entropy

Article

Thermodynamic State Machine Network

Todd Hylton

Department of Mechanical and Aerospace Engineering, University of California, San Diego, CA 92093, USA; thylton@ucsd.edu

Abstract: We describe a model system—a thermodynamic state machine network—comprising a network of probabilistic, stateful automata that equilibrate according to Boltzmann statistics, exchange codes over unweighted bi-directional edges, update a state transition memory to learn transitions between network ground states, and minimize an action associated with fluctuation trajectories. The model is grounded in four postulates concerning self-organizing, open thermodynamic systems—transport-driven self-organization, scale-integration, input-functionalization, and active equilibration. After sufficient exposure to periodically changing inputs, a diffusive-to-mechanistic phase transition emerges in the network dynamics. The evolved networks show spatial and temporal structures that look much like spiking neural networks, although no such structures were incorporated into the model. Our main contribution is the articulation of the postulates, the development of a thermodynamically motivated methodology addressing them, and the resulting phase transition. As with other machine learning methods, the model is limited by its scalability, generality, and temporality. We use limitations to motivate the development of thermodynamic computers—engineered, thermodynamically self-organizing systems—and comment on efforts to realize them in the context of this work. We offer a different philosophical perspective, thermodynamicalism, addressing the limitations of the model and machine learning in general.

Keywords: thermodynamic computing; thermodynamicalism; machine learning; scale integration; input functionalization; active equilibration

Citation: Hylton, T. Thermodynamic State Machine Network. *Entropy* **2022**, *24*, 744. https://doi.org/10.3390/e24060744

Academic Editor: Neal G. Anderson

Received: 28 March 2022
Accepted: 14 May 2022
Published: 24 May 2022

1. Introduction

The remarkable progress in machine learning systems over the past decade has led to an equally remarkable increase in the computational resources required to train them. One recent report indicates that the number of computing resources used to train state-of-the-art, large scale models has doubled every 3.4 months since 2012, as compared to a more traditional 2-year doubling rate prior to 2012 [1]. Such trends suggest the consideration of fundamentally new approaches to machine learning. We have previously outlined the vision for such an approach, *thermodynamic computing* [2–4], which is motivated by the hypothesis that the ubiquitous self-organization of the natural world is fundamentally thermodynamic [5,6], and that this capacity can be incorporated into computing hardware. In this work, we further articulate this hypothesis as four postulates concerning self-organizing, open thermodynamic systems, and build a model system—the *thermodynamic state machine network* (TSMN)—reflecting these postulates.

The TSMN is a machine learning model comprising a network of probabilistic, stateful automata that equilibrate according to Boltzmann statistics, exchange codes over unweighted bi-directional edges, update a state transition memory to learn transitions between network ground states, and minimize an action associated with fluctuation trajectories. The dynamics of the TSMN are governed by particle-like excitations whose motion is easily visualized as videos. Unlike neural network-based machine learning models, which learn long-term memory in weighted edges connecting nodes, the TSMN learns long-term memory in state transition memories at each node. After sufficient exposure to periodically changing inputs, a diffusive-to-mechanistic phase transition emerges in the network dynamics, and the evolved networks show spatial and temporal structures that look like a spiking neural network, although no such structures were incorporated into the model. Before the phase transition, the dynamics of the TSMN are diffusive, difficult to compute, and difficult to describe using an algorithm; after the transition, the dynamics are mechanistic, easy to compute, and easy to describe using an algorithm. Like other machine learning models, the TSMN is limited in its ability to generalize, scale, and represent temporal processes. We use these limitations as motivation for the development of thermodynamic computers. We also discuss several published experimental efforts that address these motivations and compare them to the postulates used to develop the TSMN.

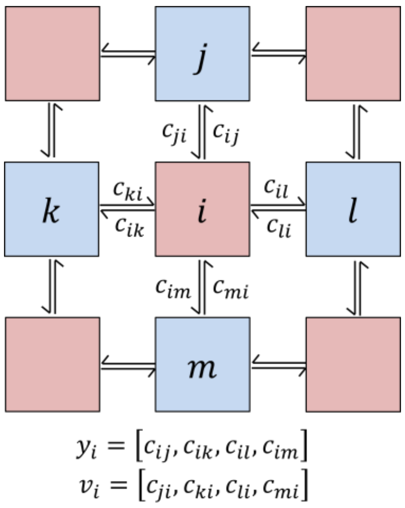
The TSMN is inspired by theories of open thermodynamic systems. The non-equilibrium fluctuation theorems of Jarzynski [7], and Crooks [8] and Still's "thermodynamics of prediction" [9], are the conceptual framework for the equilibration (Section 2.3) and adaptation (Section 2.5) methodologies of the TSMN. Ideas on "active inference" [10] influenced the inferencing methodology (Section 2.6), ideas on "dissipative adaptation" [11] influenced the adaptation methodology (Section 2.5), and ideas on "causal entropic forces" [12] influenced the state update methodology (Section 2.4). The TSMN is also inspired by other thermodynamically grounded model systems, such as Ising models [13], Hopfield networks [14], and Boltzmann machines [15], but is distinguished from them by the details of its composition (e.g., code-transport, divergence-less node states, unweighted edges, and state transition memory) and learning methodology. Like Ising models and related neural network models, the TSMN is a distributed collection of autonomous nodes that collectively self-organize through local interactions, but the TSMN is different from these models in that it has a large degeneracy of stable ground states separated by particle-like excitations. The ground state degeneracy means that there is more than one "solution" to any "problem" posed by external inputs, and that the network is generally not frustrated. Other influences on this work include "physical reservoir computers" [16] and related model systems such as "liquid state machines" [17] and "echo state networks" [18]. The TSMN shares with these approaches the goal of learning and predicting time-varying inputs, but the TSMN integrates this ability in a single, recurrent, adaptive architecture. A machine-learning method called "equilibrium propagation" [19] employs similar ideas as the TSMN but uses different methodologies and addresses neural network models. The TSMN also builds upon ideas and methods from our earlier work on a "thermodynamic neural network" model [20].

Our contributions in this article include (1) an articulation of postulates concerning the self-organization of complex, open, thermodynamic systems, (2) a new network model and machine learning methodology for state machines addressing these postulates, (3) a demonstration of a phase change in the dynamics of the TSMN when exposed to periodic inputs, and (4) a consideration of classical and thermodynamic computing in the context of the TSMN and its limitations. We hope as well that this work serves as a pedagogical example of the emergence of computation from a thermodynamic, evolutionary process.

The paper is organized as follows. In Methods (Section 2) we summarize the model approach, describe the Postulates (Section 2.1) that motivated its development, detail the model equilibration and adaptation methodologies (Sections 2.2–2.6), and describe the Visualization (Section 2.7) of the model dynamics. In Results (Section 3) we show and analyze the dynamics of networks without (Section 3.1) and with (Section 3.2) state transition memory. In Discussion (Section 4) we discuss the limitations of the model (Section 4.1) and use these limitations to motivate Thermodynamic Computing (Section 4.2) and to compare related experimental efforts, offer an interpretation of Classical Computing (Section 4.3), and speculate on a different philosophical perspective called Thermodynamicalism (Section 4.4). As an appendix, we offer a conceptual circuit model for the TSMN (Appendix A: TSMN Circuit Model).

2. Methods

The TSMN is a bi-directional, unweighted network of probabilistic, stateful automata. Figure 1 illustrates the basic network structure. We call c_{ij} the *edge code* sent from node i to node j. The collection of edge codes from node i define its output y_i, while the collection of edge codes to node i define its input v_i. c_{ij} "interacts" with edge code c_{ji} sent from node j to node i along the same $i \leftrightarrow j$ edge, such that edges with complementary codes have low energy and edges with like codes have high energy. The TSMN can be thought of a kind of "Ising model" in which the interaction of the nodes is motivated by the transportation of codes (or "charges") through them. To effectively "move" these codes, the node must configure its output to avoid a "build up" of like codes on its edges. Although more complex networks are possible, here we consider only 2-dimensional, 4-nearest-neighbor networks and binary edge codes $c_{ij} \in \{0, 1\}$ because the dynamics are easy to visualize.



j
c_{ji}
c_{ij}
k
c_{ki}
c_{ik}
i
c_{il}
c_{li}
l
c_{im}
c_{mi}
m
$y_i = [c_{ij}, c_{ik}, c_{il}, c_{im}]$
$v_i = [c_{ji}, c_{ki}, c_{li}, c_{mi}]$

Figure 1. *Network structure, edge codes, node outputs and inputs*. Nodes i and j exchange codes c_{ij} and c_{ji}. The output y_i/input v_i of node i is the vector of its output/input edge codes. In sampling the network evolution, the network is bi-partitioned in a checker-board pattern (salmon and blue nodes). The inputs of one partition are the outputs of the other partition.

This section is organized as follows. Postulates (Section 2.1) explains the assumptions behind the methods described in this section. A Node Interaction Model (Section 2.2) includes internal nodes, which conserve and transport edge codes, and external nodes, which represent inputs from environmental potentials and act as sources and sinks of edge codes to which the network must respond. An edge code interaction energy is defined such that network configurations that more effectively transport edge codes have lower energy. In simulation, the network is repeatedly sampled using a Large-Spatial-Scale, Short-Timescale Equilibration Algorithm (Section 2.3). The node interaction model is then augmented with a State Transition Memory (Section 2.4), which is accessed using a state-indexing method, and incrementally updated with every network ground state transition using a Small-Spatial-Scale, Long-Timescale Adaptation Algorithm (Section 2.5) that minimizes the free energy of the network. A method for Inference of Dormant Inputs/Generation of Outputs (Section 2.6) is also described. Methods for visualizing the dynamics implied by Sections 2.2–2.6 are described in Visualization (Section 2.7).

2.1. Postulates

We have previously integrated the thermodynamic concepts of conservation, potentiation, fluctuation, dissipation, and equilibration to create a thermodynamic neural network model [20]. In this section we articulate the result of this integration as a set of postulates concerning self-organizing, complex, open thermodynamic systems (see Figure 2). We will use these postulates to motivate the detailed methodologies of the following sections.

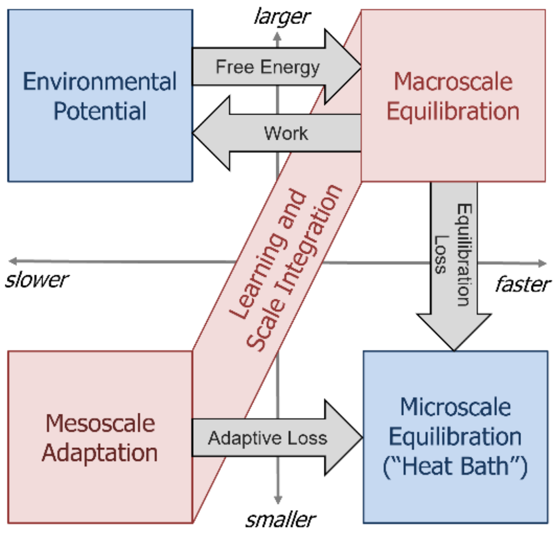


Figure 2. *Illustration of postulates.* An open thermodynamic system (red) equilibrates with an environment (blue) that provides a source of potential (**upper left**) and a reservoir of small-fast excitations, or "heat bath" (**lower right**). The open system comprises the ability to equilibrate rapidly at a large scale (**upper right**) and to adapt slowly at a small scale (**lower left**), integrated by a learning process. To maintain homeostasis, conserved quantities (e.g., energy) taken from the environment by the open system must be returned to it, which drives the learning process and integrates scales. If the open system has functionalized input potentials from the environment, then it may also interact with the environment by doing work on it to "retrieve" missing input potentials. The ability to do work on the environment implies that the open system stores energy, and that its equilibration is active.

Scale-integration—The integrity of an open system depends on its ability to rapidly change its internal, small-scale organization to address the external, large-scale changes of the potentials in its environment. We call this alignment of large- and small-scale features *scale-integration*, and the process of creating it is referred to as *learning*. In natural systems (e.g., organisms), we suppose that scale-integration is a natural consequence of multi-scale equilibration between an open system and its environment. In the TSMN, however, scale integration is created using a two-piece equilibration algorithm—one piece providing large-spatial-scale, short-timescale equilibration with an environment (Section 2.3), and another piece providing small-spatial-scale, long-time scale internal parameter adaptation (Section 2.5) that facilitates future equilibration. A familiar example of such two-piece methodology is a forward-pass through an artificial neural network providing the large-spatial-scale, short-timescale equilibration (measured as output "errors") with a training data set, and a backward-pass providing the small-spatial-scale, large-timescale internal adaptation (typically as "weight updates") to reduce future output/equilibration errors.

Transport-Driven Self-Organization—Any open system in homeostasis with its environment must return to the environment any conserved quantity that is taken from it. Imbalances in the distribution of conserved quantities in the environment create thermodynamic potentials (free energy) that can drive their transport through the open system, thereby relieving those potentials and creating entropy in the environment. Sustained entropy creation favors the evolution of organization within the open system that facilitates the transport of these conserved quantities, while also stabilizing the associated organization.

Input Functionalization—If an open thermodynamic system evolves stable internal organizations that efficiently transport conserved quantities in response to external input potentials, then we say that the system has "functionalized" its inputs. Input functionalization enables the open system to "infer" or "predict" the existence of potentials that are missing in the current context but were present during the learning process. For example, a naïve artificial neural network is first trained on stimuli-label pairs (the input potentials before functionalization) to develop its internal organization, such that the trained network can subsequently infer labels (the missing input potentials after functionalization) from stimuli (the input potentials available after functionalization). In some cases, the missing, inferred potentials are realized as "outputs" that prompt the environment to provide additional inputs. A recursive exchange of such inputs and outputs is frequently called an "interaction". For example, an artificial neural network may be used to provide product recommendations in response to a query by an online shopper. The recommendation may prompt the shopper to provide additional input that may be used to provide a new recommendation, and to improve training of the recommending system.

Active Equilibration—The capacity of an open system to interact with an environment after it has functionalized its inputs requires that it store energy internally, i.e., that the outputs supplied to its environment are themselves an aggregation of conserved quantities constituting a thermodynamic potential to which the environment must respond. Similarly, the ability of an open system to proactively change its internal organization in response to changes in the potentials in its environment requires that it store energy. We call this ability to store and use energy to interact with the environment to maintain homeostasis and to proactively change internal organization *active equilibration*.

2.2. Node Interaction Model

To implement the idea of transport driven self-organization, internal nodes are restricted to have "non-divergent" outputs: i.e., y_i are restricted to having equivalent numbers of 0 s and 1 s. Of the $2^4 = 16$ possible outputs for a node with binary edge codes and connections to four nearest neighbors, only 6 outputs satisfy this constraint. In particular, any internal node output y_i must be drawn from the set γ_i, where

$$\gamma_i = \{[1,1,0,0],[1,0,1,0],[1,0,0,1],[0,0,1,1],[0,1,0,1],[0,1,1,0]\},\ \ y_i \in \gamma_i. \tag{1}$$

We note that the assumption of unweighted, unsigned edges, and the requirement of non-divergent node outputs in the TSMN, excludes the use of binary node states commonly found in Ising and neural network models.

External nodes source and sink codes to the network and are defined as having maximal divergence. The corresponding output set is

$$\gamma_i^{ext} = \{[0,0,0,0],[1,1,1,1]\},\ \ y_i^{ext} \in \gamma_i^{ext}. \tag{2}$$

In the examples that follow, we constrain the use of external nodes such that their aggregate output is non-divergent. Hence, for every external node with output $[0,0,0,0]$, there is a complementary external node with output $[1,1,1,1]$.

We define an edge interaction energy matrix K_{ij} as

$$K_{ij} = \left(\delta_{c_{ij},c_{ji}} - 1\right) = -\delta_{c_{ij},\bar{c}_{ji}}, \tag{3}$$

where δ is the Kronecker delta function and $\bar{c}_{ji}$ is the complementary code to c_{ji}. By this definition, the edge interaction energy is low ($K_{ij} = -1$) when the edge codes are different, and high ($K_{ij} = 0$) when the edge codes are identical. We call edges with identical codes *edge-excitations*. The energy H_i of node i is the sum over the energy of its edges.

$$H_i = \sum_{j \in i_{nn}} K_{ij}, \tag{4}$$

where i_{nn} is the set of indices to the nearest neighbors of node i. A node achieves its lowest energy configuration when its output edge codes complement its input edge codes on every edge, $\bar{c}_{i_{nn},i} = c_{i,i_{nn}}$, a condition that can be met only if its inputs also have no divergence (i.e., $v_i \in \gamma_i$). The energy ε of the network is the sum over the internal node energies,

$$\varepsilon = \sum_{i=1}^{n} H_i, \tag{5}$$

where n is the number of internal nodes. A ground state of the network is obtained when the codes are complemented on every edge, a non-trivial requirement given the constraints of γ_i. Given the non-divergence of the internal nodes and maximal divergence of the external nodes, low-energy network states can be thought of as "transporting" codes between complementary external nodes. We note that the network ground states are highly degenerate, which we will show to be an important feature in the network's ability to respond to inputs from external nodes and to learn a state transition memory. From the perspective of automata theory, an internal node "accepts" inputs from its connected nodes when those inputs allow a node ground state, and the network as a whole "accepts" inputs from its external nodes when those inputs allow a network ground state. In Section 2.5 we shall use these acceptance criteria as the keys to drive learning within the network, and in Section 4.3 to discuss connections to classical computing.

2.3. Large-Spatial-Scale, Short-Timescale, Equilibration Algorithm

The probability $P(\boldsymbol{y})$ of a particular network output state $\boldsymbol{y} = (y_1, y_2 \ldots y_n)$ is assumed to be governed by a Boltzmann distribution at inverse temperature β and with partition function Z as

$$\begin{gathered} P(\boldsymbol{y}) = \tfrac{1}{Z}\cdot exp(-\beta\varepsilon(\boldsymbol{y})) = \tfrac{1}{Z}\cdot\prod_i exp(-\beta H_i(y_i)) \\ Z = \sum_{\boldsymbol{y}\in\boldsymbol{\gamma}} exp(-\beta\varepsilon(\boldsymbol{y})) = \sum_{\boldsymbol{y}\in\boldsymbol{\gamma}}\prod_i exp(-\beta H_i(y_i)) = \prod_i \sum_{y_i\in\gamma_i} exp(-\beta H_i(y_i)), \end{gathered} \tag{6}$$

where $\boldsymbol{\gamma} = (\gamma_1, \gamma_2 \ldots \gamma_n)$ is the set of all possible node outputs. We employ a Markov Chain Monte Carlo (MCMC)/Gibbs Sampling [21] algorithm that repeatedly samples Equation (6) to generate the large-spatial-scale, short-timescale dynamics of the TSMN. The algorithm partitions the nodes into two groups in a checkerboard pattern according to their placement in the 2D network, such that the inputs to the nodes in one partition are the outputs from the nodes in the opposite partition (Figure 1). We sample the two network partitions alternately by holding the node outputs in one partition fixed, while sampling the node outputs of the other partition. In this way, each node output can be sampled independently using Equation (7).

$$\begin{gathered} P(y_i|v_i) = \tfrac{1}{Z_i} exp(-\beta H_i(y_i|v_i)) \\ Z_i = \sum_{y_i\in\gamma_i} exp(-\beta H_i(y_i|v_i)) \end{gathered} \tag{7}$$

We refer to the evolution of the network as we repeatedly sample the partitions as its "equilibration dynamics". In the videos of the equilibration dynamics in Section 3, each frame in the video represents the update of the nodes in one partition.

2.4. State Transition Memory

We augment the node interaction energy H_i of each internal node with a state transition energy T_i and a weighting parameter σ, such that the total energy E_i of node i becomes

$$E_i(y_i|v_i, s_i) = H_i(y_i|v_i) - \sigma\cdot T_i(y_i|s_i). \tag{8}$$

where s_i is an index specifying the node state, which we define as a temporal sequence or "stack" of the last d outputs of node i at *a network ground state*. If we label the node output associated with the most recent network ground state as $\widetilde{y}_i^1$, the next most recent as $\widetilde{y}_i^2$, etc., then

$$s_i = \left[\widetilde{y}_i^1, \widetilde{y}_i^2 \ldots \widetilde{y}_i^d\right]. \tag{9}$$

We refer to d as the "depth" of the state-index. Every time the network reaches a new ground state, each node updates its state-index by pushing its current output onto the top of the stack and popping the oldest value off the bottom the stack. Hence, state-index updates at each node are synchronized across the network because they only occur at network (not node) ground states. Network dynamics between network ground states can be thought of as *fluctuations*, which explore the network output space but do not affect the state of the network. The node output distribution of Equation (7) used by the MCMC algorithm is modified as

$$\begin{aligned} P(y_i|v_i,s_i) &= \tfrac{1}{Z_i} exp(-\beta E_i(y_i|v_i,s_i)) \\ Z_i &= \sum_{y_i \in \gamma_i} exp(-\beta E_i(y_i|v_i,s_i)), \end{aligned} \tag{10}$$

We note that E_i comprises a component H_i, which is fundamentally spatial, and a component T_i, which is fundamentally temporal. The addition of T_i also confers upon the TSMN a "state-energy landscape": the idea that a change in state modifies the energy landscape and that a change in the energy landscape modifies the state in an iterative process that breaks time-reversal symmetry.

2.5. Small-Spatial-Scale, Long-Timescale, Adaptation Algorithm

To construct the small-spatial-scale, long-timescale adaptation algorithm, we extend the single-step (i.e., short-timescale) network energy ε of Equation (5) into multi-step (i.e., long-timescale) network action A as,

$$A\left(\boldsymbol{y}^0 \to \boldsymbol{y}^m\right) = \sum_{\boldsymbol{y}=\boldsymbol{y}^0}^{\boldsymbol{y}^m} \varepsilon(\boldsymbol{y}), \tag{11}$$

which is the sum of the network energies over a sequence of network updates where $\boldsymbol{y}^0 \to \boldsymbol{y}^m \equiv \left[\boldsymbol{y}^0, \boldsymbol{y}^1 \ldots \boldsymbol{y}^m\right]$ indicates a particular m-step *trajectory* between network outputs $\boldsymbol{y}^0$ and $\boldsymbol{y}^m$. We weight the probability of these trajectories using a Boltzmann distribution as

$$\begin{aligned} P\left(\boldsymbol{y}^0 \to \boldsymbol{y}^m\right) &= \tfrac{1}{Z} \cdot exp\left(-\beta A\left(\boldsymbol{y}^0 \to \boldsymbol{y}^m\right)\right) \\ Z &= \sum_{\boldsymbol{u} \in \{\boldsymbol{y}^0 : \boldsymbol{y}^m\}} exp(-\beta A(\boldsymbol{u})), \end{aligned} \tag{12}$$

where $\left\{\boldsymbol{y}^0 : \boldsymbol{y}^m\right\}$ indicates the set of trajectories of any length beginning with $\boldsymbol{y}^0$ and terminating with $\boldsymbol{y}^m$. Given any two endpoints and comparable ε at each step in the trajectory, shorter trajectories will have smaller action and higher probability than longer trajectories. The natural endpoints to consider for these trajectories are the network ground states, as these are the points where the network drives the evolution to converge and where automata theory indicates that the network has "accepted" its inputs (Section 2.2). As is elucidated in Section 2.1, the small-spatial-scale, long-timescale adaptation of an open system should facilitate its large-spatial-scale, short-timescale equilibration with its environment. In the TSMN model we realize this by updating the state transition memory T_i to facilitate transitions between network ground states that favor smaller action trajectories.

With these ideas in mind, at each network ground state and *before* the update of the node state-indices, the memory elements T_i are updated as

$$T_i(\gamma_i|s_i) \leftarrow T_i(\gamma_i|s_i) \cdot (1 - \alpha \cdot P(\gamma_i|\widetilde{v}_i, s_i)) \tag{13}$$

$$T_i(\gamma_i|s_i) \leftarrow T_i(\gamma_i|s_i) + \alpha \cdot P(\gamma_i|\widetilde{v}_i, s_i), \tag{14}$$

where $0 < \alpha < 1$ is a learning rate parameter, $\widetilde{v}_i$ is the node input at network ground state (i.e., the complement of $\widetilde{y}_i$), and γ_i indicates that the update is over the set of all possible node outputs. Note that the updates are at a small-spatial-scale because they are different for each node i. The physical motivation for Equation (14) derives from the observation that the negative gradient of the node free energy $F_i = -\log Z_i$ with respect to $T_i(y_i|s_i)$ is (see Equation (10))

$$\frac{-\partial F_i}{\partial T_i(y_i|s_i)} = \frac{\partial \log(Z_i)}{\partial T_i(y_i|s_i)} \sim P(y_i|v_i, s_i). \tag{15}$$

The update rule for the state transition memory T_i is therefore a regularized (Equation (13)) gradient descent (Equation (14)) of the free energy of the node over the state transition memory elements conditioned on the current state and input, implemented whenever a network ground state is obtained. Regularization bounds T_i such that for repeated application of Equations (13) and (14) for the same $\widetilde{v}_i$ and s_i, which we might expect if the network experiences periodic inputs, $T_i(\widetilde{y}_i|s_i) \to 1$ and $T_i(y_i \neq \widetilde{y}_i|s_i) \to 0$, where $\widetilde{y}_i$ is the output of node i at the *next* network ground state given the current state s_i. Additionally, to promote short trajectory dynamics, whenever a node receives an edge excitation, the state transition memory of its last output is decayed as,

$$T_i(y_i|s_i) \leftarrow T_i(y_i|s_i)\cdot(1 - \alpha\cdot P(y_i|v_i, s_i)), \tag{16}$$

which prevents edge excitations from becoming "stuck" and the long trajectories associated with these local minima. Equation (16) can be thought of as a "metabolic" cost that decays internal organization linked to less-efficient behaviors.

Referring again to the postulates of Section 2.1, Equations (13), (14), and (16) are the large-timescale, small-spatial-scale adaptation algorithm for the TSMN, and Equation (10) and the MCMC algorithm are the short-timescale, large-spatial-scale equilibration algorithm. Scale-integration results from the application of Equations (13) and (14) whenever the MCMC algorithm finds a network ground state. Additionally, because the state transition memory updates happen before the state-index updates at each network ground state, the energy of the network generally increases immediately after the state-index update, reflecting the idea of active equilibration. The effect of this active equilibration is to bias the network away from its current ground state and toward its next ground state.

2.6. Inference of Dormant Inputs/Generation of Outputs

Internal nodes infer missing or dormant inputs from connected external nodes using a slight modification of the above methodology. We divide the input vector v_i into two components, $v_i \to \left(v_i^a, v_i^d\right)$, where v_i^a and v_i^d refer to the active and dormant components of the input to node i. We modify Equation (10) as

$$\begin{aligned} P\left(y_i, v_i^d \middle| v_i^a, s_i\right) &= \tfrac{1}{Z_i} exp\left(-\beta E_i\left(y_i, v_i^d \middle| v_i^a, s_i\right)\right) \\ Z_i &= \sum_{y_i, v_i^d} exp\left(-\beta E_i\left(y_i, v_i^d \middle| v_i^a, s_i\right)\right), \end{aligned} \tag{17}$$

and infer both y_i and v_i^d by sampling Equation (17) over $y_i \in \gamma_i$ and $v_i^d \in \{0, 1\}$ for each component of v_i^d. By selection of its output y_i, node i creates an output to the dormant external nodes which can change the state of those external nodes and the larger environment of which they are a part. After a network has learned to functionalize inputs (see Section 2.1), these outputs can predict dormant inputs and be seen as functions of other, active inputs (see Section 3.2).

2.7. Visualization

Figure 3 illustrates three different ways to visualize the network, which we will employ in the videos of Section 3. Using a 3×3 array of nodes, the left-hand image shows the edge codes associated with a particular configuration of node outputs, the central image colors each node according to its output (red, green, blue, cyan, magenta, and yellow), and the right image illustrates edge-excitations among the nodes. Edge-excitations—0-0 or 1-1 code pairs—are illustrated by coloring the nodes attached to the edge in light magenta or light green, respectively. As will become apparent in the videos that follow, the equilibration dynamics of the network are most easily understood as the diffusion, annihilation, and creation of edge-excitations. Figure 4 illustrates this annihilation and creation processes by showing the underlying changes in the edge codes for an example excitation pair.

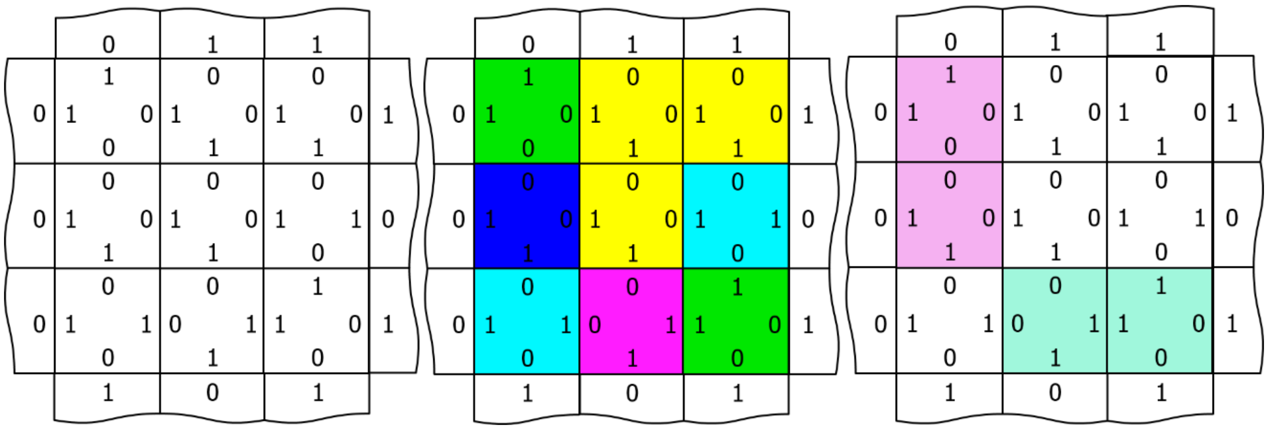


Figure 3. Three different network visualizations. On the left, an array of nine internal nodes with edge codes illustrated as 1 s and 0 s. In the center, each of the 6 node outputs is associated with a different color: red (not shown), green, blue, cyan, magenta, and yellow. On the right, edge-excitations are visualized by coloring the nodes associated with the edge; 0-0 edges are colored light magenta, and 1-1 edges are colored light green.

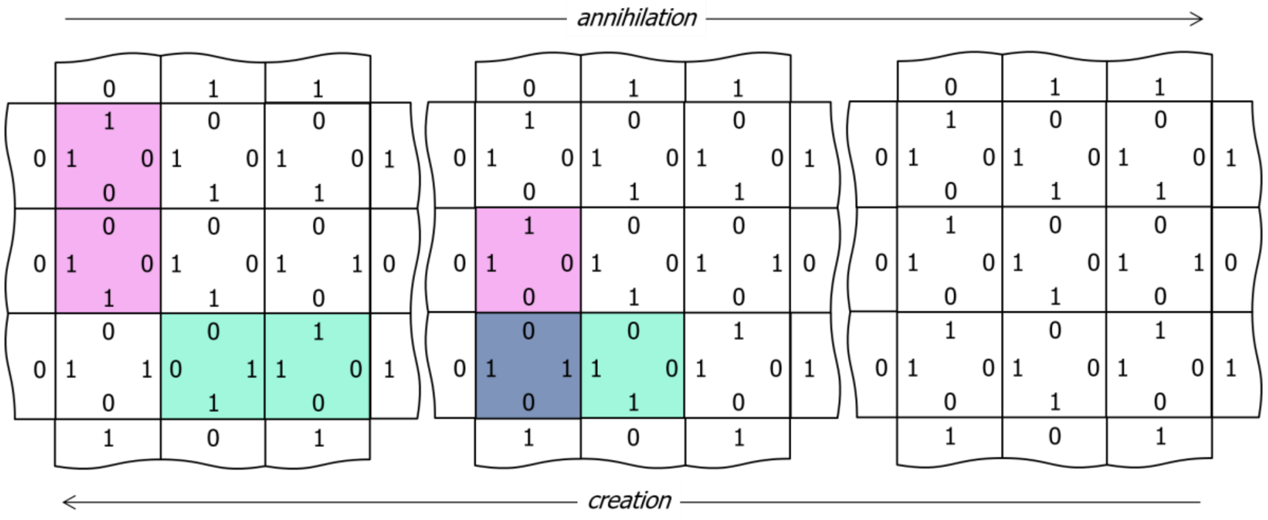


Figure 4. Pair annihilation/creation dynamics. From left to right, complementary excited edges collide and annihilate via a sequence of node output changes. From right to left, a thermal excitation of the lower left node output drives the creation of a pair of excited edges, which diffuse away from the site of their creation via the opposite sequence of node output changes.

As illustrated in Figure 5, the edge excitation dynamics of naïve networks (i.e., networks without state transition memory) are diffusive because the energies associated with node outputs are degenerate: a node with one edge excitation has three equally probable, lowest-energy outputs. Repeated sampling of the network with the MCMC algorithm randomly transfers edge excitations among adjacent nodes. By comparison, nodes receiving non-divergent inputs have a single, lowest-energy output, which makes network ground states stable at low temperatures. To make these same points differently, edge excitations represent an energy barrier separating network ground states. Updates to T_i (Equations (13) and (14)) lift the energy degeneracy in Figure 5 and drive the network from diffusive to mechanistic dynamics (see Section 3.2). Decay of T_i (Equation (16)) tends to restore the degeneracy.

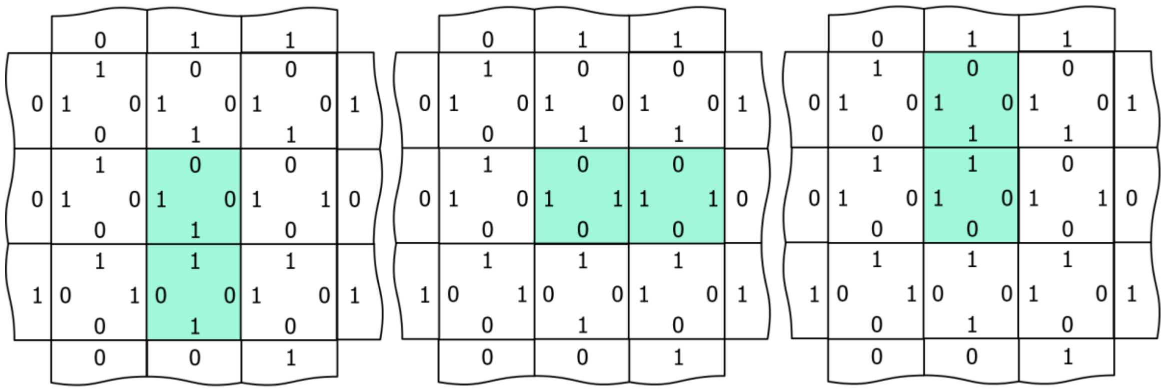


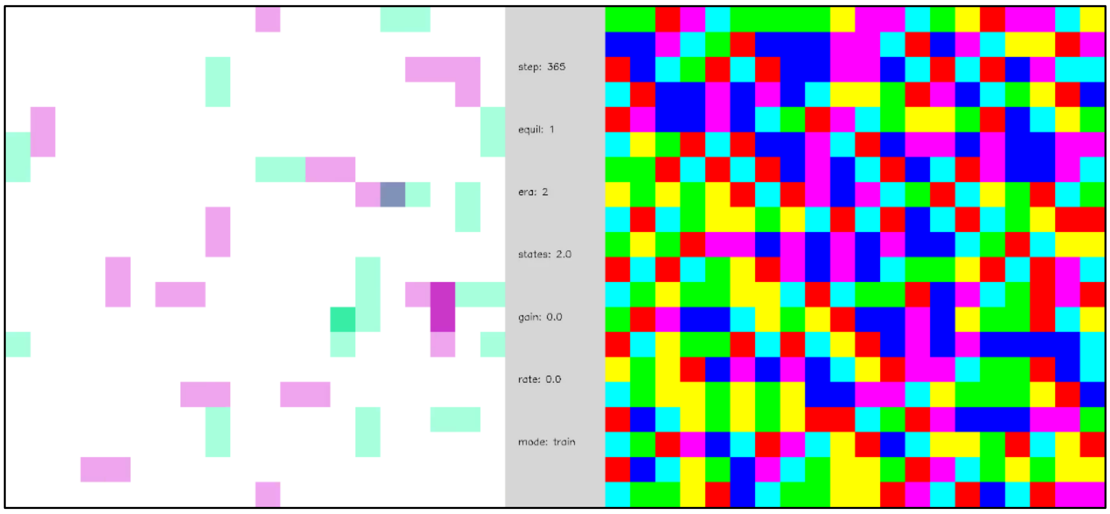
Figure 5. Excited node output degeneracy and edge excitation diffusion in naïve networks. The central node receives divergent inputs (three 1 s and one 0) and must select among three lowest-energy outputs of the same energy illustrated in three separate images. Supposing that the image on the left is the initial configuration, a sampling of the central node, resulting in the configuration of either the central or the right image, means that the edge excitation that was previously associated with the lower central node has "hopped" to another node. In this way, edge excitations diffuse randomly through the network with repeated MCMC sampling of the checkerboard partitions (Figure 1). State transition memory learning lifts this energy degeneracy (Section 2.5), making the dynamics more mechanistic.

3. Results

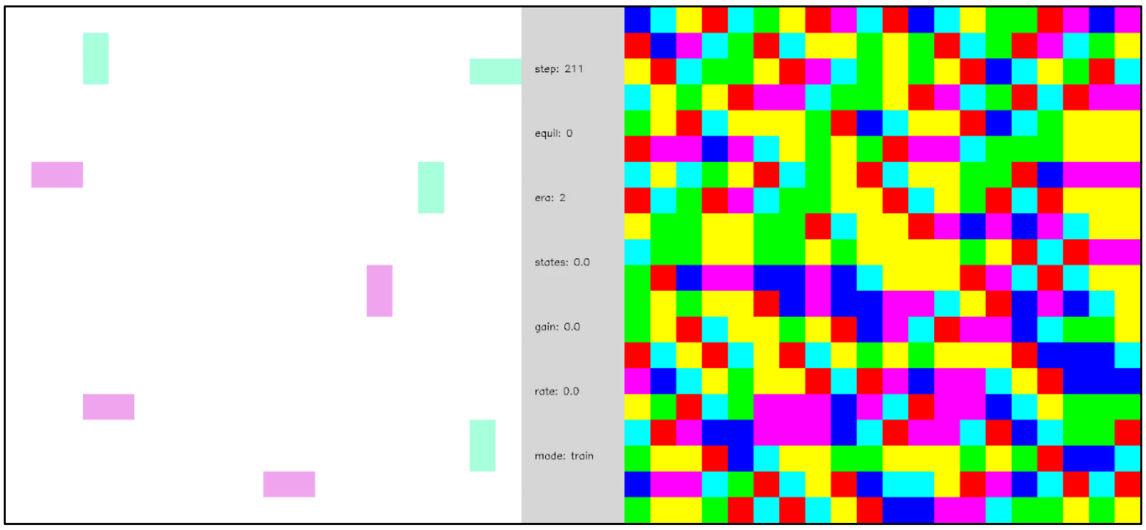
We employ the methods of Section 2 to simulate the equilibration dynamics of the TSMN, which we illustrate as a series of videos. We begin with simulations of naïve networks to illustrate diffusive dynamics and the effect of temperature (Section 3.1). These dynamics are characterized by low-energy excitations on a highly degenerate ground state with qualitative resemblance to the "quasiparticles" of Fermi liquid theory. We then simulate networks of increasing complexity that learn a state transition memory as they are exposed to external inputs that are changing periodically in time (Section 3.2). In these networks, a phase transition from diffusive to mechanistic dynamics emerges, after which the inputs are functionalized, the network is scale-integrated, and the equilibration dynamics are active. We show that the TSMN resembles a self-constructed "neural network" after this phase transition.

3.1. Networks without Memory

Video 1 presents the equilibration dynamics of a naïve network comprising 400 (20×20) internal nodes and periodic boundary conditions. Node outputs are randomized at the start of the simulation and each time the network finds a ground state. Two visualizations—a node output view and an edge-excitation view (see Section 2.7)—of the equilibration dynamics are shown side-by-side. The edge-excitation view of the network shows 1-1 and 0-0 edge excitations (created by the node output randomization at the beginning of each cycle) diffusing, colliding, and annihilating each other. The node output view reveals a patchwork of evolving domains. Although only briefly displayed (a single frame before the network is randomized), the ground states in Video 1 are stable because the simulation temperature is low. Note that the network naturally evolves to find a ground state, which motivated the choice of ground states as the terminating points in the fluctuations considered in Equation (12) and, thereby, the network adaptations of Equations (13) and (14). Video 2 shows the same network as Video 1 at a higher temperature. As in Video 1, edge excitation pairs spontaneously annihilate as they diffuse and collide within the network, but, unlike in Video 1, they also spontaneously emerge from thermal excitations. Hence, there are no long-lived ground states, and equilibrium is achieved when the annihilation and creation rates are equal.



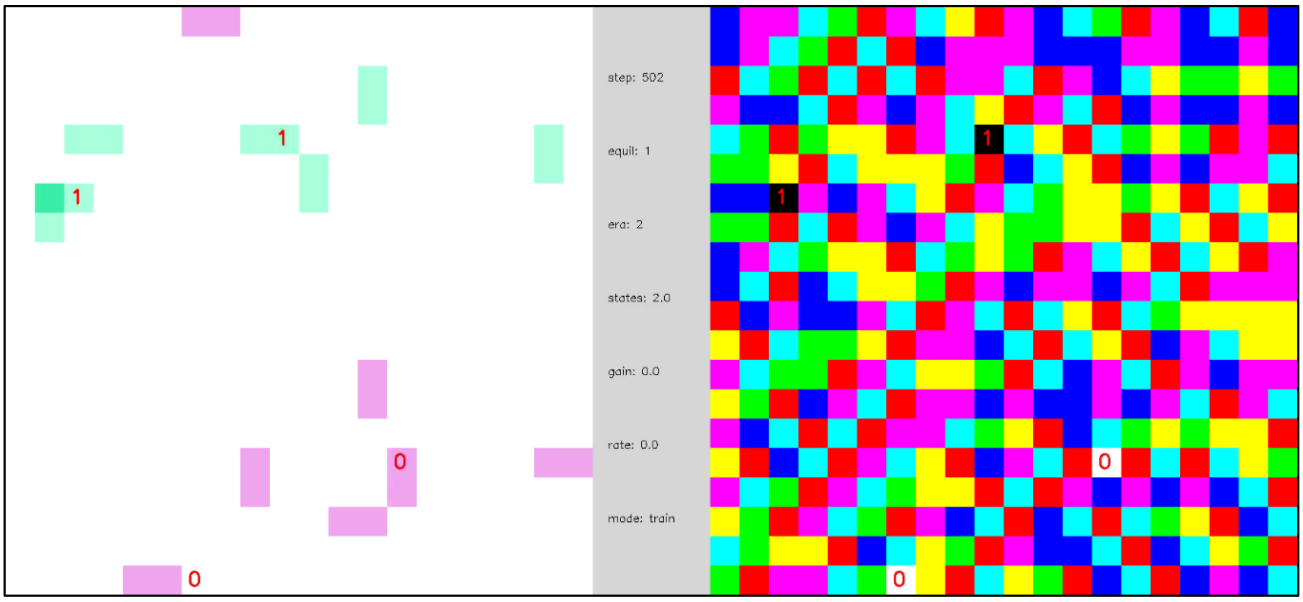
Video 1. Closed-system equilibration at a low temperature. A naïve network of 400 internal nodes with periodic boundary conditions and inverse temperature $\beta = 16$. The node outputs are randomized initially and each time the network reaches a ground state. On the left-hand side, complementary edge excitations, created when the network is randomized, annihilate as the network evolves toward a ground state. On the right-hand side, the node outputs organize into local domains that change as the network evolves toward a ground state. This simulation shows the evolution to 4 different ground states, each having the same energy. Video S1 and at https://youtu.be/bqn0qdvo4Kc (uploaded 5 October 2021).



Video 2. Closed-system equilibration at a high temperature. A naïve network of 400 randomly initialized internal nodes with periodic boundary conditions and inverse temperature $\beta = 4.5$. On the left-hand side, complementary edge excitations are both annihilated by collisions and created by thermal excitations. On the right-hand side, the local node output domains continuously evolve. Unlike in Video 1, thermal excitations prevent the formation of stable ground states. Video S2 and at https://youtu.be/cpt4Yi_3hEE (uploaded 5 October 2021).

For the remainder of this article, we consider only networks in which the temperature is low enough to inhibit thermal pair creation, which is needed to connect the TSMN to classical computing (Section 4.3). Classical computers operate in a domain where thermal fluctuations are avoided and system states are stable unless intentionally perturbed; hence, their temperature is also low.

Video 3 illustrates the evolution a network like that of Video 1, except that the network contains 2 pairs of randomly placed, complementary external nodes. Each time the network finds a ground state, the "polarity" of the external nodes is inverted ($0 \to 1$ and $1 \to 0$) and the network evolves to find another ground state. The effect of external nodes on the larger network can be thought of in three complementary ways: (1) as constraints that the network must satisfy, (2) as sources and sinks of codes that the network must transport, or (3) as sources of edge excitations that the network must annihilate. Given the large degeneracy of ground states, there are many ways that the network may configure in response to the external inputs to achieve a low energy state. If only a few ground states existed, then the likelihood of escaping local minima to find a low energy configuration would be very small—a problem that is common in some neural network and Ising models. We emphasize that the dynamics of , , and are entirely diffusive and the equilibration times are correspondingly long because there is no state transition memory and no learning to break the node output energy degeneracy of Figure 5.


step: 502

equil: 1

era: 2

states: 2.0

gain: 0.0

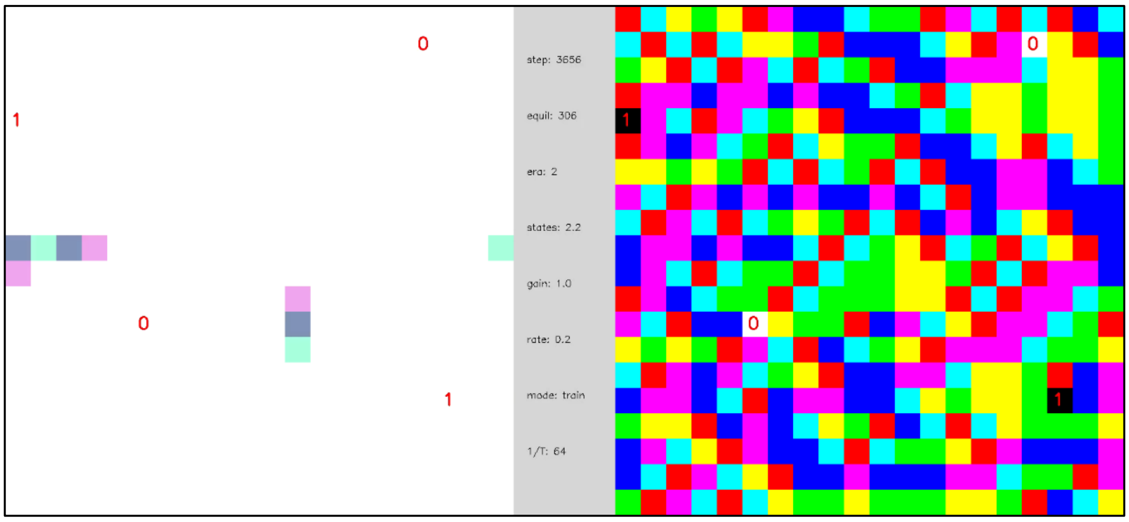
rate: 0.0

mode: train

Video 3. Open-system equilibration with periodic external inputs. A naïve network of 396 internal nodes and 2 pairs of randomly placed, complementary external nodes with periodic boundary conditions and inverse temperature $\beta = 16$. External nodes are labeled as 0 or 1 to indicate their polarity. Each time the network reaches a ground state the polarity of the external nodes is reversed. A total of four different ground states are found in this simulation. Ground states are stable because the temperature is low. Video S3 and at https://youtu.be/SQLzvfKfLgs (uploaded 5 October 2021).

3.2. Networks with Memory

Video 4 repeats the simulation of Video 3 but includes learning of the state transition memory. The initialization of the state transition memory is unbiased, $T_i(y_i|s_i) = 0$ for all i, y_i, and s_i, so the network begins in a naïve state. As in Video 3, in the early stages of the simulation the equilibration dynamics are slow and diffusive, but, as the state transition memory is learned, a phase transition emerges, and the equilibration dynamics become fast and mechanistic. To state this another way, the trajectories between network ground state equilibrations change from being random and long to being regular and short. Additionally, before the phase transition, the network dynamics cannot be described as a function, but after the phase transition the network dynamics can be described by a function—namely, as a network of (nearly) deterministic state machines.



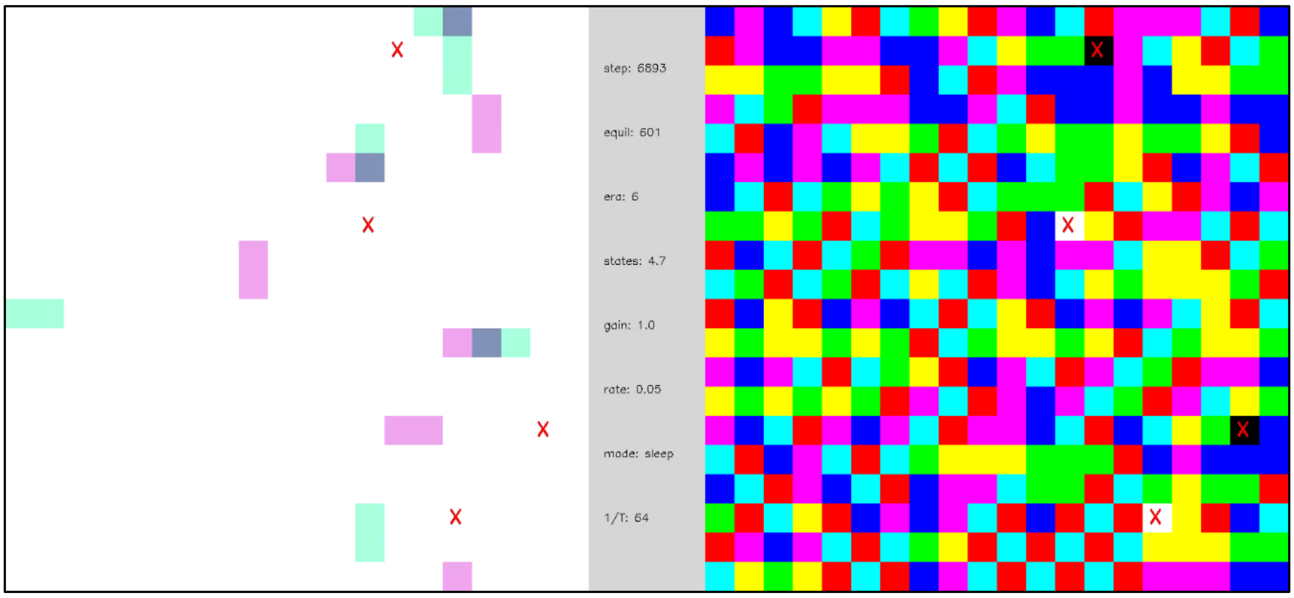
step: 3656
equil: 306
era: 2
states: 2.2
gain: 1.0
rate: 0.2
mode: train
1/T: 64

Video 4. Open system with periodic external inputs, illustrating a dynamical phase transition. A network of 396 internal nodes and 2 pairs of complementary external nodes with periodic boundary conditions, $\beta = 16$, and state transition learning with state depth $d = 1$. External nodes are labeled as 0 or 1 to indicate their polarity. Each time the network reaches a ground state the polarity of the external nodes is reversed. In the first segment of the video, the first two ground state equilibrations of the network are shown, and the dynamics are diffusive and slow. In the second segment of the video, after a gap of 300 additional ground state equilibrations (indicated as gray-colored frames in the video) in which the state transition memory is learned, the dynamics between ground states become fast and mechanistic. The central panel displays simulation parameters including the number of simulation steps ("steps"), the number of ground state equilibrations ("equil"), the average number of node states realized ("states"), the transition memory weighting parameter σ ("gain"), and the learning rate α ("rate"). Video S4 and at https://youtu.be/7H0AdQ1PxsY (uploaded 5 October 2021).

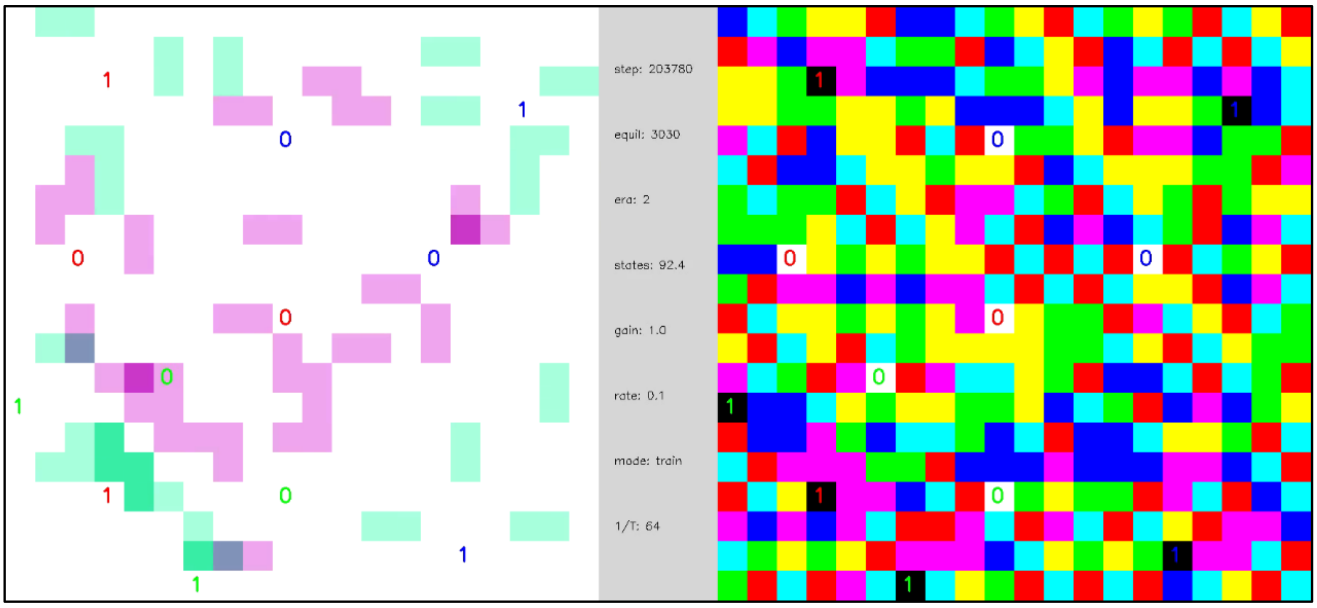
Figure 6 shows the evolution of the average node energy and the number of ground state equilibrations vs. simulation step for the network of Video 4. The rapid changes in the node energy and equilibration rate around step 1500 indicate a phase transition from diffusive to mechanistic dynamics. In this simple system, the phase transition emerges after approximately eight equilibrations. A spike in the energy occurs after each ground state equilibration, due both to an influx of new edge excitations from the external nodes and to changes in the internal network state. At each new network ground state, an updated state-index s_i addresses a new memory element T_i, which biases each node output toward its output at the *next* network ground state, and which generally *increases* the energy of the current node output (see Section 2.5). The size of these energy spikes increases as the network learns (because T_i increases) but the overall network energy decreases at the same time. In other words, the effect of T_i is to place the nodes in a metastable configuration after each ground state transition that facilitates the next ground state transition while also reducing the energy of the overall system. This storage and release of energy to equilibrate with an environment corresponds to the active equilibration postulate of Section 2.1.

After the phase transition from diffusive to mechanistic dynamics, the TSMN displays input functionalization (see Section 2.1), which we can illustrate by turning off inputs from the external nodes in the network, and allowing the internal nodes connected to these dormant external nodes to infer the missing inputs (Section 2.6). In Video 5, we explore the effects of dormant external nodes and the weighting parameter σ on the same type of network as in Video 4. As is evident in Video 5, the TSMN has functionalized its inputs after the phase transition because it can infer missing inputs from dormant external nodes and provide edge excitations as outputs to the dormant external nodes. The size of σ affects the detailed dynamics significantly by specifying the degree to which the dynamics are internally generated by the state transition memory (i.e., T_i) versus externally stimulated by the active inputs and communicated by the internal node interactions (i.e., H_i).

Video 6 shows the evolution of a network including 12 external nodes that change polarity with 3 different periods, and a corresponding state depth $d = 3$ is assumed for the state transition memory. Figure 7 captures network statics corresponding to Video 6. Because the size of the state transition memory scales as a power law with exponent d, the network of Video 6 undergoes a much larger number of ground state equilibrations before undergoing a phase transition, as compared to Video 4 (compare Figures 6 and 7). As in Video 5, various configurations of external node dormancy and state transition memory weighting are explored to illustrate input functionalization. We note that some stochasticity in the network dynamics remains even after the phase transition and that this stochasticity increases as the number of dormant inputs increases. The complexity of the network dynamics in Video 6 when all the external nodes are made dormant and $\sigma = 1$ suggests a kind of "dreaming" or "thinking" about what has been experienced in the past—an imperfect, somewhat stochastic version of that past. When the state transition memory weighting is increased to $\sigma = 2$, the network displays rapid, mechanistic dynamics that are independent of external node inputs, suggesting the ability to "think ahead" of the dynamics that initially produced it.


step: 6893
equil: 601
era: 6
states: 4.7
gain: 1.0
rate: 0.05
mode: sleep
1/T: 64

Video 5. Open system with periodic external inputs, illustrating a dynamical phase transition and input functionalization. A network of 396 internal nodes and 2 pairs of complementary external nodes with periodic boundary conditions, $\beta = 16$, and state transition learning with state depth $d = 1$. External nodes are labeled as 0 or 1 to indicate their polarity. Each time the network reaches a ground state the polarity of the external nodes is reversed. The first two ground state equilibrations are shown, and the network dynamics are diffusive and slow. After a gap of 400 ground state equilibrations, the equilibration dynamics are fast and mechanistic with $\sigma = 1$ (denoted as "gain" in the central panel). From steps 5663–5960, two external nodes are made dormant—denoted by the "X" labels—and the dynamics remain similar. From steps 5961–6441, $\sigma = 0.5$ and the dynamics become feedforward propagation of edge excitations from active to dormant external nodes. From steps 6442–6552, $\sigma = 2$ and the internal nodes spontaneously create edge excitation pairs that rapidly annihilate with other pairs. From step 6553, all external nodes become dormant and the network "makes its own dynamics" for $\sigma = 1$ and $\sigma = 2$. Video S5 and at https://youtu.be/T9tnIn-enFk (uploaded 5 October 2021).



Video 6. Open system with multi-periodic external inputs, illustrating a dynamical phase transition and input functionalization. A network of 388 internal nodes and 3 groups of 2 pairs of complementary external nodes with $\beta = 16$ and $d = 3$. External nodes are labeled as 0 or 1 to indicate their polarity. Each external node group changes polarity with a different period of ground state equilibrations—the three external node groups are distinguished by the color (red, blue, green) of the text indicating the polarity of the node. Each time the network reaches a ground state the polarities of the external nodes are updated according to their period. The first two ground state equilibrations are shown, and the network dynamics are diffusive and slow. After a gap of 3000 ground state equilibrations (gray frames), the equilibration dynamics are fast and mechanistic. From steps 203545–203793 the network dynamics are visualized with $\sigma = 1$ (denoted as "gain" in the central panel). From steps 203794–204092, six external nodes are made dormant (two per group)—denoted by the "X" labels—and the dynamics remain similar. From steps 204093–204471, all 12 external nodes are made dormant, and the network maintains similar dynamics, but with substantially greater stochasticity in the trajectories of the particles. From steps 204093–204605, the transition memory weighting is increased to $\sigma = 2$, and six external nodes are made dormant. The network spontaneously generates internal excitation pairs that rapidly annihilate to attain ground states. From step 204606 forward, all external nodes become dormant and the network effectively "makes its own dynamics". Video S6 and at https://youtu.be/TypFJ5NQJZM (uploaded 5 October 2021).

Video 7 is an edge-excitation view of a somewhat larger network after the phase transition, in which static nodes are colored gray to emphasize the structure of the network. Interestingly, the spatial structure of the evolved network looks like a neural network in which external nodes are connected by axon/dendrite-like channels communicating via spike-like edge excitations. There is even the suggestion of "integration" of these spikes before a neuron "fire". When the external nodes are made dormant, the network dynamics are more stochastic, but the underlying spatial organization remains. We note that the ground state degeneracy and stochasticity inherent in the evolution of the TSMN means that Video 7 illustrates one among many "phenotypes" that might have emerged.

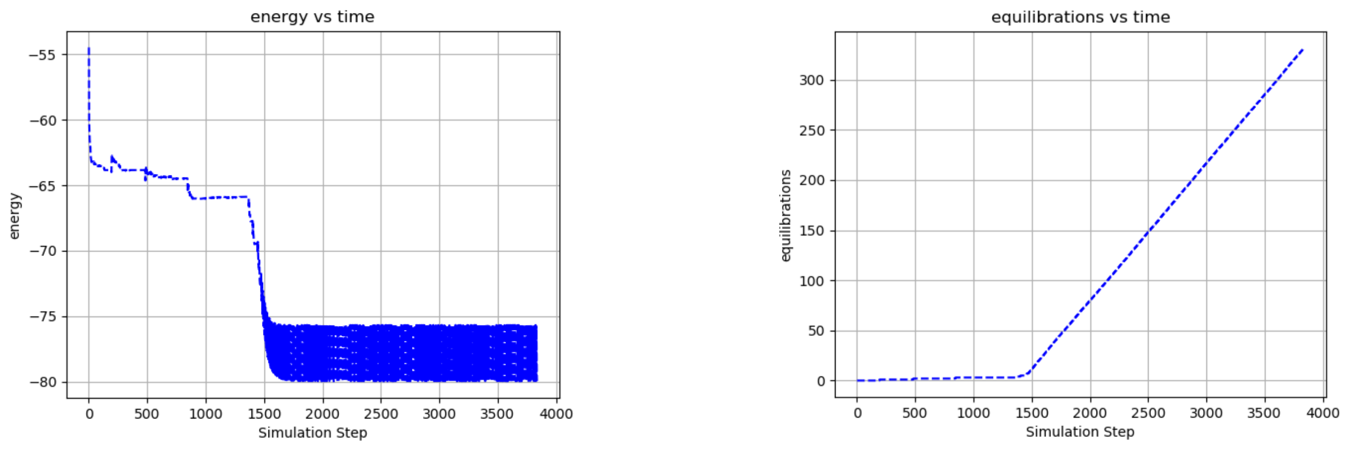


Figure 6. Network statistics for Video 4 illustrating a phase transition. Average node energy (**left**) and number of ground state equilibrations (**right**) versus simulation step. A phase transition emerges after approximately 1500 simulation steps and 8 ground state equilibrations, marking the transition from diffusive to mechanistic dynamics. The emergence of the state transition memory is illustrated in the reduction in energy as the simulation progresses. A spike in the energy occurs after each ground state equilibration, owing to the introduction of new edge excitations from the external nodes and to the change in the state transition memory. These energy spikes are dissipated as the edge excitations propagate through the network and annihilate each other.

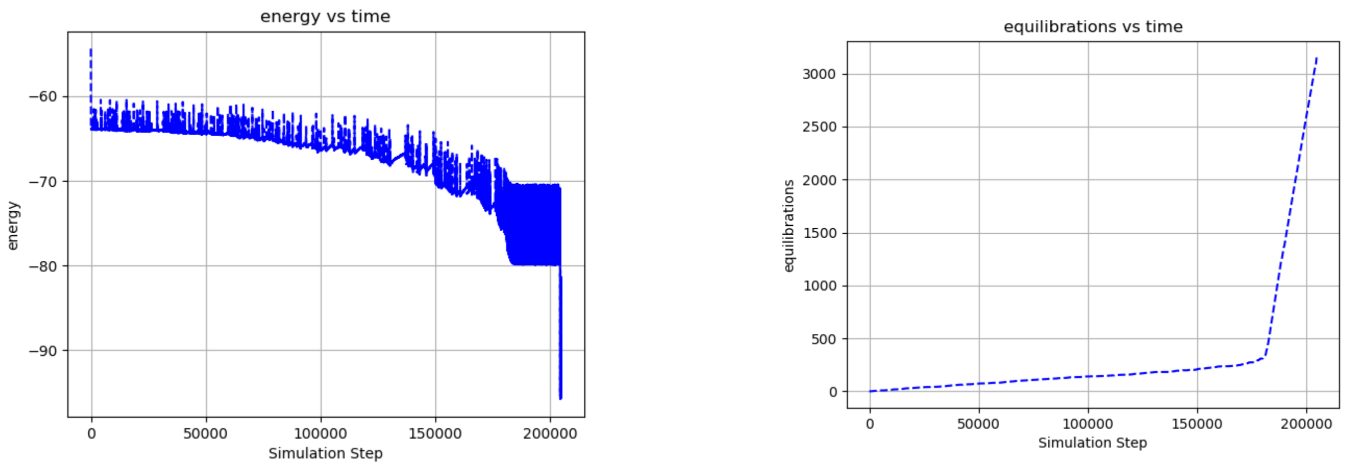
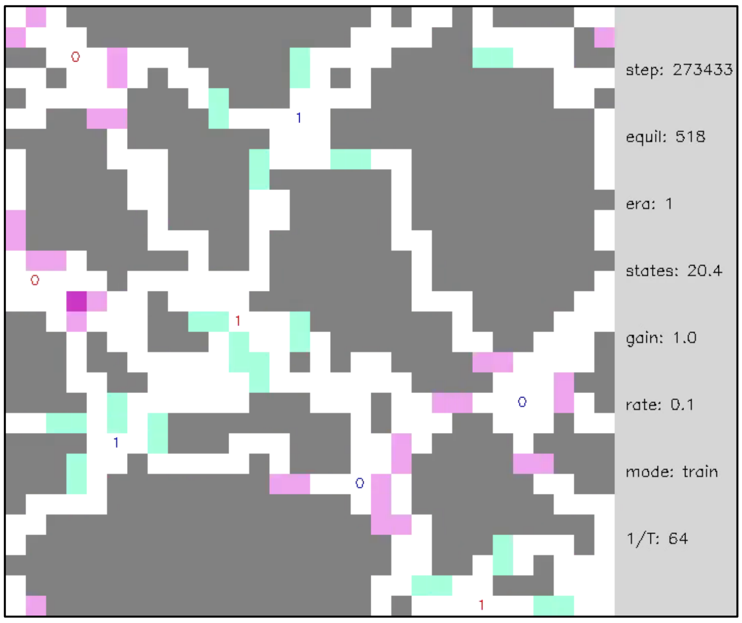


Figure 7. Network statistics of Video 6, illustrating a phase transition. Average node energy (**left**) and number of ground state equilibrations (**right**) versus simulation step. A phase transition emerges after approximately 150,000 simulation steps and 300 ground state equilibrations, marking the transition from diffusive to mechanistic dynamics. Each spike in the network energy corresponds to the introduction of new edge excitations by the external nodes, and a change in the state transition memory of the internal nodes after a ground state is achieved. The emergence of the state transition memory is illustrated by the reduction in the overall network energy and an increase in energy spikes as the simulation progresses. Although state transition memory decay (Equation (16)) is ongoing throughout the simulation, a relatively long period without ground state equilibrations around step 130,000, where the network energy is increasing, highlights this decay and the escape from a local minimum. The large change in energy at the end of the simulation is generated by a change in the weighting of the state transition memory from $\sigma = 1$ to $\sigma = 2$.



Video 7. Open system with multi-periodic external inputs, illustrating neural network structure after a dynamical phase transition. A network of 892 internal nodes and 2 groups of 2 pairs of complementary external nodes with $\beta = 16$ and $d = 2$. External nodes are labeled as 0 or 1 to indicate their polarity. Each external node group changes polarity with a different period of ground state equilibrations—the two external node groups are distinguished by the color (red, blue) of the text, indicating the polarity of the node. Each time the network reaches a ground state, the polarities of the external nodes are updated according to their period. After a gap of roughly 100 ground state equilibrations (gray frames) in which the state transition memory is learned, the equilibration dynamics are fast and mechanistic. Internal nodes that maintain fixed outputs are colored gray to emphasize the underlying spatial structure of the evolved network. From steps 273,211–273,695 the external nodes are active and from 273,696–274,650 the external nodes are dormant. External nodes resemble neurons connected by axon/dendrite-like channels communicating spike-like edge excitations. Video S7 and at https://youtu.be/IGEdvnJM4oA (uploaded 31 October 2021).

4. Discussion

In this section, we discuss the limitations of the TSMN (Section 4.1), speculate on a new approach—thermodynamic computing—to address these limitations (Section 4.2), offer an interpretation of classical computing considering the ideas presented herein (Section 4.3), and comment on an alternative philosophical perspective, thermodynamicalism (Section 4.4), addressing the concepts of the earlier sections.

4.1. Limitations of the TSMN

Generality—The TSMN has been carefully constructed such that the equilibration and adaptation algorithms of the network model are computable (see Sections 2.3 and 2.5). In particular, the ability to sample the node outputs independently (Equations (7) and (10)) is critical. We emphasize that these algorithms, while motivated by generic thermodynamic considerations, are in no way general—they work specifically for the purposes of the TSMN. In this respect, the TSMN is just one among many machine learning methods, each employing specialized algorithms of equilibration and adaptation (or, equivalently, optimization and learning) directed toward a particular task. On the other hand, the equilibration and adaptation of natural systems are seemingly not constrained by their "computability" in the same way.

Scalability—In general, the number of possible network configurations in the TSMN grows exponentially with the number of nodes, connections, edge codes, and state depth. For example, even in the 400-node networks shown above, the total number of possible network outputs ($6^{400} \cong 10^{311}$) is staggering. The effect of this very large configuration space on the computability of the TSMN is different, however, depending on whether a phase transition in the dynamics has emerged. Before the diffusive-to-mechanistic phase transition, the computational burden is large because the diffusive search for network ground states must, in principle, address the entire configuration space, and because the recursive structure of the TSMN makes the search non-deterministic. Powerful machine learning models often avoid such difficulties by eliminating recursion in the network [22], but computing resources needed to train state-of-the-art machine learning systems are still doubling every 3.4 months according to a recent report [1]. We suspect that success in finding the ground states in the examples above hinges on the relatively short diffusion distances in these small networks and the availability of many ground states ($(3/2)^{400} \cong 10^{70}$). After the phase transition, however, the network dynamics become mechanistic, and the combinatorics of the configuration space are largely irrelevant. In essence, the learning process finds and memorizes an efficient short-cut through the large combinatorial space (possible because the inputs driving it are periodic and predictable). In summary, the equilibration algorithm of the TSMN scales poorly in the learning phase, but scales well in the mechanistic phase. The same can be said of other machine learning models and computing in general.

Temporality—The dynamics of the TSMN between ground state equilibrations are nearly (Equation (16)) reversible fluctuations (Equation (10)) that do not affect the state of the network. Irreversibility in the TSMN is introduced via the updates to the state indices (Equation (9)) and learning of the state transition memory (Equations (13) and (14)) after all edge-excitations are annihilated and a network ground state is obtained. After the phase transition, this irreversibility manifests as mechanistic dynamics. Hence, the TSMN connects energy dissipation to irreversibility to the emergence of mechanistic dynamics. Additionally, the same equilibration algorithm drives both the diffusive and the mechanistic dynamics in the TSMN—there is no need to invoke separate dynamical principles a priori. Hence, the TSMN makes a clear connection between thermodynamics and temporality, but, unlike natural systems, the TSMN model is still a collection of specialized methods contrived to work well on a computer.

4.2. Thermodynamic Computing

Thermodynamic Computers (TCs) are technological systems built to address the limitations of Section 4.1. With others, we have previously described the motivation and vision for TCs [2–4]. The key hypothesis is that natural, multi-scale, thermodynamic self-organization can be an integral part of computing system hardware. In this section, we further suppose that TCs will address the postulates that motivated the TSMN (Section 2.1) and comment on the published efforts that have a bearing on these hypotheses (summarized in Table 1).

In a remarkably prescient but also very difficult exposition, Pask [23] describes most of the core concepts of a thermodynamic computer with experiments on the evolution of wires in ferrous sulfate solutions in response to external electrical potentials. Pask is concerned with the evolution of structures in physical systems and their correspondence to the evolution of concepts through "thinking". He proposes several requirements for such systems, including continuous state change, conservation of transported quantities (charge), dynamic equilibrium with an environment, organizational degeneracy, extended excited states, thermodynamic openness, component plasticity, decay of structure, and non-mechanistic behavior. In describing resistor networks with a negative temperature coefficient of resistance, he also anticipates some of the functionality of the resistive memory elements described in the examples below.

Jun & Hübler [24] have shown collections of several hundred small steel balls in a dish of castor oil, organizing in response to an electric field between the center and the periphery of the dish. From various initial placements, when the electric field is applied, the balls move to form branching networks. When one of these branches succeeds in creating a strong electrical connection between the electrodes, a phase transition in the ball dynamics is observed. Comparing this to the idea of two-component scale-integration (Section 2.1), the movement and organization of the balls plays the role of the small-spatial-scale, large-timescale adaptation, and the electron transport through these balls plays the role of the large-spatial-scale, short-timescale equilibration. The two are naturally integrated by the charges that collect on and are transported through the steel balls, which organize the collection of bearings as the system "learns", resulting in a phase transition.

Table 1. Comparison of Thermodynamic Computing-related works.

	Natural Equilibration	**Transport Driven Self Organization**	**Input Functionalization**	**Scale Integration**	**Active Dynamics**
Ferrous sulphate electrochemistry (e.g., Pask 1958)	Yes	Yes	Yes	Yes	No
Steel bearing electromigration (e.g., Jun 2015)	Yes	Yes	Yes	Yes	No
FPGA evolution (e.g., Thompson 1996)	No	Maybe	Yes	Yes	Yes
Atomic Switch Networks (e.g., Sillin 2013)	Yes	Yes	Yes	Yes	No
Ising Machines (see text)	Yes, in some cases	No	No	No	Yes
Memcomputing Machines (e.g., Traversa 2017)	Yes, in principle	Maybe	No	Yes	Yes
Chemical Protocells (e.g., Pearce 2021)	Yes	Partially	Yes	Yes	Yes
Machine Learning (e.g., TSMN model)	No	Yes	Yes	Yes	Yes

Thompson [25] demonstrate on an FPGA the evolution of a circuit that discriminates between two input frequencies ("tones"), which are much lower than the characteristic frequency of communication within the FPGA. Using an external computer to evaluate a fitness function over the FPGA output, and a genetic algorithm to sample FPGA configurations creating this output, the system evolves the ability to effectively discriminate between the tones using the underlying physics of the FPGA substrate. The method employs two timescales of evolution—the rapid dynamics of the entire FPGA and the slower, local adaptation of the genetic algorithm. Additionally, the system can functionalize its inputs and generate outputs corresponding to dormant inputs on which it was trained. Remarkably, the evolved "solution" employs physics that is entirely outside the design domain of the FPGA when used in a conventional sense.

Sillin [26] models a complex network of Ag nanowires with Ag_2S junctions ("atomic switch networks") electrically biased by external electrodes in contact with a subset of the nanowires. These models, which correspond well with experimental results, show complex network dynamics driven by the coupling between electron conduction through the network and Ag^+ ion mobility within the Ag_2S junctions. State-dependent memory effects captured within the Ag_2S junctions are also evident. In related work, some of these same authors show memory and scale integration [27] as well as dynamical phase transitions [28] in networks of Ag nanowires with polymer junctions between nanowires. Others show similar effects in networks of conducting nanoparticles near a percolation threshold [29,30]. In the language of the TSMN, low-energy configurations in the atomic switch networks are those that find strong conducting pathways through the network that connects the external electrodes [31]. The finding of such pathways creates a large flow of energy through the network, some portion of which changes the state of the Ag_2S junctions. Although these networks have been modeled for use as reservoir computers in temporal prediction tasks [32,33], they have not been shown to directly infer or predict a missing input. Perhaps a missing element within these systems as compared to the TSMN model is the ability to access or store energy internally and, thereby, to create the active dynamics needed to functionalize inputs. Another noteworthy effort [34] illustrates engineered, multiscale adaptation in networks of diffusion dominated ("neurons") and drift dominated ("synapses") memristive devices.

In other examples employing the physics of the computing substrate to accomplish a task, an engineer defines a problem by encoding certain parameters (e.g., weights or couplings) into (real or simulated) hardware, and the equilibration dynamics of the hardware solve the problem by finding a low energy configuration. Often, the problems are posed as an NP combinatorial problem, such as k-SAT, max-cut, or knapsack, which can be mapped to an Ising model [35]. Examples include Ising machines realized in CMOS annealers [36–38], coupled oscillator networks [39–41], noisy memristor crossbars [42], and quantum annealers [43]. While emphasizing the use of probabilistic magnetic bits ("p-bits") as key enabling elements, [44–47] it is possible to extend these ideas to applications of invertible logic and integer factorization in probabilistic analog hardware, digital hardware, and simulation. Lee [48] explores more complex magnetic systems, which they call "magnetic thermodynamic cores". The key task for all these Ising/Boltzmann Machines is to rapidly equilibrate toward the ground state of an Ising model given a network of coupling coefficients corresponding to a problem of interest. All these examples suffer from challenges with escaping local minima, device non-uniformity, and scaling to large networks. Among these approaches, we note the "device/circuit-level" approaches where the physics is relatively transparent, low-level hardware "computational" approaches where it is obscured, and simulations where it is entirely modeled. Hence, these Ising machines illustrate the tradeoffs associated with computation-based approaches, physics-based approaches, and the messy area between them. While all these efforts share motivation with the TSMN, they do not address the postulates of scale integration and input functionalization.

Traversa and Di Ventra [49] demonstrate the ability to evolve solutions to NP combinatorial problems with polynomial resources using a network of "self-organizing logic gates" called a "digital memcomputing machine." Like the examples in the previous paragraph, the problem to be solved is encoded by an engineer into the topology of a network, which the authors describe as the "information overhead" of the network. These self-organizing logic gates do not distinguish between inputs and outputs in the traditional sense; as networks of these gates interact, conflicts between logical assertions at each gate terminal are resolved, and an equilibrium state emerges that represents the solution to the problem encoded in the network topology. These same authors [50] describe the function of these networks in terms of dynamical systems theory, stress the intrinsic parallelism that it affords, and interpret its ability to solve NP problems using polynomial resources in terms of instantons that rapidly descend a hierarchy of critical points. Di Ventra et al. [51] generalize these ideas with a theory of topological symmetry breaking in dynamical systems. The authors argue that digital mem-computing machines can be realized in hardware as well as simulated as dynamical systems on a classical computing platform. The digital mem-computing machine approach addresses the same problem sets as in the previous paragraph and, like them, does not address all the postulates listed above. However, the mem-computing approach details important concepts that are necessary for the realization of a thermodynamic computer, including input-output equivalence, collective dynamics in classical systems, spatial non-locality stored in local memory adaptation, the role of conservation (Kirchhoff's) laws, the emergence of instanton dynamics, the role of engineered information overhead in defining a problem, and the need for intrinsic parallelism.

Given the central role of thermodynamics, chemical systems are a natural focus for the development of thermodynamic computing systems and concepts. Turing [52] perhaps anticipated this possibility in his work on pattern formation from homogeneous chemical systems. More recently, Dueñas-Díez and Pérez-Mercader [53] have shown that chemical systems can recognize words within context-sensitive languages, where the symbols in the language correspond to the introduction of aliquots into a simple, one-pot reactor. With this example, it is possible to consider any chemical reaction as a kind of thermodynamic state machine, with reactants playing the role of the self-organizing node interactions and catalysts playing the role of a state transition memory facilitating the approach to equilibrium. If the massive "parallelism" afforded by an Avogadro's number of reactants can be configured into interacting "cells", each with a much smaller number of molecules, large networks of thermodynamic automata not unlike the TSMN might be achieved. Researchers from this same group [54] have recently demonstrated important steps in this direction in more complex photochemical systems, in which cell-like vesicles—"chemical protocells"—spontaneously assemble, collapse, move, and interact with each other while carrying internal chemistry unique to their history. The various evolutionary processes in these systems naturally integrate scales and sample diverse configurations. The authors are motivated to understand the origins of life, but the connections to computing systems that organize themselves thermodynamically are a natural extension.

The approaches summarized above suggest that TCs can be constructed from hybrids of diverse electronic, electro-ionic, electro-mechanical, chemical, and electrochemical components. The integration of their diverse spatial and temporal scales also suggests an ability to create complex, evolving, self-organizing, technological systems such as the TCs envisioned here.

4.3. *Classical Computing*

Every classical computer is realized on a physical substrate (e.g., semiconductor chip) in which the transitions between states are driven by a changing energy landscape. With each clock cycle, a classical computer equilibrates to a new ground state (typically referred to as the computer's "state"), which, like the TSMN, can be associated with the acceptance of a "word" in the "language" of the computer. Like the TSMN, a classical computer has a very large number of ground states, transitions between them are driven by the transport of a conserved quantity (electric charge, typically), many different configurations can achieve the same input–output function, and the dynamics are active. Unlike the TSMN, however, the fluctuation dynamics in a classical computer are ignored: only the sequences of ground state transitions of a classical computer are considered as its dynamics. Additionally, for a classical computer, scale-integration is the result of painstaking hardware and software engineering that effectively links enormous numbers of (very small) transistors to create an output for rapid consumption by a (very large) human. Typically, this scale-integration is facilitated via a hierarchy of abstraction, or "stack", corresponding to various scales in the integration process. In other words, scale-integration in a classical computer resides within the evolutionary capacity of engineers working across many scales. With this in mind, we can draw further correspondence between the TSMN model and a classical computing system that includes its engineers. In particular, the phase transition in the TSMN dynamics corresponds to engineers obtaining a desired input/output function after an evolutionary, engineering process in which the design inputs are functionalized. In summary, classical computing works because engineers are available to perform the evolution required to integrate scales and functionalize inputs. Pablo Picasso was once quoted as saying "computers are useless. They can only give you answers" [55]. In the context argued here, we speculate that Picasso was commenting on the incompleteness of computers—namely, their inability to perform the evolutionary processes needed to solve problems or create works of art.

4.4. Thermodynamicalism

We speculate here on a different philosophical perspective that summarizes the discussion of Sections 4.1–4.3 and builds upon the postulates of Section 2.1. We call this perspective *Thermodynamicalism* and distinguish it from the perspective known as *Computationalism*, which dominates computing, engineering, and science today.

Thermodynamicalism is the philosophy that open thermodynamic systems equilibrating with external potentials will *self-organize* into *functional systems* that are *computable* after they have organized, but that the equilibrating, self-organizing process itself, which we call *thermodynamic evolution*, is not computable *in general*.

By *self-organization*, we mean a dynamic process in which a system can store energy, act upon its environment, and change its internal organization, such that equilibration with external potentials is more efficient and entropy production in the environment is sustained, even as the environment changes. By *functional systems*, we mean systems linking inputs and outputs with some internal degrees of freedom that can be described by an algorithm or mathematical function. By *computable*, we mean that the dynamics of the system can be simulated using pragmatic computing resources. The qualifier *in general* allows that certain model systems—e.g., the TSMN—have highly specialized, computable models of their evolution, while stipulating that the evolution of real physical systems of any meaningful scale is not computable. Stated differently, thermodynamicalism says variously that there is no general equilibration algorithm, no general algorithm for the second law of thermodynamics, or no general algorithm for evolution. These statements reflect the limitations of the TSMN (Section 4.1), motivate the development of thermodynamic computing (Section 4.2), and explain the incompleteness of classical computing (Section 4.3). In a nutshell, thermodynamicalism supposes that algorithms, which are the unexplained foundation of computationalism, are the artifacts of a thermodynamic, evolutionary process that is not computable in general.

5. Conclusions

Employing four postulates concerning open thermodynamic systems—transport driven self-organization, scale integration, input functionalization, and active equilibration—we have elaborated a new methodology for a new network- and machine-learning model, the Thermodynamic State Machine Network. The TSMN is a network of probabilistic automata, exchanging conserved codes over unweighted bi-directional edges, equilibrating according to Boltzmann statistics, and is characterized by a large degeneracy of ground states separated by particle-like excitations. The TSMN includes a state transition memory that learns to minimize the action of fluctuation pathways between network ground states. When exposed to periodic external inputs, the TSMN undergoes a phase transition from diffusive to mechanistic dynamics. After the phase transition, the TSMN can infer or predict missing inputs, and spontaneously generate internal dynamics representative of inputs it has functionalized. Additionally, the dynamics of the TSMN resemble those of spiking neural networks, although no such structures were incorporated into the initial model. We consider the postulates used to construct the TSMN in the context of thermodynamic computing systems and compare similarly motivated experimental efforts in the literature. We suggest that thermodynamic computing can address the fundamental limitations of the TSMN, other machine learning approaches, and classical computing in general. Given prior art, we can see no fundamental reason that thermodynamic computing systems cannot be developed. We offer speculation on a different philosophical perspective, thermodynamicalism, to address the challenges of computationalism.

Funding: This research received no external funding.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/e24060744/s1; Video S1: Closed-system equilibration at a low temperature, Video S2: Closed-system equilibration at a high temperature, Video S3: Open-system equilibration with periodic external inputs, Video S4: Open system with periodic external inputs, illustrating a dynamical phase transition, Video S5: Open system with periodic external inputs, illustrating a dynamical phase transition and input functionalization, Video S6: Open system with multi-periodic external inputs, illustrating a dynamical phase transition and input functionalization, Video S7: Open system with multi-periodic external inputs, illustrating neural network structure after a dynamical phase transition.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The author gratefully acknowledges Dharmendra Modha, Juan Perez-Mercader, Yan Yufik, Aida Todri, Behtash Behin-Aein, Neal Anderson, Zdenka Kuncic, and Joseph Friedman for their support and comments on the manuscript.

Conflicts of Interest: The author declares no conflict of interest.

Appendix A. TSMN Circuit Model

Figure A1 is a conceptual circuit for an implementation of a TSMN, showing a network, node, and state transition memory. Each node is an array of four inverters, where each inverter input is the sum of inputs from the three other inverters in the node, from the inverter of a connected node, and from a state transition memory. The idea is to create a competition among the inverters to satisfy the requirements of non-divergence, complementary edge codes, and bias learned through prior state transitions. We note that Figure A1 is not a computational substrate for the mathematical expressions above, rather, it is a circuit that captures the physical ideas that motivated the TSMN. Although Figure A1 illustrates typical CMOS inverters, stochastic inverters, such as those employing magnetic tunnel junctions [47], may be a better choice to achieve the diffusive behavior of the edge excitations shown in Figure 5.

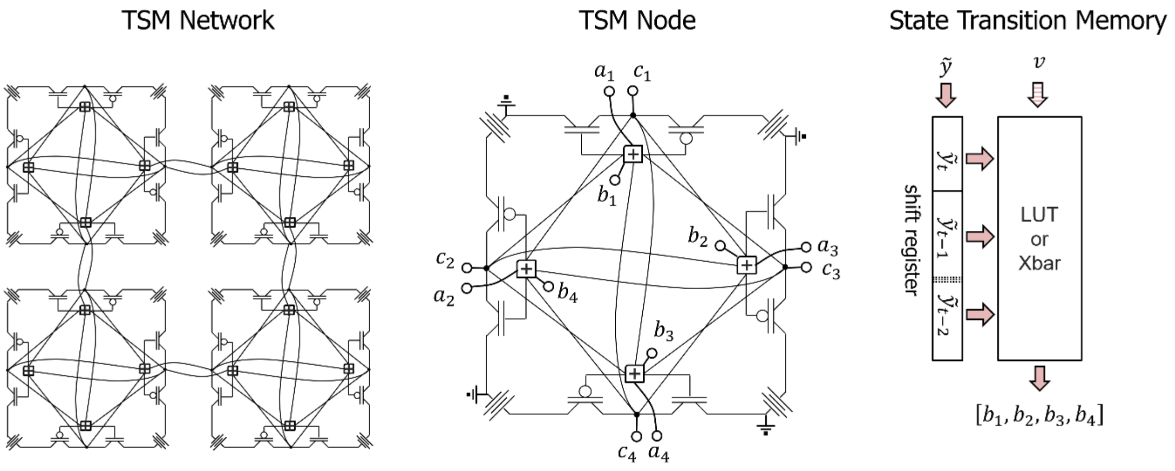


Figure A1. Conceptual circuit implementation of the TSMN. Each node is an array of four inverters that generate the output of the node, $y = [c_1, c_2, c_3, c_4]$. The inputs to each inverter are a sum of the other inverter outputs in the node and the inputs $[a_1, a_2, a_3, a_4]$ and from an inverter in another node along a corresponding edge. The circuit fosters a competition within the node to achieve a non-divergent output that complements its inputs. The state transition memory is a lookup table or crossbar array that outputs a bias vector $[b_1, b_2, b_3, b_4]$ to the summing junctions, indexed by a shift register storing the node state (the last d outputs $\tilde{y}$ at a network ground state) and trained each time a network ground state is obtained. The state transition memory is decayed whenever inputs v create an excited state. Not shown are a network signal indicating that a ground state has been reached and a memory decay concept. Details concerning the training of the bias vectors and the feasibility of the suggested implementation are still unknown.

References

1. AI and Compute. Available online: https://openai.com/blog/ai-and-compute/ (accessed on 29 April 2022).
2. Hylton, T.; Conte, T.; Still, S.; Williams, R.S. Thermodynamic Computing. In Proceedings of the Computing Community Consortium Workshop, Honolulu, HI, USA, 3–5 January 2019.
3. Hylton, T.; Conte, T.M.; Hill, M.D. A Vision to Compute like Nature: Thermodynamically. *Commun. ACM* **2021**, *64*, 35–38. [CrossRef]
4. Hylton, T. Thermodynamic Computing: An Intellectual and Technological Frontier. *Proceedings* **2020**, *47*, 23.
5. Schrödinger, E. The Physical Aspect of the Living Cell. In *What Is Life*; Camridge at the University Press: Camridge, UK, 1944.
6. Schneider, E.D.; Kay, J.J. Life as a Manifestation of the Second Law of Thermodynamics. *Math. Comput. Model.* **1994**, *19*, 25–48. [CrossRef]
7. Jarzynski, C. Nonequilibrium Equality for Free Energy Differences. *Phys. Rev. Lett.* **1997**, *78*, 2690–2693. [CrossRef]
8. Crooks, G.E. Nonequilibrium Measurements of Free Energy Differences for Microscopically Reversible Markovian Systems. *J. Stat. Phys.* **1998**, *90*, 1481–1487. [CrossRef]
9. Still, S.; Sivak, D.A.; Bell, A.J.; Crooks, G.E. Thermodynamics of Prediction. *Phys. Rev. Lett.* **2012**, *109*, 120604. [CrossRef]
10. Friston, K. Life as We Know It. *J. R. Soc. Interface* **2013**, *10*, 20130475. [CrossRef]
11. England, J.L. Dissipative Adaptation in Driven Self-Assembly. *Nat. Nanotechnol.* **2015**, *10*, 919–923. [CrossRef]
12. Wissner-Gross, A.D.; Freer, C.E. Causal Entropic Forces. *Phys. Rev. Lett.* **2013**, *110*, 168702. [CrossRef]
13. Glauber, R.J. Time-dependent Statistics of the Ising Model. *J. Math. Phys.* **1963**, *4*, 294–307. [CrossRef]
14. Hopfield, J.J. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proc. Natl. Acad. Sci. USA* **1982**, *79*, 2554–2558. [CrossRef] [PubMed]
15. Hinton, G.E.; Sejnowski, T.J. Optimal Perceptual Inference. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, Washington, DC, USA, 19–23 June 1983; Volume 448, pp. 448–453.
16. Tanaka, G.; Yamane, T.; Héroux, J.B.; Nakane, R.; Kanazawa, N.; Takeda, S.; Numata, H.; Nakano, D.; Hirose, A. Recent Advances in Physical Reservoir Computing: A Review. *Neural Netw.* **2019**, *115*, 100–123. [CrossRef] [PubMed]
17. Maass, W.; Natschläger, T.; Markram, H. Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations. *Neural Comput.* **2002**, *14*, 2531–2560. [CrossRef] [PubMed]
18. Jaeger, H. The "Echo State" Approach to Analysing and Training Recurrent Neural Networks-with an Erratum Note. *Ger. Natl. Res. Cent. Inf. Technol. GMD Tech. Rep.* **2001**, *148*, 13.
19. Scellier, B.; Bengio, Y. Equilibrium Propagation: Bridging the Gap between Energy-Based Models and Backpropagation. *Front. Comput. Neurosci.* **2017**, *11*, 24. [CrossRef]
20. Hylton, T. Thermodynamic Neural Network. *Entropy* **2020**, *22*, 256. [CrossRef]
21. Geman, S.; Geman, D. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Trans. Pattern Anal. Mach. Intell.* **1984**, 721–741. [CrossRef]
22. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2017; Volume 30.
23. Pask, G. Physical Analogues to the Growth of a Concept. In *Mechanisation of thought Processes: Proceedings of a Symposium Held at the National Physical Laboratory on 24th, 25th, 26th and 27th November 1958*; National Physical Laboratory: Teddington, UK, 1958; Volume 1958, pp. 765–794.
24. Jun, J.K.; Hübler, A.H. Formation and Structure of Ramified Charge Transportation Networks in an Electromechanical System. *Proc. Natl. Acad. Sci. USA* **2005**, *102*, 536–540. [CrossRef]
25. Thompson, A. An Evolved Circuit, Intrinsic in Silicon, Entwined with Physics. In Proceedings of the Evolvable Systems: From Biology to Hardware; Higuchi, T., Iwata, M., Liu, W., Eds.; Springer: Berlin/Heidelberg, Germany, 1997; pp. 390–405.
26. Sillin, H.O.; Aguilera, R.; Shieh, H.-H.; Avizienis, A.V.; Aono, M.; Stieg, A.Z.; Gimzewski, J.K. A Theoretical and Experimental Study of Neuromorphic Atomic Switch Networks for Reservoir Computing. *Nanotechnology* **2013**, *24*, 384004. [CrossRef]
27. Diaz-Alvarez, A.; Higuchi, R.; Sanz-Leon, P.; Marcus, I.; Shingaya, Y.; Stieg, A.Z.; Gimzewski, J.K.; Kuncic, Z.; Nakayama, T. Emergent Dynamics of Neuromorphic Nanowire Networks. *Sci. Rep.* **2019**, *9*, 14920. [CrossRef]
28. Hochstetter, J.; Zhu, R.; Loeffler, A.; Diaz-Alvarez, A.; Nakayama, T.; Kuncic, Z. Avalanches and Edge-of-Chaos Learning in Neuromorphic Nanowire Networks. *Nat. Commun.* **2021**, *12*, 1–13. [CrossRef] [PubMed]
29. Bose, S.K.; Shirai, S.; Mallinson, J.B.; Brown, S.A. Synaptic Dynamics in Complex Self-Assembled Nanoparticle Networks. *Faraday Discuss.* **2019**, *213*, 471–485. [CrossRef] [PubMed]
30. Pike, M.D.; Bose, S.K.; Mallinson, J.B.; Acharya, S.K.; Shirai, S.; Galli, E.; Weddell, S.J.; Bones, P.J.; Arnold, M.D.; Brown, S.A. Atomic Scale Dynamics Drive Brain-like Avalanches in Percolating Nanostructured Networks. *Nano Lett.* **2020**, *20*, 3935–3942. [CrossRef]
31. Kuncic, Z.; Marcus, I.; Sanz-Leon, P.; Higuchi, R.; Shingaya, Y.; Li, M.; Stieg, A.; Gimzewski, J.; Aono, M.; Nakayama, T. Emergent Brain-like Complexity from Nanowire Atomic Switch Networks: Towards Neuromorphic Synthetic Intelligence. In Proceedings of the 2018 IEEE 18th International Conference on Nanotechnology (IEEE-NANO), Cork, Ireland, 23–26 July 2018; pp. 1–3.
32. Zhu, R.; Hochstetter, J.; Loeffler, A.; Diaz-Alvarez, A.; Stieg, A.; Gimzewski, J.; Nakayama, T.; Kuncic, Z. Harnessing Adaptive Dynamics in Neuro-Memristive Nanowire Networks for Transfer Learning. In Proceedings of the 2020 International Conference on Rebooting Computing (ICRC), Atlanta, GA, USA, 1–3 December 2020; pp. 102–106.

33. Lilak, S.; Woods, W.; Scharnhorst, K.; Dunham, C.; Teuscher, C.; Stieg, A.Z.; Gimzewski, J.K. Spoken Digit Classification by In-Materio Reservoir Computing With Neuromorphic Atomic Switch Networks. *Front. Nanotechnol.* **2021**, *3*, 38. [CrossRef]
34. Wang, Z.; Joshi, S.; Savel'ev, S.; Song, W.; Midya, R.; Li, Y.; Rao, M.; Yan, P.; Asapu, S.; Zhuo, Y.; et al. Fully Memristive Neural Networks for Pattern Classification with Unsupervised Learning. *Nat. Electron.* **2018**, *1*, 137–145. [CrossRef]
35. Lucas, A. Ising Formulations of Many NP Problems. *Front. Phys.* **2014**, *2*, 5. [CrossRef]
36. Takemoto, T.; Hayashi, M.; Yoshimura, C.; Yamaoka, M. 2.6 A 2 × 30 k-Spin Multichip Scalable Annealing Processor Based on a Processing-In-Memory Approach for Solving Large-Scale Combinatorial Optimization Problems. In Proceedings of the 2019 IEEE International Solid-State Circuits Conference-(ISSCC), San Francisco, CA, USA, 17–21 February 2019; pp. 52–54.
37. Su, Y.; Kim, H.; Kim, B. 31.2 CIM-Spin: A 0.5-to-1.2V Scalable Annealing Processor Using Digital Compute-In-Memory Spin Operators and Register-Based Spins for Combinatorial Optimization Problems. In Proceedings of the 2020 IEEE International Solid-State Circuits Conference-(ISSCC), San Francisco, CA, USA, 16–20 February 2020; pp. 480–482.
38. Yamamoto, K.; Ando, K.; Mertig, N.; Takemoto, T.; Yamaoka, M.; Teramoto, H.; Sakai, A.; Takamaeda-Yamazaki, S.; Motomura, M. 7.3 STATICA: A 512-Spin 0.25M-Weight Full-Digital Annealing Processor with a Near-Memory All-Spin-Updates-at-Once Architecture for Combinatorial Optimization with Complete Spin-Spin Interactions. In Proceedings of the 2020 IEEE International Solid-State Circuits Conference-(ISSCC), San Francisco, CA, USA, 16–20 February 2020; pp. 138–140.
39. Wang, T.; Roychowdhury, J. OIM: Oscillator-Based Ising Machines for Solving Combinatorial Optimisation Problems. In Proceedings of the Unconventional Computation and Natural Computation; McQuillan, I., Seki, S., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 232–256.
40. Chou, J.; Bramhavar, S.; Ghosh, S.; Herzog, W. Analog Coupled Oscillator Based Weighted Ising Machine. *Sci. Rep.* **2019**, *9*, 14786. [CrossRef]
41. Dutta, S.; Khanna, A.; Gomez, J.; Ni, K.; Toroczkai, Z.; Datta, S. Experimental Demonstration of Phase Transition Nano-Oscillator Based Ising Machine. In Proceedings of the 2019 IEEE International Electron Devices Meeting (IEDM), San Francisco, CA, USA, 7–11 December 2019; pp. 37.8.1–37.8.4.
42. Cai, F.; Kumar, S.; Van Vaerenbergh, T.; Sheng, X.; Liu, R.; Li, C.; Liu, Z.; Foltin, M.; Yu, S.; Xia, Q.; et al. Power-Efficient Combinatorial Optimization Using Intrinsic Noise in Memristor Hopfield Neural Networks. *Nat. Electron.* **2020**, *3*, 409–418. [CrossRef]
43. Johnson, M.W.; Amin, M.H.S.; Gildert, S.; Lanting, T.; Hamze, F.; Dickson, N.; Harris, R.; Berkley, A.J.; Johansson, J.; Bunyk, P.; et al. Quantum Annealing with Manufactured Spins. *Nature* **2011**, *473*, 194–198. [CrossRef]
44. Camsari, K.Y.; Faria, R.; Sutton, B.M.; Datta, S. Stochastic P-Bits for Invertible Logic. *Phys. Rev. X* **2017**, *7*, 031014. [CrossRef]
45. Borders, W.A.; Pervaiz, A.Z.; Fukami, S.; Camsari, K.Y.; Ohno, H.; Datta, S. Integer Factorization Using Stochastic Magnetic Tunnel Junctions. *Nature* **2019**, *573*, 390–393. [CrossRef] [PubMed]
46. Pervaiz, A.Z.; Sutton, B.M.; Ghantasala, L.A.; Camsari, K.Y. Weighted P-Bits for FPGA Implementation of Probabilistic Circuits. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 1920–1926. [CrossRef] [PubMed]
47. Camsari, K.Y.; Salahuddin, S.; Datta, S. Implementing P-Bits With Embedded MTJ. *IEEE Electron Device Lett.* **2017**, *38*, 1767–1770. [CrossRef]
48. Lee, A.; Dai, B.; Wu, D.; Wu, H.; Schwartz, R.N.; Wang, K.L. A Thermodynamic Core Using Voltage-Controlled Spin–Orbit-Torque Magnetic Tunnel Junctions. *Nanotechnology* **2021**, *32*, 505405. [CrossRef] [PubMed]
49. Traversa, F.L.; Di Ventra, M. Polynomial-Time Solution of Prime Factorization and NP-Complete Problems with Digital Memcomputing Machines. *Chaos Interdiscip. J. Nonlinear Sci.* **2017**, *27*, 023107. [CrossRef] [PubMed]
50. Di Ventra, M.; Traversa, F.L. Perspective: Memcomputing: Leveraging Memory and Physics to Compute Efficiently. *J. Appl. Phys.* **2018**, *123*, 180901. [CrossRef]
51. Di Ventra, M.; Traversa, F.L.; Ovchinnikov, I.V. Topological Field Theory and Computing with Instantons. *Ann. Phys.* **2017**, *529*, 1700123. [CrossRef]
52. Turing, A.M. The Chemical Basis of Morphogenesis. *Bull. Math. Biol.* **1990**, *52*, 153–197. [CrossRef]
53. Dueñas-Díez, M.; Pérez-Mercader, J. How Chemistry Computes: Language Recognition by Non-Biochemical Chemical Automata. From Finite Automata to Turing Machines. *iScience* **2019**, *19*, 514–526. [CrossRef]
54. Pearce, S.; Perez-Mercader, J. Chemoadaptive Polymeric Assemblies by Integrated Chemical Feedback in Self-Assembled Synthetic Protocells. *ACS Cent. Sci.* **2021**, *7*, 1543–1550. [CrossRef]
55. Fifield, W. Pablo Picasso: A Composite Interview. *Paris Rev.* **1964**, *32*, 37.