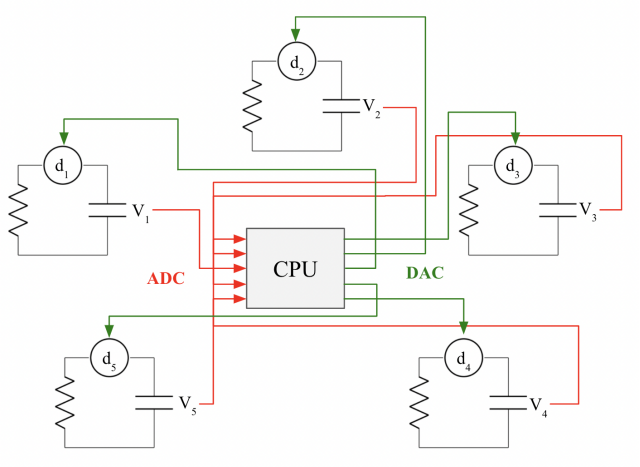

Figure 5. Illustration of how a Maxwell's Demon (stored on a CPU) could interact with a system of s-modes.

Regarding the first point, AI applications like Bayesian inference aim to approximate a posterior distribution, and it is known that such posteriors can be extremely complicated and multi-modal [45]. Similarly, generative modeling is intended to handle arbitrary data distributions. Hence, producing only Gaussian distributions, as an isolated s-mode system would do, will not suffice for these applications. Regarding the second point, we discuss how one can implement a Maxwell's Demon in hardware in the next subsection.

6.1. Digital Maxwell's Demon

We regard the Maxwell's Demon (MD) as an essential hardware component of Thermodynamic AI systems. One can construct the MD device in a variety of ways, with digital or analog approaches. A digital MD system is quite simple to describe. Namely, it can correspond to a neural network that is stored on a digital central processing unit (CPU). In this case, one would need to communicate the state vector $\mathbf{v}$ to the CPU (to be the input to the neural network), and then communicate the proposed action of the Maxwell's Demon (i.e., the output of the neural network) back to the thermodynamic hardware. Hence, one simply needs a means to interconvert signals between the thermodynamic hardware and the CPU. This is illustrated in Figure 5, with the interconversion shown as analog-to-digital converters (ADCs) and digital-to-analog converters (DACs).

6.2. Analog Maxwell's Demon

An analog MD device could allow one to integrate it more closely to the rest of the thermodynamic hardware. Moreover, this could allow one to avoid interconverting signals. Here we describe a force-based approach for the MD device, with additional approaches discussed in Ref. [36].

Recall from Newtonian mechanics that the force $\mathbf{f}(\mathbf{x})$ on a system at position $\mathbf{x}$ is given by the gradient of the potential energy function, $U(\mathbf{x})$, and the momentum $\mathbf{p}$ is proportional to the time derivative of the position:

$$\mathbf{f}(\mathbf{x}) = -\nabla_{\mathbf{x}} U(\mathbf{x}), \qquad \frac{d\mathbf{x}}{dt} = -M^{-1}\mathbf{p}\,. \tag{10}$$

Here M is the mass matrix. Equation (10) provides a recipe for how to construct an MD device. The idea would be to view the momentum $\mathbf{p}$ as the state vector associated with the s-mode system. Hence, the momentum will evolve over time according to the SDE equation associated with the s-mode system, such as Eq. (9). At a given time t, the MD system will take the momentum vector in as an input. Then the MD system will output a force that is a function of both the time t and the momentum $\mathbf{p}(t)$,

$$\text{Input from s-modes: } \mathbf{p}(t), \quad \text{Output to s-modes: } \mathbf{f}(t, \mathbf{p}(t)) \tag{11}$$

Let us discuss how the MD device performs the mapping from input to output in Eq. (11). Suppose the MD device has a latent variable or hidden variable, which corresponds to the position vector $\mathbf{x}(t)$. The latent variable $\mathbf{x}(t)$ is stored inside the MD device's memory, and it evolves over time. Specifically, it evolves over time according to the differential equation in Eq. (10). Hence, for the latent variable, we have:

$$\text{Latent variable:} \quad \mathbf{x}(t) = \mathbf{x}(0) - \int_{t'=0}^{t'=t} M^{-1}\mathbf{p}(t')dt'\,, \tag{12}$$

where $\mathbf{x}(0)$ is an initial starting point for the latent variable.

The MD device also stores a potential energy function $U_\theta(t, \mathbf{x}(t))$. For generality, we allow this potential energy function to be time-dependent. This time dependence is important for certain applications such as annealing, where one wishes to vary the potential energy function over time. In addition, we also allow the potential energy function to depend on a set of parameters θ. These parameters are trainable and will be trained during the training process, as in diffusion models in Sec. 8. Hence, the potential energy function represents the trainable portion of the MD device,

$$\text{Trainable Potential Energy Function:} \quad U_\theta(t, \mathbf{x}(t))\,. \tag{13}$$

Internally, the MD device combines (12) and (13) to produce a force. Specifically, the MD device employs Eq. (10) and computes the gradient of the potential energy function at time t and location $\mathbf{x}(t)$, as illustrated in Figure 6. The result of this computation is the output of the MD device:

$$\text{Output:} \quad \mathbf{d}_\theta(t, \mathbf{v}(t)) = \mathbf{d}_\theta(t, \mathbf{p}(t)) = -\nabla_{\mathbf{x}} U_\theta(t, \mathbf{x}(t))\,. \tag{14}$$

7. Thermodynamic Error Correction and Noise Robustness

7.1. Noise plaguing other computing paradigms

It is no secret that hardware noise has plagued particular computing paradigms. Indeed, noise is the key issue that is preventing quantum computing from being a commercially relevant technology. Noise in quantum computing can turn otherwise efficient algorithms into inefficient ones, which essentially destroys the quantum speedup that one would hope for over classical methods. Analog computing is another paradigm that suffers from hardware noise. Analog computing has a rich history, and at one point it was perhaps the dominant form of computing. However, digital computers become more precise and more economical in the 1950s to 1970s, which led to the decline of analog computing. Hardware noise in analog computing also played a role in their decline relative to digital computers.

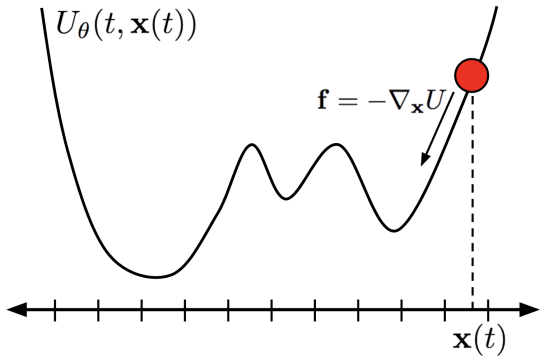

Figure 6. Force-based approach to constructing a Maxwell's Demon device.

7.2. Noise preserves the mathematical framework

In contrast, in Thermodynamic AI, noise is part of the design. It is a fundamental ingredient in the hardware. While quantum and analog computing view noise as a nuisance, Thermodynamic AI views noise as essential. While the precise physical construction of Thermodynamic AI hardware is flexible, a likely candidate would be using the same kind of analog components used in analog computing. These analog circuits will be noisy, of course. But this (unintentional) noise can be essentially combined with whatever noise sources that one is intentionally engineering into the Thermodynamic AI hardware. The final noise that results with be a combination of unintentional and intentional noise, and hence the unintentional noise becomes a non-issue.

To elaborate, suppose that one intentionally designs the hardware to have a drift matrix, drift vector, and diffusion matrix given respectively by A_0, $\mathbf{b}_0$, and C_0. Then naively the s-mode system is designed to evolve via the SDE:

$$d\mathbf{v}(t) = (A_0\mathbf{v}(t) + \mathbf{b}_0)dt + C_0 d\mathbf{w}\,. \tag{15}$$

However, in practice there will be unintentional noise. Examples of unintentional noise include capacitances of capacitors, or resistances of resistors, being off from their nominal values, or additional source of stochastic noise leading to an additional diffusion term. Mathematically, we encompass

these noise sources with a simple model, where the true values are perturbed away from A_0, $\mathbf{b}_0$, and C_0, as follows:

$$\tilde{A}_0 = A_0 + A_e, \qquad \tilde{\mathbf{b}}_0 = \mathbf{b}_0 + \mathbf{b}_e, \qquad \tilde{C}_0 = C_0 + C_e\,. \tag{16}$$

Here, A_e, $\mathbf{b}_e$, and C_e are the errors or perturbations, and the true objects implemented in hardware are $\tilde{A}_0$, $\tilde{\mathbf{b}}_0$, and $\tilde{C}_0$. Hence, the true SDE of the s-mode system is:

$$d\mathbf{v}(t) = (\tilde{A}_0\mathbf{v}(t) + \tilde{\mathbf{b}}_0)dt + \tilde{C}_0 d\mathbf{w}\,. \tag{17}$$

Equation (17) has the same mathematical form as Eq. (15), but with different hyperparameters. From a mathematical perpective, it is as if the hardware designer intentionally designed the hardware with the hyperparameters $\tilde{A}_0$, $\tilde{\mathbf{b}}_0$, and $\tilde{C}_0$. Hence, Eq. (17) still fits within the mathematical framework of Thermodyanmic AI hardware.

7.3. Maxwell's Demon learns to correct errors

In Section 6, we argued that a MD device was a key ingredient in Thermodynamic AI hardware, since it allows one to produce complicated entropy dynamics and complicated probability distributions. It turns out there is yet another benefit of employing a MD in the Thermodynamic AI hardware: error correction. Let us consider the noise model introduced above in Eq. (16). One might be worried that noise perturbing the hyperparameters would affect the overall performance of the Thermodynamic AI hardware. But that is where the MD system comes to the rescue.

Suppose that we perform *in situ* training of the MD, such that the MD is trained in the presence of the physical s-mode system. During training, one employs a loss function that is an objective, true measure of the performance on the AI task. This loss function treats the hardware as a black box that produces outputs; the loss is agnostic to the internal workings of the hardware. Hence, if one is can successfully minimize the loss function, then the system truly is performing well. It also implies that the MD system is well-suited to guide the hardware system to accomplish the desired AI task, regardless of the hardware's noise.

Hence, assume that the loss function is successfully minimized during training. Furthermore, assume that whatever noise processes that are present during training are also the same noise processes after training. In other words, assume that the perturbations in Eq. (16) do not change significantly over time, such that whatever noise is present during training is also present after training. Under these assumptions, the trained MD system will automatically correct for errors or noise. Training in the presence of noise makes the system robust to that same noise. Essentially, the MD system learns to account for the hardware noise whenever it produces its outputs, so that the MD output $\mathbf{d}(t, \mathbf{v}(t))$ will be appropriately adjusted to deal with the hardware noise. In this sense, Thermodynamic AI systems have inherent noise robustness.

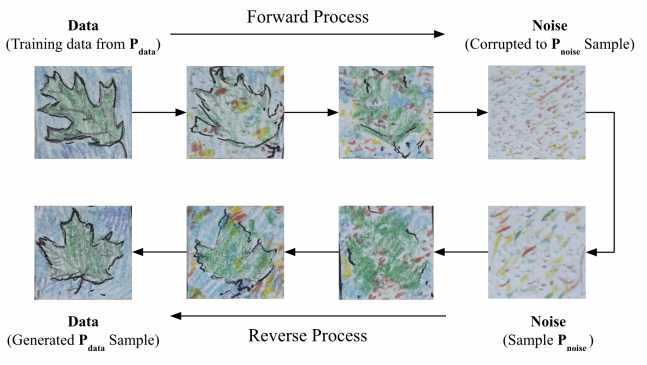

Figure 7. Diffusion models add noise to data drawn from a distribution p_{data} during a forward process, providing data to train a score network. A sample is then drawn from a noisy distribution, p_{noise}, and the process is reversed using the trained score network to generate a novel datapoint.

8. Application: Diffusion Models

8.1. Background

Diffusion models [12], [46] are a state-of-the-art method for implementing generative models. Figure 7 shows the basic idea of diffusion models. These models add noise to data drawn from a data distribution p_{data} during a forward process that evolves from from $t = 0$ to $t = T$. A sample is then drawn from a noisy distribution, p_{noise}, and the process evolves in the reverse direction, from $t = T$ to $t = 0$, to generate a novel datapoint. Both the forward and reverse processes can be described by SDEs [12]. Namely, the reverse SDE is similar to the forward SDE except that it has an additional drift term. A typical forward and reverse SDE for diffusion models has the form:

$$\mathrm{d}\mathbf{x} = f(t)\mathbf{x}\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}_t \quad \text{(Forward)} \tag{18}$$

$$\mathrm{d}\mathbf{x} = f(t)\mathbf{x}\mathrm{d}t - g(t)^2\mathbf{s}_\theta(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\overline{\mathbf{w}}_t \quad \text{(Reverse)} \tag{19}$$

where $\mathbf{x}$ is the continuous state variable. In the reverse SDE, time t runs backwards and $\mathrm{d}\overline{\mathbf{w}}_t$ is a standard Brownian motion term when time runs backwards. The vector $\mathbf{s}_\theta(\mathbf{x}, t)$ is a model for the score function $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ (the gradient of the logarithm of the probability distribution), and hence $\mathbf{s}_\theta(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$.

At a high level, the forward process provides training data to train a neural network, whose job is to output $\mathbf{s}_\theta(\mathbf{x}, t)$. This neural network is called the score network. Once the score network is trained, it can be used to guide the reverse process to generate novel samples.

In state-of-the-art diffusion models, the sampling rate still remains fairly slow. Hence, any means to speed up sampling rate could significantly improve this technology, and this is where Thermodynamic AI Hardware could help.

8.2. Fitting into our framework

Equations (18) and (19) each fall under the our framework, e.g., as special cases of Eq. (1). To make this more

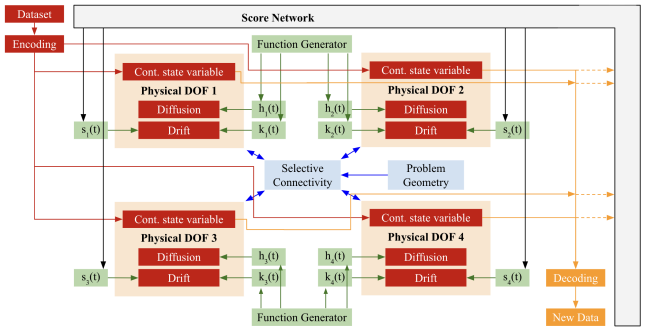

Figure 8. Schematic diagram of a Thermodynamic Diffusion Model. For simplicity, we show the case of four degrees of freedom (DOFs), where each DOF corresponds to an s-mode. These s-modes can be constructed as described in Sec. 5.

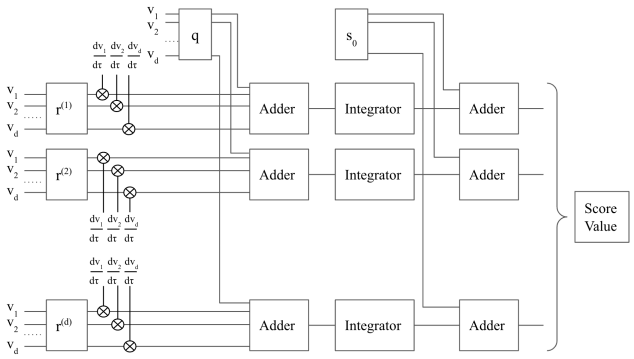

Figure 9. Schematic circuit diagram for an analog score network based on the total derivative approach. This diagram represents the process used to obtain score values during the evolution of the reverse diffusion process.

clear, one can do a change of variables $\tau = T - t$ in the reverse process and rewrite the diffusion model SDE equations as:

$$\mathrm{d}\mathbf{x} = f(t)\mathbf{x}\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}_t \quad \text{(Forward)} \tag{20}$$

$$\mathrm{d}\mathbf{x} = -f(T-\tau)\mathbf{x}\mathrm{d}\tau + g(T-\tau)^2\mathbf{s}_\theta(\mathbf{x}, T-\tau)\mathrm{d}\tau + g(T-\tau)\mathrm{d}\mathbf{w}_\tau \quad \text{(Reverse)} \tag{21}$$

In this case, both t and τ run forward in time, i.e., dt and $d\tau$ are positive increments in these equations. Note that $\mathrm{d}\mathbf{w}_\tau$ appears in this equation since $\mathrm{d}\mathbf{w}_\tau = \mathrm{d}\overline{\mathbf{w}}_t$. Since the time variables in these equations run forward in time, then they can be directly implemented in physical analog hardware, since time always runs forward in the physical world.

The key to fitting diffusion models into our framework is to make to following mapping:

$$\text{(diffusion process)} \leftrightarrow \text{(s-mode device)} \tag{22}$$

$$\text{(score network)} \leftrightarrow \text{(Maxwell's demon device)} \tag{23}$$

The mathematical diffusion process in diffusion models can be mapped to the physical diffusion process in the s-mode device. Similarly, the score vector outputted by the score network corresponds to the vector $\mathbf{d}(t, \mathbf{v}(t))$ outputted by the MD device. Hence, diffusion models fit in our framework for Thermodynamic AI systems.

8.3. Description of Diffusion Hardware

Figure 8 gives a schematic diagram of a Thermodynamic Diffusion Model. A variety of different physical paradigms can be used to implement this hardware, such as analog electrical circuits or continuous-variable optical systems. Hence, we describe the system an abstract level.

As shown in Figure 8 the physical system has multiple degrees of freedom (DOFs) - which essentially correspond to the s-modes. The number of DOFs matches the dimensionality of the data, i.e., the number of features in the data. Each DOF has a continuous state variable, and that variable evolves according to a differential equation that, in general, could have both a diffusion and drift term. A function generator can multiply these diffusion and drift terms by arbitrary time-dependent functions ($h_j(t)$ and $k_j(t)$ respectively). The problem geometry, associated with a given dataset, can be uploaded onto the device by selectively connecting the various DOFs, which mathematically couples the differential equations of the various DOFs. After some encoding, a datapoint from the dataset of interest can be uploaded to the device by initializing the values of the continuous state variables to be the corresponding feature values of the datapoint. Similarly, data can be downloaded (and decoded) from the device by measuring the values of the continuous state variables after some time evolution.

In addition, the reverse process uses a trained score network. The inputs to the score network are the values of the continuous state variables at some time t, and the output is the value of the score. The jth component of the score, $s_j(t)$, gets added as a drift term in the evolution of the jth DOF. The score network acts as a Maxwell's Demon that continuously monitors the physical system and appropriately adapts the drift term to reduce the physical system's entropy.

8.4. Analog Score Network

We described in Sec. 6 how the Maxwell's Demon can take many physical forms, and the same holds for the score network. This includes the possibility of using a digital score network in conjunction with an analog s-mode system. However, this can lead to latency issues, whereby the communication between the score network and the s-mode system has some time delay. Hence, we highlight here the possibility of using an analog architecture for the score network, which could address the latency issue. Figure 9 shows an analog circuit for the score network. This circuit is based on decomposing the score prediction based on total derivative formula [36].

9. Thermodynamic Speedups

We employ the term thermodynamic speedups for the case when Thermodynamic AI hardware accelerates an algorithm relative to standard, digital hardware. There are

several avenues by which one could obtain a thermodynamic speedup. One source of speedup arises from viewing stochasticity as a resource. Many AI algorithms involve probabilistic variables, yet must be implemented with zero-entropy states when solved on digital computers. In general, noisy physical systems provide access to states with non-zero entropy, so probabilistic aspects are injected naturally (instead of with computational effort) whenever these states are the object of computation. In other words, producing stochasticity on digital hardware is a subroutine that takes computational effort, whereas no such subroutine is necessary for algorithms run on Thermodynamic AI hardware.

A second source of speedup arises from the Maxwell demon. Oftentimes, AI algorithms formulate the mathematics of the Maxwell's demon in the language of physics, involving the gradient of the potential energy function and phase-space dynamics. These subroutines are done by brute force on digital hardware. However, by making the Maxwell's demon physical, the phase space dynamics would naturally occur (as opposed to requiring digital integration). Perhaps more importantly, the gradient of the potential energy is a physical force, as illustrated in Fig. 6. Hence, this raises the possibility of simply measuring this force instead of computing it. The potential $U_\theta(\mathbf{x}, t)$ may have a complex landscape, yet the physics of the system would ensure the gradient of this potential can be accessed instantaneously.

A third source of speedup comes during the training process. Many loss functions are formulated in terms of subroutines that analog hardware can speedup, such as computing vector norms or computing time integrals. For example, this is the case for score-matching loss functions in diffusion models [12]. Accelerating the estimation of the loss function or its gradient during training thus offers another avenue for speedup.

As a final source of speedup, one can delve into the SDE integration subroutine. In Thermodynamic AI hardware, SDE integration occurs naturally in the s-mode device. No matrix-vector multiplications are needed, no discretization of time is needed, and no numerical instabilities occur when time-evolving the s-mode device. This has the following implications: (1) Digitally calculating the terms on the right-hand side of the SDE may be computationally demanding. It involves many matrix multiplications and inversions, in some cases seen throughout the article. By solving the SDEs on physics-based hardware, one naturally avoids this problem. (2) SDEs are in general numerically unstable for large dimensions, requiring sophisticated numerical integrators and fine-tuned time-stepping schedules. With a physics-based hardware system, these difficulties disappear, as there is no time step to be chosen. (3) Unlike digital approaches, the total physical time to solve the various SDEs can be tuned and depends on the operating decay rate of the system, proportional to $1/(RC)$ for a single s-mode based on an RC circuit. This means the physical time of integration can be tuned, and when possible can be made faster, which is an advantage of physics-based computation in general.

10. Conclusions

There are several take-home messages that we would like to share with the reader:

- There is an opportunity for a new computing paradigm where the hardware is stochastic by design.
- AI applications stand to benefit most from this hardware since many such applications are inherently stochastic.
- We identified a class of algorithms called Thermodynamic AI algorithms that not only use stochasticity as a resource but also employ a Maxwell's demon subroutine to guide the random fluctuations in the right direction. This provides a mathematical unification of seemingly unrelated algorithms, such as generative diffusion models, Bayesian deep learning, and Monte Carlo inference.
- Because of this unification, these algorithms can not only be run on a unified software platform, but can they can be run on a unified hardware paradigm. We identified the key ingredients of that hardware paradigm as stochastic-unit (s-unit) system coupled to a Maxwell's demon device.
- Varying degrees of physics can be incorporated into the Maxwell's demon device. We even consider the possibility of a fully physical Maxwell's demon based on forces and phase space dynamics.
- Thermodynamic AI systems have some robustness to unintentional noise in the hardware.

In conclusion, AI algorithms like diffusion models are simultaneously breaking practical barriers and yet also not reaching their full potential due to a mismatch with the underlying hardware. Approximation-free Bayesian deep learning is currently prohibitively slow on digital hardware. Thermodynamic AI hardware appears to be a natural paradigm to unlock major speedups for these AI algorithms.

References

[1] G. Hinton, "The forward-forward algorithm: Some preliminary investigations," 2022.

[2] S. Hooker, "The hardware lottery," *Commun. ACM*, vol. 64, no. 12, pp. 58–65, 2021.

[3] S. K. Kim, L. C. McAfee, P. L. McMahon, and K. Olukotun, "A highly scalable restricted Boltzmann machine FPGA implementation," in *2009 International Conference on Field Programmable Logic and Applications*. IEEE, 2009, pp. 367–372.

[4] Y. Du, L. Du, X. Gu, J. Du, X. S. Wang, B. Hu, M. Jiang, X. Chen, S. S. Iyer, and M.-C. F. Chang, "An analog neural network computing engine using CMOS-compatible charge-trap-transistor (CTT)," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 10, pp. 1811–1819, 2018.

[5] L. G. Wright, T. Onodera, M. M. Stein, T. Wang, D. T. Schachter, Z. Hu, and P. L. McMahon, "Deep physical neural networks trained with backpropagation," *Nature*, vol. 601, no. 7894, pp. 549–555, 2022.

[6] B. E. Boser, E. Sackinger, J. Bromley, Y. Le Cun, and L. D. Jackel, "An analog neural network processor with programmable topology," *IEEE J. Solid-State Circuits*, vol. 26, no. 12, pp. 2017–2025, 1991.

[7] E. Säckinger, B. E. Boser, J. M. Bromley, Y. LeCun, and L. D. Jackel, "Application of the ANNA neural network chip to high-speed character recognition," *IEEE Trans. Neural Netw.*, vol. 3, no. 3, pp. 498–505, 1992.

[8] M. Verleysen and P. G. A. Jespers, "An analog VLSI implementation of hopfield's neural network," *IEEE Micro*, vol. 9, no. 6, pp. 46–55, 1989.

[9] J. L. England, "Statistical physics of self-replication," *J. Chem. Phys.*, vol. 139, no. 12, p. 09B623_1, 2013.

[10] J. Preskill, "Quantum computing and the entanglement frontier," *arXiv preprint arXiv:1203.5813*, 2012.

[11] V. K. Mansinghka *et al.*, "Natively probabilistic computation," Ph.D. dissertation, Citeseer, 2009.

[12] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," *arXiv preprint arXiv:2011.13456*, 2020.

[13] P. Kidger, J. Morrill, J. Foster, and T. Lyons, "Neural controlled differential equations for irregular time series," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6696–6707, 2020.

[14] X. Li, T.-K. L. Wong, R. T. Chen, and D. K. Duvenaud, "Scalable gradients and variational inference for stochastic differential equations," in *Symposium on Advances in Approximate Bayesian Inference*. PMLR, 2020, pp. 1–28.

[15] E. Goan and C. Fookes, "Bayesian neural networks: An introduction and survey," *Case Studies in Applied Bayesian Data Science: CIRM Jean-Morlet Chair, Fall 2018*, pp. 45–87, 2020.

[16] W. Xu, R. T. Chen, X. Li, and D. Duvenaud, "Infinitely deep bayesian neural networks with stochastic differential equations," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022, pp. 721–738.

[17] R. P. Feynman, "Simulating physics with computers," *Int. J. Theor. Phys.*, vol. 21, no. 6, pp. 467–488, Jun 1982. [Online]. Available: https://doi.org/10.1007/BF02650179

[18] T. Conte, E. DeBenedictis, N. Ganesh, T. Hylton, J. P. Strachan, R. S. Williams, A. Alemi, L. Altenberg, G. E. Crooks, J. Crutchfield *et al.*, "Thermodynamic computing," *arXiv preprint arXiv:1911.01968*, 2019.

[19] T. Hylton, "Thermodynamic neural network," *Entropy*, vol. 22, no. 3, p. 256, 2020.

[20] ——, "Thermodynamic state machine network," *Entropy*, vol. 24, no. 6, p. 744, 2022.

[21] N. Ganesh, "A thermodynamic treatment of intelligent systems," in *2017 IEEE International Conference on Rebooting Computing (ICRC)*, 2017, pp. 1–4.

[22] ——, "Rebooting neuromorphic design–a complexity engineering approach," in *2020 International Conference on Rebooting Computing (ICRC)*. IEEE, 2020, pp. 80–89.

[23] ——, "Thermodynamic intelligence, a heretical theory," in *2018 IEEE International Conference on Rebooting Computing (ICRC)*, 2018, pp. 1–10.

[24] A. B. Boyd, J. P. Crutchfield, and M. Gu, "Thermodynamic machine learning through maximum work production," *New J. Phys*, vol. 24, no. 8, 2022. [Online]. Available: https://dx.doi.org/10.1088/1367-2630/ac4309

[25] A. A. Alemi and I. Fischer, "Therml: Thermodynamics of machine learning," *arXiv preprint arXiv:1807.04162*, 2018. [Online]. Available: https://arxiv.org/abs/1807.04162

[26] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *Proceedings of the 32nd International Conference on Machine Learning*, vol. 37, 2015, pp. 2256–2265. [Online]. Available: https://proceedings.mlr.press/v37/sohl-dickstein15.html

[27] S. Goldt and U. Seifert, "Stochastic thermodynamics of learning," *Phys. Rev. Lett.*, vol. 118, p. 010601, 2017. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.118.010601

[28] Y. Bahri, J. Kadmon, J. Pennington, S. S. Schoenholz, J. Sohl-Dickstein, and S. Ganguli, "Statistical mechanics of deep learning," *Annual Review of Condensed Matter Physics*, vol. 11, no. 1, pp. 501–528, 2020. [Online]. Available: https://doi.org/10.1146/annurev-conmatphys-031119-050745

[29] R. Gennaro, "Randomness in cryptography," *IEEE security & privacy*, vol. 4, no. 2, pp. 64–67, 2006.

[30] J. Liu, F. Wilde, A. A. Mele, L. Jiang, and J. Eisert, "Noise can be helpful for variational quantum algorithms," *arXiv preprint arXiv:2210.06723*, 2022.

[31] K. Y. Camsari, B. M. Sutton, and S. Datta, "P-bits for probabilistic spin logic," *Appl. Phys. Rev.*, vol. 6, no. 1, 2019.

[32] J. Kaiser, S. Datta, and B. Behin-Aein, "Life is probabilistic-why should all our computers be deterministic? Computing with p-bits: Ising solvers and beyond," in *2022 International Electron Devices Meeting (IEDM)*. IEEE, 2022, pp. 21–4.

[33] N. A. Aadit, A. Grimaldi, M. Carpentieri, L. Theogarajan, J. M. Martinis, G. Finocchio, and K. Y. Camsari, "Massively parallel probabilistic computing with sparse Ising machines," *Nat. Electron.*, vol. 5, no. 7, pp. 460–468, 2022.

[34] F.-A. Croitoru, V. Hondru, R. T. Ionescu, and M. Shah, "Diffusion models in vision: A survey," *arXiv preprint arXiv:2209.04747*, 2022.

[35] J. Nelson, M. Vuffray, A. Y. Lokhov, T. Albash, and C. Coffrin, "High-quality thermal gibbs sampling with quantum annealing hardware," *Phys. Rev. Applied*, vol. 17, no. 4, p. 044046, 2022.

[36] P. J. Coles, C. Szczepanski, D. Melanson, K. Donatella, A. J. Martinez, and F. Sbahi, "Thermodynamic AI and the fluctuation frontier," *arXiv preprint arXiv:2302.06584*, 2023.

[37] N. Goodman, V. Mansinghka, D. M. Roy, K. Bonawitz, and J. B. Tenenbaum, "Church: a language for generative models," *arXiv preprint arXiv:1206.3255*, 2012.

[38] V. K. Mansinghka and E. M. Jonas, "Combinational stochastic logic," Jan. 8 2013, uS Patent 8,352,384.

[39] P. Horowitz and W. Hill, *The art of electronics; 3rd ed.* Cambridge University Press, 2015.

[40] G. Lecoy, R. Alabedra, and B. Barban, "Noise studies in internal field emission diodes," *Solid-State Electronics*, vol. 15, no. 12, pp. 1273–1276, 1972. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0038110172901190

[41] R. van Zon, S. Ciliberto, and E. G. D. Cohen, "Power and heat fluctuation theorems for electric circuits," *Phys. Rev. Lett.*, vol. 92, p. 130601, Mar 2004. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.92.130601

[42] G. Vasziová, J. Tóthová, L. Glod, and V. Lisỳ, "Thermal fluctuations in electric circuits and the Brownian motion," *J. Electr. Eng.*, vol. 61, no. 4, pp. 252–256, 2010.

[43] M. D. Vidrighin, O. Dahlsten, M. Barbieri, M. Kim, V. Vedral, and I. A. Walmsley, "Photonic Maxwell's demon," *Phys. Rev. Lett*, vol. 116, no. 5, p. 050401, 2016.

[44] L. B. Kish and C.-G. Granqvist, "Electrical Maxwell demon and Szilard engine utilizing Johnson noise, measurement, logic and control," *PLOS One*, 2012.

[45] P. Izmailov, S. Vikram, M. D. Hoffman, and A. G. G. Wilson, "What are Bayesian neural network posteriors really like?" in *International conference on machine learning*. PMLR, 2021, pp. 4629–4640.

[46] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020.