UNIVERSITY OF CALIFORNIA SAN DIEGO

The Physics of Computing with Memory

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Physics

by

Yuanhang Zhang

2024

University of California San Diego

2024

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my PhD advisor, Professor Massimiliano Di Ventra, for his invaluable guidance and unwavering support throughout my PhD journey. His expertise and insights have been fundamental to my development and the success of my research.

I am immensely thankful to my fiancée, Yu Pei, for her love, patience, and understanding. Her companionship during this journey has been a source of comfort and strength.

My heartfelt thanks go to my family and friends, who have provided endless encouragement and support. Their belief in me has been a constant motivation and has made this journey all the more meaningful.

Chapter 1.2, in full, is a reprint of the material as it appears in *Nature Communications* [1]. Yuan-Hang Zhang, Chesson Sipling, Erbin Qiu, Ivan K Schuller, Massimiliano Di Ventra, 2024. The dissertation author was the primary investigator and author of this paper.

Chapter 2.2, in full, is a reprint of the material as it appears in *Chaos: An Interdisciplinary Journal of Nonlinear Science* [2]. Yuan-Hang Zhang, Massimiliano Di Ventra, 2021. The dissertation author was the primary investigator and author of this paper.

Chapter 2.3, in full, is a reprint of the material as it appears in *International Journal of Circuit Theory And Applications* [3]. Yuan-Hang Zhang, Massimiliano Di Ventra, 2024. The dissertation author was the primary investigator and author of this paper.

Chapter 3.1, in full, is a reprint of the material as it appears in *Physical Review A* [4]. Yuan-Hang Zhang, Massimiliano Di Ventra, 2022. The dissertation author was the primary investigator and author of this paper.

Chapter 3.2, in full, is a reprint of the material as it appears in *Physical Review B* [5]. Yuan-Hang Zhang, Massimiliano Di Ventra, 2023. The dissertation author was the primary investigator and author of this paper.

VITA

2019	Bachelor of Science in Physics, University of Science and Technology of China
2020	Teaching Assistant, Department of Physics, University of California San Diego
2020–2024	Graduate Student Researcher, University of California San Diego
2024	Doctor of Philosophy in Physics, University of California San Diego

PUBLICATIONS

[1] Zhang, Y. H., Sipling, C., Qiu, E., Schuller, I.K., & Di Ventra, M. (2024). *Collective dynamics and long-range order in thermal neuristor networks.* Nature Communications, 15(1), 6986.
[2] Zhang, Y. H., & Di Ventra, M. (2024). *Implementation of digital MemComputing using standard electronic components.* International Journal of Circuit Theory and Applications.
[3] Qiu, E., Zhang, Y. H., Di Ventra, M., & Schuller, I. K. (2023). *Reconfigurable cascaded thermal neuristors for neuromorphic computing.* Advanced Materials, 2306818.
[4] Primosch, D., Zhang, Y. H., & Di Ventra, M. (2023). *Self-averaging of digital memcomputing machines.* Physical Review E, 108(3), 034306.
[5] Nguyen, D.C., Zhang, Y. H., Di Ventra, M. & Pershin, Y.V. (2023). *Hardware implementation of digital memcomputing on small-size FPGAs.* 2023 IEEE 66th International Midwest Symposium on Circuits and Systems.
[6] Zhang, Y. H., & Di Ventra, M. (2023). *Transformer quantum state: A multipurpose model for quantum many-body problems.* Physical Review B, 107(7), 075147.
[7] Zhang, Y. H., & Di Ventra, M. (2022). *Efficient quantum state tomography with mode-assisted training.* Physical Review A, 106(4), 042420.
[8] Zhang, Y. H., & Di Ventra, M. (2021). *Directed percolation and numerical stability of simulations of digital memcomputing machines.* Chaos: An Interdisciplinary Journal of Nonlinear Science, 31(6), 063127.
[9] Zhang, Y. H., Zheng, P. L., Zhang, Y., & Deng, D. L. (2020). *Topological quantum compiling with reinforcement learning.* Physical Review Letters, 125(17), 170501.
[10] Zhao, J., Zhang, Y. H., Shao, C. P., Wu, Y. C., Guo, G. C., & Guo, G. P. (2019). *Building quantum neural networks based on a swap test.* Physical Review A, 100(1), 012334.
[11] Jia, Z. A., Zhang, Y. H., Wu, Y. C., Kong, L., Guo, G. C., & Guo, G. P. (2019). *Efficient machine-learning representations of a surface code with boundaries, defects, domain walls, and twists.* Physical Review A, 99(1), 012307.
[12] Zhang, Y. H., Jia, Z. A., Wu, Y. C., & Guo, G. C. (2018). *An efficient algorithmic way to construct Boltzmann machine representations for arbitrary stabilizer code.* arXiv preprint arXiv:1809.08631.

ABSTRACT OF THE DISSERTATION

The Physics of Computing with Memory

by

Yuanhang Zhang

Doctor of Philosophy in Physics

University of California San Diego, 2024

Professor Massimiliano Di Ventra, Chair

The evolution of computing technologies has perpetually intersected with the fundamental principles of physics. In this dissertation, we explore the frontier of computational paradigms through the lens of memory-augmented physical systems. Conventional computing, constrained by the architecture of Turing machines, can be substantially evolved by incorporating elements of memory into the physical computing substrates.

We demonstrate that time non-local interactions, conceptualized as "memory," can induce spatial long-range order by correlating distant computational units despite their spatially local interactions. Such long-range order is critical for solving complex optimization problems as it enables strategies that transcend local moves to escape local minima. MemComputing, embodying this methodology, solves target problems by following the trajectory of a dynamical system embedded with memory. This system is meticulously designed so that its equilibrium points align with the solutions of the problem, and it explores distant configurations through instantonic tunneling.

We provide a comprehensive demonstration of the MemComputing framework through applications in complex problem-solving scenarios, including the efficient simulation of quantum systems and tackling NP-complete problems like the SAT problem. Our findings indicate significant enhancements over traditional computing methods, spotlighting the profound potential of integrating memory with physical systems for next-generation computing.

This research not only deepens our understanding of the intersection between memory and physical laws in computational processes but also establishes a foundational basis for the development of next-generation computing technologies. These technologies are poised to be more efficient and scalable, representing a significant leap over conventional computational frameworks.

Introduction

Computing, fundamentally a mathematical concept, becomes tangible through the application of physics. Consider a hand-held calculator where I input: $59 \times 97 =$, and it displays: 5723. Mathematically, this is the product of 59 and 97, but physically, the process is more complex. Inside the calculator, each number is transformed into a binary string, represented as high and low voltages across a network of transistors. A binary multiplication circuit, composed of CMOS logic gates, processes these strings to produce an output voltage corresponding to the result, which is subsequently converted back to base-10. The design and operation of the entire circuitry, governed fundamentally by Maxwell's equations, illustrate the deep integration of physical laws in computing processes.

Despite the exponential growth in computing power over the past fifty years, as predicted by Moore's Law, the fundamental physics underlying modern computers remains unchanged since the ENIAC, the first digital computer. These systems essentially manipulate symbols on a tape, a concept introduced by Alan Turing in his design of the Turing machines, and thus inherit similar constraints.

Consider the inverse problem: identifying the factors of 5723. Within Turing's computational framework, there is currently no efficient solution to integer factorization; the approach is essentially brute force. For smaller numbers, simple trial divisions might quickly yield factors; however, for larger numbers, even the best known classical algorithm, the general number field sieve, is nothing more than enhancing the brute-force trial divisions with modular operations and strategic grouping.

However, computing transcends the mathematical confines of Turing machines, constrained only by the laws of physics. When employing quantum mechanics as opposed to electrodynamics, Shor's algorithm demonstrates the potential to efficiently factor large numbers using polynomial resources. This pivotal discovery catalyzed the burgeoning field of quantum computing.

In this thesis, we delve into the computational potential of classical statistical mechanics, particularly through the lens of complex systems. As Philip Anderson stated, "More is different." Within complex systems, while individual components may adhere to simplistic rules, their collective behavior can exhibit emergent properties that introduce both unexpected challenges and exciting opportunities. A prime example is the neuron—whether biological or artificial, the behavior of a singular neuron is comprehensively understood. However, in aggregation, billions of neurons contribute to the most intricate yet profound computation, culminating in the emergence of intelligence. A critical, yet often overlooked, component in this process is memory, or the capacity for time non-local interactions. This thesis will demonstrate how a system's ability to recall its past dynamics facilitates spatial long-range order, leading towards a novel computational paradigm known as MemComputing.

The thesis is structured into three parts: First, we examine the role of memory in complex systems, illustrating how memory can foster long-range order and its implications for computational tasks. Subsequently, we detail the MemComputing paradigm, focusing on the properties and implementation of MemComputing machines. Finally, we investigate the applications of MemComputing within machine learning, with a particular emphasis on simulating quantum systems.

In conclusion, we aim to broaden the horizons of computational theory beyond conventional frameworks, venturing into the realms governed by the laws of classical statistical mechanics. By exploring the profound impact of memory and emergent behaviors in complex systems, we aim to not only advance our understanding of computation but also propose practical frameworks that could revolutionize how we approach complex computational challenges. MemComputing, a paradigm that leverages the intrinsic properties of memory within systems, represents a potential breakthrough in our ability to process and analyze information at unprecedented scales. As we navigate through these discussions, it is our hope that this work will illuminate new pathways for integrating physical principles with computational technologies, offering insights that are as practical as they are theoretical.

Chapter 1

Memory-induced long-range order

1.1 Introduction: Long-range order arising from memory

Consider a maze where ants travel from a nest to a feeding site, leaving pheromones that act as memory degrees of freedom (DOFs). These pheromones, which persist much longer than an ant's traversal time, provide crucial navigational information and reinforce the shortest path. This results in strong, long-range spatial correlations among ants, as even those far apart can influence each other's routes through the enduring pheromone trails. Thus, memory in the form of pheromones not only facilitates non-local communication across time and space but also fundamentally alters the dynamics of the system.

This scenario illustrates the concept of memory-induced long-range order (MILRO), where "memory" specifically denotes time non-locality rather than mere storage. This form of memory enables increasingly broad couplings despite the inherently local nature of interactions. MILRO has been observed in diverse systems, including neuromorphic systems [1], spin-glasses [6], and dynamical systems endowed with memory [7]. Such order can be elicited and precisely adjusted by manipulating the memory degrees of freedom within these systems.

A concept closely related yet distinctly different is criticality, which emerges at the boundaries of phase transitions. These are characterized by fluctuations that span all scales and correlations that endure indefinitely. In the realm of neuroscience, the critical brain hypothesis suggests that the brain functions at a state of criticality, situated at the brink between distinct phases, which is believed to be optimal for processing information.

The rapid advancements in artificial intelligence underscore the significance of the critical brain hypothesis in the development of efficient machine learning algorithms and neuromorphic systems. However, the hypothesis remains controversial. On one hand, the evidence for criticality in brain activity is equivocal, leading some to doubt whether the brain truly operates in a critical state. On the other hand, the absence of criticality in contemporary deep learning architectures prompts questions about its relevance and practicality in both the theoretical and applied aspects of machine learning.

In light of these considerations, MILRO offers a fresh perspective [1, 6, 8]. Unlike criticality, which demands precise calibration to maintain a system at a transition point and is inherently fragile, MILRO characterizes a robust phase that is easier to achieve and significantly more resilient to perturbations. Building on the notion that criticality enhances computational efficiency, it becomes clear that MILRO not only serves this purpose but also offers greater practicality and advantageous properties.

Next, we will delve into a neuromorphic system to explore how long-range order, influenced by memory, arises and to discuss its implications for computing tasks.

1.2 Collective dynamics and long-range order in thermal neuristor networks

In the pursuit of scalable and energy-efficient neuromorphic devices, recent research has unveiled a novel category of spiking oscillators, termed “thermal neuristors.” These devices function via thermal interactions among neighboring vanadium dioxide resistive memories, emulating biological neuronal behavior. Here, we show that the collective dynamical behavior of networks of these neurons showcases a rich phase structure, tunable by adjusting the thermal coupling and input voltage. Notably, we identify phases exhibiting long-range order that, however, does not arise from criticality, but rather from the time non-local response of the system. In addition, we show that these thermal neuristor arrays achieve high accuracy in image recognition and time series prediction through reservoir computing, without leveraging long-range order. Our findings highlight a crucial aspect of neuromorphic computing with possible implications on the functioning of the brain: criticality may not be necessary for the efficient performance of neuromorphic systems in certain computational tasks.

1.2.1 Introduction

Neuromorphic computing, a field inspired by brain functionality, represents a powerful approach to tackle a wide range of information processing tasks that are not instruction-based, such as those typical of artificial intelligence and machine learning [9, 10, 11]. Unlike traditional computers that use the von Neumann architecture, separating memory and computing, neuromorphic systems utilize artificial neurons and synapses. These components can be implemented using diverse physical systems, such as photonics [12], spintronics [13], resistive switching materials [14, 15], and electrochemical devices [16].

In neuromorphic systems, regardless of the underlying physical framework, information processing is executed via a spiking neural network [17]. Neurons in this network emit spikes in response to specific external stimuli. These spikes travel through synapses, either exciting or inhibiting downstream neurons. During training for a particular task, synaptic weights are iteratively updated, guided by either biologically-inspired algorithms like spike timing-dependent plasticity [18] and evolutionary algorithms [19] or adaptations of traditional machine learning algorithms like backpropagation [20].

The collective, as opposed to the individual behavior of the neurons in the network, facilitates the aforementioned tasks. This collective behavior may also be essential for the functioning of the animal brain. For instance, the "critical brain hypothesis" suggests that the brain operates in a state of "criticality"; namely, it is poised at a transition point between different phases [21, 22, 23, 24, 25]. This critical state is believed to be optimal for the brain's response to both internal and external stimuli, due to its structural and functional design. Yet, despite the popularity of the hypothesis, questions and doubts remain, and some argue that the brain is not truly critical or not critical at all [23, 26, 27, 28].

In our present study, we do not aim to directly tackle the critical brain hypothesis. Rather, we approach the subject from a different angle: we examine a neuromorphic system that exhibits brain-like features. With similar working principles, one can then naturally extend the critical brain hypothesis to neuromorphic systems and question whether spiking neural networks also function at a critical state. This topic remains contentious, and arguments supporting [29, 30] and opposing [31] the notion have been reported, each presenting slightly different definitions and perspectives.

In this work, we show that a neuromorphic system may support long-range ordered (LRO) phases, without criticality. The origin of this LRO is the time non-local (memory) response of the system to external perturbations. On the other hand, we show that such LRO is not necessary for certain computational tasks, such as classification and time series predictions. These results may provide some hints on the functioning of biological brains.

As a specific example, we consider a neuromorphic system comprised of thermal neuristors [32, 15], based on vanadium dioxide (VO_2) spiking oscillators that communicate via heat signals. The properties of the individual oscillators (which take advantage of the hysteric metal-insulator transition of VO_2) and their mutual interactions have been experimentally validated earlier [32, 15]. These earlier studies form the basis of our numerical model of a large-scale network, which allows us to numerically analyze the collective dynamics of the system. We find that the different phases can be tuned by varying the thermal coupling between the neurons and the input voltage. We apply this system to image recognition tasks using reservoir computing [33] and explore the relationship between performance and collective dynamics. We find that LRO does not necessarily enhance the performance in tasks like image recognition, a result in line with the findings of Ref. [31].

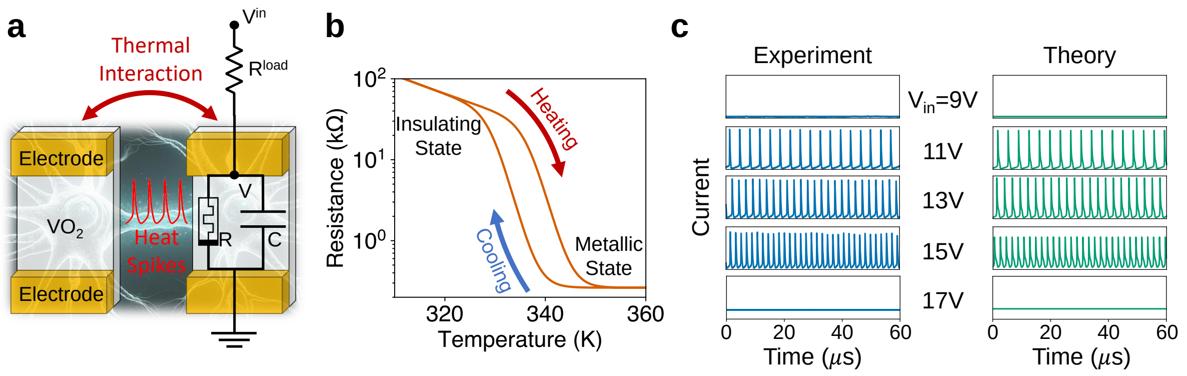


Figure 1.1. Overview of the thermal neuristor model. (a) Schematic and circuit diagram of two neighboring thermal neuristors. Each neuristor is modeled as an RC circuit, which undergoes stable spiking oscillations with proper external input. Neighboring neuristors are electrically isolated but communicate with each other through thermal interactions. (b) The resistance-temperature characteristic of the VO_2 film, denoted by the variable resistor R in panel (a). VO_2 exhibits an insulator-to-metal transition at approximately 340 K, characterized by distinct heating and cooling trajectories, thus forming a hysteresis loop. (c) Illustration of stable spiking oscillations in a single neuristor across various input voltages, with the y-axis range for each plot set between 0 and 5 mA. Numerical simulations based on Eqs. (1.1) and (1.2) align well with experimental data, demonstrating stable spiking patterns within a certain input voltage range and an increase in spiking frequency proportional to the input voltage.

1.2.2 Results

VO_2-based oscillators have been utilized as artificial neurons in many previous studies [34, 35, 36, 37, 38, 32, 15], each featuring slightly different designs, mechanisms, and applications. In particular, we focus on thermal neuristors, a concept pioneered in [15], which effectively reproduces the behavior of biological neurons. These neuristors are not only straightforward to manufacture experimentally but also exhibit advantageous properties such as rapid response times and low energy consumption.

Fig. 1.1(a) presents the design and circuitry of the thermal neuristor, featuring a thin VO_2 film connected in series to a variable load resistor. VO_2 undergoes an insulator-to-metal transition (IMT) at approximately 340 K [39], with different resistance-temperature heating and cooling paths, which leads to a hysteresis loop, as depicted in Fig. 1.1(b). Additionally, the system includes a parasitic capacitance resulting from the cable connections, which is vital for the neuristor's operation.

The behavior of the circuit displayed in Fig. 1.1(a) closely resembles a leaky integrate-and-fire neuron [40]. The capacitor C is charged up by the voltage source, V^{in}, and slowly leaks current through R. When the voltage across VO_2 reaches a threshold, joule heating initiates the IMT, drastically reducing resistance in the VO_2 which causes C to discharge, leading to a current spike. At the same time, the reduced resistance leads to reduced joule heating, which is then insufficient to maintain the metallic state, causing the VO_2 film to revert to its insulating phase. This process repeats, producing consistent spiking oscillations.

We have experimentally fabricated and evaluated this system of VO_2-based thermal neuristors. The spiking behavior of a single neuristor is shown in Fig. 1.1(c). With insufficient heating, the neuristor does not switch from the insulating state whereas excessive heating keeps it perpetually in the metallic state. As a consequence, no spiking patterns emerge when the input voltage is too low or too high. Numerical simulations, using the model described in the next section, corroborate this behavior, mirroring the experimental findings.

Distinct from biological neurons that communicate via electrical or chemical signals, thermal neuristors interact through heat. As illustrated in Fig. 1.1(a), adjacent neuristors, while electrically isolated, can transfer heat via the substrate. Each current spike produces a heat spike, which spreads to nearby neuristors, reducing their IMT threshold voltage, thereby causing an excitatory interaction. Conversely, excessive heat can cause neighboring neuristors to remain metallic and cease spiking, akin to inhibitory interactions between neurons. Further experimental insights on neuristor interactions are detailed in the Appendix.

Although we have experimentally shown that a small group of thermal neuristors can mirror the properties of biological neurons, effective computations require a vast network of interacting neurons. Before building a complex system with many neuristors, we first simulate a large array of thermal neuristors, providing a blueprint for future designs.

Theoretical model

The theoretical model builds upon the framework established in [15], with some minor adjustments. The system is built of identical neuristors, uniformly spaced in a regular 2-dimensional array. Their behavior is governed by the following equations:

$$C\frac{dV_i}{dt} = \frac{V_i^{\text{in}}}{R_i^{\text{load}}} - V_i\left(\frac{1}{R_i} + \frac{1}{R_i^{\text{load}}}\right), \tag{1.1}$$

$$C_{\text{th}}\frac{dT_i}{dt} = \frac{V_i^2}{R_i} - S_{\text{e}}(T_i - T_0) + S_{\text{c}}\nabla^2 T_i + \sigma\eta_i(t). \tag{1.2}$$

Equation (1.1) describes the current dynamics, with each variable corresponding to those shown in Fig. 1.1(a). Equation (1.2) describes the thermal dynamics, including the coupling between nearest-neighbor neuristors. Here, T_0 represents the ambient temperature, C_{th} is the thermal capacitance of each neuristor, S_{e} denotes the thermal conductance between each neuristor and the environment, and S_{c} refers to the thermal conductance between adjacent neuristors. $\eta_i(t)$ represents a Gaussian white noise variable for each neuristor that satisfies $\langle\eta_i(t)\eta_j(t')\rangle = \delta_{i,j}\delta(t-t')$, and σ is the noise strength. Detailed values of these constants are provided in the methods section. R_i is the resistance of the VO_2 film, which depends on temperature and its internal state, or memory, following the hysteresis loop depicted in Fig. 1.1(b). This memory factor is pivotal in determining the collective behavior of thermal neuristors. We utilize the hysteresis model formulated in [41], with comprehensive details available in the methods section.

Numerical results

We used the theoretical model to simulate an $L \times L$ square lattice comprised of identical thermal neuristors, whose dynamics are governed by Eqs. (1.1) and (1.2). Different input voltages V^{in} produce a diverse array of oscillation patterns, as illustrated in Fig. 1.2. At very low (9V) or high (15V) input voltages, the system remains inactive, as found in individual neuristors. With

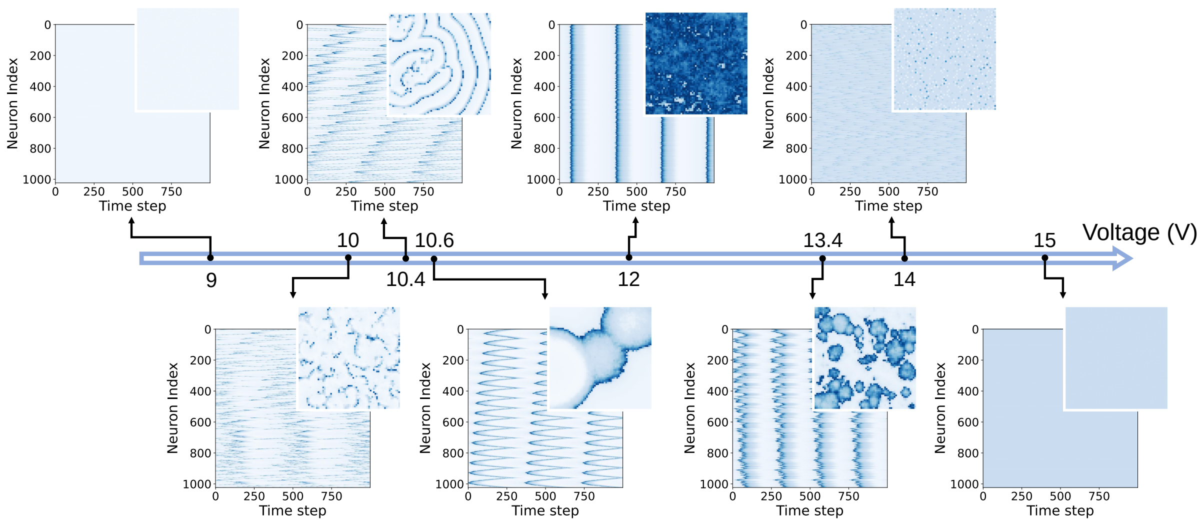


Figure 1.2. Snapshots of different oscillation patterns in a 64×64 array of thermal neuristors. In each panel, color indicates current level: white signifies no current, while shades of blue denote current spikes. The main panels show collective current-time plots for the first 1024 neuristors (concatenated from the first 16 rows), and each inset captures a specific moment in the 64×64 array. The system exhibits no activity at very low input voltages. As the voltage increases, a sequence of dynamic phases unfolds, including correlated clusters (10 V and 13.4 V), system-wide waves (10.4 V and 10.6 V), synchronized rigid states (12 V), and uncorrelated spikes (14 V), culminating again in inactivity at excessively high voltages. The thermal capacitance, C_{th}, is fixed at the experimentally estimated value. Detailed simulation parameters can be found in the methods section.

a 12 V input voltage, synchronization develops, with nearly all neuristors spiking in unison, creating a phase of rigid states. A phase transition occurs slightly below $V^{\text{in}} = 10$ V, where clusters of correlated spikes start to form, then gradually turn into system-wide activity waves (10.4 V and 10.6 V). Another phase transition occurs slightly above $V^{\text{in}} = 13.4$ V, where the synchronized rigid oscillations start to fracture into smaller clusters until the individual spikes become uncorrelated (14 V).

Analytical understanding

The emergence of a broad range of phases and long-range correlations in our system, despite only diffusive coupling existing between neurons, is a point of significant interest. Diffusive coupling is typically associated with short-range interactions, making the discovery of long-range correlations particularly intriguing.

It is well-established that long-range correlations can emerge from local interactions in various systems such as sandpiles [42], earthquake dynamics [43], forest fires [44], and neural activities [45]. These systems exhibit "avalanches"—cascades triggered when one unit's threshold breach causes successive activations—manifesting as power-law distributions of event sizes, indicative of scale-free or near scale-free behaviors.

Such spontaneously emerging long-range correlations are often described under the framework of "self-organized criticality" [42, 43]. However, this term may be misleading. "Criticality" suggests a distinct boundary, characterized by a phase above and below it, as seen in the sandpile model where an appropriately defined order parameter undergoes a second-order phase transition [46, 47]. In contrast, systems like earthquakes, while displaying power-law behaviors, do not exhibit true scale-invariance [48] and can be described as undergoing continuous phase transitions without clear critical boundaries [47].

We argue that the observed LRO in our system, similar to those in systems without genuine scale-free behaviors, is induced by memory (time non-local) effects stemming from a separation of time scales: a slow external drive contrasts sharply with fast avalanche dynamics.

In our system, we identified three distinct time scales: the metallic RC time ($\tau_{\text{met}} = R_{\text{met}}C \sim 187$ ns), the insulating RC time ($\tau_{\text{ins}} = R_{\text{ins}}C \sim 7.57\ \mu$s), and the thermal RC time ($\tau_{\text{th}} = R_{\text{th}}C_{\text{th}} = C_{\text{th}}/(S_{\text{c}} + S_{\text{e}}) \sim 241$ ns). We observe that $\tau_{\text{met}} \lessapprox \tau_{\text{th}} \ll \tau_{\text{ins}}$. As the spiking and avalanche dynamics are primarily controlled by τ_{met} and τ_{th}, and the driving dynamics by τ_{ins}, our system does exhibit an approximate separation of time scales.

This separation allows us to conceptualize the slower time scale as memory, which retains long-term information about past states and remains relatively constant within the faster time scale, capable of preserving non-local temporal correlations. As a consequence, neuristors that are spatially distant are progressively coupled, resulting in long-range spatial correlations. This concept is systematically explored in a spin glass-inspired model [6], and similar behavior is also observed in a class of dynamical systems with memory (memcomputing machines) used to solve combinatorial optimization problems [49]. In the Appendix, we provide an analytical derivation of this phenomenon using a slightly simplified version of our model.

Consequently, altering the memory strength, specifically through adjustments of the thermal time scale τ_{th} by varying C_{th} (the thermal capacitance of each neuristor), should result in changes to the oscillation patterns and the presence or absence of long-range correlations. Indeed, we find that by modifying C_{th}, we can control the rate of heat dissipation, effectively influencing the memory's response time. Additionally, in the Appendix, we present another example where increasing the ambient temperature reduces the insulating RC time, thereby diminishing memory and minimizing long-range correlations.

Avalanche size distribution

To verify the presence of LRO in our system, we analyzed the avalanche size distribution of current spikes. Here, we define an avalanche as a contiguous series of spiking events occurring in close spatial (nearest neighbor) and temporal (400 ns) proximity. The heat generated by each spiking event transfers to the neighboring neuristors, making their IMT more likely and thus triggering a cascade of spikes. Fig. 1.3(a)(b) shows examples of avalanche size distributions in which a power-law distribution is observed, indicative of LRO. The methodology for identifying these avalanches is detailed in the methods section.

We varied the input voltage and thermal capacitance, C_{th}, to generate the phase diagram depicted in Fig. 1.3(c). Here, the y-axis reflects C_{th}'s relative value against the experimentally estimated one. Similar to observations in Fig. 1.2, both a synchronized rigid state, characterized by collective neuristor firing, and a quiescent state, with no spiking activity, are found. Around the phase boundaries, a wide range of parameters leads to a power-law distribution in avalanche sizes across several orders of magnitude, confirming the existence of LRO. This is further supported in Fig. 1.3(d), where we compute avalanche sizes for each point in the parameter space and plot the absolute value of the exponent from the fitted power-law distribution. Areas without a colored box indicate an unsuccessful power-law fit, with the maximum exponent limited to 6 to remove outliers.

While we empirically observe power-law scaling in avalanche sizes, one might question if this implies criticality and scale-invariance. The numerical evidence presented here suggests otherwise. First, the power-law distributions in Fig. 1.3(a)(b) do not align with the finite-size scaling ansatz [50, 47], which predicts diminishing finite-size effects with increasing system size. Furthermore, a rescaling based on the system size should collapse all curves onto one [47, 29] for scale-invariant systems, but such an effect is notably missing in our system, contradicting finite-size scaling expectations. In Fig. 1.9 in the appendix, we present the results of attempted finite-size scaling, which clearly imply a lack of scale-invariance.

Despite the absence of criticality, can the system still perform some computing tasks effectively? Is the LRO observed in these thermal neuristor arrays even necessary for such tasks? We demonstrate in the following section that for classification, LRO, let alone criticality, is not necessary, as anticipated in [31].

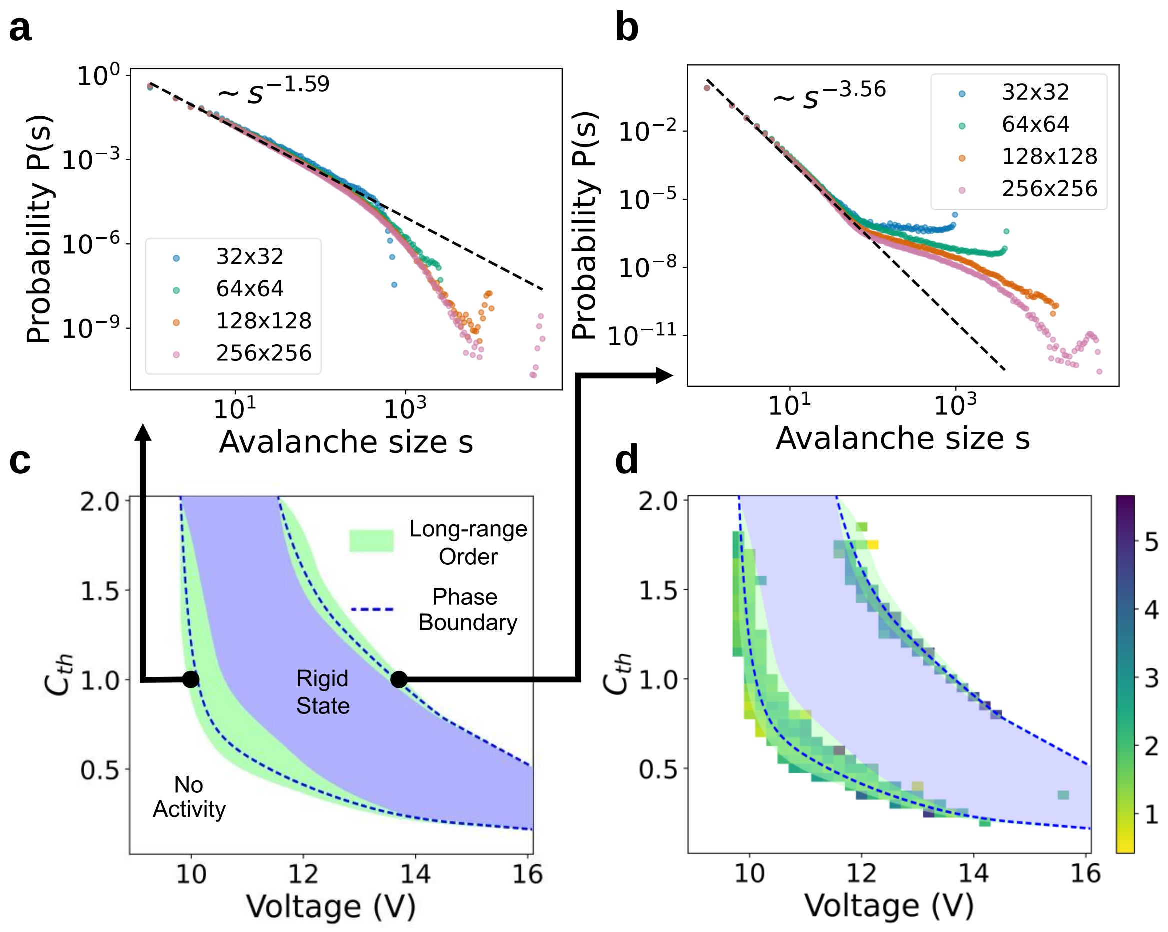


Figure 1.3. Avalanche size distributions and phase structures in 2D thermal neuristor arrays of different sizes. (a)(b) Two different avalanche size distributions at phase boundaries, with both distributions obtained at $C_{\text{th}} = 1$, but different input voltages ($V^{\text{in}} = 9.96$ V for (a), and 13.46 V for (b)). (c) Phase diagram of the thermal neuristor array, with the y-axis depicting the relative value of C_{th} compared to its experimentally estimated level. We observe synchronized rigid states with collective spiking and quiescent states with no spikes (no activity). Near the phase boundaries, a robust power-law distribution in avalanche sizes is noted across various parameters, signaling the existence of LRO. (d) Exponents of the power-law fit of the avalanche size distributions (omitting the negative signs for clarity). The phase diagram from panel (c) is superimposed for enhanced visualization. Regions lacking a colored box signify a failed power-law fit, and exponents are capped at 6 to exclude outliers.

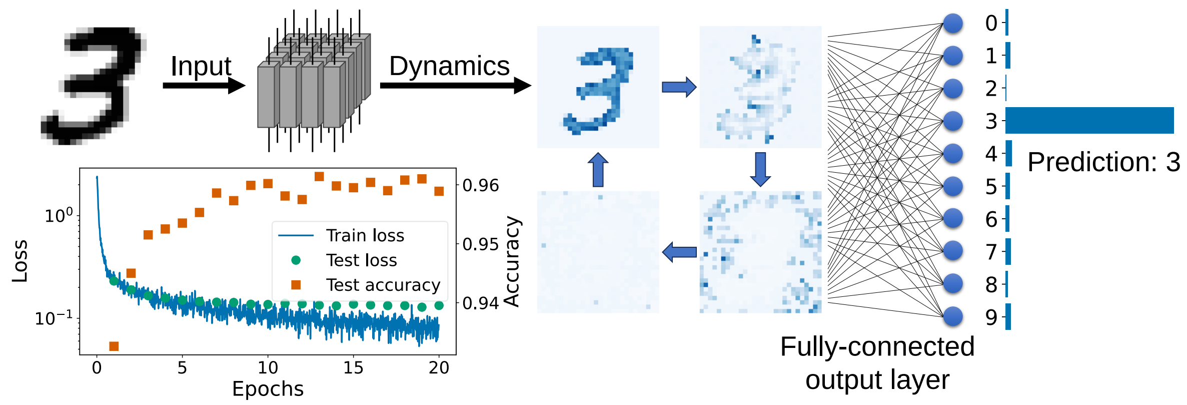


Figure 1.4. Overview of our reservoir computing implementation with a 2D thermal neuristor array, using the MNIST handwritten digit dataset [51] as a benchmark. Each image from the dataset is translated into input voltages for a 28×28 thermal neuristor array. The array's spiking dynamics are gathered as the reservoir output. A fully connected output layer, enhanced with softmax nonlinearity, is trained to classify the digit. The bottom-left panel illustrates the training process, displaying both loss and accuracy, culminating in a final test set accuracy of 96%.

Role of LRO in reservoir computing classification tasks

We apply our thermal neuristor array to reservoir computing (RC) to answer the above questions. RC differentiates itself from traditional neural network models by not requiring the reservoir – the network's core – to be trained. The reservoir is a high-dimensional, nonlinear dynamical system. It takes an input signal, $\mathbf{x}$, and transforms it into an output signal, $\mathbf{y} = f(\mathbf{x})$. A simple output function, usually a fully connected layer, is then trained to map this output signal, $\mathbf{y}$, to the desired output, $\hat{\mathbf{z}} = g(\mathbf{y})$. Training typically involves minimizing a predefined loss function between the predicted output $\hat{\mathbf{z}}$ and the actual label $\mathbf{z}$, associated with the input $\mathbf{x}$, using backpropagation and gradient descent. If the output function is linear, training can be reduced to a single linear regression.

The reservoir's transfer function f can be arbitrary, with its main role being to project the input signal $\mathbf{x}$ into a high-dimensional feature space. Since the reservoir doesn't require training, employing an experimentally designed nonlinear dynamical system like our thermal neuristor array for RC is both effective and straightforward.

As a practical demonstration, we applied RC using thermal neuristors to classify handwritten digits from the MNIST dataset [51]. Each 28×28 grayscale pixel image, representing digits 0 to 9, is converted into input voltages through a linear transformation. The system is then allowed to evolve for a specific time, during which we capture the spiking dynamics as output features from the reservoir. Subsequently, a fully connected layer with softmax activation is trained to predict the digit. This process is schematically represented in Fig. 1.4.

The output layer was trained over 20 epochs, as shown in the bottom-left panel of Fig. 1.4. The test loss stabilized after approximately 10 epochs, and due to the network's simple architecture, overfitting was avoided. Ultimately, the test set accuracy reached 96%. Further training details can be found in the methods section.

In this experiment, we treated the voltage transformation, thermal capacitance C_{th}, and noise strength σ as adjustable hyperparameters. This allowed us to check which region of phase space would produce optimal results. We found that the parameters that yielded optimal performance were an input voltage range between 10.5 V and 12.2 V, $C_{\text{th}} = 0.15$, and noise strength $\sigma = 0.2\ \mu\text{J}\cdot\text{s}^{-1/2}$. These settings placed us within the synchronized rigid phase, not the LRO one, with the input voltage variations introducing complex oscillatory patterns. In fact, choosing the parameters in the LRO phase produced worse results. We show this in the appendix. The phase diagram relating to noise strength can be found in Fig. 1.10.

To further explore the role of LRO in reservoir computing tasks, the Appendix details our efforts to eliminate LRO within the reservoir by either removing interactions between neurons altogether or by reducing memory. We quantified LRO using the avalanche size distribution under these various settings. The findings reveal that even when the reservoir operates in a rigid or non-interacting state, long-range structures inherited from the dataset are still apparent. However, no relation between LRO and computational performance was observed.

As further verification, the Appendix documents an additional experiment involving the prediction of chaotic dynamics governed by the 2D Kuramoto-Sivashinsky equations [52]. The results corroborate our primary findings: optimal performance in reservoir computing is achieved without the presence of LRO within the reservoir.

In conclusion, the spiking dynamics of the optimally performing reservoir in our study are not characterized by an LRO state. This observation aligns with the findings in [31], and challenges the well-accepted critical brain hypothesis [22] and theories suggesting that near-critical states enhance computational performance [53, 54]. However, our results do not directly contradict the critical brain hypothesis, since it is possible that long-range correlations are effectively encapsulated within the feed-forward layer. Despite this possibility, our findings highlight a crucial aspect: criticality is not a prerequisite for effective computational performance in such tasks.

1.2.3 Discussion

In this study, we have developed and experimentally validated VO_2-based thermal neuristors that exhibit brain-like features. We then formulated a theoretical model grounded in our experimental findings to facilitate large-scale numerical simulations. These simulations revealed a variety of phase structures, notably those with LRO, across a broad spectrum of parameters. Our analysis suggests that this LRO stems from the time-nonlocal response of the system and is not associated with criticality. Significantly, we demonstrate that this feature does not impair the system's computational abilities. In fact, it does not even seem to be necessary in some tasks, such as classification and time series prediction, as we have shown by using our thermal neuristor array in reservoir computing.

The thermal neuristor represents an innovative artificial neuron model, and our research offers insights into the collective dynamics of artificial neuronal activities. Our findings suggest that criticality is not a prerequisite for effective information processing in such systems. This challenges the critical brain hypothesis and its applicability to neuromorphic systems, indicating that even non-critical systems can excel in some computational tasks. We then advocate for a broader exploration of non-critical dynamical regimes that might offer computational capabilities just as powerful, if not more so, than those found at or near a critical state.

Moreover, our work highlights the potential of VO_2-based thermal neuristors in computing applications, setting the stage for more extensive experiments. Given the growing need for innovative hardware in neuromorphic computing, our VO_2-based thermal neuristor system is a promising candidate for advancing next-generation hardware in artificial intelligence.

1.2.4 Methods

Fabrication of VO_2 thermal neuristor arrays

Epitaxial VO_2 thin film growth

We employed reactive RF magnetron sputtering to deposit a 100-nm thick VO_2 film onto a (012)-oriented Al_2O_3 substrate. Initially, the substrate was placed in a high vacuum chamber, achieving a base pressure of around 10^{-7} Torr, and heated to 680°C. The chamber was then infused with pure argon at 2.2 s.c.c.m and a gas mix (20% oxygen, 80% argon) at 2.1 s.c.c.m. The sputtering plasma was initiated at a pressure of 4.2 mTorr by applying a forward power of 100 W to the target, corresponding to approximately 240 V. Post-growth, the sample holder was cooled to room temperature at a rate of 12°C/min. Specular x-ray diffraction analysis of the film revealed textured growth along the (110) crystallographic direction.

VO_2 thermal neuristor arrays fabrication

For patterning the VO_2 neuristor arrays, Electron Beam Lithography (EBL) was employed. Each neuristor, sized at 100×500 nm^2, was delineated with 500 nm gaps. The initial lithography pattern defined electrodes by depositing a 15 nm Ti layer followed by a 40 nm Au layer. To investigate thermal interactions between neuristors, a second lithography and etching step was necessary. We utilized a reactive-ion etching system to etch the exposed VO_2 films between devices, as per the second-step lithography patterns, while the negative resist shielded the electrodes and devices from etching.

Transport measurements

Transport measurements were conducted in a TTPX Lakeshore probe station equipped with a Keithley 6221 current source, a Keithley 2812 nanovoltmeter, a Tektronix Dual Channel Arbitrary Function Generator 3252C, and a Tektronix Oscilloscope MSO54. The current source and nanovoltmeter were utilized to gauge the device's resistance versus temperature. The Arbitrary Function Generator (AFG) was employed to apply either DC or pulse voltage bursts, while the oscilloscope monitored the output signals. Notably, the impedance for the channel assigned to measure voltage dynamics was set at 1 MΩ, and the channel for capturing spiking current dynamics was configured to 50 Ω.

Details of numerical simulations

Model details and constant parameters

The constants in Eqs. (1.1) and (1.2) are crucial in our simulations, as they depend on specific experimental setups. Following the approach in [15], we optimized these parameters to closely replicate the experimental results. The chosen values are summarized in Table 1.1.

The resistance of the VO$_2$ film, R, is modeled based on the hysteresis model introduced in [41], described by the equations:

$$\begin{aligned}
R(T) &= R_0 \exp\left(\frac{E_\mathrm{a}}{T}\right) F(T) + R_\mathrm{m}, \\
F(T) &= \frac{1}{2} + \frac{1}{2}\tanh\Bigg(\beta\Bigg\{\delta\frac{w}{2} + T_\mathrm{c} \\
&\quad - \left[T + T_\mathrm{pr} P\left(\frac{T - T_\mathrm{r}}{T_\mathrm{pr}}\right)\right]\Bigg\}\Bigg), \\
T_\mathrm{pr} &= \delta\frac{w}{2} + T_\mathrm{c} - \frac{1}{\beta}[2F(T_\mathrm{r}) - 1] - T_\mathrm{r}, \\
P(x) &= \frac{1}{2}(1 - \sin\gamma x)\left[1 + \tanh\left(\pi^2 - 2\pi x\right)\right].
\end{aligned} \tag{1.3}$$

Each component of Eq. (1.3) is detailed in [41]. The term T_r denotes the reversal temperature, marking the most recent transition between heating and cooling processes. Here, δ equals 1 on the heating branch and -1 on the cooling branch, with all other symbols representing constant parameters. These constants were selected in accordance with [15] to accurately reflect the experimentally observed hysteresis loop, and their values are compiled in Table 1.1.

The noise strength σ was chosen to facilitate a diverse range of phase structures. We conducted preliminary tests on the phase diagram by varying σ, with the results detailed in the

Table 1.1. Parameters utilized in the numerical simulations for Eqs. (1.1)-(1.3). R^{load} and T_0 are taken from experiment, while other parameters are optimized to align the numerical model as closely as possible to the experimentally measured data.

Param	Value	Physical meaning
C	145 pF	Capacitance
R^{load}	12.0 kΩ	Load resistance
C_{th}	49.6 pJ/K	Thermal capacitance
		Note: Figures show relative value to this
S_{e}	0.201 mW/K	Thermal conductance to environment
S_{c}	4.11 μW/K	Thermal conductance to neighbor
T_0	325 K	Ambient temperature
σ	1 μJ·s$^{-1/2}$	Noise strength
R_0	5.36 mΩ	Insulating resistance prefactor
E_{a}	5220 K	VO_2 activation energy
R_{m}	1286 Ω	Metallic resistance
w	7.19 K	Width of the hysteresis loop
T_{c}	332.8 K	Center of the hysteresis loop
β	0.253	Fitting parameter in hysteresis
γ	0.956	Fitting parameter in hysteresis

Appendix.

Numerical methods

For the numerical integration of Eqs. (1.1) and (1.2), we employed the Euler-Maruyama method [55] with a fixed time step of $dt = 10$ ns. The current-time trajectories were recorded, and current spikes were identified by locating the local maxima within these trajectories.

To analyze the avalanche size distribution, we first defined an "avalanche" as a contiguous series of spiking events occurring within a certain spatial and temporal proximity. We determined a specific window length for both spatial and temporal dimensions and then coarse-grained the spiking trajectories, categorizing each spiking event into a corresponding window. This process resulted in a $D+1$-dimensional lattice (D spatial and 1 temporal dimensions), where each lattice site denoted the number of spikes within its window. Following this, the Hoshen-Kopelman algorithm [56] was applied to identify clusters of spiking activities within the lattice. Each identified cluster was considered as one distinct avalanche, in line with our defined criteria.

The avalanche size distribution is influenced by the chosen window size. Generally, the temporal window length should be significantly longer than the duration of each spike but shorter than the interval between consecutive spikes. For all results presented in this paper, the temporal window length was set at 400 ns. In terms of spatial window length, we focused on immediate neighbors (length = 1) of each neuristor for cluster identification.

After identifying the avalanches, we computed the histogram of avalanche sizes using a logarithmic binning scheme [57], where bins are uniformly distributed on a logarithmic scale. The sizes of these bins were determined according to Scott's normal reference rule [58]. To characterize the avalanche size distributions presented in Fig. 1.3, we applied a power-law fit to each histogram, excluding the tails for more accurate modeling.

Reservoir setup

In employing thermal neuristors for reservoir computing, we consider the entire neuristor array as the reservoir. The input voltages serve as the reservoir input, and the resultant spike trains are recorded as the output.

For the MNIST dataset [51], the reservoir's parameters are detailed in the main text. To record the spike trains, we simulate the system dynamics for 10 μs, extracting spikes using the method outlined in the previous section. These spike trains are then coarse-grained with a time window of $\Delta t = 500$ ns. Each time window is assigned a binary value indicating the presence or absence of a spike. This process results in a $28 \times 28 \times 20$ binary array representing the reservoir's spike train output. This array is then flattened into a one-dimensional sequence of 15680 elements. A fully connected layer with dimensions 15680×10 is trained to map the reservoir output to the ten digit classes. At the final stage, a softmax nonlinearity is applied to transform the output layer's results into predicted probabilities. Although activation functions are not typically standard in reservoir computing tasks, we still implemented the softmax activation in conjunction with negative log-likelihood loss, as it demonstrated enhanced performance compared to mean-square-error loss without an activation function.

For training this fully connected output layer, we utilized the Adam optimizer [59] with a learning rate of 10^{-3}. The corresponding loss curve is depicted in the bottom-left panel of Fig. 1.4.

1.2.5 Appendix

Emergence of long-range order

In this section, we delve deeper into Eqs. (1.1) and (1.2) to explore the emergence of long-range order (LRO). In particular, we show that it arises due to the time non-local (memory) response of the system.

Heuristically, the relatively slowly varying temperature field gradually couples spatially separated neuristors, giving rise to long-range correlations. This intuition suggests that only terms in these equations that couple V_i and T_i or introduce thermal diffusive coupling are necessary to induce LRO. As our analysis continues, it will become more clear that these are, in fact, the only terms that are necessary to make the existence of LRO manifest. For clarity, we rewrite Eqs. (1.1) and (1.2) below, with "…" representing the terms in the original equations which are now irrelevant for the purposes of our analysis:

$$\begin{aligned} \dot{V}_i &= \frac{-V_i}{\tau_{V_i}} + \dots, \\ \dot{T}_i &= \frac{1}{\tau_\mathrm{T}}\left(\frac{V_i^2}{S_\mathrm{c} R_i} + \nabla^2 T_i\right) + \dots. \end{aligned} \tag{1.4}$$

Here, $\tau_{V_i} \equiv CR_i$ represents a site-dependent voltage timescale, and $\tau_\mathrm{T} \equiv C_\mathrm{th}/S_\mathrm{c} \approx 50\tau_\mathrm{th}$ is the intersite thermal diffusion timescale. The latter differs from τ_th by a factor of about 50, as the thermal coupling between sites is significantly weaker than with the environment. To simplify our analysis, we assume $\tau_{V_i}^{-1}$ is a linear interpolation between the inverse timescale in the insulating state (τ_ins^{-1}) and the metallic state (τ_met^{-1}), across a range within the hysteresis loop (see Fig. 1.1(b) in the main text), governed by the following equations:

$$\frac{1}{\tau_{V_i}} = \frac{\alpha(T_i)}{\tau_{\text{ins}}} + \frac{1-\alpha(T_i)}{\tau_{\text{met}}}\,, \tag{1.5}$$

$$\alpha(T_i) = \frac{1}{2}\left(1 - \left(\frac{T_i - T_{\text{c}}}{w}\right)\right). \tag{1.6}$$

Here, T_{c} and w characterize the center and width of the hysteresis loop, respectively.

This approximation, which defines $\alpha(T_i)$, is inspired by Matthiessen's rule [60], traditionally applied in scattering contexts. By focusing solely on T_i-dependent terms in $\tau_{V_i}^{-1}$, we explicitly couple the current and thermal dynamics without reference to R_i:

$$\begin{aligned} \dot{V}_i &= \frac{1}{\tau_V}\left(\frac{-T_i V_i}{T_{\text{c}}}\right) + \ldots\,, \\ \dot{T}_i &= \frac{1}{\tau_T}\left(\zeta T_i V_i^2 + \nabla^2 T_i\right) + \ldots\,. \end{aligned} \tag{1.7}$$

In (1.7), $\tau_V^{-1} \equiv \frac{T_{\text{c}}(\tau_{\text{ins}} - \tau_{\text{met}})}{2w\tau_{\text{ins}}\tau_{\text{met}}} \approx \frac{T_{\text{c}}}{2w\tau_{\text{met}}}$ (since $\tau_{\text{ins}} \gg \tau_{\text{met}}$) and $\zeta \equiv \frac{C}{2w\tau_{\text{met}}S_{\text{c}}}$. Additionally, we confirm that both $T_i/T_{\text{c}} \sim 1$ and $\zeta V_i^2 \sim 1$ with our chosen parameter values (see Table 1.1, and note that $V_i \sim 5$V during spiking dynamics). Therefore, τ_V and τ_T are representative *site-independent* timescales of the relevant current and thermal dynamics, respectively. Given our chosen parameter values, $\tau_V \sim 10$ ns and $\tau_T \sim 10$ μs, indicating a separation of timescales.

Considering these characteristic timescales τ_V and τ_T, we can treat the contributions to $\dot{T}_i$ written in (1.7) as small over intervals $\Delta t \lesssim \tau_T$. Thus, the relevant contributions to T_i over each such interval will be approximately constant. Although experiment/simulation suggests a more comprehensive thermal timescale might be shorter (the inter-spike interval is $\sim 1\mu$s), recall that we are only considering terms that couple T_i and V_i or introduce diffusive coupling. While T_i does implicitly depend on voltage coupling, thermal coupling with adjacent sites and the environment, *and* noise, contributions to T_i which only depend on the time-evolution terms kept in Eq. (1.7) will always exist. For clarity, we call these relevant contributions $\tilde{T}_i$ and $\tilde{V}_i$.

We now show that by iteratively integrating $\tilde{T}_i$ and $\tilde{V}_i$ over a prolonged period of time, the presence of long-range current couplings becomes manifest. This "time coarse-graining" approach is analogous to methods used in other complex systems studies [61, 62].

First, we evolve $\tilde{T}_i$ over intervals of length τ_T, during which $\tilde{T}_i$ is approximately constant:

$$\begin{aligned}\tilde{T}_i(\tau_T) &= T_i(0) + \int_0^{\tau_T} \dot{\tilde{T}}_i(t)dt \\ &\approx \left(1 + \zeta \overline{V_{i,1}^2} + \nabla^2\right) T_i(0) \equiv \hat{L}_0 T_i(0)\,,\end{aligned} \tag{1.8}$$

$$\begin{aligned}\tilde{T}_i(2\tau_T) &= \tilde{T}_i(\tau_T) + \int_{\tau_T}^{2\tau_T} \dot{\tilde{T}}_i(t)dt \\ &\approx \left(1 + \zeta \overline{V_{i,2}^2} + \nabla^2\right) \tilde{T}_i(\tau_T) \equiv \hat{L}_1 \hat{L}_0 T_i(0)\,.\end{aligned} \tag{1.9}$$

Above, we've defined $\overline{V_{i,l}^2}$ to be the average of V_i^2 over a time interval $[(l-1)\tau_T, l\tau_T)$ (since $\tau_V << \tau_T$, V_i^2 is not approximately constant over intervals of length τ_T, and there is no need to distinguish between V_i and $\tilde{V}_i$). We introduce the operator $\hat{L}_p$, which time-evolves $\tilde{T}_i(p\tau_T)$ to $\tilde{T}_i((p+1)\tau_T)$, for compactness. This generalizes to

$$\tilde{T}_i(l\tau_T) \approx \left(\prod_{p=0}^{l-1} \hat{L}_p\right) T_i(0)\,. \tag{1.10}$$

For $l \geq 2$, notice that $\tilde{T}(l\tau_T)$ will have terms $\sim \nabla^2 \overline{V_{i,l-1}^2}$ to $\sim \nabla^2 \ldots \nabla^2 \overline{V_{i,1}^2}$, the latter of which has $l-1$ ∇^2's. These terms implicitly depend on V_j through the diffusive coupling, where j can range from the 1st to $(l-1)$th nearest-neighbor of the ith site. Applying a similar technique for $\tilde{V}_i$,

$$\begin{aligned}\tilde{V}_i(\tau_T) &= V_i(0) + \int_0^{\tau_T} \dot{\tilde{V}}_i(t)dt \\ &\approx V_i(0) - \frac{\tau_T}{\tau_V}\frac{1}{T_\text{c}}\left(T_i(0)\overline{V_{i,1}^2}\right),\end{aligned} \tag{1.11}$$

$$\begin{aligned}\tilde{V}_i(2\tau_T) &\approx \tilde{V}_i(\tau_T) - \frac{\tau_T}{\tau_V}\frac{1}{T_\text{c}}\left(\tilde{T}_i(\tau_T)\overline{V_{i,2}^2}\right) \\ &\approx V_i(0) - \frac{\tau_T}{\tau_V}\frac{1}{T_\text{c}}\left(T_i(0)\overline{V_{i,1}^2} + \tilde{T}_i(\tau_T)\overline{V_{i,2}^2}\right).\end{aligned} \tag{1.12}$$

Again, this generalizes for $\tilde{V}_i(N\tau_T)$:

$$\tilde{V}_i(N\tau_T) \approx V_i(0) - \frac{\tau_T}{\tau_V}\frac{1}{T_\text{c}}\sum_{l=0}^{N-1}\tilde{T}_i(l\tau_T)\overline{V_{i,l+1}^2}. \tag{1.13}$$

As is suggested by Eq. (1.10), $\tilde{T}_i(l\tau_T)$ will implicitly depend on voltages $l-1$ sites away from the i^{th} lattice site. Since there is feedback between V_i and T_i, this means that $\tilde{V}_i(N\tau_T)$ will also depend on other V_j in a highly non-local manner. Additionally, after a sufficiently large time $N\tau_T$, these non-local couplings will have arbitrarily large order (the number of V_j which multiply one another). This spatial non-locality, induced by the interplay between V_i-T_i couplings and thermal site diffusion, is indicative of the LRO that we observe numerically.

Experimentally measured thermal interactions

Here, we provide some additional experimentally measured interactions between two neighboring thermal neuristors. Despite only having two neuristors, the interactions observed here offer valuable insights into the various phases of the thermal neuristor array.

In Fig. 1.5, we show the current-time curves of two adjacent neuristors under varying input voltages. Note that these devices are different from those discussed in the main text, possessing distinct threshold voltages, though their qualitative behaviors are consistent.

In panels (a) to (c) of Fig. 1.5, neuristor B is set with a 2.6 V input voltage, slightly below

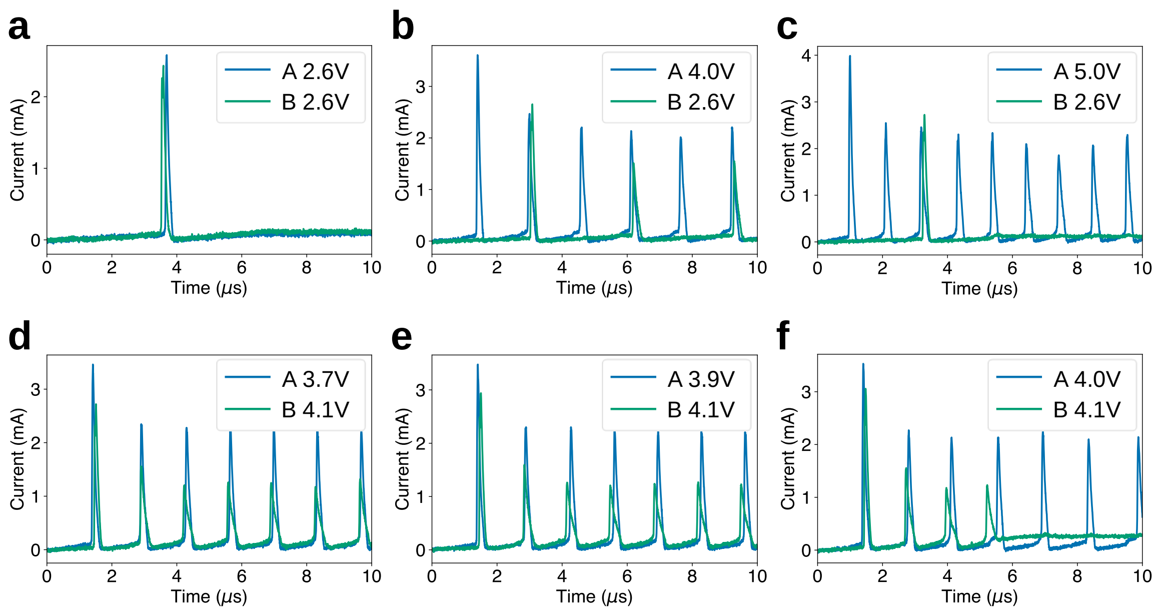


Figure 1.5. Experimental demonstration of thermal interactions between two adjacent thermal neuristors. Panels (a)-(c) feature neuristor B under a 2.6 V input, just below its lower spiking threshold. Panels (d)-(f) depict neuristor B with a 4.1 V input, marginally below the upper spiking threshold. (a) A 2.6 V input to neuristor A triggers a single synchronized spike in both neuristors. (b) Increasing A's input to 4 V establishes a stable 2:1 spiking pattern. (c) Further increasing A's input to 5 V disrupts the synchronization, resulting in only one spike in B. (d) A 3.7 V input to A leads to 1:1 stable spiking oscillations. (e) A slight increase in A's input voltage introduces phase lags in the synchronization. (f) A further increase in A's voltage breaks the synchronization, causing B to stop spiking.

the threshold for independent spiking. However, the influence of neuristor A initiates spiking activities. Panel (a) shows that a 2.6 V input to neuristor A, also below its spiking threshold, results in mutual heating and a single synchronized spike in both neuristors. Increasing A's driving voltage leads to a stable spike train in A, as seen in panel (b), which in turn induces stable oscillations in B, forming a 2:1 spiking pattern. Further increasing A's voltage disrupts this synchronization, resulting in only a single spike in B, as depicted in panel (c).

Conversely, in panels (d) to (f), neuristor B operates under a 4.1 V input, capable of sustaining stable oscillations alone but sensitive to additional stimuli. Panel (d) shows that an optimal input to A yields a 1:1 synchronized pattern. However, a slight increase in A's voltage, as shown in panel (e), introduces phase lags between A and B. Eventually, as illustrated in panel (f), the significant phase lag disrupts B's stable spiking pattern, causing it to cease spiking.

In this example, both boundaries of the spiking thresholds exhibit first-order phase transitions, where minor changes in external stimuli lead to markedly different behaviors. This underpins the phase boundaries identified in Fig. 1.3 in the main text. Slight perturbations are amplified and propagated through the lattice, resulting in the diverse phase structures and LRO we observed.

Spiking patterns and avalanche size distributions under different conditions

In this section, we provide an intuitive understanding and visualization of spiking patterns and their corresponding avalanche size distributions under various conditions. The settings used here are consistent with those in Fig. 1.2 and Fig. 1.3, and the figures presented retain the same meanings as those referenced.

First, to elucidate the phase structures depicted in Fig. 1.3, Fig. 1.6 illustrates the avalanche size distributions and corresponding spiking patterns at a point within the rigid phase ($V^{\text{in}} = 12.5$ V, $C_{\text{th}} = 1$) and at a point near the no activity phase ($V^{\text{in}} = 13$ V, $C_{\text{th}} = 1.25$).

Furthermore, to highlight the significance of thermal coupling, Fig. 1.7 compares spiking patterns with and without thermal coupling at $V^{\text{in}} = 12$ V and $C_{\text{th}} = 1$. Given that thermal

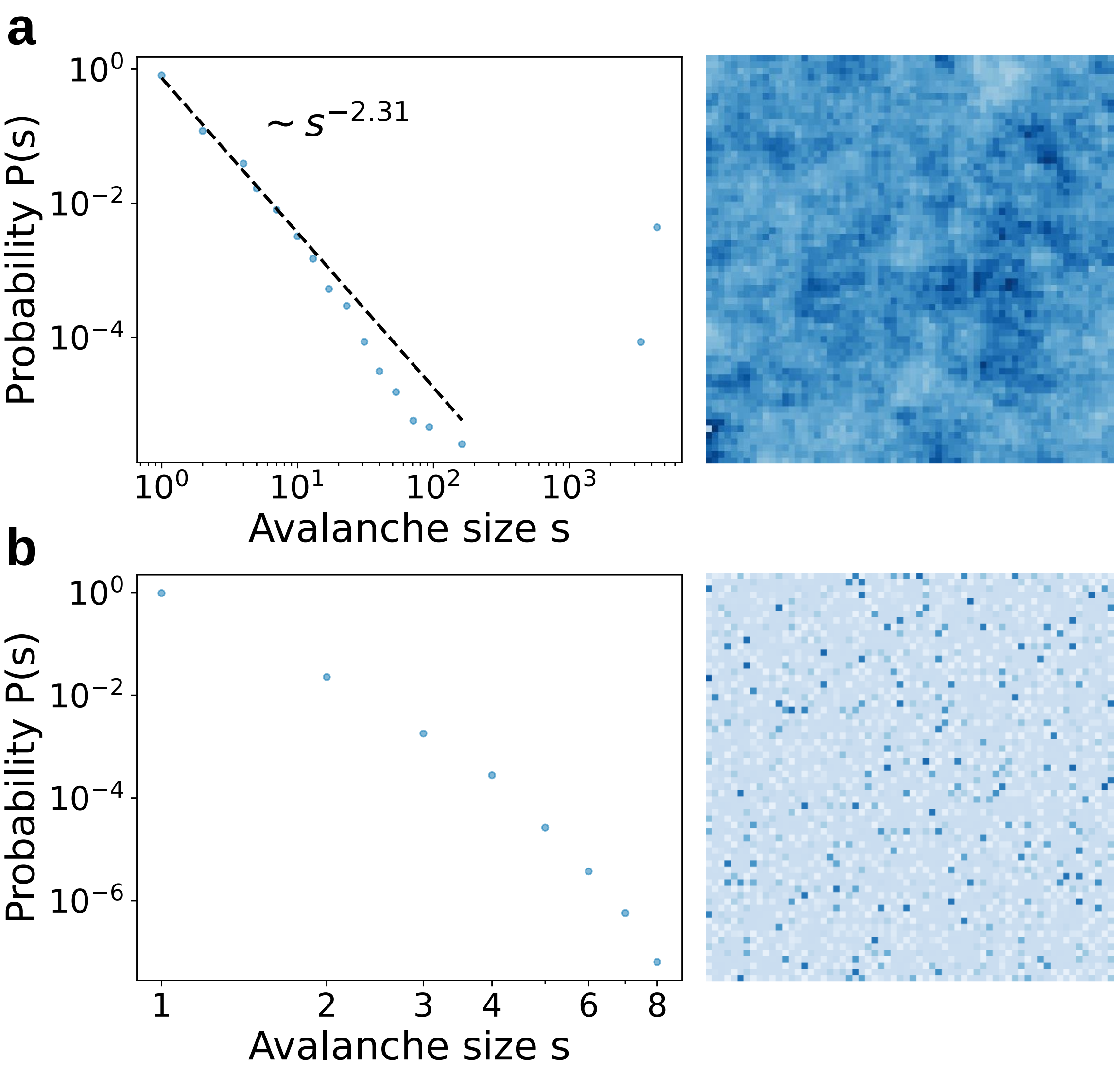


Figure 1.6. Examples of avalanche size distributions and corresponding spiking patterns for settings consistent with Fig. 1.2 and Fig. 1.3 in the main text: (a) $V^{\text{in}} = 12.5$ V, $C_{\text{th}} = 1$. This setting corresponds to the rigid phase as depicted in Fig. 1.3(c). The right panel illustrates synchronized spiking behavior across nearly all neuristors, with the avalanche size distribution showing a power-law for smaller events. Notably, a prominent peak at the right end of the distribution indicates the prevalence of system-wide avalanches. (b) $V^{\text{in}} = 13$ V, $C_{\text{th}} = 1.25$. This setting is at the boundary of the "no activity" phase. The snapshot reveals random, uncorrelated spikes, and the avalanche size distribution is predominantly characterized by single or very small events.

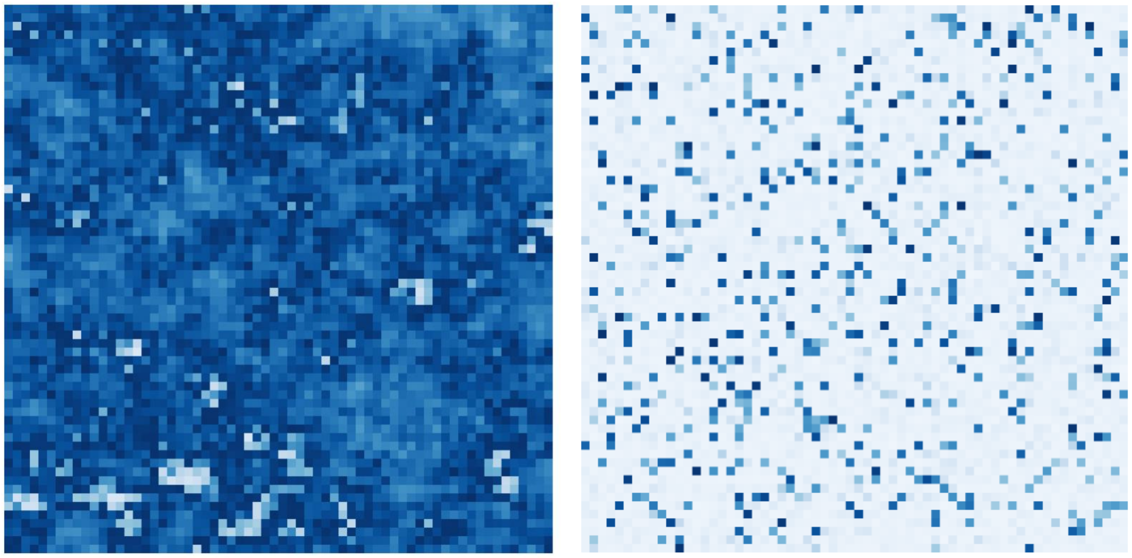


Figure 1.7. Comparison of spiking patterns with (left) and without (right) thermal coupling at $V^{\text{in}} = 12$ V, $C_{\text{th}} = 1$. With thermal coupling, the setting induces a rigid state where almost all neuristors spike simultaneously, demonstrating highly correlated behavior. Conversely, without thermal coupling, the neuristors spike independently, resulting in uncorrelated, isolated spiking patterns.

coupling is the only means through which neuristors exchange information, the absence of thermal coupling naturally results in random, uncorrelated spikes.

Additionally, in Fig. 1.8, we compare the spiking patterns and avalanche size distributions under periodic and open boundary conditions at $V^{\text{in}} = 9.96$ V and $C_{\text{th}} = 1$. This comparison shows that the type of boundary condition has minimal impact on the spiking behaviors, demonstrating the robustness of the spiking dynamics against boundary effects.

Finite-size scaling

In the main text (Fig. 1.3), we observed power-law distributions of avalanches across various system sizes, with closely matched power-law exponents at specific parameter points.

According to finite-size scaling theory [50], at criticality, avalanche size distributions for different system sizes should conform to a common scaling rule:

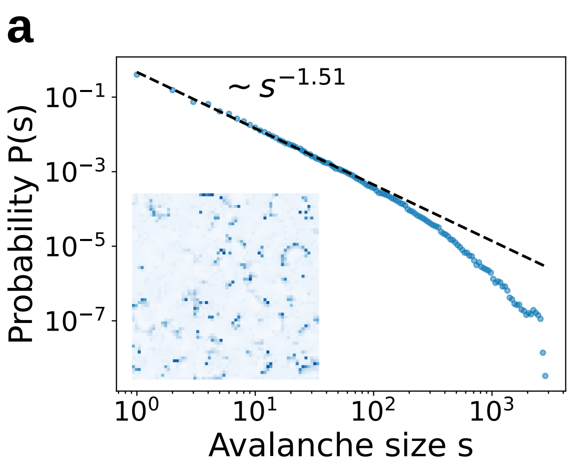


Figure 1.8. Comparison of spiking patterns and avalanche size distributions under (a) periodic and (b) open boundary conditions at $V^{\text{in}} = 9.96$ V and $C_{\text{th}} = 1$, both settings within the LRO phase. The figure demonstrates that the type of boundary condition has minimal impact on the spiking behaviors and avalanche size distributions, indicating robustness of the spiking dynamics against boundary effects.

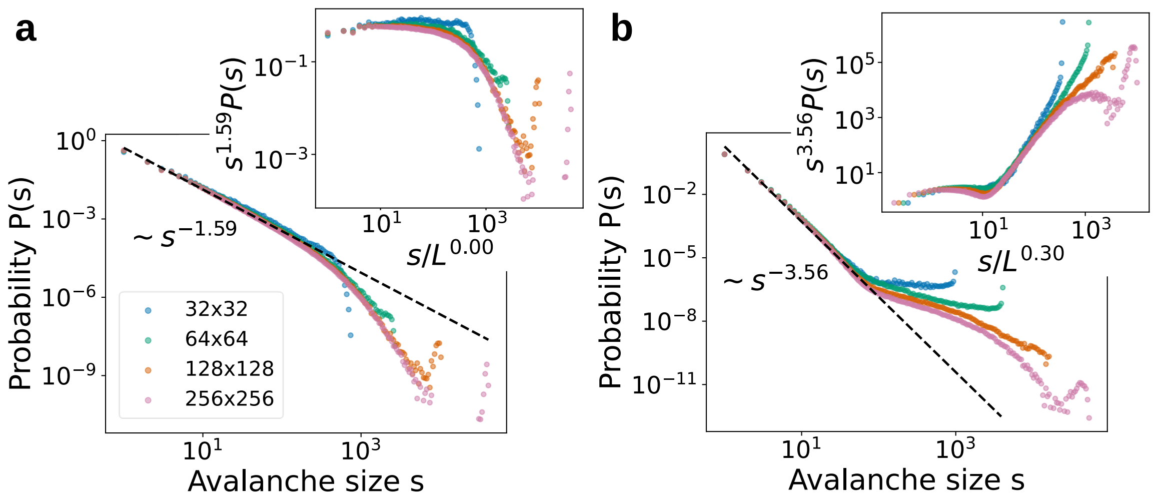


Figure 1.9. Rescaling of the avalanche size distributions from Fig. 1.3 in the main text, based on the finite-size scaling ansatz, Eq. (1.14). In the inset, $L = \sqrt{N}$ is the length of the lattice. Criticality should result in an overlap of curves from different system sizes, demonstrating scale-invariance. However, this scale-invariance is notably absent in our system, suggesting a lack of criticality.

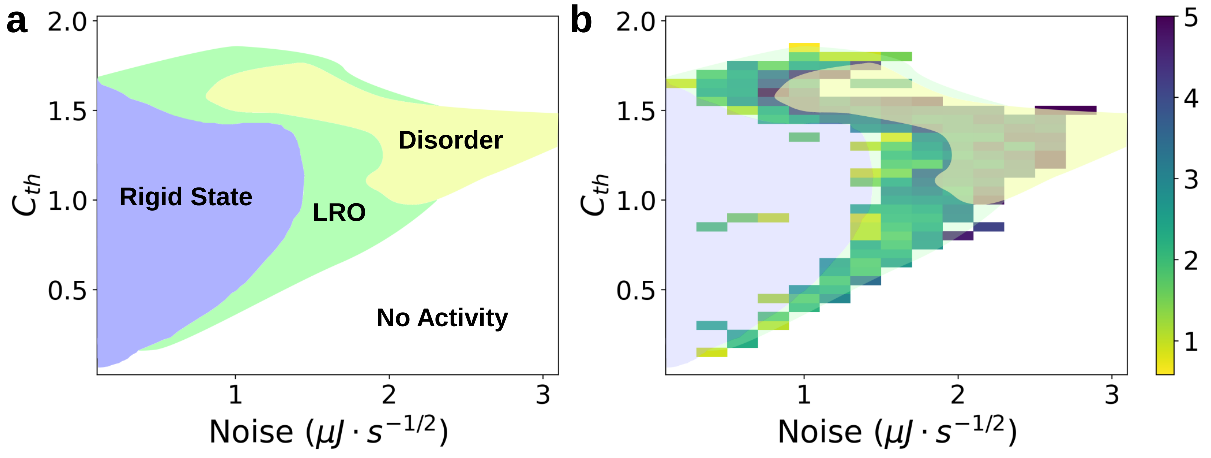


Figure 1.10. Phase diagram of a 32×32 thermal neuristor array under varying noise strength and thermal capacitance, with a fixed input voltage of 12 V. As noise strength approaches zero, a rigid state prevails, characterized by synchronized firing of all neuristors. With increasing noise, a phase of LRO emerges, eventually giving way to a disordered phase at higher noise levels, characterized by uncorrelated spiking. (b) The slope of the avalanche size distributions at each parameter point, capped at 5 and excluding the negative sign for clarity. A lack of color in a box signals an unsuccessful power-law fit. The phase diagram from panel (a) is overlaid for context.

$$P(s,N) \sim s^{\alpha} \exp\left(-s/N^{\beta}\right), \tag{1.14}$$

where s is the avalanche size, N is the system size, α is the critical exponent of the avalanche size distribution, and β is the cutoff exponent.As emphasized in several studies, such as [29, 47], the rescaling of avalanche size distributions according to Eq. (1.14) should result in all distributions collapsing onto a single curve at criticality, characterized by a near-perfect overlap of all rescaled curves. Indeed, this phenomenon is typical in scale-free systems, with examples of near-perfect rescaling plots documented in references like [29, 47, 6].

Fig. 1.9 attempts this rescaling for the distributions presented in Fig. 1.3 from the main text. Here, we optimized the exponent β to align the four curves corresponding to different system sizes. However, our findings show a clear departure from Eq. (1.14), suggesting the absence of scale-invariance and criticality.

Effect of noise strength

In the main text, we set the noise strength σ in Eq. (1.2) to 1 μJ·s$^{-1/2}$. This section examines how varying the noise strength influences the phase structures of the system.

Fig. 1.10(a) presents the phase diagram for a 32×32 thermal neuristor array, exploring the interplay between thermal capacitance and noise strength. Here, the input voltage is consistently held at 12 V, while all other parameters remain as described in the main text. Fig. 1.10(b) plots the slopes of the avalanche size distributions for each set of parameters, paralleling the approach of Fig. 1.3 from the main text.

It is evident that noise strength significantly impacts the phase structures in this model. Near zero noise strength, when all neuristors are identical and begin with the same initial conditions, a rigid phase emerges across most parameters. This phase is characterized by synchronized spiking in all neuristors. Increasing the noise strength reveals a phase of LRO, situated between the rigid phase and the inactive phase. As the noise strength further escalates, the rigid phase vanishes, giving way to a disordered phase. In this phase, correlations between neuristors fade exponentially, resulting in isolated, uncorrelated spikes. In our main text simulations, we chose a noise strength of 1 μJ·s$^{-1/2}$, leading to a wide variety of oscillation patterns.

Hyperparameter selection in reservoir computing

Hyperparameter tuning is a critical aspect of machine learning algorithms. In our study, the parameters of the thermal neuristor array serve as hyperparameters for the reservoir computing algorithm.

As detailed in the main text, we use a 28×28 array of thermal neuristors for handwritten digit recognition on the MNIST dataset. While Bayesian optimization algorithms can efficiently select hyperparameters [63], we opt for a grid search to gain deeper insight into the reservoir's physics. Fig. 1.11 illustrates the accuracy of predictions after training for one epoch. In panel (a), we adjusted the thermal capacitance C_{th} and the input voltage for black pixels ($V_{\max}$), while keeping the input voltage for white background pixels ($V_{\min}$) at 10.5 V, and the noise strength σ

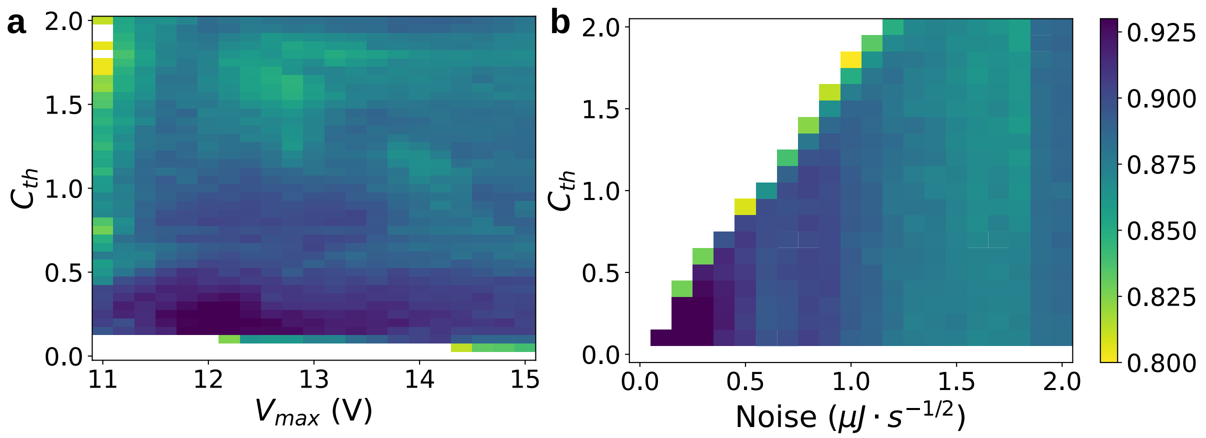


Figure 1.11. Classification accuracy on the MNIST dataset after one epoch of training with various parameters. Areas in white indicate regions lacking spiking dynamics. (a) Varying the maximum input voltage $V_{\max}$ and thermal capacitance C_{th}, while fixing the minimum input voltage $V_{\min}$ to be 10.5 V and the noise strength σ to be 0.2 μJ$\cdot$s$^{-1/2}$. Optimal performance is observed at the point $C_{\text{th}} = 0.15$, $V_{\max} = 12.2$ V. (b) Fixing $V_{\min} = 10.5$ V and $V_{\max} = 12$ V. Optimal performance is observed at the point $\sigma = 0.2$ μJ$\cdot$s$^{-1/2}$ and $C_{\text{th}} = 0.1$.

at 0.2 μJ$\cdot$s$^{-1/2}$. Similarly, in panel (b), we fixed $V_{\min} = 10.5$ V and $V_{\max} = 12$ V, and varied σ and C_{th}.

In both panels, most parameter combinations yield reasonable performance, provided some level of spiking dynamics is present in the system. The best performance is achieved near the phase transition boundary between no activity and the rigid state, with a small thermal capacitance ($C_{\text{th}} \sim 0.15$), moderate input voltage ($V_{\max} \sim 12$ V), and small noise strength ($\sigma \sim$ 0.2 μJ$\cdot$s$^{-1/2}$). A lower thermal capacitance facilitates a quicker response in the thermal neuristor array, while a smaller noise makes the system more predictable, enhancing the classification task.

Further analysis of LRO in reservoir computing

In the preceding section, we determined that the optimal parameters place us within the synchronized rigid phase, as depicted in Fig. 1.10. However, this configuration does not necessarily imply that the reservoir operates in a rigid state, given that variations in the input data may introduce additional structures and correlations within the reservoir.

To rigorously test this hypothesis and explicitly quantify LRO within the reservoir, we

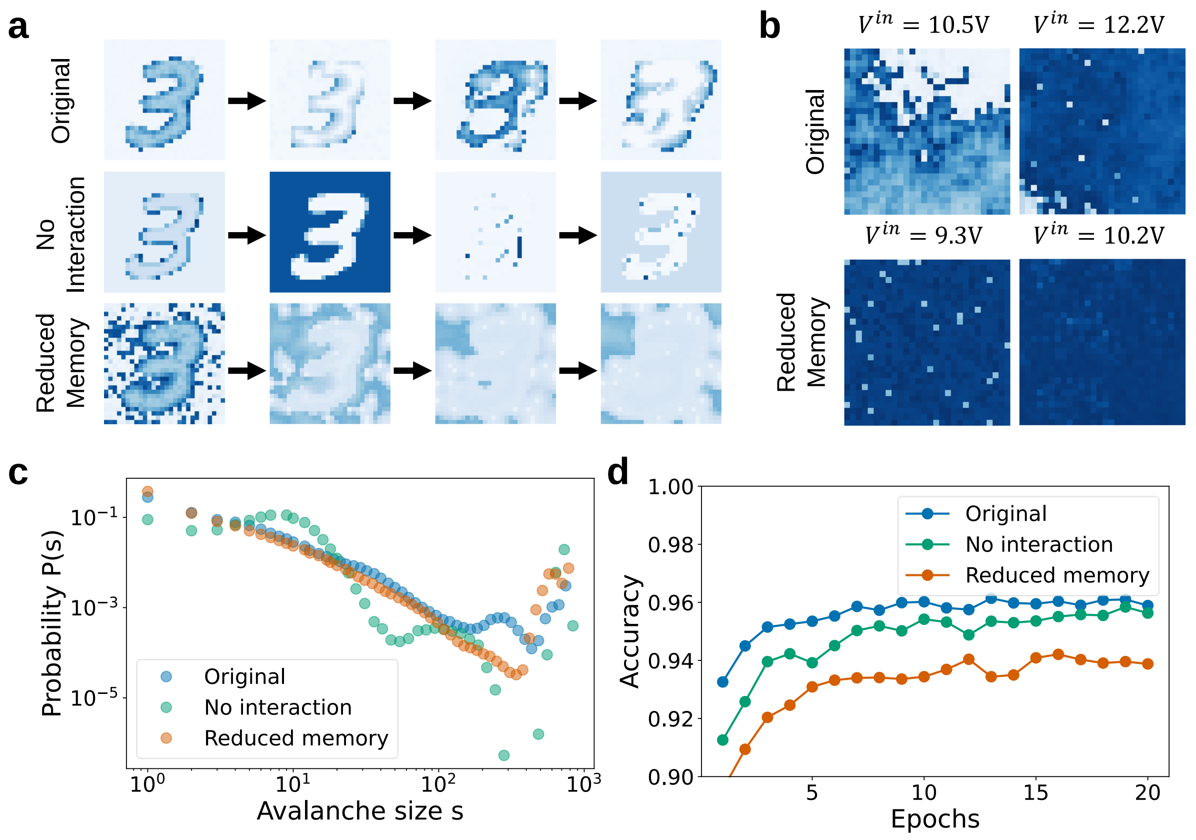


Figure 1.12. Comparison of MNIST handwritten digit classification using reservoir computing under three different settings. **Original:** $V_{\min} = 10.50\text{V}$, $V_{\max} = 12.20\text{V}$, $C_{\text{th}} = 0.15$, $\sigma = 0.2\mu\text{J}\cdot\text{s}^{-1/2}$. **No interaction:** Thermal interactions between neighboring neuristors are removed. $V_{\min} = 11.99\text{V}$, $V_{\max} = 13.37\text{V}$, $C_{\text{th}} = 0.18$, $\sigma = 0$, and $S_c = 0$, replacing the original $4.11\mu\text{W/K}$. **Reduced memory:** The slower time scale is minimized by raising the ambient temperature. $V_{\min} = 9.30\text{V}$, $V_{\max} = 10.16\text{V}$, $C_{\text{th}} = 0.15$, $\sigma = 0.2\mu\text{J}\cdot\text{s}^{-1/2}$, and $T_0 = 330\text{K}$, up from 325K. Each setting is optimized to achieve the highest possible classification accuracy for the conditions specified. **(a) Snapshots of Reservoir Dynamics:** This panel showcases reservoir dynamics under three distinct settings, with each snapshot taken approximately 4μs apart. **(b) Dynamics under Uniform Input:** This panel displays reservoir dynamics under uniform input voltages set to $V_{\min}$ and $V_{\max}$, confirming that the reservoir operates within a rigid state. **(c) Avalanche Size Distribution:** Avalanche size distribution of current spikes using the MNIST dataset's 60,000 training images as input. **(d) Classification Accuracy:** Classification accuracy on the MNIST test set after 20 epochs of training. The original configuration achieves the highest accuracy at 96.1%. Remarkably, the no-interaction scenario still performs well, achieving 95.8% accuracy without any internal information transfer within the reservoir. With reduced memory, where the slower time scale is lessened, long-range order within the system is substantially reduced, leading to a lower accuracy of 94.2%. These results further illustrate that there is no straightforward correlation between LRO and computational performance.

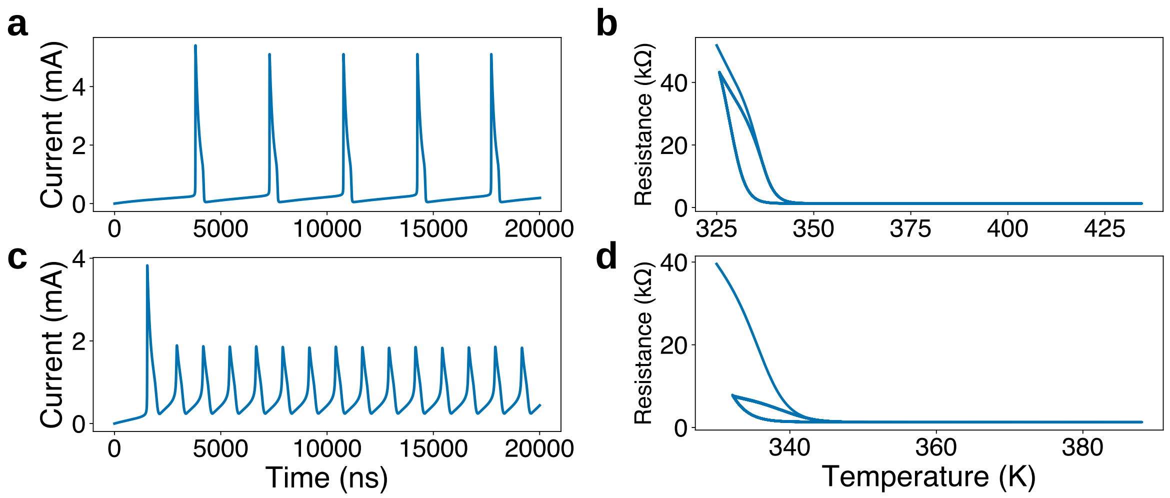


Figure 1.13. Demonstration of how increasing ambient temperature, T_0, reduces the slower time-scale. (a) Spiking dynamics and (b) the resistance-temperature curve of a single neuristor at input voltage $V^{\text{in}} = 10.2\text{V}$, $C_{\text{th}} = 0.15$, and $T_0 = 325\text{K}$. The dynamics are stable, with the VO_2 device consistently reverting to its insulating state at $R_{\text{ins}} = 43\text{k}\Omega$ after each spike. (c) Spiking dynamics and (d) the resistance-temperature curve for the same neuristor under identical settings but with the ambient temperature increased to $T_0 = 330\text{K}$. The elevated temperature results in the VO_2 device reverting to a much lower resistance of $7\text{k}\Omega$ after each spike, significantly reducing the insulating RC time, τ_{ins}, to approximately 1μs. This adjustment leads to smaller spiking amplitudes, a higher frequency, and more unstable spiking dynamics. Further increases in temperature or minor external perturbations can cause the VO_2 device to remain in its metallic state, ceasing to spike.

computed the avalanche size distribution when the MNIST dataset served as the input. The experimental setup adheres to the methodology outlined in Sec. 1.2.2 of the main text, employing identical parameters. We recorded the current spikes to calculate the avalanche size distribution using the approach detailed in Sec. 1.2.4.

The avalanche size distribution, encompassing all 60,000 training images from the MNIST dataset, is depicted in Fig. 1.12(c). Although the system is in a rigid state, the distribution reveals long-range structures: a power-law distribution is evident for smaller clusters up to $s \sim 10^2$, followed by two prominent bumps, which correspond to avalanches that span across the entire system and those limited to the black pixels comprising the digits.

Superficially, the distinct structure observed within the MNIST dataset's avalanche size distribution may seem to suggest that LRO plays a crucial role in the classification of MNIST images. Instead, we now demonstrate that the emergence of these long-range structures originates from the dataset itself, rather than the reservoir, and that long-range correlations are not essential for effective performance in this task.

First, we eliminated all interactions within the reservoir by disconnecting the thermal coupling between adjacent neuristors. Consequently, each thermal neuristor responds solely to its designated input pixel, devoid of any contextual awareness. After fine-tuning the hyperparameters via Bayesian optimization [63] to maximize classification accuracy, we observed the resultant reservoir dynamics, as depicted in Fig. 1.12(a). The corresponding avalanche size distribution is illustrated in Fig.1.12(c). In this configuration, the distribution reveals more defined structures: system-wide avalanches represent the white background, while smaller avalanches delineate the fractures within the digits. Given the absence of inter-neuristor interactions, it is evident that these structures are derived *solely* from the dataset. When compared to the original distribution involving interactions, it becomes apparent that internal interactions within the reservoir tend to smooth out smaller avalanches while retaining the principal structures inherent to the dataset.

Upon training the output layer, the classification accuracy achieved on the MNIST dataset's test set is presented in Fig. 1.12(d). Remarkably, even in the absence of interactions within the reservoir, we attained an accuracy of 95.8%, which is nearly equivalent to the performance under the optimized setup that included interactions. This outcome substantiates the assertion that LRO within the reservoir is not essential for achieving high computational effectiveness.

In another experiment, we aimed to minimize LRO in the neuristor array as much as possible. Previous findings, as discussed in the main text, suggest that LRO is influenced by the separation of time scales. To address this, we attempted to reduce the memory within the system and eliminate the slower time scale by decreasing the resistance in the insulating state of the VO_2 device. This adjustment involved increasing the ambient temperature, T_0, from 325 K to 330 K. Fig. 1.13 illustrates the impact of this temperature increase on a single neuristor: the VO_2 device begins closer to the metallic state with reduced resistance, and after each spiking event, it resets to a lower resistance state. Further increase of T_0 ultimately leads the neuristors to remain in the metallic state, thereby ceasing to spike.

As illustrated in Fig.1.13(d), the maximum resistance in the insulating state during spiking, denoted as R_{ins}, is approximately 7kΩ. This value results in an insulating time constant, $\tau_{\text{ins}} = R_{\text{ins}}C \sim 1\mu$s, significantly reduced from $\tau_{\text{ins}} = 7.57\mu$s mentioned in the main text. Consequently, both memory and LRO within the system are diminished. Furthermore, as shown in Fig.1.13(c), the spiking frequency increases, while the amplitudes of the spikes decrease. This alteration in dynamics leads to a spiking pattern that is more susceptible to disruption by external perturbations.

Once again, we optimized the hyperparameters using Bayesian optimization [63] to achieve the best classification accuracy on the MNIST dataset. Fig.1.12(a) captures snapshots of the reservoir dynamics. These visuals demonstrate how the digit patterns rapidly blur and fade, indicating a rapid loss of memory within the system. Fig.1.12(b) shows that synchronized system-wide spikes from uniform input confirm the reservoir's operation in a rigid state under these conditions. However, when using the MNIST dataset as input, the avalanche size distribution, depicted in Fig.1.12(c), shows a power-law distribution for smaller avalanches and system-wide activities, mirroring the original setup. As noted previously, this pattern mostly originates from structural features inherent in the dataset, particularly since the modified reservoir struggles to maintain LRO. The classification accuracy recorded on the test set, as depicted in Fig.1.12(d), ultimately reaches 94.2%. Although the classification task is executed effectively, the neuristors' unstable spiking behavior remains less than ideal for achieving higher accuracy.

From the two experiments described above, it is clear that LRO may not necessarily emerge from the reservoir itself, rather it can be inherited from the dataset. Across all three experiments, there was no discernible correlation between LRO and computational performance. While the original setup, which included the possibility of LRO, performed the best, the non-interacting setup yielded nearly comparable results without any long-range interactions. Meanwhile, the experiment with reduced memory, although showing the lowest performance, still exhibited similar long-range structures as the original setup. Therefore, we conclude that LRO is not essential for effective computational performance in these scenarios.

Predicting chaotic dynamics with reservoir computing

In this section, we describe an experiment designed to predict chaotic dynamics governed by the 2D Kuramoto-Sivashinsky (KS) equations [52] using a reservoir computing framework implemented with a thermal neuristor array.

The 2D KS equation is expressed as:

$$\frac{\partial u}{\partial t} + \frac{1}{2}|\nabla u|^2 + \Delta u + \Delta^2 u = 0 \tag{1.15}$$

where the boundary conditions are spatially periodic. This equation has been extensively studied [52] and is known for its chaotic behavior, which poses significant challenges in predicting long-term dynamics.

We discretized Eq. (1.15) on a 16×16 square lattice with a unit bond length and simulated the dynamics numerically using a 4th order Runge-Kutta method with a time step of $\Delta t = 0.05$.

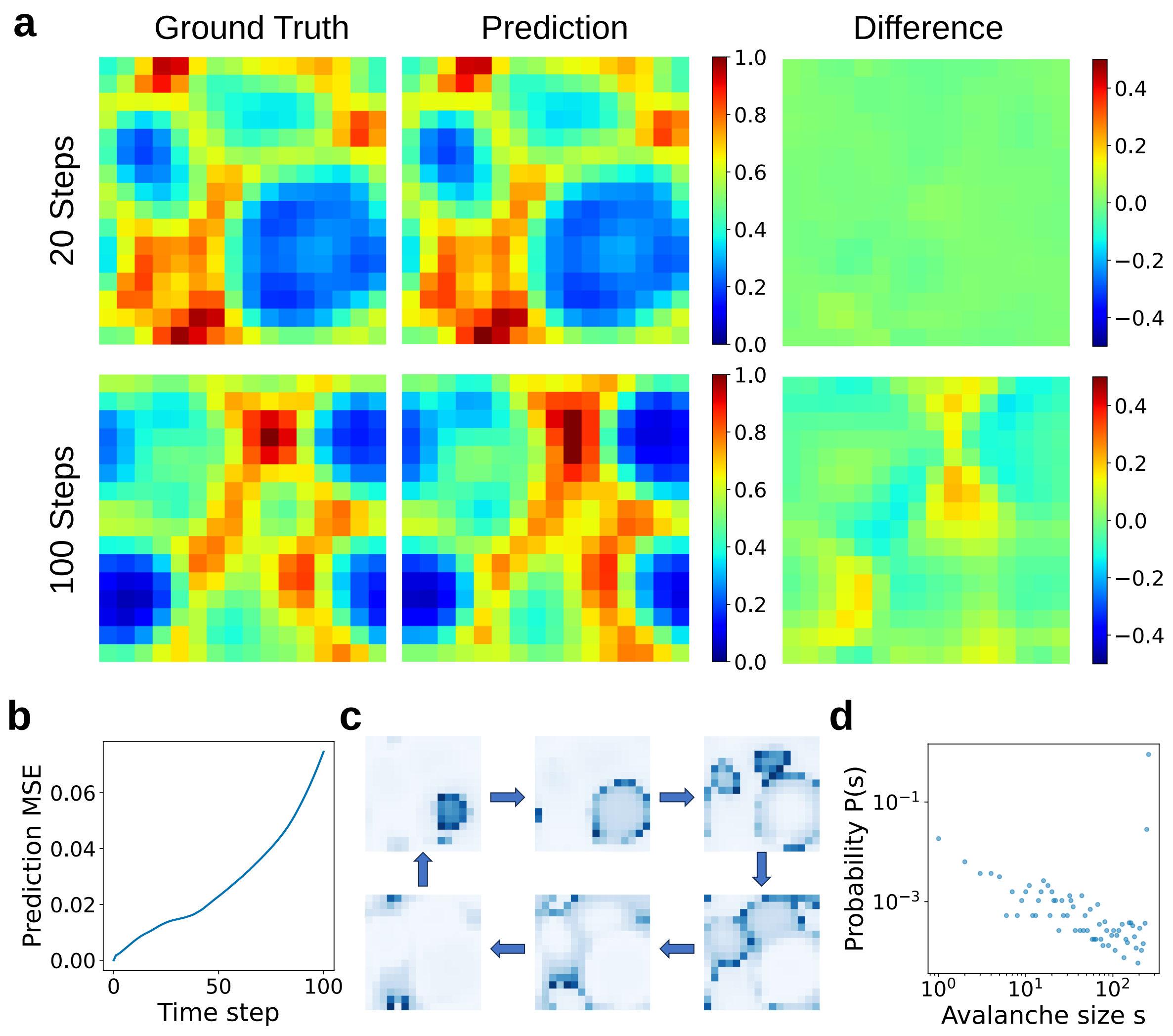


Figure 1.14. Predicting the chaotic dynamics governed by the 2D Kuramoto-Sivashinsky (KS) equations using reservoir computing. (a) Comparison of the ground truths and predictions, after 20 and 100 time steps, with each time step corresponding to $\Delta t = 0.05$ (arbitrary unit). The two rightmost panels illustrate the differences between the predictions and the ground truths, highlighting the model's accuracy over time. (b) The mean-square-error (MSE) of the prediction as a function of the number of time steps predicted. Although the prediction MSE gradually increases with more extended simulation of the dynamics, the predictions remain reasonably accurate for up to 100 time steps, equivalent to $\Delta t = 5$ (arbitrary unit). (c) An example of reservoir dynamics is presented, showing patterns that loosely resemble the evolution observed in the KS equation dynamics. (d) Avalanche size distribution with KS dynamics as input to the reservoir. A prominent peak at the right end of the distribution highlights the predominance of system-wide avalanches, confirming that the reservoir operates in a rigid state.

The simulation began from random initial conditions and, after allowing for the decay of initial transients, the dynamics of the u field were recorded as training data.

To predict the chaotic dynamics described by the KS equation, we utilized reservoir computing with an array of thermal neuristors. This setup is similar to that used in the MNIST classification experiment detailed in the main text, allowing for a direct comparison of the techniques' effectiveness across different types of machine learning tasks.

In our approach, noting that the mean value of the u field is nonzero and decreases over time, yet the KS equation (1.15) depends only on the gradient and not the magnitude of u [52], we performed a preprocessing step. This involved subtracting the mean from the u field and normalizing it between 0 and 1 to yield the transformed field, $\tilde{u}$. Snapshots of the $\tilde{u}$ field are shown in Fig. 1.14(a). Each $\tilde{u}(x,y)$ value is then linearly transformed into an input voltage for the corresponding thermal neuristor, with the spiking dynamics of the neuristor array serving as the output feature from the reservoir. An output layer is subsequently trained to predict the incremental change $u(x,y,t+\Delta t) - u(x,y,t)$.

Reservoir hyperparameters were optimized using the hyperopt library [63]. The optimized parameters included $C_{\text{th}} = 1.073$, noise strength $\sigma = 0$, and the transformation formula $V^{\text{in}} = (11.83 - 0.48\tilde{u})\text{V}$. Notably, the optimization yielded zero noise, underscoring that noise is detrimental in this reservoir computing task.

To enhance performance on this dataset, we implemented several modifications to the reservoir:

1. The output from the reservoir consists of the magnitudes of the current spikes, rather than merely indicating the presence of a spike.

2. Reflecting the local interaction nature of Eq. (1.15), we replaced a fully-connected output layer with a convolution-like layer, where each output neuron is connected to the 5×5 nearest neighbors in the reservoir.

3. We eliminated the softmax nonlinearity in the output layer and employed linear regression

for training.

This modified procedure predicts $u(x,y,t+\Delta t)$ from $u(x,y,t)$ at time t. Long-time predictions are iteratively performed by using the predicted u field as the input for the subsequent prediction step. The prediction results after 20 and 100 steps with $\Delta t = 0.05$ are illustrated in Fig. 1.14(a), and the mean-square-error of the prediction is plotted in Fig. 1.14(b), demonstrating good agreement with the ground truth. An example of the dynamics within the reservoir is depicted in Fig. 1.14(c), which loosely resembles the patterns in the $\tilde{u}$ field.

To verify the presence of LRO, we calculated the avalanche size distribution within the reservoir using the transformed $\tilde{u}$ field from the KS dynamics as input. The results, depicted in Fig. 1.14(d), are characterized predominantly by system-wide avalanches, indicating that the system maintains a rigid state even with non-uniform inputs. Moreover, the implementation of a locally connected output layer demonstrates that local information alone is sufficient for predicting dynamics, further reinforcing the notion that LRO is not necessary for achieving optimal computational performance.

This section, in full, is a reprint of the material as it appears in Nature Communications [1]. Yuan-Hang Zhang, Chesson Sipling, Erbin Qiu, Ivan K Schuller, Massimiliano Di Ventra, 2024. The dissertation author was the primary investigator and author of this paper.

Chapter 2

The MemComputing paradigm

2.1 Introduction to MemComputing

The term “MemComputing” was coined by Prof. Massimiliano Di Ventra in 2013 [64], describing physical systems that compute in and with memory in a massively parallel manner. Here, the prefix ”Mem” signifies ”memory,” not in the traditional sense of mere storage, but as temporal interactions—specifically, the system’s capacity to retain information about its past dynamics.

MemComputing bears a resemblance to brain functions: each memory unit (analogous to a neuron in the brain) not only receives information from its neighbors but also processes and transmits the outcome back to them. This approach starkly contrasts with the traditional von Neumann architecture, which centralizes processing in the CPU, separate from storage units. In MemComputing, each memory unit acts as an independent processor, fostering a massively parallel computing paradigm.

As discussed in the previous chapter, the ability of memory to induce spatial long-range order is especially beneficial for tackling optimization problems. This capability allows for approaches that transcend local moves, facilitating escapes from local minima— a principle that is integral to MemComputing.

In our discussions, we focus on digital MemComputing machines (DMMs), which handle inputs and outputs as digital signals (finite strings of symbols like 0s and 1s). However, the transition function of a DMM typically involves a continuous dynamical system, where equilibrium points reflect logically consistent configurations of input and output variables. This design renders DMMs terminal-agnostic, treating inputs and outputs equivalently—a feature particularly valuable for specific problem types. For instance, a multiplication circuit built for DMMs can compute the product of two integers and, intriguingly, can be reversed to perform integer factorization using the same setup. Research shows [65, 66] that DMM dynamics can be characterized as instantons in the extended phase space, with the number of instantonic jumps to equilibrium being proportional to the system's dimension, thus providing an efficient method for various combinatorial optimization problems.

As dynamical systems, ideal DMMs operate in continuous time, necessitating the physical construction of the corresponding system. Alternatively, DMMs can be simulated numerically on modern computers; however, this approach requires discretizing time, which breaks time-translational symmetry and disrupts the instantonic nature of DMM dynamics. Upcoming sections will detail how numerical noise impacts the simulation of DMMs, drawing theoretical parallels to directed percolation. We also explore hardware designs for DMMs that overcome the challenges posed by numerical noise, thereby offering new avenues for addressing increasingly complex computational problems.

2.2 Directed percolation and numerical stability of simulations of digital MemComputing machines

Digital memcomputing machines (DMMs) are a novel, non-Turing class of machines designed to solve combinatorial optimization problems. They can be physically realized with continuous-time, non-quantum dynamical systems with memory (*time non-locality*), whose ordinary differential equations (ODEs) can be numerically integrated on modern computers. Solutions of many hard problems have been reported by numerically integrating the ODEs of DMMs, showing substantial advantages over state-of-the-art solvers. To investigate the reasons behind the robustness and effectiveness of this method, we employ three explicit integration schemes (forward Euler, trapezoid and Runge-Kutta 4th order) with a constant time step, to solve 3-SAT instances with planted solutions. We show that, *(i)* even if most of the trajectories in the phase space are destroyed by numerical noise, the solution can still be achieved; *(ii)* the forward Euler method, although having the largest numerical error, solves the instances in the least amount of function evaluations; and *(iii)* when increasing the integration time step, the system undergoes a "solvable-unsolvable transition" at a critical threshold, which needs to decay at most as a *power law* with the problem size, to control the numerical errors. To explain these results, we model the dynamical behavior of DMMs as directed percolation of the state trajectory in the phase space in the presence of noise. This viewpoint clarifies the reasons behind their numerical robustness and provides an analytical understanding of the solvable-unsolvable transition. These results land further support to the usefulness of DMMs in the solution of hard combinatorial optimization problems.

2.2.1 Introduction

MemComputing is a recently proposed (non-Turing) computing paradigm in which *time non-locality* (memory) accomplishes both tasks of information processing and storage [64]. Its *digital* (hence scalable) version (digital memcomputing machines or DMMs) has been introduced specifically to solve combinatorial optimization problems [67, 68].

The basic idea behind DMMs is that, instead of solving a combinatorial optimization problem in the traditional algorithmic way, one maps it into a specifically designed dynamical system with memory, so that the only equilibrium points of that system represent the solutions of the given problem. This is accomplished by writing the problem in Boolean (or algebraic) form, and then replacing the traditional (unidirectional) gates with *self-organizing gates*, that can satisfy their logical (or algebraic) relation *irrespective* of the direction of the information flow, whether from the traditional input or the traditional output: they are *terminal-agnostic gates*. [67, 68, 69, 49]. In addition, the resulting dynamical systems can be designed so that no periodic orbits or chaos occur during dynamics [70, 71].

DMMs then map a *finite* string of symbols into a *finite* string of symbols, but operate in *continuous time*, hence they are distinct from Turing machines that operate in discrete time. (Note that universal memcomputing machines have been shown to be Turing-complete, but not Turing-equivalent [72].) As a consequence, they seem ideally suited for a hardware implementation. In fact, they can be realized in practice with non-linear, *non-quantum* dynamical systems with memory, such as electrical circuits implemented with conventional complementary metal–oxide–semiconductor technology [67].

On the other hand, DMMs, being classical dynamical systems, are such that their state trajectory belongs to a topological manifold, known as *phase space*, whose dimension, D, scales *linearly* with the number of degrees of freedom [73]. In addition, their state dynamics are described by *ordinary differential equations* (ODEs) [67, 68]. One can then attempt to *numerically* integrate these ODEs on our traditional computers, using any integration scheme [74]. However, naively, one would expect that the computational overhead of integrating these ODEs, coupled with large numerical errors, would require an unreasonable amount of CPU time as the problem size grew, hence limiting the realization of DMMs (like quantum computers) to only hardware.

To make things worse, the ODEs of DMMs are *stiff* [74], meaning that they have several, quite distinct, intrinsic time scales. This is because the memory variables of these machines have a much slower dynamics than the degrees of freedom representing the logical symbols (variables) of the problem [67, 68]. Stiff ODEs are notorious for requiring *implicit* methods of integration, which, in turn, require costly root-finding algorithms (such as the Newton's method), thus making the numerical simulations very challenging [74].

This leads one to expect poor performance when using *explicit* methods, such as the simplest *forward Euler* method, on the ODEs of the DMMs. Instead, several results using the forward Euler integration scheme on DMMs have shown that these machines still find solutions to a variety of combinatorial optimization problems, including, Boolean satisfiability (SAT) [71], maximum satisfiability [75, 76], spin glasses [77], machine learning [78, 79], and integer linear programming [69].

For instance, in Ref. [71], the numerical simulations of the DMMs, using forward Euler with an adaptive time step, substantially outperform traditional algorithms [80, 81, 82], in the solution of 3-SAT instances with planted solutions. The results show a *power-law* scaling in the number of integration steps as a function of problem size, avoiding the exponential scaling seen in the performance of traditional algorithms on the same instance classes. Furthermore, in Ref. [71], it was found that the average size of the adaptive time step decayed as a *power-law* as the problem size grew.

It is then natural to investigate how the power-law scaling of integration steps varies with the numerical integration scheme employed to simulate the DMMs. In particular, by employing a constant integration time step, Δt, we would like to investigate if the requisite time step to solve instances and control the numerical errors, continues to decay with a power-law, or will require exponential decay as the problem grows.

In this paper, we perform the above investigations while attempting to solve some of the planted-solution 3-SAT instances used as benchmarks in Ref. [71]. We will implement three *explicit* integration methods (forward Euler, trapezoid, and Runge-Kutta 4th order) while numerically simulating the DMM dynamics to investigate the effect on scalability of the number of integration steps versus the number of variables at a given clause-to-variable ratio.

Our numerical simulations indicate that, regardless of the explicit integration method used, the ODEs of DMM are robust against the numerical errors caused by the discretization of time. The robustness of the simulations is due to the *instantonic dynamics* of DMMs [65, 66], which connect critical points in the phase space with increasing stability. Instantons in dissipative systems are specific heteroclinic orbits connecting two distinct (in index) critical points. Both the critical points and the instantons connecting them are of topological character [83, 84]. Therefore, if the integration method preserves critical points (in number and index) then the solution search is "topologically protected" against reasonable numerical noise [1].

For each integration method, we find that when increasing the integration time step Δt, the system undergoes a "solvable-unsolvable transition" [2] at a critical Δt_c, which decays at most as a *a power law* with the problem size, avoiding the undesirable exponential decay that severely limits numerical simulations. We also find that, even though higher-order integration schemes are more favorable in most scientific problems, the first-order forward Euler method works the best for the dynamics of DMMs, providing best scalability in terms of function evaluations vs. the size of the problem. We attribute this to the fact that the forward Euler method preserves critical points, irrespective of the size of the integration time step, while higher order methods may introduce "ghost critical points" in the system [85], disrupting the instantonic dynamics of DMMs.

Finally, we provide a physical understanding of these results, by showing that, in the presence of noise, the dynamical behavior of DMMs can be modeled as a *directed percolation* of the state trajectory in the phase space, with the inverse of the time step Δt playing the role of percolation probability. We then analytically show that, with increasing percolation probability, the system undergoes a *permeable-absorbing phase transition*, which resembles the "solvable-unsolvable transition" in DMMs. All together, these results clarify the reasons behind the numerical robustness of the simulations of DMMs, and further reinforce the notion that these dynamical systems with memory are a useful tool for the solution of hard combinatorial optimization problems, not just in their hardware implementation but also in their numerical simulation.

This paper is organized as follows. In Sec. 2.2.2 we review the DMMs used in Ref. [71] to find satisfying assignments to 3-SAT instances. (Since our present paper is not a benchmark paper, the interested reader is directed to Ref. [71] for a comparison of the DMM approach to traditional algorithms for 3-SAT.) In Sec. 2.2.3 we compare the results on the scalability of the

[1]Incidentally, this is also the reason why the *hardware* implementation of DMMs would be robust against reasonable *physical* noise [65, 66].

[2]Note that all instances can be solved, as they have planted solutions.

number of steps to reach a solution for three explicit methods: forward Euler, trapezoid, and Runge-Kutta 4th order. For all three methods, our simulations show that increasing Δt (thereby increasing numerical error) induces a sort of "solvable-unsolvable phase transition," that becomes sharper with increasing size of the problem. However, we also show that the "critical" time step, Δt_c, decreases as a *power law* as the size of the problem increases for a given clause-to-variable ratio. In Sec. 2.2.4, we model this transition as a *directed percolation* of the state trajectory in phase space, with Δt playing the role of the inverse of the percolation probability. We conclude in Sec. 2.2.6 with thoughts about future work.

2.2.2 Solving 3-SAT with DMMs

In the remainder of this paper, we will focus on solving satisfiable instances of the 3-SAT problem [86], which is defined over N Boolean variables $\{y_i = 0,1\}$ constrained by M clauses. Each clause consists of 3 (possibly negated) Boolean variables connected by logical OR operations, and an instance is solved when an assignment of $\{y_i\}$ is found such that all M clauses evaluate to TRUE (satisfiable).

For completeness, we briefly review the dynamical system with memory representing our DMM. The reader is directed to Ref. [71] for a more detailed description and its mathematical properties.

To find a satisfying assignment to a 3-SAT instance with a DMM, we transform it into a Boolean circuit, where the Boolean variables y_i are transformed into continuous variables and represented with terminal voltages $v_i \in [-1,1]$ (in arbitrary units), where $v_i > 0$ corresponds to $y_i = 1$, and $v_i < 0$ corresponds to $y_i = 0$. We use $(l_{m,i} \vee l_{m,j} \vee l_{m,k})$ to represent the m-th clause, where $l_{m,i} = y_i$ or $\bar{y}_i$ depending on whether y_i is negated in this clause. Each Boolean clause is also transformed into a continuous constraint function:

$$C_m(v_i, v_j, v_k) = \frac{1}{2}\min\left(1 - q_{m,i}v_i, 1 - q_{m,j}v_j, 1 - q_{m,k}v_k\right), \tag{2.1}$$

where $q_{m,i} = 1$ if $l_{m_i} = y_i$ and $q_{m,i} = -1$ if $l_{m_i} = \bar{y}_i$. We can verify that the m-th clause evaluates to true if and only if $C_m < 0.5$ [71].

The idea of DMMs is to propagate information in the Boolean circuits 'in reverse' by using self-organizing logic gates (SOLGs) [67, 68] (see [69] for an application of *algebraic* ones).

SOLGs are designed to work bidirectionally, a property referred to as "terminal agnosticism" [49]. The terminal that is traditionally an output can now receive signals like an input terminal, and the traditional input terminals will self-organize to enforce the Boolean logic of the gate. This extra feature requires additional (memory) degrees of freedom within each SOLG. We introduce two additional memory variables per gate: "short-term" memory $x_{s,m}$ and "long-term" memory $x_{l,m}$. For a 3-SAT instance, the dynamics of our self-organizing logic circuit (SOLC) are governed by Eq. (2.2) [71]:

$$\begin{aligned} \dot{v}_i &= \sum_m x_{l,m} x_{s,m} G_{m,i} + \left(1 + \zeta x_{l,m}\right)(1 - x_{s,m}) R_{m,i}, \\ \dot{x}_{s,m} &= \beta \left(x_{s,m} + \varepsilon\right)(C_m - \gamma), \\ \dot{x}_{l,m} &= \alpha \left(C_m - \delta\right), \end{aligned} \tag{2.2}$$

where $x_s \in [0,1]$, $x_l \in [1, 10^4 M]$, $\alpha = 5, \beta = 20, \gamma = 0.25, \delta = 0.05, \varepsilon = 10^{-3}, \zeta = 0.1$. The "gradient-like" term ($G_{m,i} = \frac{1}{2} q_{m,i} \min(1 - q_{m,j} v_j, 1 - q_{m,k} v_k)$) and the "rigidity" term ($R_{m,i} = \frac{1}{2}(q_{m,i} - v_i)$ if $C_m = \frac{1}{2}(1 - q_{m,i} v_i)$, and $R_{m,i} = 0$, otherwise) are discussed in Ref. [71] along with other details.

2.2.3 Numerical scalability with different integration schemes

Numerical details

To simulate Eq. (2.2) on modern computers, we need an appropriate numerical integration scheme, which necessitates the discretization of time. In Ref. [71] we applied the first-order forward Euler method, which, in practice, tends to introduce large numerical errors in the long-time integration. Yet, the massive numerical error in the *trajectory* of the ODE solution does not translate into *logical* errors in the 3-SAT solution. The algorithm in Ref. [71] was capable of showing *power-law* scalability in the typical-case of clause distribution control (CDC) instances [86], though, using an adaptive time step.

To further understand this robustness, we study here the behavior of Eq. (2.2) under different *explicit* numerical integration methods using a *constant* time step, Δt. We will investigate the effect of increasing the time step. Writing the ODEs as $\dot{\mathbf{x}}(t) = F(\mathbf{x}(t))$, with $\mathbf{x}$ the collection of voltage and memory variables, and F the flow vector field [68], we use the following explicit Runge-Kutta time step [74]:

$$x_{n+1} = x_n + \Delta t \sum_{i=1}^{q} \omega_i k_i \tag{2.3}$$

where $k_i = F(x_n + \Delta t \sum_{j=1}^{i-1} \lambda_{ij} k_j)$. Specifically, we apply three different integration schemes: *forward Euler* method with $q = 1$, $\omega_1 = 1$; *trapezoid* method with $q = 2$, $\omega = (\frac{1}{2}, \frac{1}{2})$, $\lambda_{21} = 1$; *4th order Runge-Kutta* method (RK4) with $q = 4$, $\omega = (\frac{1}{6}, \frac{1}{3}, \frac{1}{3}, \frac{1}{6})$, and

$$\lambda = \begin{pmatrix} 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{2.4}$$

Note that Eqs. (2.2) are *stiff*, meaning that the explicit integration methods we consider here should diverge quite fast, within a few integration steps, irrespective of the problem instances we choose [74]. However, as we will show below, the explicit integration methods actually work very well for these equations, and the integration time step can even be chosen quite large.

As an illustration, we focus on planted-solution 3-SAT instances at clause-to-variable ratio $\alpha_r = 8$ [3]. These instances require exponential time to solve using the local-search algorithm

[3]Using the method of [86] with $p_0 = 0.08$

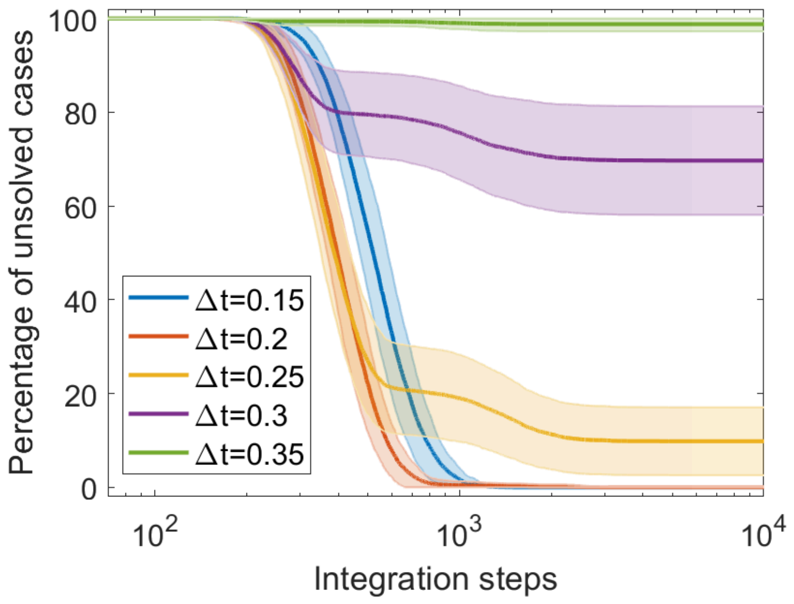


Figure 2.1. Solution of clause distribution control 3-SAT instances with 10^4 variables at clause-to-variable ratio $\alpha_r = 8$ by numerically integrating Eqs. (2.2) with the forward Euler method with different time steps, Δt. We performed 10^4 solution trials (100 instances and 100 different initial conditions for each instance) for each Δt. We observe the number of unsolved cases decays as integration steps increase, with the solid line representing the result for all 10^4 trials, and shaded area representing one standard deviation over the 100 curves calculated separately using the 100 instances, with each curve covering 100 different initial conditions using one specific instance. At large number of integration steps, the number of unsolved cases reaches a plateau, whose height only depends on the integration time step Δt. The fraction of solved instances represents the size of the basin of attraction of Eqs. (2.2), which is similar for all instances at a certain Δt and shrinks as Δt increases. This indicates that rather than being problem-specific, our numerical algorithm is a general incomplete solver for 3-SAT problems.

walk-SAT [86] and other state-of-the-art solvers [71]. In the Appendix, we will show similar results for $\alpha_r = 6$.

We start with 100 CDC 3-SAT instances with the number of variables $N = 10^4$, and solve them by numerically integrating Eqs. (2.2) using the forward Euler method with different values of Δt. For each 3-SAT instance, we made 100 individual attempts with different random initial conditions, so that the total number of solution attempts is 10^4. In Fig. 2.1, we see the number of unsolved instances decreases rapidly until reaching a plateau. When the simulations are performed again with a smaller time step, the plateau height decreases. Once reached, the

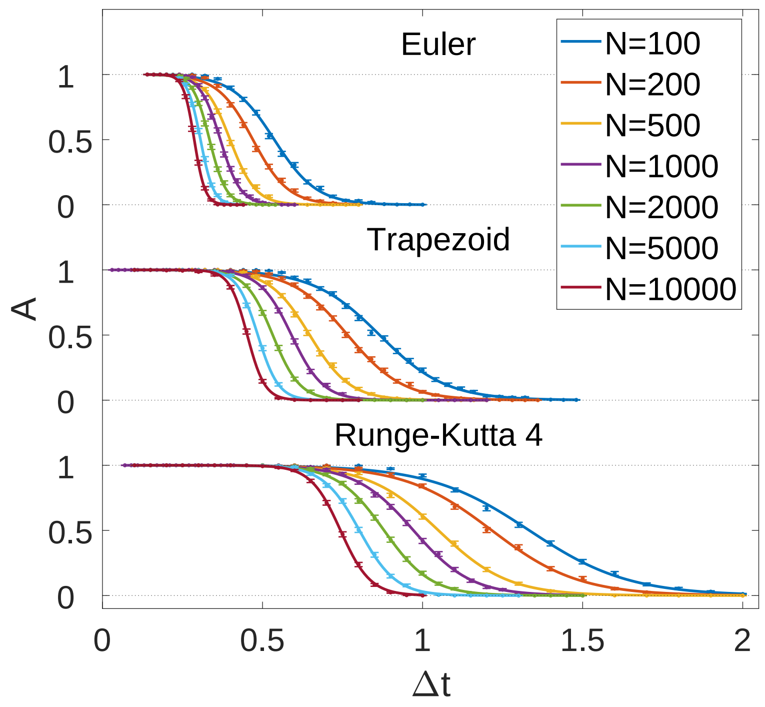


Figure 2.2. The size of the basin of attraction, A, versus Δt, for different numbers of variables N and different explicit integration schemes, is well fitted by sigmoid-like curves, that become sharper as N increases. This indicates a "solvable-unsolvable phase transition" at a critical Δt_c. Each data point is calculated based on 1000 solution trials (100 instances and 10 initial conditions per instance), and the solid curves are fitted using the function $A = 1/[1+\exp(-c(\Delta t - d))]$, with c and d fitting parameters. The error bars, estimated with the help of data from Fig. 2.1, represent one standard deviation.

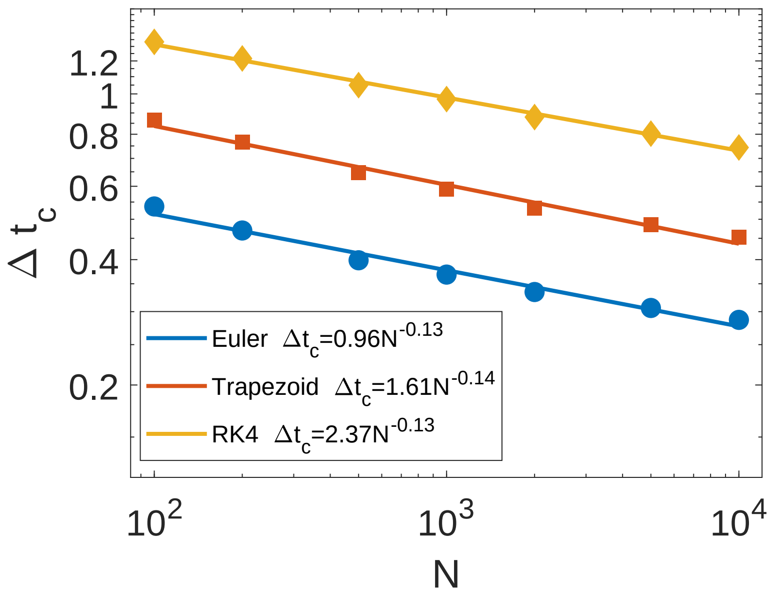


Figure 2.3. Scaling of the critical time step, Δt_c. In this figure, Δt_c is extracted from Fig. 2.2 where $A(\Delta t_c) = 1/2$. Δt_c shows a *power-law* scaling with number of variables N, and, as expected, integration methods with higher orders have a larger Δt_c.

plateau persists while increasing the number of integration steps.

Basin of attraction

We argue that the plateaus in Fig. 2.1 are caused by a reduction of the basin of attraction in Eqs. (2.2), created by the numerical integration method, rather than the hardness of the instances themselves. To show this, first note that *all* solution attempts succeed when $\Delta t = 0.15$ (basin of attraction is the entire phase space), while almost all attempts fail when $\Delta t = 0.35$ (basin of attraction shrinks to zero size). When Δt falls in between, for each 3-SAT instance, the number of unsolved attempts is centered around the shaded area in Fig. 2.1. That is, at a certain Δt, the size of the basin of attraction for different instances is approximately the same.

We use A to denote the ratio between the volume of the basin of attraction and the volume of the entire phase space. In Fig. 2.2, we estimate this quantity by calculating the fraction of solved instances for each number of variables N and time step Δt for 100 different 3-SAT instances per size N, and 10 initial conditions per instance. At smaller Δt, all instances with all

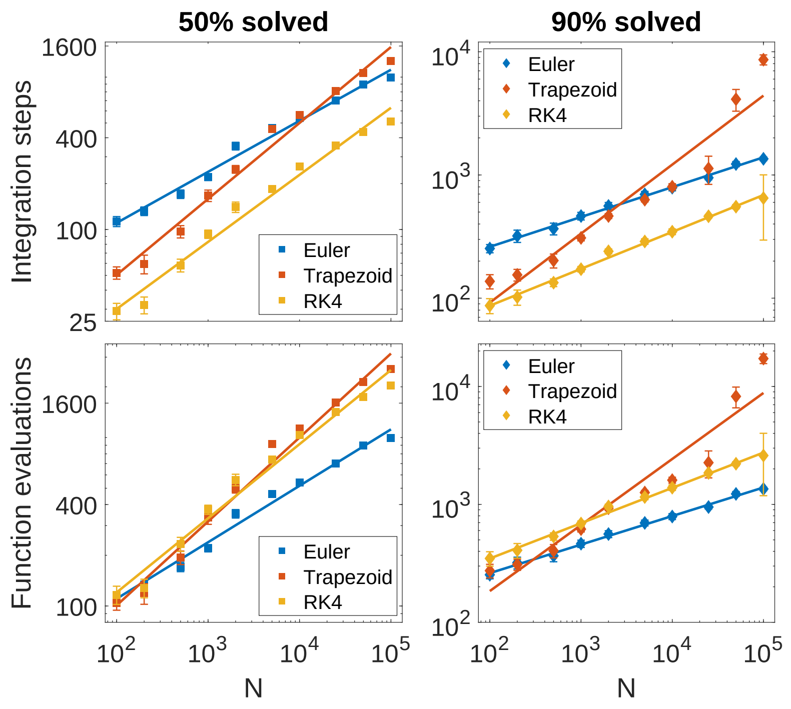


Figure 2.4. The scalability curves for the three different explicit integration schemes considered in this work, in which the integration time step Δt is calculated using the power-law fit of Δt where $A(\Delta t) = 0.95$, times an additional "safety factor" of 0.6 (see text). All three methods show a power-law scaling, and the values of the scaling exponents are 0.34 (Euler 50%), 0.50 (Trapezoid 50%), 0.44 (RK4 50%), 0.24 (Euler 90%), 0.56 (Trapezoid 90%), 0.30 (RK4 90%). The scaling is the same, for each integration scheme, in terms of either number of steps or function evaluations; only the pre-factor changes. The error bars are estimated with bootstrapping and represent one standard deviation. For $N \in [10^2, 10^5]$, Runge-Kutta 4th order (RK4) requires the least number of integration steps. However, in terms of function evaluations, the simplest forward Euler method is the most efficient one.

different initial conditions are solved, i.e., the basin of attraction is the entire phase space, $A \to 1$. This is in agreement with theoretical predictions for continuous-time dynamics [71]. On the other hand, when Δt is too large, the structure of the phase space gets modified by the numerical error introduced during discretization, and $A \to 0$. This is a well known effect that has been studied also analytically in the literature [87].

Plotting A versus Δt for different number of variables N and different integration schemes, we get a series of sigmoid-like curves (Fig. 2.2), that become sharper as N increases. This suggests the existence of a "phase transition": when going beyond a critical Δt_c, there is a drastic change in A, and the system undergoes a *solvable-unsolvable transition*.

Integration step scalability

The above results allow us to determine how the integration step, Δt, for the three integration schemes we have chosen, scales as the size of the problem increases. In Fig. 2.3, we define Δt_c such that $A(\Delta t_c) = 1/2$, and determine the relation between Δt_c and N.

We find Δt_c scales as a *power law* with the number of variables N. This is a major result because it shows that we do not need Δt to decrease exponentially to have the same success rate. In Ref. [71], it was demonstrated, using topological field theory [65, 66], that in the ideal (noise-free) *continuous-time* dynamics, the physical time required for our dynamical system to reach the solution scales as $O(N^\gamma)$, with $\gamma \leq 1$.

Our numerical results show that, when simulating the dynamics with numerical integration schemes on classical computers, $\Delta t_c \sim N^{-\delta}$. Coupled with the previous analytical results [71], this means that the number of integration steps scales as $O(N^{\gamma+\delta})$. In other words, discretization only adds a $O(N^\delta)$ overhead to the complexity of our algorithm, indicating that DMMs can be efficiently simulated on classical computers [4].

In Fig. 2.3, note that $\Delta t_c > 10^{-1}$ for all three integration methods. This is quite unexpected for a stiff system simulated with explicit integration methods, because a large time step introduces

[4]Of course, this is *not* an analytical proof of the efficiency of the *numerical* simulations, just an empirical result.

large local truncation errors at each step of the integration. This error accumulates and should destroy the *trajectory* of the ODEs we are trying to simulate. However, we can still solve *all* tested instances with Eqs. (2.2) even at such a large Δt. In Sec. 2.2.4, we will provide an explanation of why this is possible.

Here, we note that choosing an appropriate Δt could speed up the algorithm significantly, as smaller Δt leads to excessive integration steps, while larger Δt may cause the solver to fail. To find an appropriate time step for our scalability tests, we choose Δt from Fig. 2.2 such that 95% of the trials have found a solution, and multiply that Δt by a "safety factor" of 0.6 to make sure we hit the region of $A(\Delta t) \approx 1$.

By employing the resulting value of Δt, we plot in Fig. 2.4 the typical-case and 90th percentile scalability curves up to $N = 10^5$. For each variable size, N, we perform 1000 3-SAT solution trials (100 instances and 10 initial conditions each), and report the number of integration steps when 50% and 90% of the trials are solved. Additionally, we report the number of *function evaluations* (namely, the number of times the right-hand side of Eqs. (2.2) is evaluated), which differs from the number of integration steps by only a constant.

Both quantities show a *power-law* scalability for all three integration schemes. Since Δt_c is larger for higher-ordered ODE solvers, the latter ones typically require fewer steps to solve the problems. However, when taking the total number of function evaluations into account, the simplest forward Euler method becomes the most efficient integration scheme for our ODEs.

2.2.4 Directed percolation and noise in DMMs

As anticipated, our results are unexpected in view of the large truncation errors introduced during the simulations of the ODEs of the DMMs. However, one might have predicted the numerical robustness upon considering the type of dynamics the DMM executes in phase space.

Instantonic dynamics

Instantons are topologically non-trivial solutions of the equations of motion connecting two different critical points (classical vacua in field-theory language) [88]. In dissipative systems, as the ones considered in this work, instantons connect two critical points (such as saddle points, local minima or maxima), that differ in index (number of unstable directions). In fact, they always connect a critical point with higher index to another with lower index [83, 84].

It was shown in Refs. [65, 66] that, by stochastically quantizing the DMM's equation of motion, one obtains a supersymmetric topological field theory, from which it can be deduced that the *only* "low-energy", "long-wavelength" dynamics of DMMs is a collection of elementary *instantons* (a composite instanton). Critical points are also of topological character. For instance, their number cannot change unless we change the topology of phase space [89].

In addition, given two distinct (in terms of indexes) critical points, there are several (a family of) trajectories (instantons) that may connect them, since the unstable manifold of the "initial" critical point may intersect the stable manifold of the "final" critical point at several points in the phase space. In the ideal (noise-free) continuous time dynamics, if the only equilibria (fixed points of the dynamics) are the solutions of the given problem (as shown, e.g., in [67, 71]), then the state trajectory is *driven* towards the equilibria by the voltages that set the input of the problem the DMM attempts to solve.

Physical noise

In the presence of (physical) noise instead, *anti-instantons* are generated in the system. These are (time-reversed) trajectories that connect critical points of *increasing* index. However, anti-instantons are *gapped*, in the sense that their amplitude is exponentially suppressed with respect to the corresponding amplitude of the instantons [90]. This means that the "low-energy", "long wave-length" dynamics of DMMs is still a succession of instantons, even in the presence of (moderate) noise.

Nevertheless, suppose an instanton connects two critical points, and immediately after an anti-instanton occurs for the *same* critical points (even if the two trajectories are different). For all practical purposes, the "final" critical point of the original instanton has never been reached, and that critical point can be called *absorbing*, in the sense that the trajectory would "get stuck" on that one, or wanders in other regions of the phase space. In other words, in the presence of noise, there is a finite probability for some state trajectory to explore a much larger region of the phase space. The system then needs to explore some other (anti-)instantonic trajectory to reach the equilibria, solutions of the given problem. Nevertheless, due to the fact that anti-instantons are gapped, and the topological character of both critical points and instantons connecting them, if the physical noise level is not high enough to change the topology of the phase space, the dynamical system would still reach the equilibria. [5]

On the other hand, if the noise level is too high, our system may experience a condensation of instantons and anti-instantons, which can break supersymmetry dynamically [91]. In turn, this supersymmetry breakdown, can produce a noise-induced chaotic phase even if the flow vector field remains integrable. Although we do not have an analytical proof of this yet, we suspect that this may be related to the unsolvable phase we observe.

Directed percolation

If we visualize the state trajectory as the one traced by a liquid in a corrugated landscape, the above suggests an intriguing analogy with the phenomenon of *directed percolation* (DP) [92], with the critical points acting as 'pores', and instantons as 'channels' connecting 'neighboring' (in the sense of index difference) critical points.

DP is a well-studied model of a non-equilibrium (continuous) phase transition from a fluctuating permeable phase to an absorbing phase [92]. It can be intuitively understood as a liquid passing through a porous substance under the influence of a field (e.g., gravity). The field restricts the direction of the liquid's movement, hence the term "directed". In this model, neighboring pores are connected by channels with probability p, and disconnected otherwise.

[5] In fact, moderate noise may even help *accelerate* the time to solution, by reducing the time spent on the critical points's stable directions.

When increasing p, the system goes through a phase transition from the absorbing phase into a permeable phase at a critical threshold p_c.

Numerical noise

In numerical ODE solvers, discrete time also introduces some type of noise (truncation errors), and the considerations we have made above on the presence of anti-instantons still hold. However, numerical noise could be substantially more damaging than physical noise. The reason is twofold.

First, as we have already anticipated, numerical noise *accumulates* during the integration of the ODEs. In some sense, it is then always *non-local*. Physical noise, instead, is typically local (in space and time), hence it is a *local* perturbation of the dynamics [93]. As such, if it is not too large, it cannot change the phase space topology.

Second, unlike physical noise, integration schemes may change the topology of phase space *explicitly*. This is because, when one transforms the continuous-time ODEs (2.2) into their discrete version (a *map*), this transformation can introduce *additional* critical points in the phase space of the map, which were not present in the original phase space of the continuous dynamics. These extra (undesirable) critical points are sometimes called *ghosts* [85]. For instance, while the forward Euler method can *never* introduce such ghost critical points, irrespective of the size of Δt (because both the continuous dynamics and the associated map have the same flow field F), both the trapezoid and Runge-Kutta 4th order may do so if Δt is large enough. These critical points would then further degrade the dynamics of the system.

With these preliminaries in mind, let us now try to quantify the analogy between the DMM dynamics in the presence of numerical noise and directed percolation.

2.2.5 Paths to solution

First of all, the integration step, Δt, must be inversely related to the percolation probability p: when Δt tends to zero, the discrete dynamics approach the ideal (noise-free) dynamics for which $p \to 1$. In the opposite case, when Δt increases, p must decrease.

In directed percolation, the order parameter is the density of active sites [92]. This would correspond to the density of "achievable" critical points in DMMs. However, due to the vast dimensionality of their phase space (even for relatively small problem instances), this quantity is not directly accessible in DMMs.

Instead, what we can easily evaluate is the ratio of successful solution attempts: starting from a random point in the phase space (initial condition of the ODEs (2.2)), and letting the system evolve for a sufficiently long time, a path would either successfully reach the solution (corresponding to a permeable path in DP) or fail to converge (corresponding to an absorbing path in DP) . By repeating this numerical experiment for a large number of trials, the starting points of the permeable paths essentially fill the basin of attraction of our dynamical system. The advantage of considering this quantity is that the ratio of permeable paths in bond DP models can be calculated analytically.

Now we set up the correspondence of paths between DP and DMM. In DP, a permeable path is defined to be a path that starts from the top of the lattice and ends at the bottom, and corresponds to a successful solution attempt in DMMs. Similarly, an absorbing path starts from the top and terminates within the lattice in DP, and corresponds to a failed solution attempt in DMMs. Consider then a 3-SAT problem with only one solution. A DMM for such a problem can reach the solution from several initial conditions. This translates into a D-dimensional cone-shaped lattice for the DP model (see Fig. 2.5). The (top) base of the cone would represent the starting points, and the apex of the cone would represent the solution point. A permeable path connects the base to the apex, and an absorbing path ends in the middle of the lattice with all bonds beneath it disconnected.

Note that, in the DP model, it is possible to have different permeable paths starting from the same initial point. However, from the perspective of DMMs, there is no randomness in the dynamics, and each initial condition corresponds to a unique path. Since DP is a simplified theoretical model to describe numerical noise in DMMs, in this model we assume that the trajectory “randomly chooses” a direction when reaching a critical point. This is a reasonable assumption since there can be a large number of instantons starting from some critical point in the phase space, and it is almost impossible for us to actually monitor which instantonic path the trajectory follows.

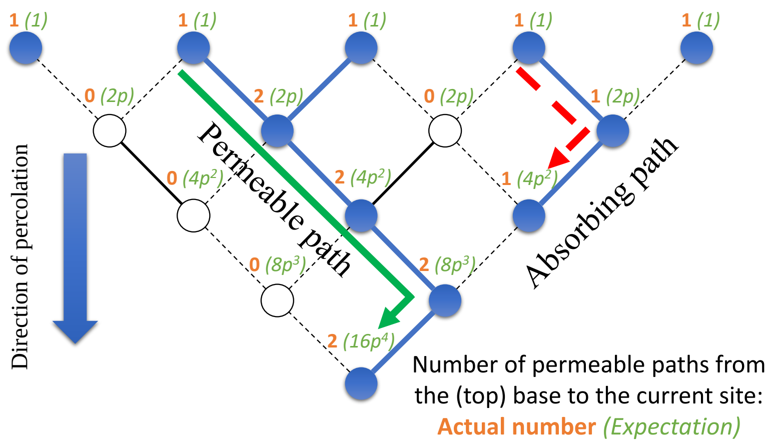


Figure 2.5. Illustration of permeable and absorbing paths in DP on a cone-shaped 2d lattice. Connected bonds are represented with solid lines, and disconnected ones are represented with dotted lines. A permeable path connects the base to the apex, and an absorbing path ends in the middle of the lattice with all bonds beneath it disconnected. The number of permeable paths n_p is calculated iteratively: n_p at a given site equals the sum of n_p at its connected predecessor sites, and the expectation $\langle n_\mathrm{p} \rangle$ at a given site equals the sum of n_p at all its predecessor sites times p, the percolation probability.

With this correspondence in mind, the ratio of permeable paths in DP is analogous to the ratio of successful solution attempts in DMMs. Below, we will calculate this quantity exactly in DP, and use this calculation to model the solvable-unsolvable transition we found in DMMs.

To begin with, we assume that all starting points are occupied with equal probability, and calculate the expectation of the number of permeable and absorbing paths. This can be done iteratively: the expected number of permeable paths at a given site equals the sum of permeable paths at its predecessor sites, times p, the percolation probability. Assuming the number of time steps required to reach the apex is T, then the expected number of permeable paths

$$\langle n_{\mathrm{p},T}\rangle = (Dp)^T. \tag{2.5}$$

An illustration of the calculation on a cone-shaped 2d lattice is shown in Fig. 2.5.

The number of absorbing paths $\langle n_{a,T}\rangle$ is trickier to compute. Details of the calculation can be found in Appendix 2.2.7, here we only give the approximate result (valid for $Dp > 1$) near the transition:

$$\langle n_{\mathrm{a},T}\rangle = \frac{(Dp)^{T+1}}{2}\left(\frac{1-p}{\ln Dp}\right)^D \mathrm{erfc}(\sqrt{D}\frac{1-\ln Dp}{\sqrt{2\ln Dp}}). \tag{2.6}$$

The ratio of permeable paths r is simply $\langle n_{\mathrm{p},T}\rangle/(\langle n_{\mathrm{p},T}\rangle + \langle n_{\mathrm{a},T}\rangle)$:

$$r = \frac{1}{1+\frac{1}{2}Dp(\frac{1-p}{\ln Dp})^D \mathrm{erfc}(\sqrt{D}\frac{1-\ln Dp}{\sqrt{2\ln Dp}})}. \tag{2.7}$$

We observe that the transition occurs at $p_c \sim \frac{e}{D}$. Near the transition, let us define $p = \frac{e+\delta}{D}$, where δ is small. Then, $\ln Dp \approx 1+\frac{\delta}{e}$. Further using $\lim_{x\to\infty}(1+\frac{1}{x})^x = e$, to order $O(\delta)$, Eq. (2.7) becomes

$$r = \frac{1}{1+\frac{1}{2}e^{1-e-\delta-D\delta/e}\mathrm{erfc}\big(-\sqrt{\frac{D}{2}\frac{\delta}{e}}\big)}. \tag{2.8}$$

In the limit of $D\to\infty$, the divergence comes from the D in the exponent. Therefore, the transition happens exactly at $\delta = 0$. When $\delta > 0$, $r\to 1$; when $\delta < 0$, $r \to 0$.

Note that when $\delta \ll 0$, the erfc term dominates over the $e^{-D\delta/e}$ term instead. However, in this case, p is too small, and some approximations we made in Appendix 2.2.7 to derive $\langle n_{\mathrm{a},T}\rangle$ no longer hold. In this sense, Eqs. (2.7) and (2.8) are only valid near the transition point $p_c = \frac{e}{D}$.

The behavior of Eq. (2.7) is plotted in Fig. 2.6 for different dimensions D. Comparing to Fig. 2.2, we can already see a few similarities: both figures exhibit a sigmoidal behavior.

However, as Δt in DMMs is inversely related to p in DP, they are curving in opposite directions.

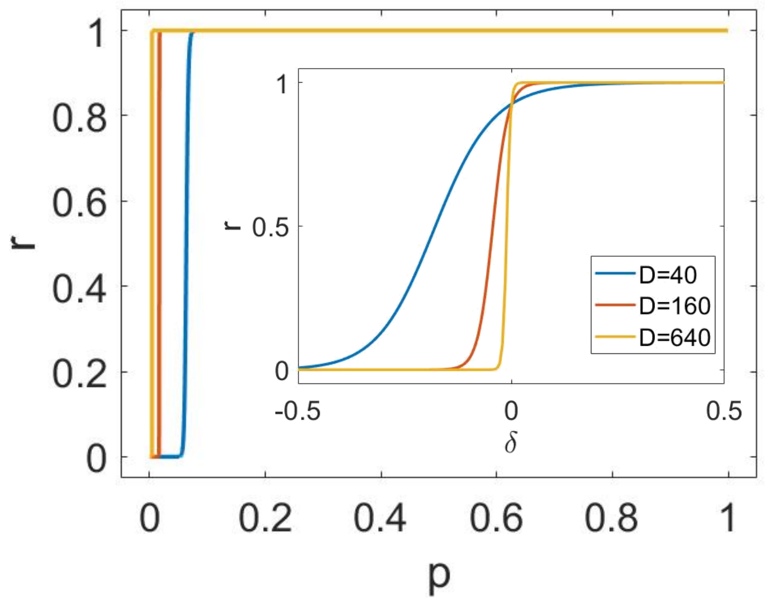


Figure 2.6. The ratio of permeable paths, calculated with Eq. (2.7), plotted for different values of the dimension D. The inset shows the same curves near their transition points, with $\delta = Dp - e$. The curves exhibit a sigmoidal behavior, which is similar to the one in Fig. 2.2. However, note that, since Δt in DMMs is inversely related to the probability p in DP, they are curving in opposite directions.

Recall that the size of the basin of attraction, A, in DMMs corresponds to the ratio r in DP. To model their relation, we then use the *ansatz* $\delta \equiv Dp - e = a\left(\frac{1}{N\Delta t} - b\right)$, with a and b some real numbers, and fit A to Δt using Eq. (2.7). (Note that this trial function has a meaning only near Δt_c.)

We find that we can fit the curves reasonably well by fixing $a = 5$, and Fig. 2.7 shows the fitting result of b and D, where the number of variables N ranges between 10^2 and 10^4, and each data point is obtained by numerically integrating Eqs. (2.2) for 1000 3-SAT solution attempts (100 instances and 10 initial conditions each), until a plateau, as in Fig. 2.1, has been reached. Note that the horizontal axis represents $1/(N\Delta t)$.

The fitted parameters b and D are shown in Fig. 2.8, where both parameters exhibit a power-law scaling. At the transition threshold $\delta = 0$, we have $b = \frac{1}{N\Delta t}$. Therefore, the power-law

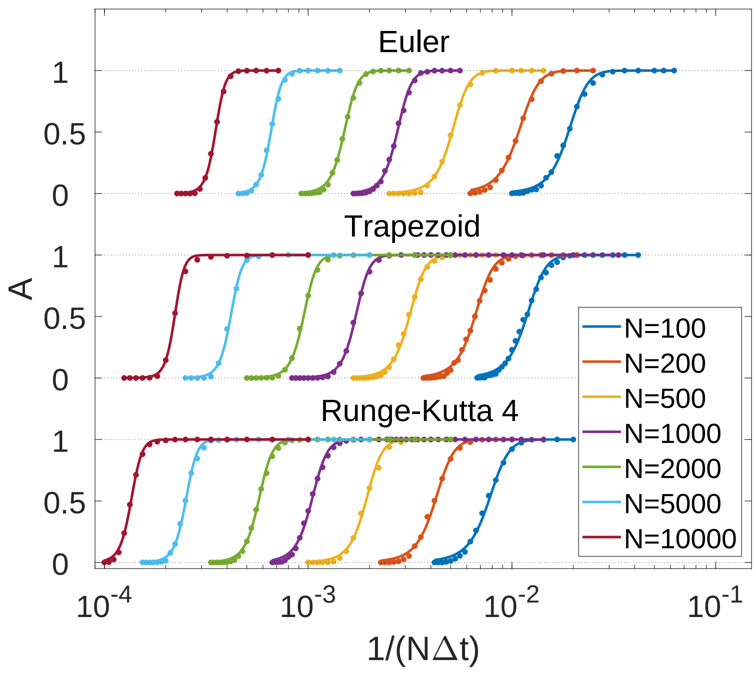


Figure 2.7. Relation, close to Δt_c, between the size of the basin of attraction A and the discretization time step Δt, plotted for different system sizes N, for the three explicit methods used in this work. Each data point is obtained by numerically integrating Eqs. (2.2) over 1000 3SAT solution attempts (100 instances and 10 initial conditions each), until reaching a plateau. The solid curves are fitted using Eq. (2.7), with A corresponding to r and $\delta \equiv Dp - e = a\left(\frac{1}{N\Delta t} - b\right)$, with a and b being fitting parameters.

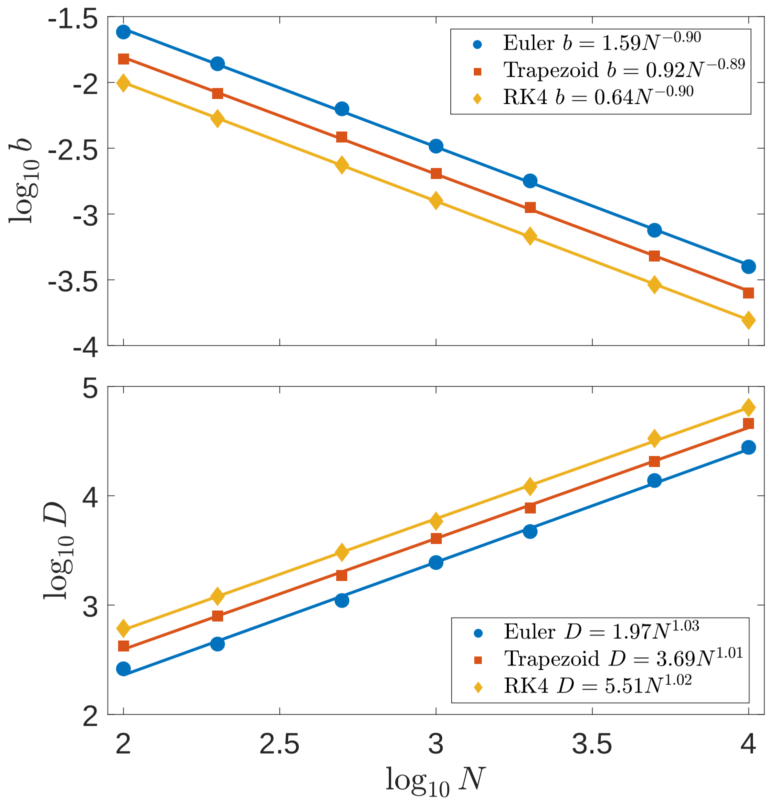


Figure 2.8. The fitted parameters b and D for different variable size N for the trial function $\delta \equiv Dp - e = a\left(\frac{1}{N\Delta t} - b\right)$, which connects the time step Δt to the percolation probability p. (a is fixed to 5.) D is the dimensionality of the DP lattice, which corresponds to the dimensionality of the composite instanton.

scaling of b is closely related to the power-law scaling of Δt_c in Fig. 2.3, and these two scalings show similar trends.

The dimensionality D is proportional to the dimensionality of the composite instanton (namely the number of elementary instantons to reach the solution). In Ref. [66], this dimensionality was predicted to scale (sub-)linearly with the number of variables N (in the noise-free case), and it is smaller than the dimensionality of the actual phase space ($(1+2\times\alpha_r)N = 17N$ in the present case). This prediction agrees with our numerical results.

Finally, using the correspondence between DMMs in the presence of noise and DP, we can qualitatively explain why the DMMs numerically integrated still provide solution to the problem they are designed to solve even with such large numerical errors introduced during integration. In DMMs, the dimensionality of the composite instanton, D, is usually very large, and DP tells us that the percolation threshold $p_c = e/D \sim e/N$.

Therefore, even if most of the paths in the phase space are destroyed by noise, we can still achieve the solution as long as the probability of an instantonic path is larger than $e/D \sim e/N$. This argument, in tandem with the fact that critical points and instantons have a topological character [83, 84], ensures the robustness of DMMs in numerical simulations.

2.2.6 Conclusions

In conclusion, we have shown that the dynamics of digital memcomputing machines (DMMs) under discrete numerical solvers can be described as a directed percolation of state trajectory in the phase space. The inverse time step, $1/\Delta t$, plays the role of percolation probability, and the system undergoes a "solvable-unsolvable phase transition" at a critical Δt_c, which scales as a *power law* with problem size. In other words, for the problem instances considered, we have numerically found that the integration time step does *not* need to decrease exponentially with the size of the problem, in order to control the numerical errors.

This result is quite remarkable considering the fact that we have only employed *explicit* methods of integration (forward Euler, trapezoid, and Runge-Kutta 4th order) for *stiff* ODEs. (In fact, the forward Euler method, although having the largest numerical error, solves the instances in the least amount of function evaluations.) It can be ultimately traced to the type of dynamics the DMMs perform during the solution search, which is a composite instanton in phase space. Since instantons, and the critical points they connect, are of topological character, perturbations to the actual trajectory in phase space do not have the same detrimental effect as the changes of topology of the phase space.

Since numerical noise is typically far worse than physical noise, these results further reinforce the notion that these machines, if built in hardware, would be topologically protected against moderate physical noise and perturbations.

However, let us note that we did not prove that the dynamics of DMMs with numerical (or physical) noise belong to the DP universality class. In fact, according to the DP-conjecture [94, 95], a given model belongs to such a universality class if *(i)* the model displays a continuous phase transition from a fluctuating active phase into a unique absorbing state; *(ii)* the transition is characterized by a non-negative one-component order parameter; *(iii)* the dynamic rules are short-ranged; *(iv)* the system has no special attributes such as unconventional symmetries, conservation laws, or quenched randomness.

It is easy to verify that DMMs under numerical simulations satisfy properties *(iii)* and *(iv)*. However, verifying property *(i)* and *(ii)* is not trivial, as the relevant order parameters are not directly accessible due to the vast phase space of DMMs. Still, the similarities we have outlined in this paper, between DMMs in the presence of noise and DP, help us better understand how these dynamical systems with memory work, and why their simulations are robust against the unavoidable numerical errors.

2.2.7 Appendix

Results for $\alpha_r = 6$

Here, we show that the results presented in the main text hold also for other clause-to-variable ratios. As examples, we choose $\alpha_r = 6$. (Similar scalability results have been already reported in Ref. [71] for $\alpha_r = 4.3$.) In particular, we find similar scaling behavior for Δt_c for all clause-to-variable ratios, with of course, different power laws.

Figure 2.9 have been obtained as discussed in the main text, and show Δt_c vs. number of variables for $\alpha_r = 6$.

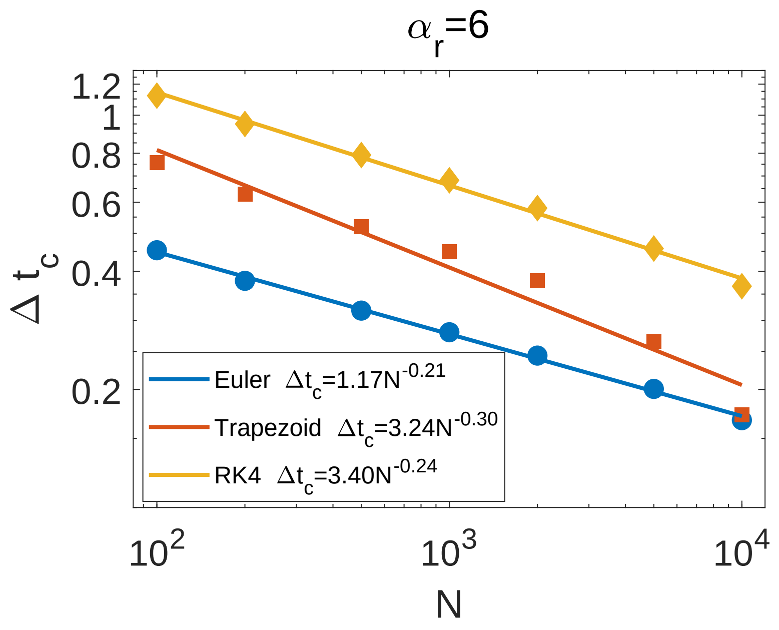


Figure 2.9. Critical Δt_c defined as $A(\Delta t_c) = 1/2$ (see main text) for a clause-to-variable ration of $\alpha_r = 6$. Δt_c shows a *power-law* scaling with N, and, as expected, integration methods with higher orders have a larger Δt_c.

In Fig. 2.10, instead we show the scalability curves for $\alpha_r = 6$, considering all three explicit integration methods. As in the main text, we find that the forward Euler method, although having the largest numerical error, solves the instances in the least amount of function evaluations for $\alpha_r = 6$. Every data point in the curves is obtained with 100 3-SAT instances, with 10 solution trials for each instance.

Calculating the number of absorbing paths in directed percolation

Here, we give a detailed calculation of the number of absorbing paths we outlined in Sec. 2.2.5. We use D to denote the dimension of the lattice where percolation takes place (corresponding to the dimension of the DMM phase space), T to denote the number of steps to

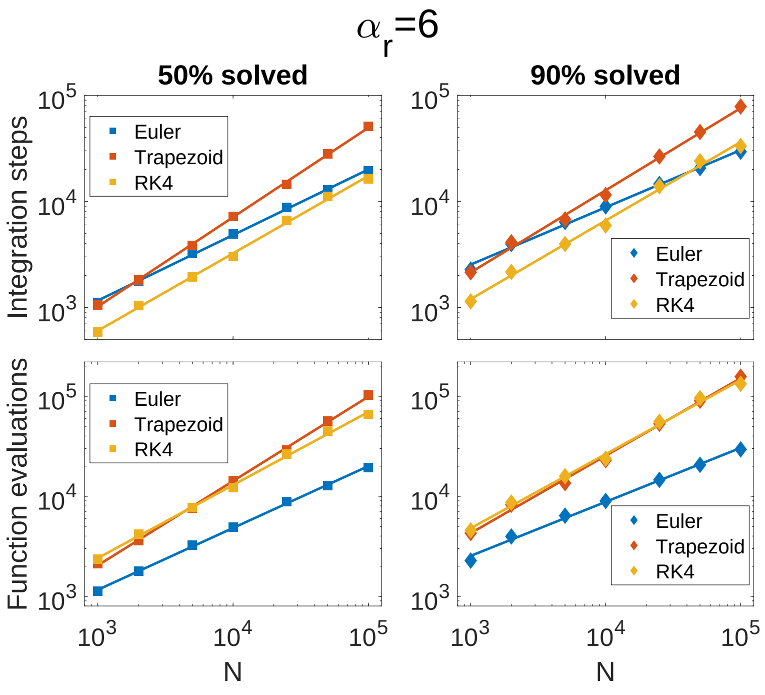


Figure 2.10. The scalability curves at $\alpha_r = 6$ for the three different explicit integration schemes considered in this work, in which the constant integration time step Δt is calculated using the power-law fit of Δt where $A(\Delta t) = 0.95$, times an additional "safety factor" of 0.6 (see main text). All three methods show a power-law scaling. The values of the scaling exponents are 0.62 (Euler 50%), 0.84 (Trapezoid 50%), 0.73 (RK4 50%), 0.54 (Euler 90%), 0.77 (Trapezoid 90%), 0.74 (RK4 90%). The scaling is the same, for each integration scheme, in terms of either number of steps or function evaluations; only the pre-factor changes. For $N \in [10^3, 10^5]$, Runge-Kutta 4th order (RK4) requires the least number of integration steps. However, in terms of function evaluations, the simplest forward Euler method is the most efficient one.

reach the bottom of the lattice (corresponding to the number of instantonic steps to reach the solution in DMMs), and p to denote the percolation probability. Throughout the calculation, we use the approximation that $D \to \infty, T \to \infty$, as both D and T are very large in DMMs.

As we illustrated in Sec. 2.2.5, the expected number of permeable paths at time step i (i.e., the i-th level in the lattice in Fig. 2.5, where i starts from 0), is

$$\langle n_{\mathrm{p},i} \rangle = (Dp)^i \tag{2.9}$$

The number of absorbing paths at each site is the number of permeable paths at that site, times $(1-p)^D$, the probability that all bonds beneath it are disconnected. Note that this probability has a different expression at the boundary of the lattice, but as D and T are large, we ignore the boundary effect here. Meanwhile, the number of sites m_i at each time step i, to the leading order, equals to the volume of a $(D-1)$-dimensional hyperpyramid,

$$m_i = \frac{(T+1-i)^{D-1}}{(D-1)!}. \tag{2.10}$$

Then, the total number of absorbing paths is

$$\begin{aligned}\langle n_{\mathrm{a},T} \rangle &= (1-p)^D \sum_{i=0}^{T-1} \frac{(T+1-i)^{D-1}}{(D-1)!} (Dp)^i \\ &= \frac{(1-p)^D (Dp)^{T+1}}{(D-1)!} \sum_{i=0}^{T-1} (T+1-i)^{D-1} \frac{1}{(Dp)^{T+1-i}} \\ &= \frac{(1-p)^D (Dp)^{T+1}}{(D-1)!} \sum_{N=2}^{T+1} N^{D-1} \frac{1}{(Dp)^N}\end{aligned} \tag{2.11}$$

To get rid of the summation in Eq. (2.11), we approximate it by adding the negligible $N=1$ term, and replace the summation with an integral:

$$\begin{aligned}\langle n_{\mathrm{a},T}\rangle &\approx \frac{(1-p)^D(Dp)^{T+1}}{(D-1)!}\int_0^{T+1} x^{D-1}\frac{1}{(Dp)^x}\mathrm{d}x \\ &= \frac{(1-p)^D(Dp)^{T+1}}{(D-1)!}\int_0^{T+1} x^{D-1}e^{-\ln(Dp)x}\mathrm{d}x \\ &= \frac{(Dp)^{T+1}}{(D-1)!}\left(\frac{1-p}{\ln Dp}\right)^D\int_0^{(T+1)\ln Dp} x^{D-1}e^{-x}\mathrm{d}x \\ &= \frac{(Dp)^{T+1}}{(D-1)!}\left(\frac{1-p}{\ln Dp}\right)^D\gamma(D,(T+1)\ln Dp)\end{aligned} \tag{2.12}$$

where $\gamma(s,x)=\int_0^x t^{s-1}e^{-t}\mathrm{d}t$ is the lower incomplete gamma function, and the result is valid for $Dp>1$.

To further simplify $\langle n_{\mathrm{a},T}\rangle$, we use the relation [96]

$$\Gamma(n+1,x)\equiv\Gamma(n+1)-\gamma(n+1,x)=n!e_n(x)e^{-x} \tag{2.13}$$

where $e_n(x)=\sum_{k=0}^{n}\frac{x^k}{k!}$ is the truncated exponential series. Then,

$$\begin{aligned}&\gamma(D,(T+1)\ln Dp)\\ =&(D-1)!\left(1-\frac{e_D\big((T+1)\ln Dp\big)}{e^{(T+1)\ln Dp}}\right)=\alpha(D-1)!\,,\end{aligned} \tag{2.14}$$

where

$$\alpha=\left(1-\frac{e_D\big((T+1)\ln Dp\big)}{e^{(T+1)\ln Dp}}\right), \tag{2.15}$$

is a number between 0 and 1.

Let us define

$$g(k,x)=\frac{x^k}{k!}e^{-x}. \tag{2.16}$$

Using Stirling's formula, we have

$$\begin{aligned} g(k,x) =& \frac{e^{-x}}{\sqrt{2\pi k}}\left(\frac{ex}{k}\right)^k \\ =& \frac{1}{\sqrt{2\pi k}} e^{k(1+\ln x - \ln k)-x}. \end{aligned} \tag{2.17}$$

We can check that $g(k,x)$, as a function of k, reaches its maximum near $k = x$. Let us expand $\ln k$ near $k = x$:

$$\ln k = \ln x + \frac{k-x}{x} - \frac{(k-x)^2}{2x^2} + O((\frac{k-x}{x})^3). \tag{2.18}$$

Then,

$$\begin{aligned} g(k,x) =& \frac{1}{\sqrt{2\pi k}} e^{k(1+\ln x - \ln x - \frac{k-x}{x} + \frac{(k-x)^2}{2x^2} + O((\frac{k-x}{x})^3))-x} \\ =& \frac{1}{\sqrt{2\pi k}} e^{-k^2/x+2k-x+k(k-x)^2/(2x^2)+O((\frac{k-x}{x})^3)} \\ =& \frac{1}{\sqrt{2\pi k}} e^{-\frac{(k-x)^2}{2x}(2-k/x+O(\frac{k-x}{x}))}. \end{aligned} \tag{2.19}$$

Near the maximum $k = x$, we have

$$g(k,x) \approx \frac{1}{\sqrt{2\pi x}} e^{-(k-x)^2/2x}, \tag{2.20}$$

which is a normal distribution with mean value x and standard deviation $\sqrt{x}$. Figure 2.11 shows the comparison of the original $g(k,x)$ and its approximation. We can see that the approximation is very good.

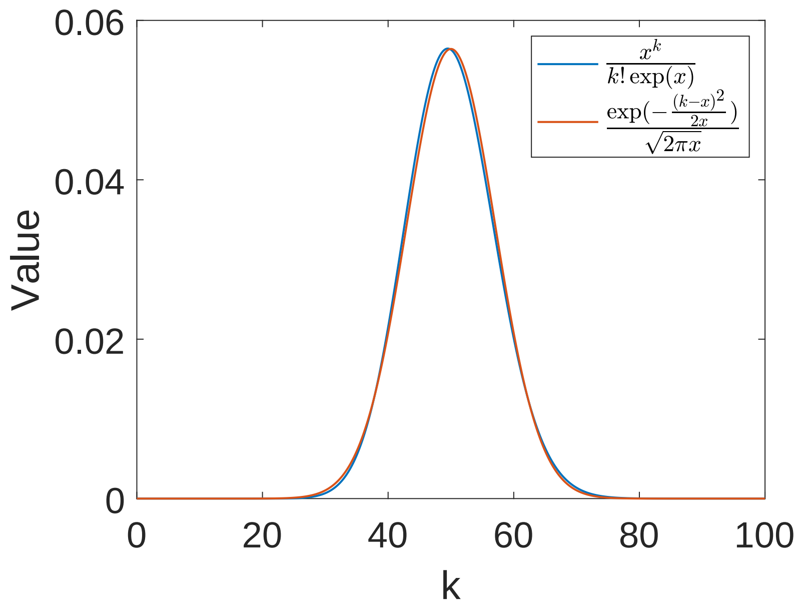


Figure 2.11. Comparison of each term in the truncated exponential series and the normal distribution. In this figure $x = 50$.

Then, we have

$$\begin{aligned}\frac{e_n(x)}{e^x} =& \sum_{k=0}^{n} g(k,x) \\ \approx& \int_0^n \frac{1}{\sqrt{2\pi x}} e^{-(k-x)^2/2x} \mathrm{d}k \\ \approx& \int_{-\infty}^n \frac{1}{\sqrt{2\pi x}} e^{-(k-x)^2/2x} \mathrm{d}k \\ =& \Phi(\frac{n-x}{\sqrt{x}}),\end{aligned} \tag{2.21}$$

where $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2} \mathrm{d}t$ is the cumulative distribution function of the standard normal distribution.

Back to Eq. (2.15), we finally have

$$\begin{aligned}\alpha =& 1 - \frac{e_D\big((T+1)\ln Dp\big)}{e^{(T+1)\ln Dp}} \\ =& 1 - \Phi(\frac{D-(T+1)\ln Dp}{\sqrt{(T+1)\ln Dp}}) \\ =& \frac{1}{2}\mathrm{erfc}(\frac{D-(T+1)\ln Dp}{\sqrt{2(T+1)\ln Dp}}),\end{aligned} \tag{2.22}$$

where erfc is the complementary error function.

In DMMs, each instanton connects two critical points whose indices (number of unstable directions) differ by 1 [71]. Since the index of a critical point equals at most the dimension D, and the index of the equilibrium point is 0, we have $T+1=D$, and Eq. (2.22) simplifies to

$$\alpha = \frac{1}{2}\mathrm{erfc}(\sqrt{D}\frac{1-\ln Dp}{\sqrt{2\ln Dp}}). \tag{2.23}$$

Therefore,

$$\langle n_{\mathrm{a},T}\rangle = \frac{(Dp)^{T+1}}{2}\left(\frac{1-p}{\ln Dp}\right)^{D}\mathrm{erfc}(\sqrt{D}\frac{1-\ln Dp}{\sqrt{2\ln Dp}}), \tag{2.24}$$

which is Eq. (2.6) in the main text.

This section, in full, is a reprint of the material as it appears in Chaos: An Interdisciplinary Journal of Nonlinear Science [2]. Yuan-Hang Zhang, Massimiliano Di Ventra, 2021. The dissertation author was the primary investigator and author of this paper.

2.3 A hardware implementation of digital MemComputing machines using standard electronic components

Digital MemComputing machines (DMMs), which employ nonlinear dynamical systems with memory (time non-locality), have proven to be a robust and scalable unconventional computing approach for solving a wide variety of combinatorial optimization problems. However, most of the research so far has focused on the numerical simulations of the equations of motion of DMMs. This inevitably subjects time to discretization, which brings its own (numerical) issues that would be otherwise absent in actual physical systems operating in continuous time. Although hardware realizations of DMMs have been previously suggested, their implementation would require materials and devices that are not so easy to integrate with traditional electronics. Addressing this, our study introduces a novel hardware design for DMMs, utilizing readily available electronic components. This approach not only significantly boosts computational speed compared to current models but also exhibits remarkable robustness against additive noise. Crucially, it circumvents the limitations imposed by numerical noise, ensuring enhanced stability and reliability during extended operations. This paves a new path for tackling increasingly complex problems, leveraging the inherent advantages of DMMs in a more practical and accessible framework.

2.3.1 Introduction

Over the past half century, the rapid development of computers has mirrored the prediction of Moore's law, with the number of transistors doubling approximately every two years [97]. As we near the physical limits of this growth, formidable challenges such as quantum tunneling, heat dissipation, and escalating production costs [98] necessitate a pivot towards novel computational paradigms. Unconventional computing, encompassing areas like quantum [99], neuromorphic [100], optical [101], and molecular computing [102], promises to not only sustain computational growth but also foster more energy-efficient and adaptable systems, capable of tackling problems currently beyond the reach of classical computers.

One key idea in unconventional computing is to harness physical principles to greatly accelerate the solution of certain problems. For instance, an ideal (decoherence-free) quantum computer could in principle solve integer factorization in polynomial time utilizing quantum entanglement and interference [103]. A new class of machines, dubbed "MemComputing" [104], leverages time non-locality (memory) to solve a wide variety of computational problems efficiently [105, 7]. In particular, their digital version (digital MemComputing machines or DMMs) have been designed to solve combinatorial optimization problems [106].

We stress here that the prefix "Mem" stands for "memory", which is generally intended as "time non-locality", not necessarily storage [107]. Time non-locality is the *non-equilibrium* property of a physical system that when perturbed, the perturbation affects the system's state at a later time [7]. Under appropriate conditions, it induces spatial non-locality [107]. This, in turn, initiates dynamical long-range order in the system [108, 109], which is exploited by DMMs to scrutinize the structure of the target problem, thereby solving it efficiently.

In the present paper we focus on DMMs [106], which have a finite set of input and output states that can be written/read with finite precision, hence are scalable to large problem sizes. The dynamics of a DMM are governed by a set of ordinary differential equations (ODEs) [106, 110], whose equilibria correspond to the solutions (if they exist) of the target problem. In continuous (physical) time, it has been proved that if such systems have equilibria (solutions), periodic orbits and chaos can be avoided [111, 112], and the convergence to the equilibrium point(s) can be reached with a number of jumps (more precisely, instantons, which are abrupt transitions in the trajectories, connecting critical points of the dynamics of different stability) that scales polynomially with problem size [108, 109].

Studies have illustrated that numerical simulations of DMM's equations can effectively solve a plethora of complex combinatorial optimization problems [106, 113, 114, 115, 110], significantly surpassing traditional algorithms in performance (see also industrial case studies performed at MemComputing, Inc. [116]). Recently, it was numerically shown that MemComputing can solve the hardest integer factorization problems with quadratic complexity up to 300 bits [117], with projections to factor 2048-bit RSA keys [118] within weeks in software.

However, the numerical simulation of ODEs inevitably necessitates time discretization, leading to the accumulation and possible amplification of numerical errors over time [87]. Given that DMMs typically require extended simulation times to reach the solution of very large problems (in the hundreds of thousand or millions of variables), the numerical errors may ultimately cause the simulations of DMMs to fail [2]. Although better numerical methods and careful control of such simulations can mitigate this issue, it comes with additional numerical overhead.

The research gap, therefore, lies in overcoming these limitations inherent in numerical simulations of DMMs. A promising solution is the construction of a *physical* DMM that operates in continuous time. Towards this objective, attempts have been made to realize DMMs using hardware components. For instance, Ref. [106] proposed a design employing resistive memories, but since these are not standard elements, their fabrication and integration into circuits could present a challenge. A similar issue pertains to the self-organizing logic gates of DMMs discussed in [119], where nanomagnets have been suggested as possible memory materials. Alternatively, Ref. [120] realized a DMM in hardware using field-programmable gate arrays, which achieved considerable speed-up compared to simulations. However, this implementation is still based on the discretization of time. As such, it essentially falls under the category of numerical simulations (albeit directly in hardware), implying that the previously mentioned challenges persist.

Addressing this gap, our work aims to design a physical, hardware-based DMM that operates in continuous time, exclusively employing *standard* electronic components. The strategy is to start from a set of ODEs describing a DMM solving, e.g., a combinatorial problem, and then look for standard hardware components that would reproduce such dynamics. As a prototypical example we consider the Boolean satisfiability (SAT) problem with three literals per clause. Any other combinatorial problem can, in principle, be mapped into it [121].

Given that we are essentially conducting hardware-based ODE simulations, our design concept draws inspiration from analog computers prevalent in the 1960s, predominantly utilizing operational amplifier (op-amp)-based circuits to perform mathematical operations.

However, a key issue still persists: by moving from software to hardware we have exchanged numerical noise for physical noise. Analog computers largely fell out of favor in the 1970s in part due precisely to physical noise, which limited their scalability. Factors like component variability in resistors and capacitors, temperature sensitivity, parasitic effects, and jitter all contribute to this physical noise, compromising the accuracy of analog signals. While careful design can mitigate some of these noise sources, they can never be completely eliminated.

While physical noise is detrimental for analog computers, they do not pose much of a problem for DMMs. This is because, just like modern digital computers, despite the use of physical signals, a DMM is a *digital* machine, since the input and output states of a DMM are finite and can be read/written with *finite precision*. Moreover, it was demonstrated that, the transition function between the input and the output states of a DMM is of topological character, making them robust to perturbative noise [108, 109]. This is key for their scalability as a function of problem size, a feature that is not shared by analog machines. In fact, it has also been shown that introducing some (Gaussian) physical noise can, in certain cases, even aid the solution [122].

One additional key distinction between physical and numerical noise is worth noting: physical noise is typically localized in space and time and does not accumulate over extended periods [7]. In contrast, numerical noise can accumulate and may even amplify exponentially over time [87]. Therefore, for large-scale, long-time simulations, a physical DMM, even with reasonably high noise levels, is expected to eventually outperform numerical simulations.

In this paper, we outline a prototype of such a physical DMM. We are aware that some of the design elements we have used may not be optimal. However, our goal was not to achieve optimality in design. Rather, we hope that our work would serve as a foundational demonstration that can inspire future developments.

This paper is organized as follows. In Sec. 2.3.2, we outline the formulation of the DMM equations and introduce a circuit that implements these equations. Results from hardware emulations using LTspice and numerical simulations with Python are presented in Section 2.3.3, and noise resistance of our model is also demonstrated. We conclude the paper with a few remarks in Sec. 2.3.4. The technical details necessary to reproduce our paper can be found in the Appendix.

2.3.2 Solving 3-SAT with MemComputing

In a SAT problem, the task is to determine whether there exists an assignment of truth values to a set of Boolean variables, such that the given Boolean equation evaluates to true. Despite its simple formulation, the solution of SAT problems is often required in various industrial applications such as circuit design, logistics, scheduling, etc. [123, 124], and an efficient solver is desirable. In particular, we focus on the three-satisfiability (3-SAT) problem, where the Boolean equation is a conjuction of clauses, each of which is a disjuction of three literals, where a literal is either a Boolean variable or its negation.

MemComputing equations for 3-SAT

The approach for solving 3-SAT problems using DMMs was already reported in [110], where the dynamics of the DMM was written as a set of ordinary differential equations (ODEs) and solved numerically. For an easier hardware implementation, we slightly modified the equations used in Ref. [110]. For a 3-SAT problem with *N* variables and *M* clauses, the resulting equations are presented below:

$$\dot{v}_n = \sum_m \Big(\eta \,\text{softmax}(\mathbf{x}_l^n)_m x_{s,m} G_{n,m}(v_n, v_j, v_k) \tag{2.25}$$
$$+ (1 + \zeta \eta \,\text{softmax}(\mathbf{x}_l^n)_m)\,(1 - x_{s,m})\, R_{n,m}(v_n, v_m, v_k) \Big)$$

$$\dot{x}_{s,m} = \beta\,(x_{s,m} + \varepsilon)\,\big(C_m(v_i, v_j, v_k) - \gamma\big)\,, \tag{2.26}$$

$$\dot{x}_{l,m} = \alpha e^{-x_{l,m}}\,\big(C_m(v_i, v_j, v_k) - \delta\big)\,, \tag{2.27}$$

$$C_m = 1 - \max(\tilde{v}_{i,m}, \tilde{v}_{j,m}, \tilde{v}_{k,m}), \tag{2.28}$$

$$G_{n,m} = q_{n,m} C_m(v_n, v_j, v_k), \tag{2.29}$$

$$R_{n,m} = \begin{cases} q_{n,m} C_m(v_n, v_j, v_k), \\ \qquad \text{if } C_m(v_n, v_j, v_k) = 1 - q_{n,m} v_n, \\ 0, \quad \text{otherwise.} \end{cases} \tag{2.30}$$

$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{2.31}$$

Here, $v_n \in [0,1]$ represent the continuously relaxed Boolean variables in the 3-SAT problem ($n = 1, \cdots, N$), while $x_{l,m} \in [0, M]$ and $x_{s,m} \in [0,1]$ are the long-term and short-term memory variables, respectively ($m = 1, \cdots, M$). Whenever a variable exceeds the bounds, it is reset to the bound value, which is handled by the circuit detailed in the next section. $\alpha, \beta, \gamma, \delta, \varepsilon, \zeta$ and η are constants.

Each clause is represented by the clause function, $C_m(v_i, v_j, v_k)$, indicating that the i-th, j-th and k-th Boolean variables are present in the m-th clause. If the i-th literal in the m-th clause is negated, then $q_{i,m} = -1$ and $\tilde{v}_{i,m} = 1 - v_i$, otherwise $q_{i,m} = 1$ and $\tilde{v}_{i,m} = v_i$. The definition of C_m in Eq. (2.28) gives it the range [0, 1], and the clause is closer to satisfaction if C_m is closer to 0.

The gradient-like term, $G_{n,m}$ (Eq. (2.29)), aims to push all literals in a clause towards satisfaction uniformly, while the rigidity term, $R(n,m)$ (Eq. (2.30)), tries to hold the literal closest to being satisfied in place. The short-term memory, $x_{s,m}$, functions to alternate between the gradient-like and the rigidity terms, while the long-term memory, $x_{l,m}$, assigns a dynamic weight to each of the clauses. A more comprehensive interpretation of each of these parameters and terms can be found in [7, 110]. Here, the parameters are chosen to be $\alpha = 5, \beta = 20, \gamma = 0.25, \delta = 0.05, \varepsilon = 10^{-3}$, and $\eta = 3000$. The optimal ζ decreases according to a power law in a logarithmic scale with system size N, and the details can be found in Appendix 2.3.5.

Comparing to the original equations in Ref. [110], we adjusted Eq. (2.25) by rescaling and computing the softmax of the long-term memories $\mathbf{x}_l$. This bounds the range of the multiplier, making the circuit implementation easier.

Hardware implementation of the equations

Eqs. (2.25)-(2.30) contain basic arithmetic operations, exponentiation, maximum and softmax functions. Linear operations can be implemented using op-amp based arithmetic circuits, while the exponential function can be realized using the exponential relation between a bipolar junction transistor's emitter current and base voltage. We will now illustrate how to realize these ODEs using these basic components, with the detailed design of each operation provided in 2.3.5.

The variables v_n, $x_{s,m}$ and $x_{l,m}$ are represented as voltages across capacitors, and their time derivatives are simulated using currents charging and discharging the capacitors.

Fig. 2.12 presents the circuit implementing the short-term memory variable, x_s. The value of x_s is represented by the voltage across a 10 nF capacitor, C_1. A very large resistor, R_1, is connected in parallel to C_1 for stability, and its impact on the dynamics is negligible. A voltage-controlled current source, G_1, charges C_1. The control voltage, dx_s, which is the right-hand side of Eq. (2.26), is pre-computed in another segment of the circuit.

Before controlling the current source G_1, dx_s is first passed into a bidirectional switch regulated by two control signals, ctrl$_+$ and ctrl$_-$. Each of these signals selectively allows or blocks input signals of a specific polarity. As x_s is confined within the [0, 1] range, we use two open-loop operational amplifiers functioning as comparators to maintain these bounds, generating the control signals. Specifically, ctrl$_\pm$ equals 5V when the voltage across C_1 falls within the [0V, 1V] range and -5V when the voltage surpasses its respective limit.

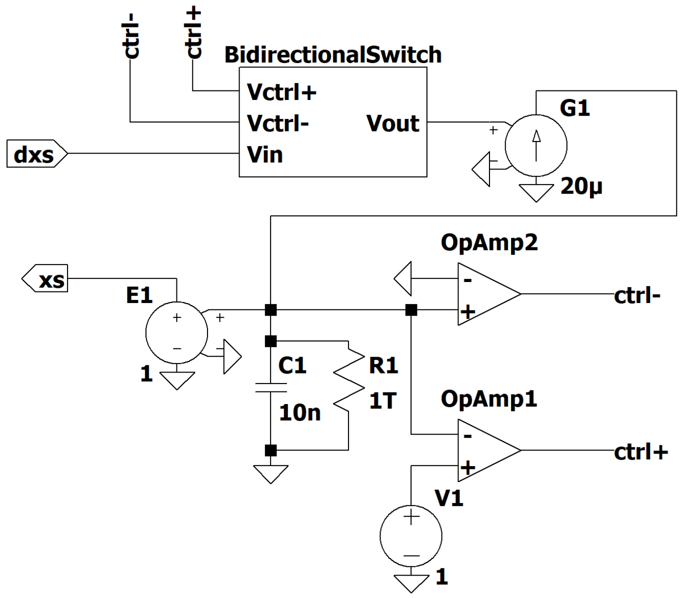


Figure 2.12. Circuit for the implementation of the short-term memory variable, x_s. The circuit takes in the pre-computed voltage, dx_s, integrates it, and subsequently produces the updated value of x_s.

In operation, when the voltage representing x_s remains within the bounds (0V to 1V), the bidirectional switch allows the input signal, dx_s, to pass through. This signal influences the current source G_1, which then charges or discharges the capacitor C_1 as per Eq. (2.26). If x_s exceeds 1V, the control signal ctrl$_+$ becomes negative, prompting the bidirectional switch to block positive dx_s signals, thus allowing only negative dx_s to pass. This mechanism enforces the upper boundary for x_s. The same process, in reverse, handles any voltage drops below 0V to uphold the lower boundary.

To maintain the capacitor's charge and avoid any additional input or output current to C_1, we use a voltage-controlled voltage source with unity gain, E_1, for isolation and to output the value of x_s. This isolation can be practically achieved through a voltage follower circuit made with a single op-amp.

To implement Eq. (2.25) for the variables v_n, we use a similar setup, depicted in Fig. 2.13. As each literal in 3-SAT can be either negated or not, this circuit simultaneously generates two values, v and $\bar{v} = 1 - v$. Likewise, inputs are also grouped by their polarities (the sign mismatches in Fig. 2.13 result from the inverted adders in the previous step). Although the constant $\eta = 3000$ in Eq. (2.25) (represented by $G_1 = 3\text{mA}/\text{V}$ in Fig. 2.13) renders the rigidity term without it seemingly negligible (represented by $G_2 = 1\mu\text{A}/\text{V}$ in Fig. 2.13), numerical experiments corroborate that the inclusion of a small rigidity term aids in convergence towards the solution.

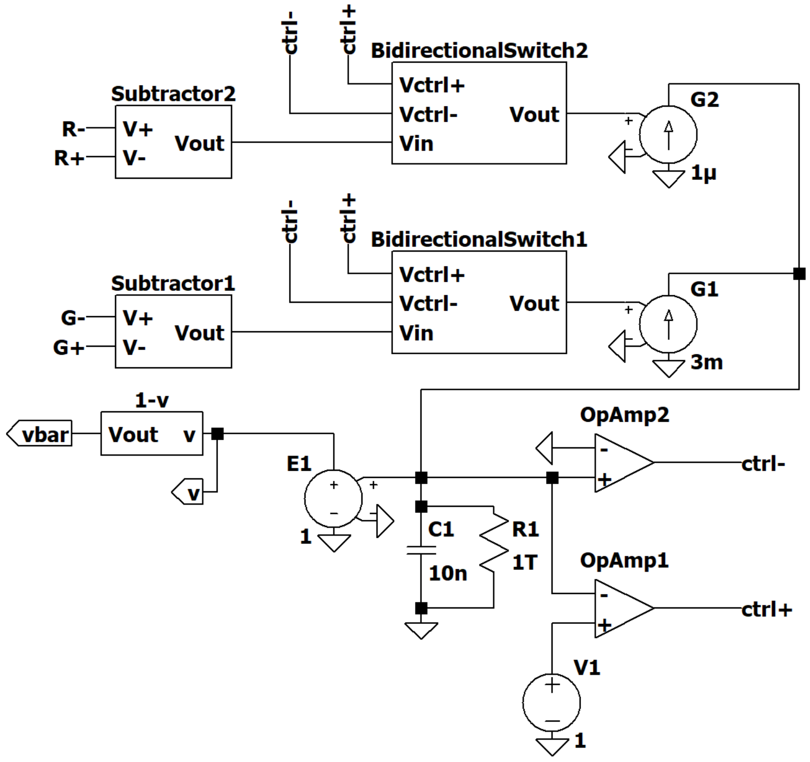


Figure 2.13. Implementation of the voltage dynamics, Eq. (2.25), with a built-in negation mechanism.

Implementing the long-term memory dynamics accurately, as per Eq. (2.27), requires more consideration due to the variability of the exponential term across several orders of magnitude. To address this, we choose to work in the logarithmic space and rewrite Eq. (2.27) as:

$$\dot{x}_{l,m} = \alpha\left(\exp(\log(C_m+\lambda)-x_l)-\exp(\log(\delta+\lambda)-x_l)\right). \tag{2.32}$$

Given that $C_m - \delta$ can be either positive or negative, taking the logarithm directly would pose issues. Therefore, we split it into two terms, $C_m+\lambda$ and $\delta+\lambda$, with $\lambda = 0.1$ to ensure both terms are always positive. We then evaluate each product separately using log-sum-exp computations, and compute their difference to restore Eq. (2.27). Necessary constants are incorporated to rescale the signals to suitable ranges, and the details can be found in 2.3.5.

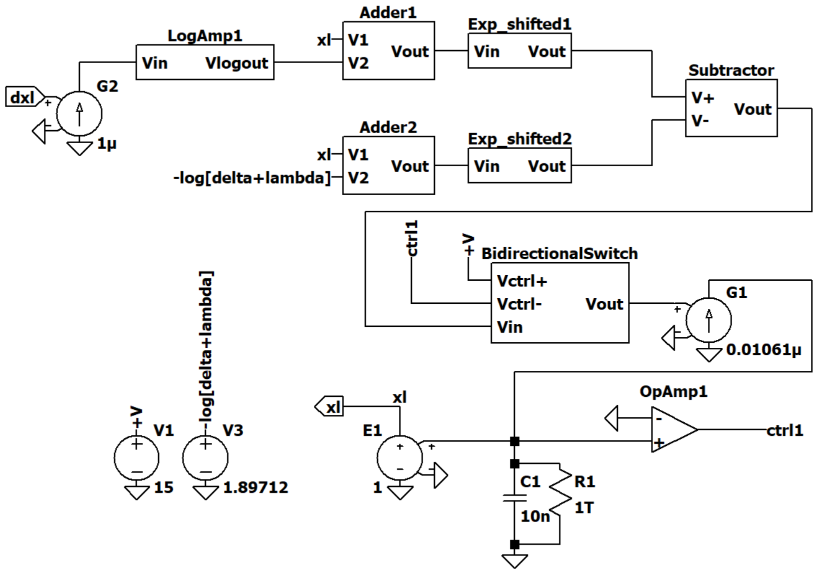


Figure 2.14. Implementation of the long-term memory dynamics, Eq. (2.27). Here, the input $dx_l = C_m + \lambda$, and log-sum-exp computation is employed to restore Eq. (2.27). Note that our implementation of the logarithm and exponential amplifiers contains a negative sign (see 2.3.5).

Finally, Fig. 2.15 shows the evaluation of the clause function, gradient-like and rigidity terms, and how they contribute to the computation of the time derivatives in Eqs. (2.25)-(2.27). The comparator module initially calculates the maximum of the three input (possibly negated) voltages, $v_{\max}$, along with three control signals, b_1, b_2 and b_3, indicating which input voltage attains the maximum value. The clause function is then computed as $C_m = 1 - v_{\max}$, which is further employed to evaluate dx_s and dx_l according to Eq. (2.26) and Eq. (2.32). The rigidity term is calculated with the assistance of the control signals b_i, setting the specified values to 0. Finally, the right-hand-side of Eq. (2.25) is divided into two parts, $dv_{n,1} = x_{s,m}G_{n,m} + \zeta(1 - x_{s,m})R_{n,m}$, which is later multiplied with the softmax module (Fig. 2.24); and $dv_{n,2} = (1 - x_s)R_{n,m}$, which is added at the end. This final processing step is not depicted in Fig. 2.15.

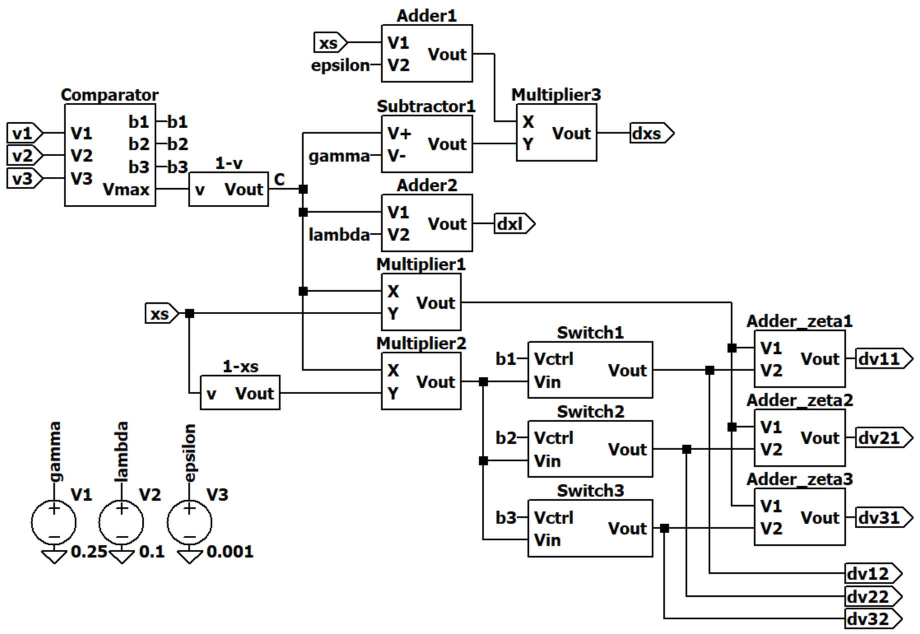


Figure 2.15. The module computing the time derivatives of the variables. It accepts the possibly negated variables, v_1, v_2 and v_3, and the short-term memory x_s, and outputs the intermediate results of their time derivatives, dx_s, dx_l, $dv_{n,1}$ and $dv_{n,2}$.

By integrating all the modules in accordance with the topology of the target 3-SAT problem, we successfully develop a continuous-time, hardware-accelerated DMM. We stress here that in large-scale applications, the topology, or connectivity, of the 3-SAT problem (and its conjunctive normal form representation), can be programmed using cross-point switches or field-programmable gate arrays. This would allow switching from one type of problem to another with the same type of hardware design. Given our selection of capacitance and current source, it can be verified that one second of circuit dynamics equates approximately 100 units of time in Eqs. (2.25)-(2.30). With high-speed op-amps, it is possible to further reduce the characteristic time scale of the circuit to accelerate the solution. Limited by the numerical accuracy and convergence in our circuit emulations, we leave this as a future work.

2.3.3 Results

To investigate the accuracy and efficiency of our methods, we emulated Eqs. (2.25)-(2.30) in LTspice [125, 126] using the realistic circuit elements as we have discussed in the preceeding Sec. 2.3.2. Furthermore, we also developed a Python program to simulate the same ODEs directly using numerical solvers. The full model files are available at [127].

To generate challenging 3-SAT instances, we utilized the unbiased generator with a planted solution as proposed in [128], specifically targeting instances near the complexity peak at a clause-to-variable ratio of 4.3.

As a proof-of-concept test, we compared both methodologies using a 3-SAT problem consisting of 10 variables and 43 clauses. Fig. 2.16(a)(b) illustrates the trajectories of the first five variables. Both simulations started with identical initial conditions, and we can see that the initial part of their trajectories are the same. However, due to noise and minor discrepancies in different implementations, their trajectories soon differ. Nevertheless, despite following completely different trajectories, both simulations promptly converged to the same solution.

To better understand the noise sensitivity of our design, we introduced white noise of varying strength to all relevant voltage sources in LTspice emulations, except for those where noise is unlikely to make a significant impact (e.g., the power supplies for op-amps). Of course, this added noise compounds the numerical one which is intrinsic in the discretization of time of the simulations. The findings are displayed in Fig. 2.16(c)-(d). In these figures, the added noise strength is set at 10% of the corresponding voltage for (c), and 20% for (d). Again, we see that despite different trajectories, all simulations converge to the same solution, regardless of noise (both physical or numerical).

This example serves to illustrate the inherent robustness of DMMs against noise [7]. To understand this robustness, note that, with the values of the voltage sources entering computations via arithmetic circuits, the injected white noise can affect the parameters $\gamma, \delta, \varepsilon$ in Eqs. (2.25)-

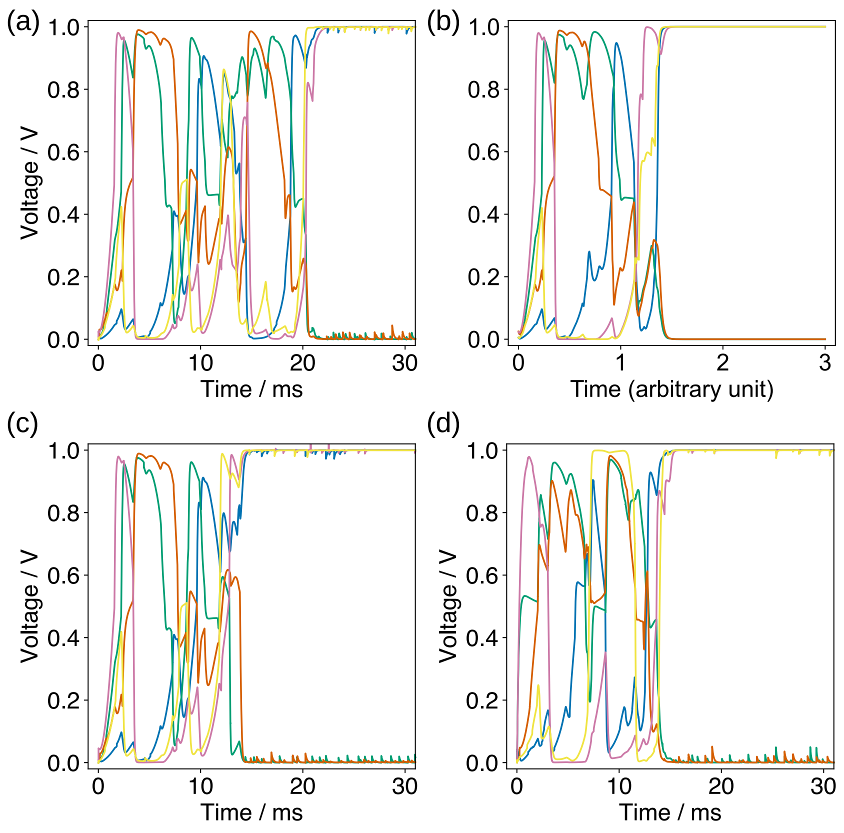


Figure 2.16. Comparison of the trajectories of the first five variables in a 3-SAT problem with 10 variables and 43 clauses for (a) LTspice circuit emulation, with no noise added, (b) Python numerical simulation, (c) LTspice circuit emulation, with 10% white noise, and (d) LTspice circuit emulation, with 20% white noise. With the same initial conditions, the trajectories are initially similar, but soon differ due to differences in noise (both physical and numerical). However, despite the paths are different, eventually they all converge to the same solution.

(2.30), but cannot change the equations' functional form. According to our analysis in Appendix 2.3.5, although an optimal set of parameters exist, the DMM still works consistently for a wide range of parameters and always converge to the solution if the parameters do not deviate too much from their optimal values.

In fact, as observed in Fig. 2.16, minor perturbations can cause the trajectory to diverge significantly. Yet, all paths ultimately converge towards the solution of the problem, demonstrating the model's resilience. Furthermore, this example also affirms the functional equivalence between the physical circuit built using realistic elements and the numerical Python-based implementation.

To extend the scope of our assessment, we generated additional 3-SAT instances with varying problem sizes and solved them through the DMM circuit simulation. Given the substantial resource consumption of realistic circuit emulations in LTspice, we opted for the Python simulation in this test, which allowed us to simulate much larger systems.

Fig. 2.17(a) illustrates how the median integration time (in arbitrary units) scales with the number of variables in the 3-SAT problem. These problems were generated near the complexity peak with a clause-to-variable ratio of 4.3. Each data point was determined by solving 100 distinct 3-SAT instances, with the median being recorded once 51 of them were solved. A power-law fit of the median solution time against the number of variables yielded an exponent of 2.23 ± 0.17. This result marginally outperforms the findings reported in [110].

Similarly, Fig. 2.17(b) displays the median wall time, where 100 instances are simulated in parallel on a single NVIDIA TITAN RTX GPU, and Fig. 2.17(c) shows the average wall time per integration step. For smaller cases, the program initialization time dominates, resulting in a nearly constant wall time and a greater wall time per step. However, as the number of variables grows, the initialization time becomes insignificant, and the wall time per step increases linearly with the problem size.

Finally, Fig. 2.17(d) computes the ratio between the median integration time and the median wall time, which indicates how many units of time we can integrate per real-world second.

In our previously detailed hardware implementation, one second corresponds to approximately 100 units of time (as represented by the red dotted line). Compared to this, the ratio in the Python simulation reaches a peak of around 100 and gradually diminishes as the system size expands. If we assume the hardware solution shares the same scaling properties of the numerical simulation, this means that for a 3-SAT problem with 2000 variables, our circuit design would achieve an 8-fold acceleration compared to the numerical simulations.

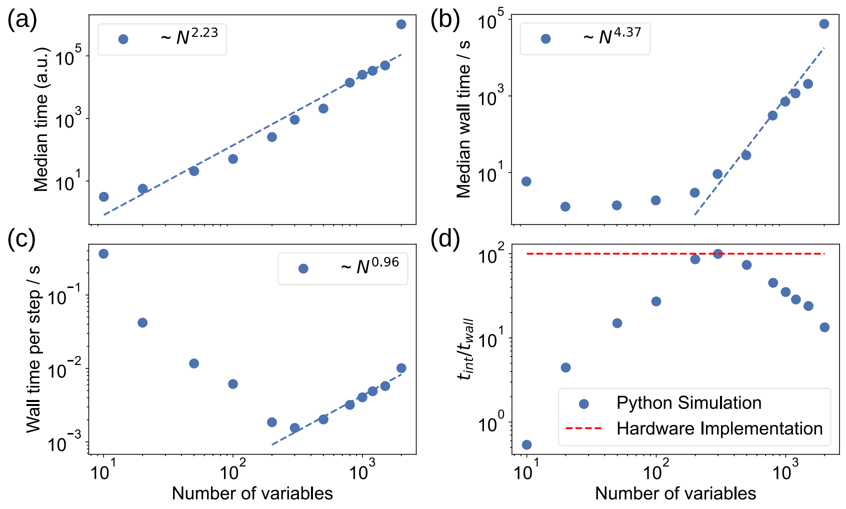


Figure 2.17. Scaling of solution time versus the number of variables. (a) Median integration time (in arbitrary units) as a function of the number of variables in the 3-SAT problem, which were generated near the complexity peak at a clause-to-variable ratio of 4.3. The curve approximates a power-law fit, yielding an exponent of 2.23 ± 0.17. (b) The progression of the median wall time for solving problems, exhibiting near-constant times for smaller problems and scaling approximately as $N^{4.37\pm0.55}$ for larger problems. (c) Wall time per integration step. Excluding the initialization time, this grows linearly with the system size. (d) The ratio of integrated time to real-world seconds, peaking around 100 for the Python simulation, and declining as the system size expands. Excluding the smaller cases where the Python simulation's initialization time is dominant, our hardware-accelerated DMM would achieve up to an $8\times$ speed increase compared to the numerical simulations.

However, we expect the hardware solution to scale better compared to numerical simulations [7]. In the Python simulations presented above, the discretized time step Δt is of the order of 10^{-1}. For more challenging problems, Δt needs to be reduced, resulting in an increase in the number of integration steps needed [110]. However, our envisioned hardware design, which operates in continuous time, integrates the ODEs at a consistent rate, regardless of the problem size or difficulty. This intrinsic characteristic of the hardware, once practically implemented and verified, is expected to lead to substantial speed improvements.

Moreover, since numerical noise is eliminated in hardware, we expect this approach to offer enhanced stability over extended periods of dynamics. This is another key advantage of our hardware-based approach over numerical simulations.

Despite promising results depicted in Fig. 2.17, the initial Python simulations did not account for physical imperfections that are typically present in realistic settings, such as component tolerance, capacitor leakage, op-amp input offset voltage, and temperature variations. These factors are crucial as they can significantly influence the dynamics and reliability of our system.

Among these factors, the tolerance of resistors stands out as potentially the most impactful on the system dynamics. Consider, for example, the adder circuit shown in Fig. 2.19, which is governed by the equation:

$$V_{\text{out}} = \frac{V_1 R_2 + V_2 R_1}{R_1 + R_2} \cdot \frac{R_3 + R_4}{R_4}. \tag{2.33}$$

With $R_1 = R_2 = R_3 = R_4$, we have

$$\delta V_{\text{out}} = \frac{1}{2} V_1 \left(-\frac{\delta R_1}{R_1} + \frac{\delta R_2}{R_2} + \frac{\delta R_3}{R_3} + \frac{\delta R_4}{R_4} \right) + \frac{1}{2} V_2 \left(\frac{\delta R_1}{R_1} - \frac{\delta R_2}{R_2} + \frac{\delta R_3}{R_3} + \frac{\delta R_4}{R_4} \right). \tag{2.34}$$

In practical terms, if each resistor has a tolerance of 1%, this would result in an approximate 2% error in the computed value of V_{out}. Such errors are likely to accumulate through cascaded addition and multiplication operations, significantly magnifying deviations from the ideal dynamics.

On the other hand, temperature variations affect the thermal voltage, $V_T = k_B T / q$, which influences the characteristics of transistors crucial to the design of logarithmic amplifiers and multipliers. Appropriate designs incorporate temperature compensations based on paired transistors, cancelling out the impact of temperature fluctuations across a broad range of operating temperatures, rendering it negligible compared to the influence of resistor tolerance. Nonetheless, the error induced by temperature variations is multiplicative, similar to that caused by resistor tolerance, and we model them together, as detailed below.

Comparatively, factors such as capacitor leakage and op-amp input offset voltage present less risk to the system's dynamics. For instance, capacitor leakage currents are generally around $10^{-2}\mu\text{A}/(\mu\text{F}\cdot\text{V})\cdot CV$ for aluminum electrolytic capacitors [129]. Similarly, typical input offset voltages for op-amps are on the order of 100μV. These values are relatively insignificant when compared to the operational voltages and currents in our system, thereby having a lesser impact on the system's performance.

To validate our system's resilience against physical imperfections, we incorporated simulations of component tolerance and capacitor leakage into our Python model. We modeled resistor tolerance and temperature variation together as a multiplicative noise, η, affecting each addition and multiplication operation described in Eqs. (2.25)-(2.30). Since resistor tolerance is dominant, based on Eq. (2.34), η is approximately twice the actual resistor tolerance. Since an error η is present in every addition and multiplication operation, $\eta = 1\%$ will result in an estimated 5% error in the computed derivatives.

Capacitor leakage was integrated as an additional term in Eqs. (2.25)-(2.27), modifying the derivatives according to:

$$\tilde{\dot{v}}_n = \dot{v}_n - \kappa v_n \tag{2.35}$$

and similarly for x_l and x_s. We selected $\kappa = 10^{-3}$, representing a leakage current $I_{\text{leak}} = 10^{-1}\mu\text{A}/(\mu\text{F}\cdot\text{V})\cdot CV$, which is ten times the standard rate for aluminum electrolytic capacitors [129].

The simulation results for low noise levels, $\eta = 0$ and 0.01, are shown in Fig. 2.18. We observe that such minimal noise levels do not impair the solution capabilities of our system,

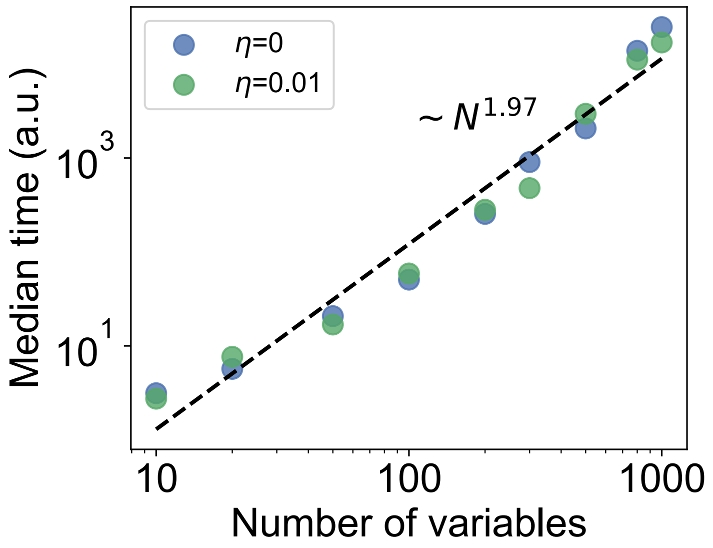


Figure 2.18. Median integration time (in arbitrary units) as a function of the number of variables, N, for 3-SAT problems at clause-to-variable ratio 4.3. This simulation incorporates an error η in every addition and multiplication operation, attributable to resistor tolerances, and a capacitor leakage current $I_{\text{leak}} = 10^{-1}\mu\text{A}/(\mu\text{F}\cdot\text{V})\cdot CV$. For each data point, at least 51 out of 100 distinct 3-SAT instances are solved to calculate the median integration time. The simulation here indicates that capacitor leakage and a small resistor tolerance has a minimal impact on the dynamics, and the median integration time scales essentially quadratically with the problem size.

which continues to solve hard 3-SAT problems effectively. Importantly, the median time-to-solution scales essentially quadratically with the problem size, indicating robust performance under these conditions.

In Appendix 2.3.5, we present the complete scaling curves for η values up to 0.2 and variable counts (N) up to 1500. The results demonstrate that our system maintains reliable performance in solving hard 3-SAT problems for $N \leq 500$, even at higher noise levels. However, as expected, beyond this threshold, the median solution time starts to exhibit exponential growth for much larger η. These findings highlight the robustness of our system's design, even under moderate noise conditions.

2.3.4 Conclusion

In this paper, we have introduced a hardware design of a DMM that leverages only standard electronic components, which can be readily built using available technology, without the need of special materials or devices. By means of both LTspice and Python simulations, we have validated the reliability of our approach in tackling difficult 3-SAT problems even in the presence of physical noise under realistic conditions. Operating in continuous time, this hardware methodology completely removes numerical noise, aligning closely with the *physical* DMM concept, which has been suggested as a powerful alternative to solve hard combinatorial optimization problems [106].

In finalizing our study, it is crucial to acknowledge the potential disparities between our simulated design and its real-world hardware implementation. Factors like parasitic effects at high frequencies and the distinct characteristics of physical noise all pose challenges often unaccounted for in simulations. These elements highlight the imperative for rigorous experimental validation and adjustments to ensure the practical robustness and performance of our circuit design. Moving forward, our future research will focus on the experimental implementation of this design, aiming to bridge the gap between simulations and real continuous-time dynamics.

2.3.5 Appendix

Details of the circuit design

In this Appendix, we present details of the hardware design to realize Eqs. (2.25)-(2.30), illustrating the individual building blocks of the circuit that performs different arithmetic functions.

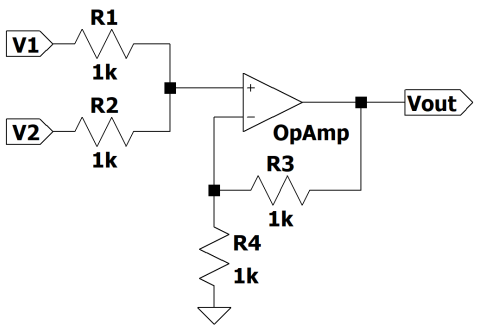


Figure 2.19. The adder circuit, realized using an op-amp feedback loop. It computes the function $V_{\text{out}} = V_1 + V_2$.

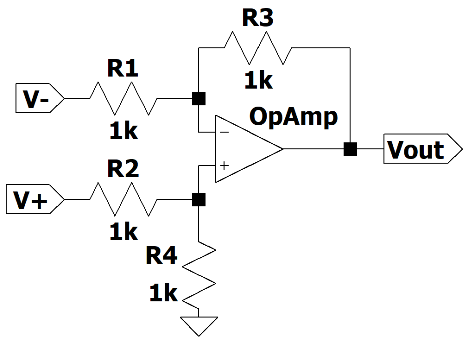


Figure 2.20. The subtractor circuit. Computes the function $V_{\text{out}} = V_+ - V_-$.

Fig. 2.19, 2.20 illustrate the circuit designs for an adder and a subtractor, respectively. These are conventional designs that leverage the feedback loop of an op-amp. The circuit architecture for a multiplier is more intricate, and we have chosen to use the commercially available model, AD834, by Analog Devices [130]. Figure 2.21 demonstrates the external circuit connections using the AD834 chip. This chip calculates the product of two input voltages, X and

Y, and the result is rendered as a current, W, according to the equation:

$$W = \frac{XY}{(1\text{V})^2} \times 4\text{mA}. \tag{2.36}$$

We have employed an op-amp feedback loop to translate the output current back into a voltage signal. As a result, the complete transfer function of the multiplier module is given as:

$$V_{\text{out}} = \frac{XY}{1\text{V}}. \tag{2.37}$$

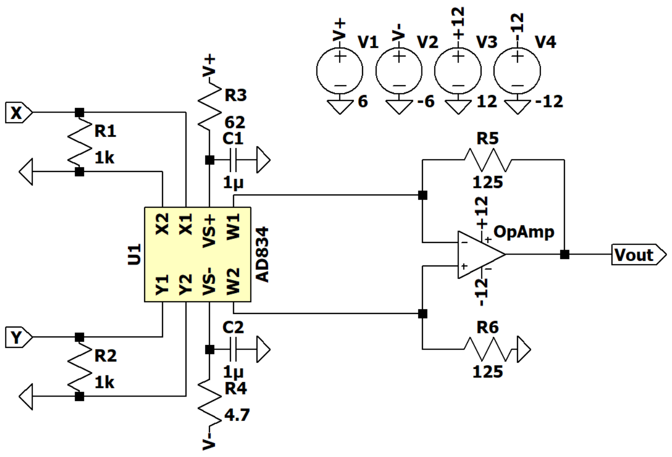


Figure 2.21. The multiplier circuit, constructed using the commercially available AD834 chip. It computes the function $V_{\text{out}} = XY/1\text{V}$.

Fig. 2.22 showcases the design of the logarithm amplifier. This particular circuit computes the logarithm function as follows:

$$V_{\text{logout}} = (-0.375\text{V})\log_{10}\frac{V_{\text{in}}}{1\mu\text{A}}. \tag{2.38}$$

Again, we have made use of a commercially available model, LOG114, produced by Texas Instruments [131]. The circuit calculates the logarithm by utilizing the exponential relation between the emitter current and base voltage of a bipolar junction transistor (BJT). Two matching

BJTs are used within this circuit design to effectively cancel out any temperature dependencies.

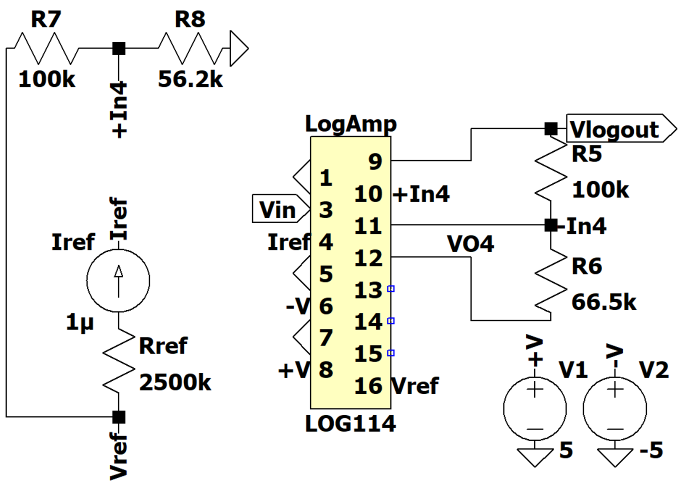


Figure 2.22. The logarithm amplifier, implemented using the commercially available design, the LOG114 chip. It computes the function $V_{\text{logout}} = (-0.375\text{V})\log_{10}(V_{\text{in}}/1\mu\text{A})$.

Fig. 2.23 shows the design of the anti-log amplifier, which computes the exponential function,

$$V_{\text{out}} = 30\text{mV}\exp\left(-\frac{V_{\text{in}}}{30\text{mV}}\right). \tag{2.39}$$

Due to the absence of commercially suitable designs for the anti-log amplifier, we had to implement this circuit ourselves. The final design bears similarity to the internal structure of the LOG114 circuit, but has been appropriately modified to accommodate for the exponential function.

Fig. 2.24 presents a specially designed circuit for computing the softmax function, given by $\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$. This circuit design builds upon pre-existing models [132, 133]. Rather than calculating multiple exponential functions, an array of BJTs with common emitters are employed. By setting a constant sum for the emitter currents, each base voltage regulates the distribution of current that flows through each respective BJT. The resulting expression precisely matches the softmax function. The current is then converted into a voltage measurement by observing the voltage drop across a resistor, after which a subtractor generates the desired final

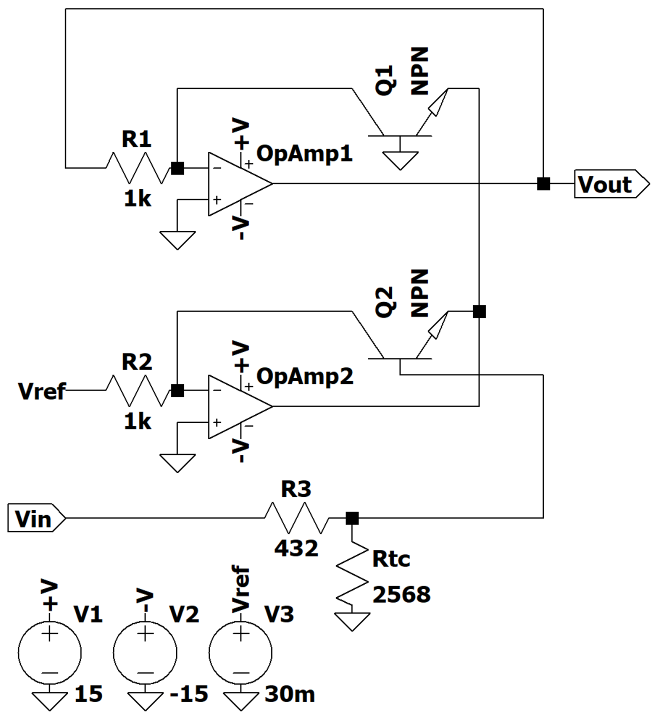


Figure 2.23. The anti-log amplifier. It computes the function $V_{\text{out}} = 30\text{mV}\exp(-V_{\text{in}}/30\text{mV})$.

output. The transfer function of this circuit can be expressed as:

$$y_i = (1V)\exp(x_i/V_T)/\sum_j \exp(x_j/V_T) \tag{2.40}$$

where $V_T = \frac{k_B T}{e}$ denotes the thermal voltage. Assuming a standard room temperature of $T =$ 25°C, $V_T = 25.68$mV.

This circuit design permits the computation of the softmax function with an arbitrary number of inputs by incorporating additional BJTs configured in the same manner. It is crucial to note, however, that for the circuit to operate correctly, no current can flow into the z_i ports of Fig. 2.24. This isolation can be achieved with the use of a voltage follower - an op-amp configured with unity gain - which does not affect the voltage output.

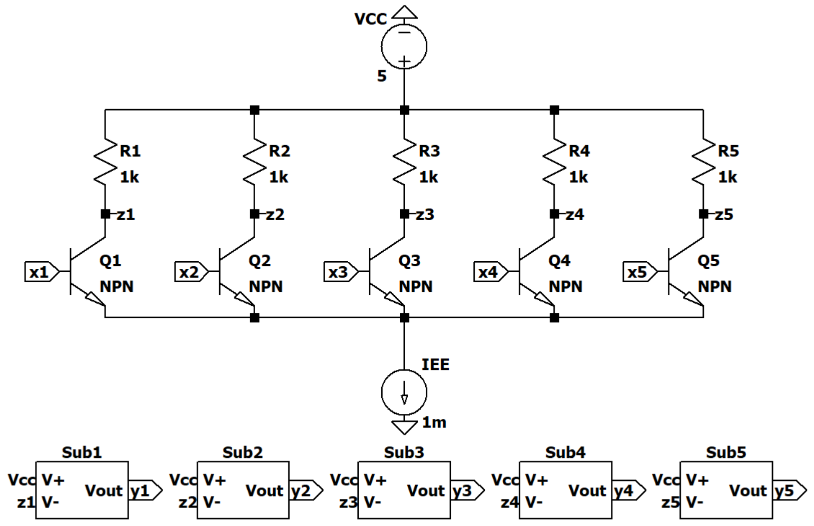


Figure 2.24. The circuit computing the softmax function. It outputs $y_i = (1V)\exp(x_i/V_T)/\sum_j \exp(x_j/V_T)$, where $V_T = k_B T/e$ is the thermal voltage.

To compute the maximum function, we employed the comparator module depicted in Fig. 2.25. This circuit is composed of two subcircuits. The feedback loop, which includes op-amps 1, 3, 5 and the three diodes, calculates the maximum voltage from the inputs V_1, V_2 and V_3, and outputs this value at the terminal $V_{\max}$. This maximum voltage, $V_{\max}$, is then fed back into three open-loop op-amps for comparison with the three input voltages. The results of these comparisons are output as b_1, b_2 and b_3, each of which indicates whether its corresponding input voltage is the maximum.

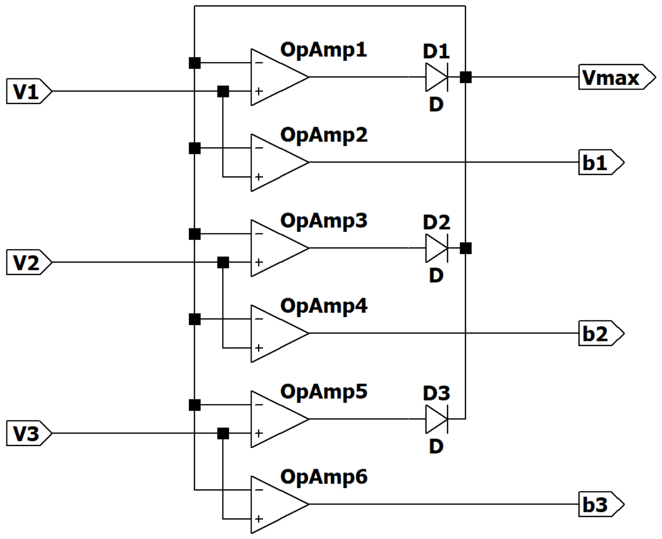


Figure 2.25. The comparator module. This circuit calculates $V_{\max} = \max(V_1, V_2, V_3)$, and the control signals b_i. If V_i is the maximum, then $b_i = V_{\max} + V_D$, where V_D is the voltage drop across the diodes. Otherwise, $b_i = -5$V.

Fig. 2.26 illustrates the design of a bidirectional switch with two control signals. This mechanism modulates the current flow using the unidirectional conductivity properties of diodes and MOSFETs. Each MOSFET governs the signal flow in a specific direction. By independently managing the two opposing MOSFETs, we can selectively permit signals of certain polarities to pass through. This can be leveraged to set boundaries on a variable. For instance, positive currents to a capacitor can be blocked when its voltage surpasses the upper limit, while negative currents can be inhibited when the voltage falls below the lower threshold.

Determining the optimal parameters

The equations presented in Eqs. (2.25)-(2.30) include seven constant parameters: α, β, γ, δ, ε, ζ, and η. These parameters can be fine-tuned to accelerate the numerical simulations. Following the guidelines from Ref. [110], we fixed $\alpha = 5$, $\beta = 20$, $\gamma = 0.25$, $\delta = 0.05$, and

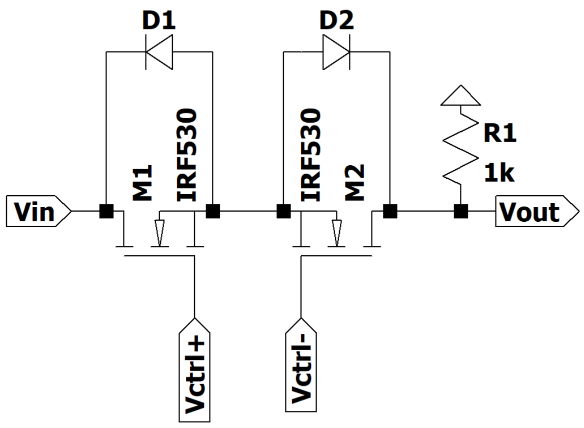


Figure 2.26. The bidirectional switch. It allows a positive signal to pass when $V_{\text{ctrl}+}$ is positive and a negative signal to pass when $V_{\text{ctrl}-}$ is positive. Conversely, it blocks a positive signal when $V_{\text{ctrl}+}$ is negative and a negative signal when $V_{\text{ctrl}-}$ is negative. The resistor R_1 facilitates the current flow when V_{out} is connected to a large impedance.

$\varepsilon = 10^{-3}$.

To identify the optimal values for ζ, η, and the integration time step Δt, we employed Bayesian optimization, leveraging the Hyperopt library [63]. Our process entailed solving 100 3-SAT instances, each comprising 1000 variables. We aimed at the maximization of the number of solved 3-SAT instances within a predefined step limit. The optimization process delivered the following optimal values: $\eta = 3000$, $\zeta = 3 \times 10^{-3}$, and $\Delta t = 0.14$.

However, in practice, the optimal values of these parameters may be problem-specific. To simplify the analysis, we vary one parameter at a time and scrutinize how the optimal parameter fluctuates in accordance with the number of variables in the problem.

Fig. 2.27(a) plots the number of successfully solved instances against Δt for a set of 100 3-SAT problems, each composed of 1500 variables. The curve can be approximately fitted with a Gaussian, with the peak indicating the optimal Δt. By repeating this process for varying system sizes, we derive the relationship between the optimal Δt and system size N. The results are demonstrated in Fig. 2.27(b). The optimal Δt adheres to a power-law distribution, expressed as $\Delta t_{\text{optimal}} = 0.230N^{-0.069}$.

A similar analysis is performed for the parameter ζ. Fig. 2.27(c) presents the results for $N = 1500$. By replicating this process for various N, we obtain the optimal ζ, as displayed in Fig. 2.27(d). This can be approximated with a polynomial in logarithmic space: $\ln\zeta_{\text{optimal}} = 6.83(\ln N)^{-1.10} - 6.53$. The parameter values derived from this analysis are applied to the numerical simulations discussed in the main text.

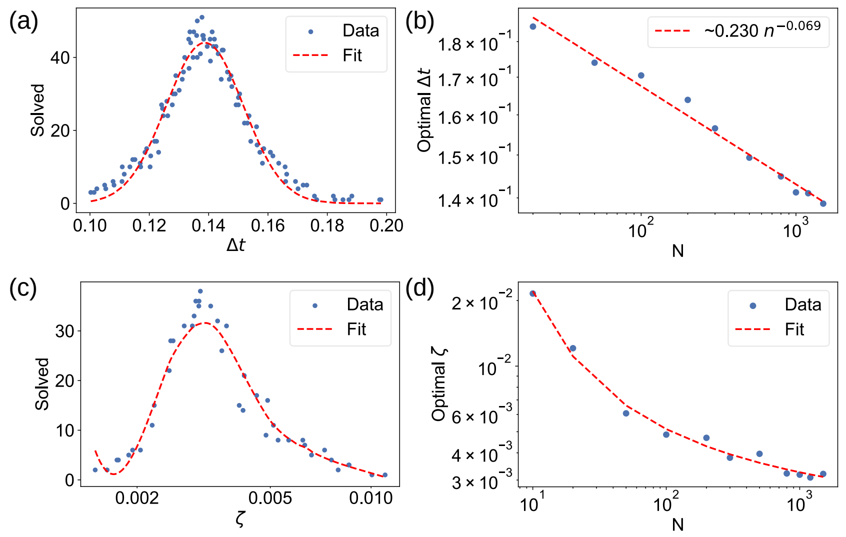


Figure 2.27. Determining the optimal parameters in the equations. (a) The number of solved 3-SAT instances vs. Δt within a given number of steps, for 100 3-SAT instances with 1500 variables. The optimal Δt is chosen to be the peak of the fit. (b) The optimal Δt as a function of the system size N: $\Delta t_{\text{optimal}} = 0.230N^{-0.069}$. (c) A similar analysis on the parameter ζ. Again, 100 3-SAT instances with 1500 variables are repeatedly solved with different parameter ζ. (d) The optimal ζ as a function of the system size N: $\ln\zeta_{\text{optimal}} = 6.83(\ln N)^{-1.10} - 6.53$.

Effects of physical imperfections

In this appendix, we explore the impact of physical imperfections, including component tolerances and capacitor leakage, on our system's ability to solve hard 3-SAT problems.

In the results section of the main text, we discussed how various physical factors affect performance and demonstrated our system's tolerance to low noise levels in resistor values in Fig. 2.18. Here, we extend this analysis with additional scaling curves in Fig.2.28, examining noise levels, η, up to 0.2 and variable counts, N, up to 1500. The experimental setup remains consistent with that described in Fig. 2.18, unless specified otherwise.

As η and N increase, we observe a significant growth in solution time. We evaluated 100 distinct 3-SAT problems for each size. To manage computational costs, simulations were terminated after 2×10^6 integration steps. If more than half of the instances are solved within this limit, the median solution time is directly calculated. If fewer than half are solved, we estimate the median solution time based on the number of solved instances at 2×10^6 integration steps, assuming a uniform distribution of solution times across all instances.

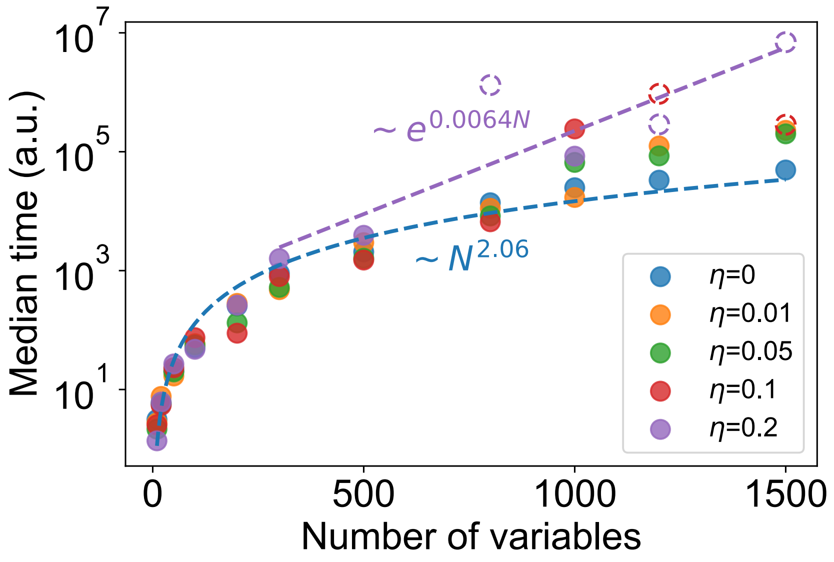


Figure 2.28. Extended analysis from Fig. 2.18, illustrating the median integration time (in arbitrary units) as a function of the number of variables, N, for 3-SAT problems with a clause-to-variable ratio of 4.3. This simulation includes an error, η, in every addition and multiplication operation due to resistor tolerances, and accounts for a capacitor leakage current of $I_{\text{leak}} = 10^{-1}\mu\text{A}/(\mu\text{F}\cdot\text{V})\cdot CV$. Solid data points represent conditions where at least 51 out of 100 distinct 3-SAT instances are solved, allowing direct calculation of the median integration time. Dashed circles indicate estimated median times for scenarios where fewer than half of the instances are resolved. The findings underscore that while capacitor leakage minimally affects the dynamics, resistor tolerances notably impact system behavior. Noise levels, η, slightly influence solution times for smaller instances ($N \leq 500$); however, as expected, for larger N, the median solution time shows exponential growth, at much higher η values.

Fig. 2.28 presents the median integration time as a function of N and η. Estimated medians for cases where fewer than half of the instances are solved are depicted with dashed circles. Remarkably, even at η values as high as 0.2—which could potentially double or halve outcomes in the computed derivatives—our system reliably solves hard 3-SAT problems for

$N \leq 500$. Beyond this threshold, however, the median solution time begins to show exponential growth for large η. Despite this, for lower η values, our results consistently demonstrate effective resolution of these challenging problems, with the median solution time displaying a quadratic scaling for N up to 1500.

This section, in full, is a reprint of the material as it appears in International Journal of Circuit Theory And Applications [3]. Yuan-Hang Zhang, Massimiliano Di Ventra, 2024. The dissertation author was the primary investigator and author of this paper.

Chapter 3

Applications

Having explored the role of memory in inducing long-range order and its evolution into the MemComputing paradigm, we now turn our attention to practical applications where this paradigm excels.

Machine learning stands out in this regard. Over the past decade, machine learning has proven to be a powerful tool for efficiently solving diverse problems across multiple disciplines, from natural language processing and autonomous driving to the discovery of new drugs and proteins. With the help of MemComputing, we examine a technique known as ”mode-assisted training” [134, 135], illustrating how it can significantly expedite the training of certain types of neural networks.

One particular area of application is quantum many-body problems, which are known for their exponentially growing Hilbert spaces that render brute-force solutions impractical for large systems. Recent advancements have shown that machine learning can address these challenges through effective dimensionality reduction [136]. We will demonstrate how MemComputing, especially through the mode-assisted training algorithm, can be utilized to efficiently solve a variety of problems in quantum many-body systems.

3.1 Quantum state tomography with mode-assisted training

Neural networks (NNs) representing quantum states are typically trained using Markov chain Monte Carlo based methods. However, unless specifically designed, such samplers only consist of local moves, making the slow-mixing problem prominent even for extremely simple quantum states. Here, we propose to use mode-assisted training that provides global information via the *modes* of the NN distribution. Applied to quantum state tomography using restricted Boltzmann machines, this method improves the quality of reconstructed quantum states by orders of magnitude. The method is applicable to other types of NNs and may efficiently tackle problems previously unmanageable.

3.1.1 Introduction

With the ability to compress and extract information from high-dimensional data, machine learning has become a useful tool in a wide variety of fields [137]. Physics is no exception.

For instance, neural networks (NNs) have been used with reasonable success as variational wavefunctions of quantum many-body systems [136, 138, 139, 140, 141, 142, 143, 144, 145]. Irrespective of the type of NN employed as variational state, the vast majority of methods to train NNs rely on Markov chain Monte Carlo (MCMC) sampling [137], which, unless specifically designed, only consists of local moves. As a result, the slow-mixing problem arises, significantly slowing down the algorithm, sometimes causing the training to fail completely, even for very simple systems. Countless efforts have been devoted to solving this problem, and various improved MCMC routines have been proposed, aiming at accelerating the mixing of the Markov chain [146, 147, 148, 149, 150, 151]. Yet, they all serve one purpose: to increase the quality of the MCMC samples for a more accurate gradient estimation.

In fact, the cost function of an NN defines a non-convex landscape, and as any non-convex landscape, convergence to the global minimum with gradient-based methods can never be guaranteed. In some cases this may not be of concern, since proper design of the NN, weight initialization and/or learning rate scheduling empirically seem to guarantee a smooth convergence to some local minimum, close enough to the global one. However, as we will show below, conventional sampling methods can easily lead to bad local minima for certain types of quantum states with strongly non-local features, which could become a serious problem, causing complete failure of the training.

In this paper, we tackle this issue from a new perspective: we design an *off-gradient* training step (i.e., a training step that does not align with the direction of the gradient), constructed using the *mode* of the NN distribution, which we call mode-assisted training [134, 135]. This method supplements the regular gradient descent with mode updates, which explicitly inject global information to the training process, leading to better estimations of the global minimum.

As an example of NNs, we will employ the well-known restricted Boltzmann machines (RBMs), and focus on the challenging task of reconstructing a quantum state with repeated measurements on its identical copies. This is called quantum state tomography (QST) [152, 153]. While traditional, brute-force methods require tens of thousands of measurements to reconstruct even small quantum states [154], recent advancements in machine learning methods have greatly improved the efficiency of such a task, making it feasible to perform QST on states with tens or even hundreds of qubits [140, 142, 155, 144]. Yet, as we will show below, such methods are still inefficient when the quantum states showcase strongly non-local features. Instead, mode-assisted training significantly improves the quality of reconstructed quantum states while reducing the number of required measurements by orders of magnitude. This opens up the possibility of efficiently tackling other types of quantum problems previously unmanageable with these types of approaches.

3.1.2 Restricted Boltzmann machines

In this section, we outline the basics of the RBM, and leave the detailed calculations in Appendix 3.1.6. As illustrated in Fig. 3.1, an RBM is a two-layered neural network with n visible nodes $\mathbf{v} \in \{0,1\}^n$, m hidden nodes $\mathbf{h} \in \{0,1\}^m$, and trainable weights W_{ij} and biases a_i,

b_j. Together, they define a joint distribution,

$$p(\mathbf{v},\mathbf{h}) = \frac{1}{Z}\exp\left(\sum_{i=1}^{n} a_i v_i + \sum_{j=1}^{m} b_j h_j + \sum_{i=1}^{n}\sum_{j=1}^{m} W_{ij} v_i h_j\right), \tag{3.1}$$

where Z is the partition function. The marginal distribution of the visible nodes,

$$p(\mathbf{v}) = \sum_{\mathbf{h}} p(\mathbf{v},\mathbf{h}) = \frac{1}{Z} e^{\sum_i a_i v_i} \prod_j \left(1 + e^{b_j + \sum_i v_i W_{ij}}\right), \tag{3.2}$$

is used to model the unknown data distribution.

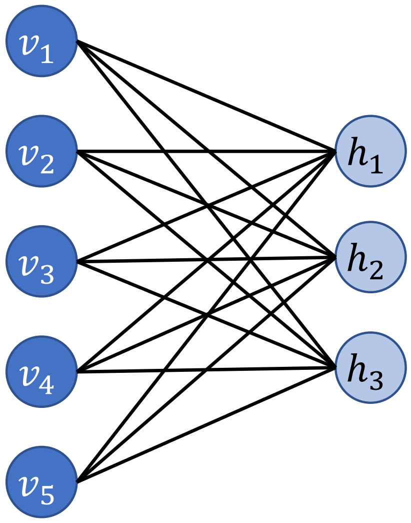


Figure 3.1. The structure of a typical RBM. The visible nodes v_i and hidden nodes h_j form a bipartite graph, with no intra-layer connections.

To represent a quantum state, $|\psi\rangle$, we map the wavefunction to such a probability distribution. For instance, if the wave function is positive, we can simply set $\psi(\mathbf{v}) = \sqrt{p(\mathbf{v})}$. For a general complex wavefunction, one can either allow the RBM weights to have complex values [136], or use a second RBM to model the phase [138]. Yet another approach is to model the quantum state with informationally complete positive operator-valued measurements (IC-POVM) [142], whose outcome is an ordinary probability distribution instead of a quasi-probability distribution.

Irrespective, the standard method of training an RBM is to minimize the KL divergence [156, 157],

$$\mathrm{KL}(q||p) = \sum_{\mathbf{v}} q(\mathbf{v}) \log \frac{q(\mathbf{v})}{p(\mathbf{v})}, \tag{3.3}$$

between the target distribution $q(\mathbf{v})$ and the RBM distribution $p(\mathbf{v})$. Computing the gradient with respect to the RBM parameters, we obtain the formula of weight updates:

$$\Delta W_{ij} = \eta \left(\langle v_i h_j \rangle_{q(\mathbf{v})p(\mathbf{h}|\mathbf{v})} - \langle v_i h_j \rangle_{p(\mathbf{v},\mathbf{h})} \right), \tag{3.4}$$

in which η is the learning rate, and $\langle \cdot \rangle_p$ denotes expectation over the probability distribution p. A similar expression holds for the biases [1]. See Appendix 3.1.6 for detailed derivations.

In Eq. (3.1), the partition function Z involves a summation over an exponential amount of terms, making it impossible to evaluate $p(\mathbf{v},\mathbf{h})$ efficiently. Instead, Z cancels out in the conditional probabilities, $p(\mathbf{h}|\mathbf{v})$ and $p(\mathbf{v}|\mathbf{h})$, making them efficiently computable [157]:

$$\begin{aligned} p(\mathbf{h}|\mathbf{v}) &= \prod_{j=1}^{m} \frac{e^{h_j(b_j + \sum_{i=1}^{n} v_i W_{ij})}}{1 + e^{b_j + \sum_{i=1}^{n} v_i W_{ij}}} \\ p(\mathbf{v}|\mathbf{h}) &= \prod_{i=1}^{n} \frac{e^{v_i(a_i + \sum_{j=1}^{m} W_{ij} h_j)}}{1 + e^{a_i + \sum_{j=1}^{m} W_{ij} h_j}} \end{aligned} \tag{3.5}$$

Therefore, in the expression of the gradient, Eq. (3.4), the first expectation can be evaluated exactly and efficiently, while the second expectation is usually approximated with a sampling algorithm.

[1]Throughout this work, $\eta = 0.01$ and reduces by half whenever performance doesn't improve for 10^4 iterations, and the minibatch size for computing the expectation is N^2, where N is the number of qubits.

3.1.3 Local Samplers

Contrastive divergence (CD) [137] is the most widely adopted sampling algorithm for RBMs. CD-k starts from a sample $\mathbf{v}^0$ from the dataset and constructs a Markov chain of samples,

$$\mathbf{v}^0 \to \mathbf{h}^0 \to \mathbf{v}^1 \to \mathbf{h}^1 \to \cdots \to \mathbf{v}^k, \tag{3.6}$$

using the conditional distributions $p(\mathbf{h}|\mathbf{v})$ and $p(\mathbf{v}|\mathbf{h})$, alternating between the visible nodes $\mathbf{v}$ and hidden nodes $\mathbf{h}$ for k times. When $k \to \infty$, the distribution of $\mathbf{v}^k$ converges to $p(\mathbf{v})$, and we can approximate the second term in Eq. (3.4) with an expectation over a batch of sampled $\mathbf{v}^k$.

CD, among many other Markov chain based samplers, like persistent contrastive divergence (PCD) [146] and parallel tempering (PT) [147, 148, 149], belongs to the category of "local samplers". Unless specifically designed, they only contains "local moves" and does not include global information on the probability distribution. In Appendix 3.1.6, we rigorously define the concept of locality with respect to Markov chains, and visualize the spatial proximity of basis states over an example RBM. Here, we first demonstrate the potential issues that can arise with local samplers.

As a simple example, let us consider the Greenberger-Horne-Zeilinger (GHZ) state) [158],

$$|\Psi_{\text{GHZ}}\rangle = \frac{1}{\sqrt{2}}\left(|00\cdots 0\rangle + |11\cdots 1\rangle\right), \tag{3.7}$$

a prototypical N-qubit entangled state with two modes that has wide applications in quantum information theory, and is also used for benchmarking different QST algorithms [142, 143].

Superficially, this state seems trivial and has an exact RBM representation with only one hidden neuron [140]. On the other hand, training an RBM with CD to represent this state faces immediate failure. Starting from an ideal 10-qubit GHZ state, we obtain training data by performing projective measurements in the $\{|0\rangle, |1\rangle\}$ basis on 10^4 copies of the state. Fig. 3.2 (red dotted curve) shows the reconstruction fidelity for this state between the exact state $|\psi_{\text{exact}}\rangle$

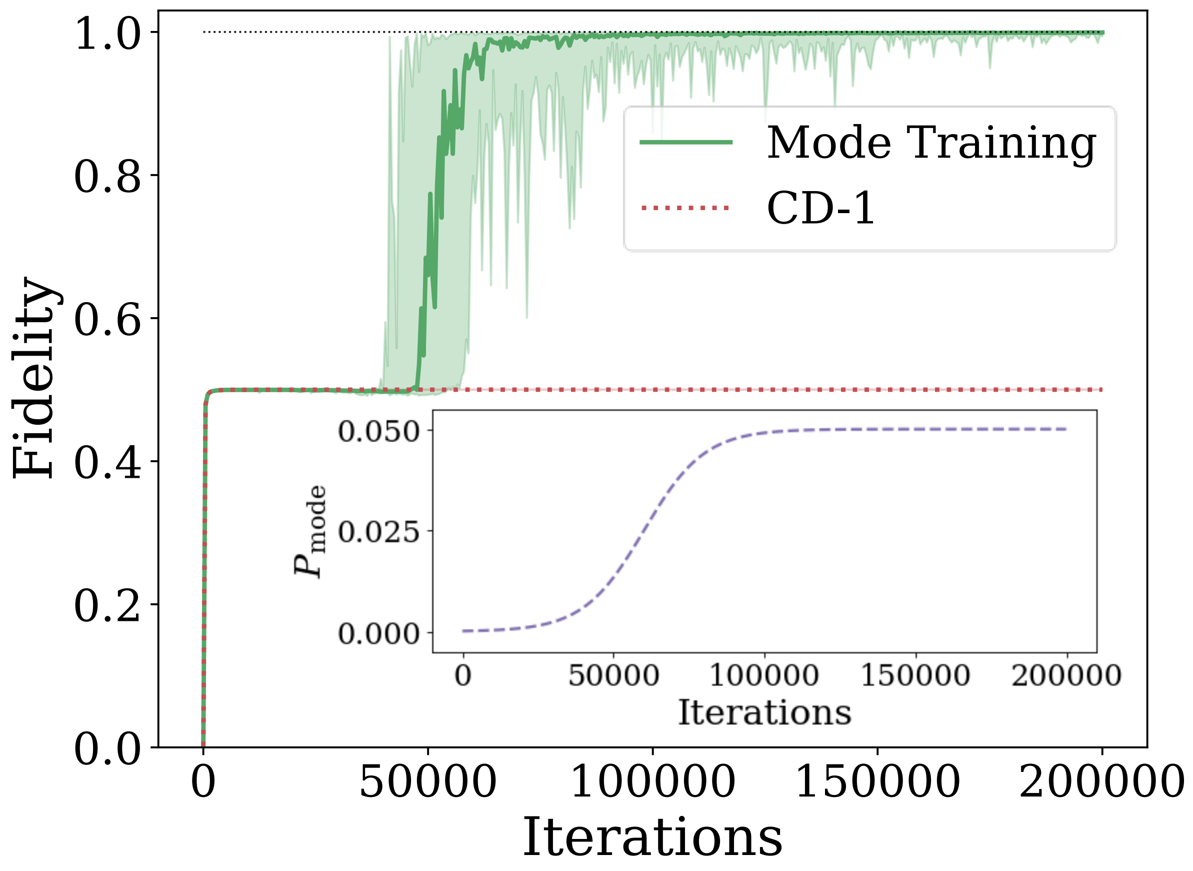


Figure 3.2. Training an RBM to represent a 10-qubit GHZ state. Training data is obtained by performing projective measurements in the $\{|0\rangle, |1\rangle\}$ basis on 10^4 copies of the state. The inset shows the frequency of mode updates, Eq. (3.9) (with parameters $\alpha = 20$, $\beta = 6$ and $P_{\max} = 0.05$), as training goes on. Curves represent the medians of 20 runs, and the shaded regions are enclosed by the maximum and minimum.

and the reconstructed one $|\psi\rangle$, $f(\psi_{\text{exact}}, \psi) = |\langle\psi_{\text{exact}}|\psi\rangle|^2$, for 10 qubits. CD can only learn one of the two modes, reaching a final fidelity of 1/2. As already mentioned, this is of no surprise, since CD is a local sampling algorithm and has difficulty mixing between different modes [150]. If a CD chain starts from one mode, it is (almost) trapped there forever. This sampling bias is amplified over time, causing the RBM to eventually converge towards one of the two modes.

To solve this problem, we need to properly incorporate global information into the training process. One simple yet efficient approach is mode-assisted training [134].

3.1.4 Mode-assisted training

Algorithm

To explicitly inject global information into the training process, we design an off-gradient training step that supplements the regular gradient updates, using the mode of the RBM distribution [134]. This means replacing the formula of weight updates, Eq. (3.4), as follows:

$$\Delta W_{ij}^{\text{mode}} = \eta\left(\langle v_i h_j\rangle_{q_{\text{mode}}(\mathbf{v})p(\mathbf{h}|\mathbf{v})} - [v_i h_j]_{p(\mathbf{v},\mathbf{h})}\right) \tag{3.8}$$

where $[\cdot]$ indicates expectation over the mode of the RBM distribution $p(\mathbf{v},\mathbf{h})$, and q_{mode} is a uniform distribution over all possible modes of the data distribution q. We call Eq. (3.8) the "mode-assisted training", or "mode training" for short.

We train the NN for $n_{\max}$ iterations ($n_{\max} = 2 \times 10^5$ throughout this work), and the schedule of when to perform mode updates is determined by calculating the probability of replacing Eq. (3.4) with Eq. (3.8) at each training iteration step n as:

$$P_{\text{mode}}(n) = P_{\max}\sigma\left(\alpha\frac{n}{n_{\max}} - \beta\right), \tag{3.9}$$

where σ is the sigmoid function, and $0 < P_{\max} \leq 1$ is the maximum probability of a mode update. At the beginning of the training, P_{mode} is then small, but it increases gradually with the number of updates, according to the parameters α and β (see inset of Fig. 3.2) [2].

To find $\mathbf{v}_{\text{mode}}$, the mode of the RBM distribution, for distributions with a simple data structure, it suffices to evaluate the amplitudes of mode candidates from the dataset (e.g., $|00\cdots 0\rangle$ and $|11\cdots 1\rangle$ for the GHZ state). For more complicated distributions where identifying the mode candidates is hard, $q_{\text{mode}}(\mathbf{v})$ can be approximated with $q(\mathbf{v})$, and $\mathbf{v}_{\text{mode}}$ can be sampled from the joint distribution $p(\mathbf{v},\mathbf{h})$, by employing an optimization solver. This involves minimizing the RBM energy,

$$E(\mathbf{v},\mathbf{h}) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} W_{ij} v_i h_j, \tag{3.10}$$

which is a quadratic unconstrained binary optimization problem [159] and is generally NP-hard. We employ the MemComputing solver [113, 160, 134, 7], which can efficiently generate good approximations of $\mathbf{v}_{\text{mode}}$, and an empirical polynomial scaling for typical RBMs is observed [134].

We now show that when properly combined with regular CD updates, this off-gradient mode update greatly increases the stability of the training. This is clearly seen in Fig. 3.2. In the initial phase of the training, CD is able to learn the support of the distribution but not much else. As training goes on, the frequency of mode updates is increased to balance between the multiple modes and bring the RBM distribution as close to the data distribution as possible. In fact, as the probability of mode updates increases, mode-assisted training easily jumps out of the local minimum and learns the full distribution with near perfect fidelity.

W-state

In order to show even more clearly that mode-assisted training provides global (long-range) information during training, we consider the W-state (named after Wolfgang Dür) [161],

[2]The choice of the sigmoid in Eq. (3.9) is arbitrary: other types of functions can be chosen to accomplish the same task.

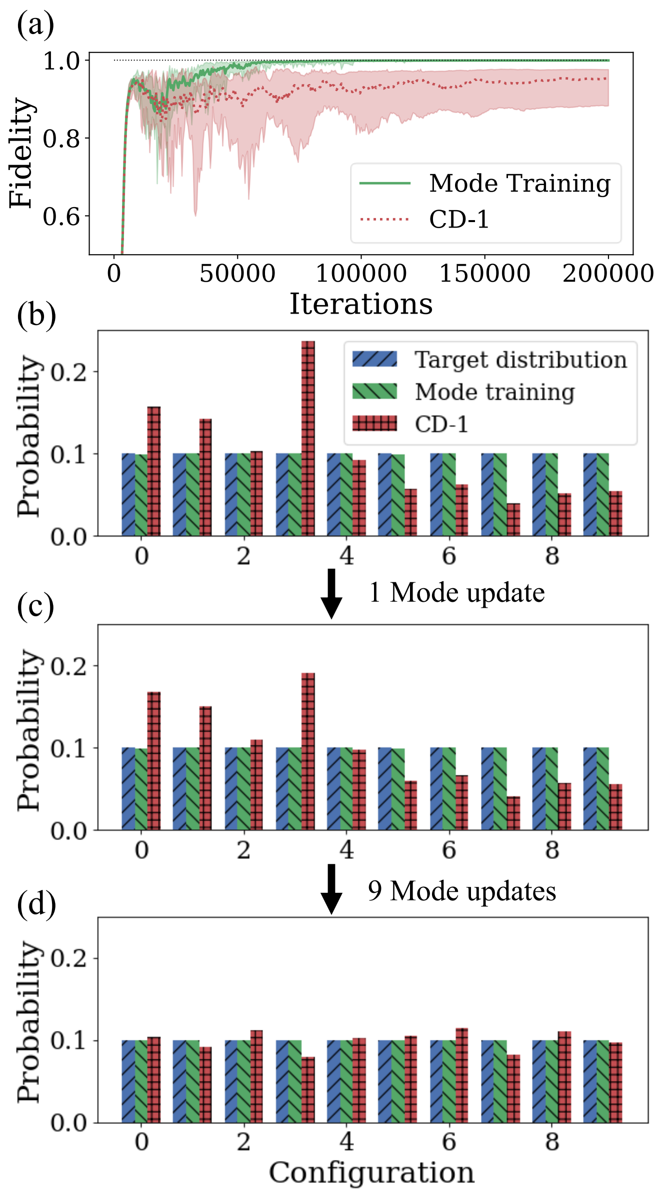


Figure 3.3. Training an RBM on the 10-qubit W state. Training data is obtained by performing projective measurements in the $\{|0\rangle, |1\rangle\}$ basis on 10^4 copies of the state. (a) The fidelity curve during training. Curves represent medians of 20 runs, and the shaded regions are enclosed by the maximum and minimum. Mode-assisted training converges quickly to $f = 1$ with vanishing variance, leaving a significant gap compared to CD-1. (b) Example of the distribution learned by the RBM, after training completes. Among all 2^{10} possible configurations, we only plotted the 10 most important ones in the W-state, $|10\cdots0\rangle$ through $|00\cdots1\rangle$. While the result of mode-assisted training is almost perfect, CD-1 produces far less satisfying results. (c) and (d) Smoothing the noisy distribution learned with CD-1 using additional mode updates. Each mode update locates the global maximum and "pushes it" down, while all other states "pop up" a bit. Repeated mode updates would eventually enforce uniformity over multiple modes.

another N-qubit entangled state given by

$$|\Psi_W\rangle = \frac{1}{\sqrt{N}}\left(|100\cdots 0\rangle + |010\cdots 0\rangle + \cdots + |000\cdots 1\rangle\right). \tag{3.11}$$

Similar to the GHZ state, we constructed a synthetic dataset by taking 10^4 projective measurements in the $\{|0\rangle, |1\rangle\}$ basis on a 10-qubit W-state, and trained two RBMs using CD-1 and mode-assisted training, respectively. Fig. 3.3(a) compares their reconstruction fidelity. In Fig. 3.3(b), we show the amplitude of the state after training: mode-assisted training quickly converges to the target distribution almost perfectly, but the result for CD-1 is a lot more noisy. Again, while CD-1 can correctly locate the mode states (the support of the distribution), it does a terrible job at matching their amplitudes to the target.

At this point, if we correct the noisy distribution learned with CD-1, we can visualize how mode-assisted training works. Figs. 3.3(b), (c) and (d) show this procedure, where we applied 10 mode updates to the distribution learned with CD-1. The correction is already significant after a single mode update: from (b) to (c), the global maximum of the CD-1 distribution is pushed down, while all other states pop up a bit. Repeating this procedure for an additional 9 times, the resulting distribution is already very close to uniformity. The direct access to global maxima is what local samplers like CD lack, and this deficiency cannot be solved by increasing the length of the sampling chain. This is explicitly shown Appendix 3.1.6. In Appendix 3.1.6, we compare the performance of mode-assisted training against persistent contrastive divergence [146] and parallel tempering [147, 148, 149], two advanced (albeit still local) samplers that are designed to alleviate the slow-mixing problem. Several additional numerical experiments are presented in Appendix 3.1.6, showing the performance of mode training under different scenarios.

Scalability

So far, we have only considered small systems. Now, we scale up the system size and show that mode training requires orders of magnitude less number of measurements compared to MCMC sampling. As an explicit example we consider the W-state up to 50 qubits, and focus on two physical quantities: fidelity and number of measurements. To study the effect of measurements on reconstruction quality, we mimic experiments by first performing a fixed number of measurements on the exact state, then use the measured results as dataset to train the RBM model. As shown in Fig. 3.4 for a W-state with *N* qubits, when fixing the number of measurements, mode training outperforms CD-1 by one to two orders of magnitude.

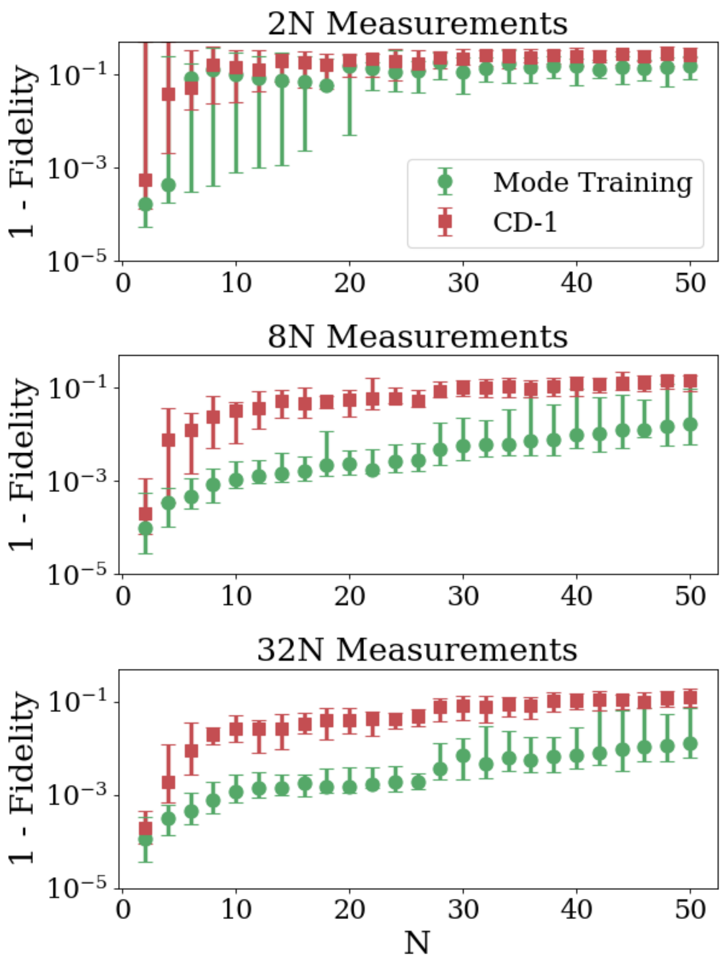


Figure 3.4. Comparing the reconstruction fidelity of CD-1 and mode-assisted training for the W-state with *N* qubits and fixed amount of measurements. Data points are computed using the median of 20 runs, and error bars represent corresponding maximum and minimum values. In all cases, mode-assisted training outperforms CD-1, and the difference is increasing as we increase the number of measurements, up to two orders of magnitude.

Next, we fix a target fidelity and estimate the amount of measurements required to reach that fidelity. For better comparison, we also included results from the maximum likelihood method, a brute-force approach for quantum state tomography [153]. The difference is drastic: maximum likelihood has a hard time reaching any fidelity using reasonable resources already for less than 10 qubits. This is due to the exponential space complexity of storing and manipulating the full density matrix, which allows us to show results only up to 7 qubits in Fig. 3.5.

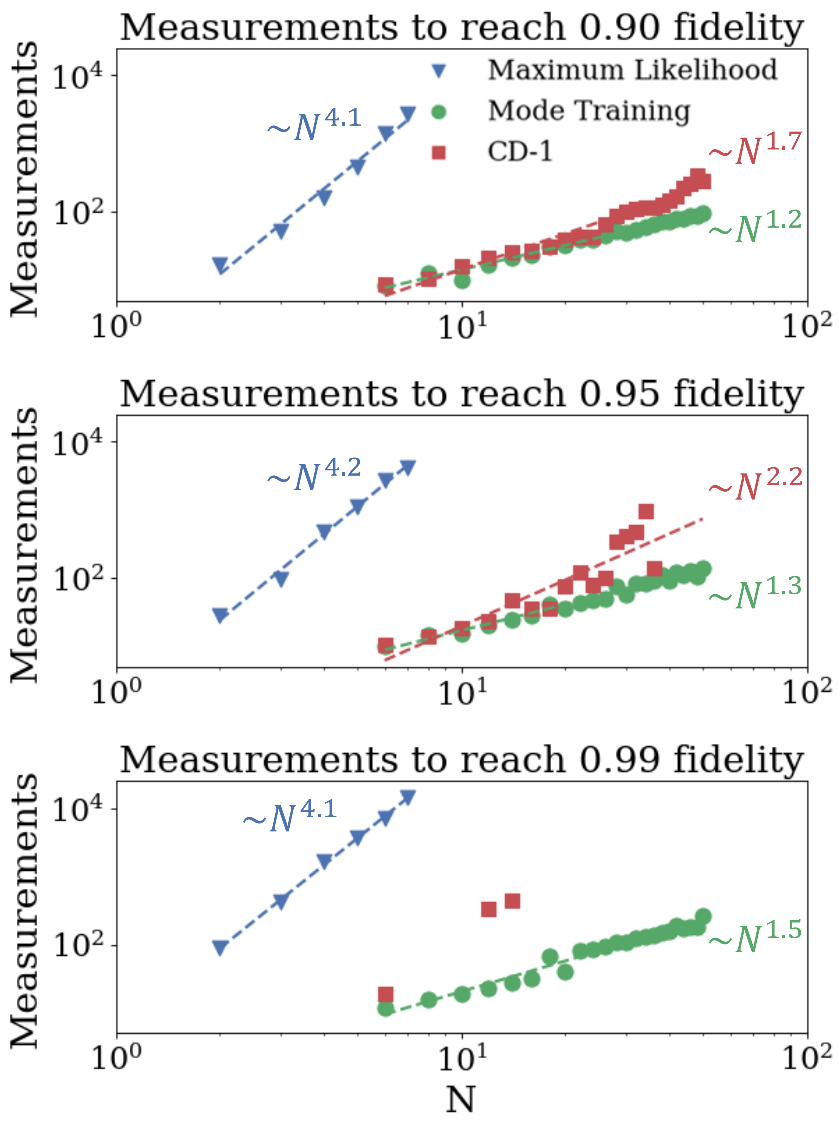


Figure 3.5. Number of measurements required to reach a certain fidelity for the W state with N qubits with maximum likelihood, CD-1, and mode-assisted training. Data points are computed with interpolation, using the best result from 20 runs with fixed number of measurements.

When the fidelity target is not too high ($f \lesssim 0.95$), CD-1 shows a performance comparable to mode-assisted training. However, as the target fidelity increases, CD increasingly struggles to reach the same fidelity, eventually disappearing from the plot due to its inability to reach the target with less than 10^4 measurements. Mode-assisted training, on the other hand, performs consistently throughout the size range, and shows a sub-quadratic scaling (see Fig. 3.5) in the number of measurements to reach the target up to the size considered, without much sensitivity to the target fidelity.

3.1.5 Conclusion

In this work, we have shown that providing global information to the training of an NN representing quantum states, in the form of the modes of its probability distribution, improves significantly the reconstruction of such states. The improvement also translates into orders of magnitude reduction in the number of required measurements. We have employed RBMs as example, but the method is applicable to other types of NNs [135].

We have also shown that the mode-assisted training method scales very favorably in terms of number of measurements required to reach a target fidelity as a function of number of qubits. This result, coupled with optimization methods, like MemComputing [7], to efficiently sample the mode(s) of multi-dimensional probability distributions, paves the way to solve a wide variety of quantum problems classically and with considerably less resources.

3.1.6 Appendix

An introduction to restricted Boltzmann machines

For completeness, in this section, we systematically introduce the restricted Boltzmann machine (RBM) [137].

Fig. 3.1 shows the structure of a typical RBM, where two sets of binary nodes, $\{v_i, h_j\}$, lies on a bipartite graph. One can view the RBM as a classical Ising model, with each binary node as an individual spin. The weight matrix W_{ij} parameterizes the interaction between the spins, and each spin has an external field, a_i or b_j, acting on it. Combining everything, we can define the energy of the RBM:

$$E(\mathbf{v},\mathbf{h}) = -\left(\sum_{i=1}^{n} a_i v_i + \sum_{j=1}^{m} b_j h_j + \sum_{i=1}^{n}\sum_{j=1}^{m} W_{ij} v_i h_j\right) \tag{3.12}$$

At equilibrium, the distribution of the spins is characterized by the Boltzmann distribution (hence the name Boltzmann machine):

$$p(\mathbf{v},\mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v},\mathbf{h})} \tag{3.13}$$

where $Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}$ is the partition function. To model an unknown distribution, we use the marginal distribution of $\mathbf{v}$,

$$p(\mathbf{v}) = \sum_{\mathbf{h}} p(\mathbf{v},\mathbf{h}). \tag{3.14}$$

There are two conventions when choosing the values of the binary nodes, either $\{0,1\}$ or $\{+1,-1\}$, and conversion between the two representations can be easily carried out via a transformation on the weights and biases [162]. We stick to the former convention, $\{\mathbf{v},\mathbf{h}\} \in \{0,1\}^{n+m}$. Then, the summation in Eq. (3.14) can be carried out explicitly:

$$\begin{aligned} p(\mathbf{v}) =& \frac{1}{Z} \sum_{\mathbf{h}\in\{0,1\}^m} e^{-E(\mathbf{v},\mathbf{h})} \\ =& \frac{1}{Z} e^{\sum_{i=1}^{n} a_i v_i} \prod_{j=1}^{m} \left(1 + e^{b_j + \sum_{i=1}^{n} v_i W_{ij}}\right) \end{aligned} \tag{3.15}$$

The partition function Z involves a summation over an exponential amount of terms, making it impossible to evaluate Eq. (3.15) exactly. Instead, the conditional probabilities, $p(\mathbf{h}|\mathbf{v})$ and $p(\mathbf{v}|\mathbf{h})$, can be computed efficiently:

$$
\begin{aligned}
p(\mathbf{h}|\mathbf{v}) =& \frac{p(\mathbf{v},\mathbf{h})}{p(\mathbf{v})} \\
=& \prod_{j=1}^{m} \frac{e^{h_j(b_j+\sum_{i=1}^{n} v_i W_{ij})}}{1+e^{b_j+\sum_{i=1}^{n} v_i W_{ij}}} \\
=& \prod_{j=1}^{m} p(h_j|\mathbf{v})
\end{aligned} \tag{3.16}
$$

with

$$
\begin{aligned}
p(h_j = 1|\mathbf{v}) &= \text{sigmoid}(b_j + \sum_{i=1}^{n} v_i W_{ij}), \\
p(h_j = 0|\mathbf{v}) &= \text{sigmoid}(-b_j - \sum_{i=1}^{n} v_i W_{ij}).
\end{aligned} \tag{3.17}
$$

Thanks to the bipartite structure of RBM, with fixed $\mathbf{v}$, different h_j are independent of each other, and the conditional probability $p(\mathbf{h}|\mathbf{v})$ factors into a product form. Similarly,

$$
\begin{aligned}
p(\mathbf{v}|\mathbf{h}) &= \prod_{i=1}^{n} p(v_i|\mathbf{h}), \\
p(v_i = 1|\mathbf{h}) &= \text{sigmoid}(a_i + \sum_{j=1}^{m} W_{ij} h_j), \\
p(v_i = 0|\mathbf{h}) &= \text{sigmoid}(-a_i - \sum_{j=1}^{m} W_{ij} h_j).
\end{aligned} \tag{3.18}
$$

Eqs. (3.17), (3.18) can be efficiently computed and are frequently used in the training and sampling procedures.

To model the target distribution $q(\mathbf{v})$, we train the RBM by minimizing the KL divergence [156, 157],

$$
\text{KL}(q||p) = \sum_{\mathbf{v}} q(\mathbf{v}) \log \frac{q(\mathbf{v})}{p(\mathbf{v})}. \tag{3.19}
$$

Explicitly computing the gradients with respect to the RBM weights, we have:

$$\begin{aligned}
&\frac{\partial \mathrm{KL}(q||p)}{\partial W_{ij}} \\
&= -\sum_{\mathbf{v}} q(\mathbf{v}) \frac{1}{p(\mathbf{v})} \frac{\partial p(\mathbf{v})}{\partial W_{ij}} \\
&= -\sum_{\mathbf{v}} q(\mathbf{v}) \frac{1}{p(\mathbf{v})} \sum_{\mathbf{h}} \frac{\partial p(\mathbf{v},\mathbf{h})}{\partial W_{ij}} \\
&= -\sum_{\mathbf{v},\mathbf{h}} q(\mathbf{v}) \frac{p(\mathbf{v},\mathbf{h})}{p(\mathbf{v})} v_i h_j + \frac{1}{Z} \frac{\partial Z}{\partial W_{ij}} \\
&= -\sum_{\mathbf{v},\mathbf{h}} q(\mathbf{v}) p(\mathbf{h}|\mathbf{v}) v_i h_j + \sum_{\mathbf{v},\mathbf{h}} p(\mathbf{v},\mathbf{h}) v_i h_j \\
&= -\langle v_i h_j \rangle_{q(\mathbf{v})p(\mathbf{h}|\mathbf{v})} + \langle v_i h_j \rangle_{p(\mathbf{v},\mathbf{h})}
\end{aligned} \tag{3.20}$$

Similarly,

$$\begin{aligned}
\frac{\partial \mathrm{KL}(q||p)}{\partial a_i} &= -\langle v_i \rangle_{q(\mathbf{v})} + \langle v_i \rangle_{p(\mathbf{v},\mathbf{h})} \\
\frac{\partial \mathrm{KL}(q||p)}{\partial b_j} &= -\langle h_j \rangle_{q(\mathbf{v})p(\mathbf{h}|\mathbf{v})} + \langle h_j \rangle_{p(\mathbf{v},\mathbf{h})}
\end{aligned} \tag{3.21}$$

With the analytic expression of the gradients, we can use algorithms such as stochastic gradient descent to train the RBM. However, another problem remains: without access to the partition function Z, we cannot compute the expectation with respect to $p(\mathbf{v},\mathbf{h})$ efficiently.

To compute the likelihood gradient, Eq. (3.20), one can use a sampling algorithm to approximate the expectation with respect to $p(\mathbf{v},\mathbf{h})$. The most widely adopted algorithm, contrastive divergence (CD) [137], starts from a sample $\mathbf{v}^0$ from the dataset and constructs a Markov chain of samples,

$$\mathbf{v}^0 \to \mathbf{h}^0 \to \mathbf{v}^1 \to \mathbf{h}^1 \to \cdots \to \mathbf{v}^k, \tag{3.22}$$

using the conditional distributions $p(\mathbf{h}|\mathbf{v})$ and $p(\mathbf{v}|\mathbf{h})$. When $k \to \infty$, the distribution of $\mathbf{v}^k$ converges to $p(\mathbf{v})$, and we can approximate the second term in Eq. (3.20) with an expectation over a batch of sampled $\mathbf{v}^k$.

In practice, k can be chosen to be very small, and the performance of CD-k is reasonable even with $k = 1$. In this case, the samples are biased from the actual RBM distribution, and we are actually minimizing the difference between two KL-divergences [156] (hence the name contrastive divergence),

$$\mathrm{KL}(q|p) - \mathrm{KL}(p_k|p) \tag{3.23}$$

where p_k is the distribution of the visible nodes after k steps of the Markov chain. As $k \to \infty$, $p_k \to p$, and the approximation becomes exact.

However, as we will show next, in some cases, the Markov chain does not reach stationarity even with very large k. When this happens, CD practically fails as a useful sampler, and mode-assisted training would be necessary to train the RBM successfully [134].

Distance measure on the RBM

In the main text, we mentioned that CD only utilizes *local*, short-range information, while mode-assisted training can incorporate *global*, long-range information into the training procedure. Here we define precisely what kind of distance we refer to.

Recall that CD utilizes a Markov chain, Eq. (3.22), to sample the state space. At the i-th step $\mathbf{v}^i \to \mathbf{h}^i \to \mathbf{v}^{i+1}$, the achievable states $\mathbf{v}^{i+1}$ might be limited. In this case, the achievable states $\{\mathbf{v}^{i+1}\}$ become the "neighborhood" of $\mathbf{v}^i$. With respect to the RBM, we define the distance between two configurations, $\mathbf{v}^i$ and $\mathbf{v}^j$, as

$$d(\mathbf{v}^i \to \mathbf{v}^j) = -\log\left(\sum_{\mathbf{h}} p(\mathbf{v}^j|\mathbf{h})p(\mathbf{h}|\mathbf{v}^i)\right), \tag{3.24}$$

which is the negative log transition probability from $\mathbf{v}^i$ to $\mathbf{v}^j$. With this definition, we can verify that $d(\mathbf{v}^i \to \mathbf{v}^j) + d(\mathbf{v}^j \to \mathbf{v}^k)$ leads to the transition probability $p(\mathbf{v}^i \to \mathbf{v}^j \to \mathbf{v}^k)$.

Note that the distance defined in Eq. (3.24) is not symmetric and doesn't necessarily satisfy the triangle inequality. Rather, if we view each basis state in the Hilbert space as a vertex on a graph, Eq. (3.24) will act as the directed graph distance between two vertices. In this way, we can define the concept of locality on the graph.

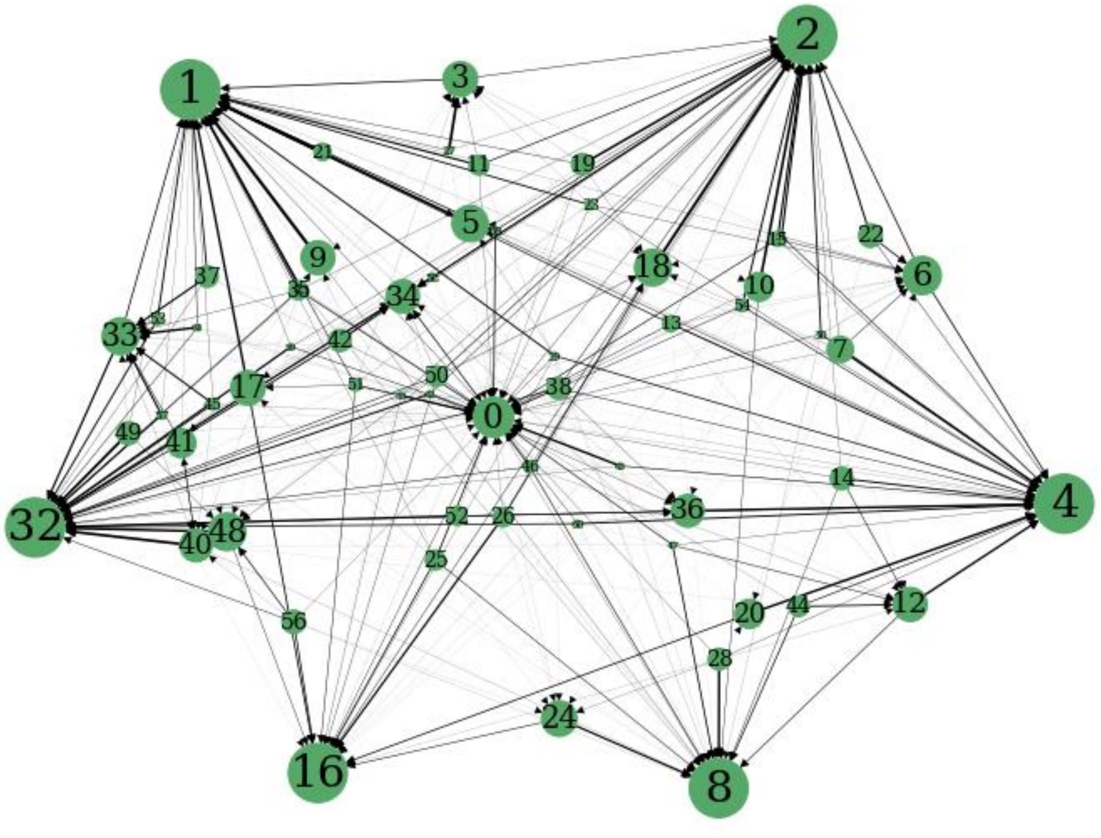


Figure 3.6. Visualization of the graph structure of an RBM, trained on the 6-qubit W-state. Sizes of vertices represent probabilities in the RBM distribution (not proportional), and width of edges are proportional to the transition probabilities in the Markov chain during sampling. With this graphical representation, we can view CD as random walk on the graph.

In Fig. 3.6, we plot the weighted directed graph defined above using the NetworkX python package [163], with the reference RBM trained on a 6-qubit W-state. Each vertex is numbered using the decimal representation of its corresponding binary basis vector, with larger vertices representing larger probabilities in the RBM distribution, and widths of the edges proportional to the transition probabilities. Edges with transition probabilities less than 0.01 are omitted.

The layout of the vertices are computed using the Fruchterman-Reingold force-directed algorithm [164], which treats the vertices as repelling objects and edges as springs holding them close. At equilibrium, the spatial proximity of vertices would more or less characterize the distance between basis vectors in the Hilbert space.

Recall that $|000001\rangle$ through $|100000\rangle$ are the six most important bases in the W-state. In Fig. 3.6, the corresponding nodes $1, 2, 4, 8, 16$ and 32 form a hexagon, enclosing all other nodes in it. None of them have an edge connecting each other —in fact, none of them even have an outgoing edge at all! According to our defined distance measure Eq. (3.24), they are indeed far apart from each other.

Now, we can view CD-k as random walk on the graph for k steps. Then, it is immediately obvious that, CD is a *local sampler*, in the sense that it can only access a small neighborhood of the starting vertex. What is worse, in Fig. 3.6, the six out-most vertices act as sinks for the random walker: once it gets in one of them, escaping it is almost impossible. In this case, ergodicity is practically lost, and CD fails as a useful sampler.

A further look into CD-*k*

We further demonstrate via a sampling experiment that, access to global minima cannot be cured with a local sampler, like CD, by simply increasing the length of the Markov chain.

Again, using the RBM trained on the 6-qubit W-state, we start the sampling chain from one of the modes, and perform CD-k sampling for 10^4 times. The normalized transition probability is plotted in Fig. 3.7: the (i, j)-th location in each plot represents $p_{i\to j}/p_j$, the probability of starting from the i-th state and ending in the j-th state, divided by the probability of the j-th state in the original distribution. In the ideal case where the Markov chain has sufficiently mixed, the sampled distribution should converge to the exact distribution and be independent of the starting point, resulting in $p_{i\to j}/p_j = 1$.

However, as we clearly see in Fig. 3.7 this is not the case. Figures 3.7(a), (b) and (c) show the results of CD-k, with $k = 1, 32, 1024$, respectively. Even with CD-1024, most sampling chains are still stuck at their starting point. Practically, CD fails as a sampler in this case, as it cannot properly explore the phase space: when falling into a mode, it cannot easily escape from it.

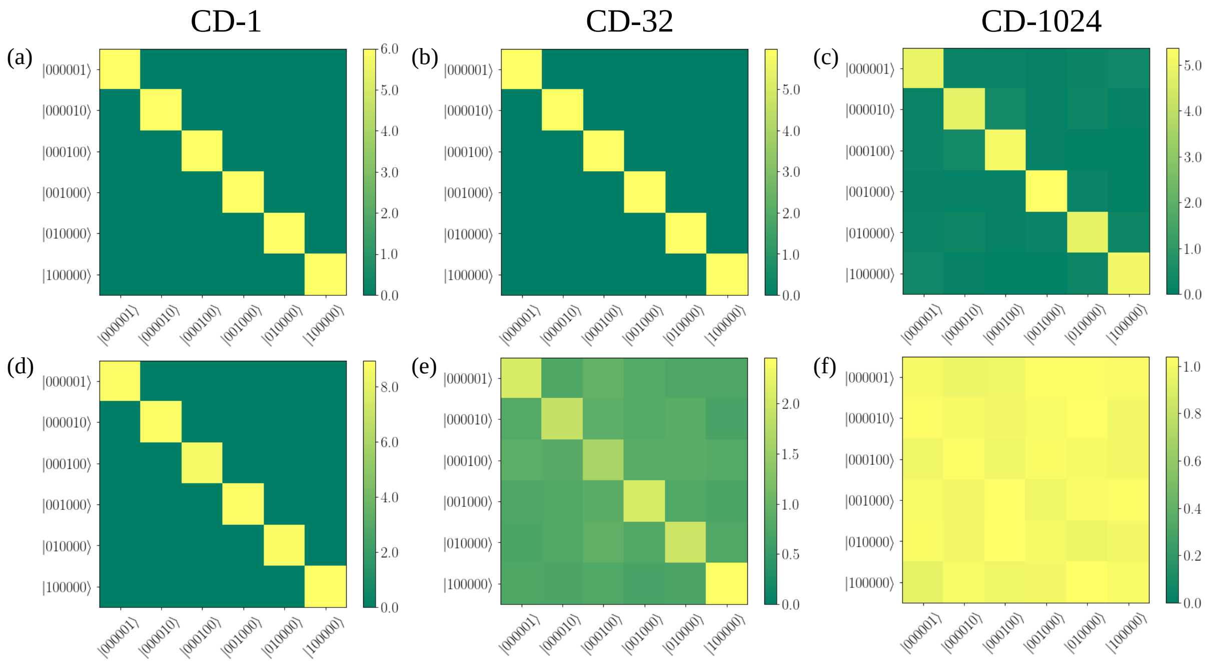


Figure 3.7. Normalized transition probability when sampling a trained RBM with CD-k. Location (i, j) represents the normalized transition probability, $p_{i\to j}/p_j$, of starting the Markov chain from configuration i and ending in configuration j. We observe a strong correlation between the initial and final configuration. (a), (b), and (c) The 6-qubit pure W-state. With isolated modes and zero amplitude on all other basis, the energy barriers between different modes is so high that even CD-1024 cannot escape them. (d), (e), and (f) The depolarized 6-qubit W-state with $p = 0.4$. With a background noise, CD can properly escape from each local minimum and explore other parts of the phase space.

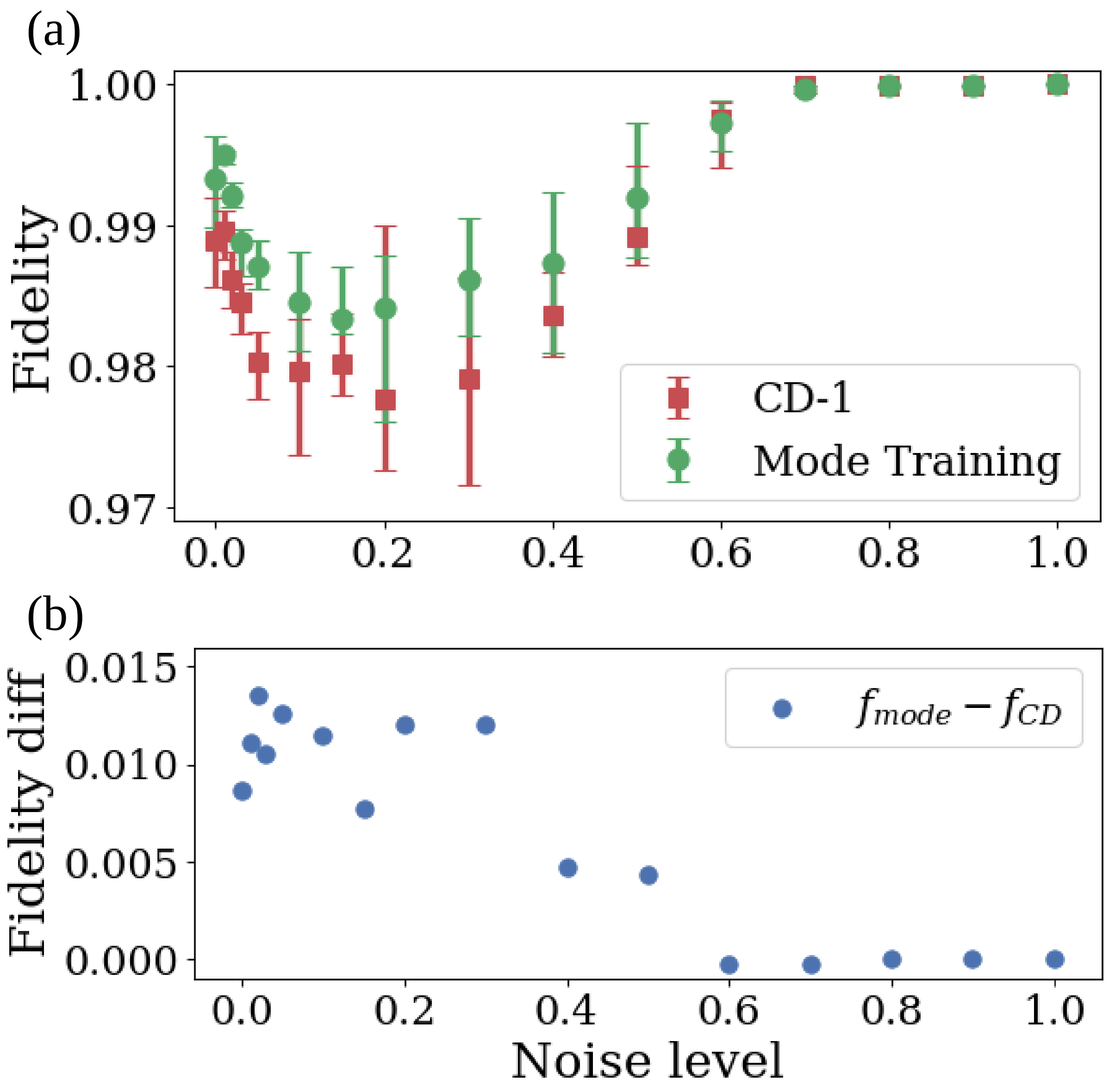


Figure 3.8. Quantum state tomography on the depolarized W-state. (a) Reconstruction of the depolarized W-state $\rho_W = (1-p)|\Psi_W\rangle\langle\Psi_W| + p\mathbb{1}/2^N$. Data points are medians over 20 runs, and error bars represent corresponding maximum and minimum values. Reconstruction is most difficult at a moderate noise level $p \sim 0.15$. (b) Fidelity difference between mode-assisted training and CD-1. As sampling gets easier with the introduction of the background noise, the advantage of mode-assisted training gradually disappears.

We can then naturally ask: for what type of distributions does CD work well? To this end, we consider the depolarized W-state, whose density matrix is given by:

$$\hat{\rho}_W = (1-p)|\Psi_W\rangle\langle\Psi_W| + p\mathbb{1}/2^N \tag{3.25}$$

where p controls the noise level, and N is the number of qubits. To get comparable results to the pure W-state $|\Psi_W\rangle$, we synthesize a similar dataset by perform 2-outcome POVMs described by the measurement operators $M_i = \{|0\rangle\langle 0|, |1\rangle\langle 1|\}$. To be specific, in each measurement, the full measurement operator is the tensor product $M = M_1 \otimes M_2 \otimes \cdots \otimes M_N$, and the outcome of the measurement can be written as a bitstring $v_1 v_2 \cdots v_N$. In the noiseless limit, this is the projective measurement in the computational basis, and the outcome bitstrings precisely correspond to the basis vectors $|v_1 v_2 \cdots v_N\rangle$. With the depolarization noise, the second term in Eq. (3.25) acts as a uniform background noise to the distribution of the measurement outcome. Note, however, that this measurement is not informationally complete.

In Figs. 3.7 (d), (e) and (f), we show the same sampling experiment, with $p = 0.4$. Thanks to the background noise which lowers the energy barrier between different modes, it now becomes possible for local moves to jump out of the local minima, and the CD chains can properly converge with increasing chain length k.

Fig. 3.8 shows this effect from another perspective, where we plot the final fidelity after training versus the noise level p, on the depolarized 10-qubit W state. Unlike some works claiming that the noiseless limit is the hardest to learn [142], here we find that the difficulty peaks near $p \sim 0.15$. We suspect this is due to the fact that the model has to learn the exact amplitude of a small but nonzero background noise, which is more difficult than simply setting the background to zero.

Irrespective, since the background noise greatly eases the burden of sampling, the advantage of mode-assisted training gradually disappears as p increases, until both methods converge to $f = 1$ as $p \to 1$. Such strongly noisy states are easier for local samplers, as local moves would be sufficient to explore the entire phase space, and this difficulty is tunable as we change the parameter p. This further reinforces the notion that approaches providing non-local information to the training (as the one we have discussed here) are very important for quantum states with strongly non-local features.

Advanced samplers

To overcome the weaknesses of CD, many advanced sampling algorithms have been proposed, aiming at alleviating the slow mixing problem of the Markov chain. In this section, we examine two notable examples, persistent contrastive divergence (PCD) [146] and parallel tempering (PT) [147, 148, 149].

The quality of the samples drawn by CD depends strongly on the length of the Markov chain. Generally, CD-k with a large k performs better than CD-1, at the expense of much longer running time.

PCD builds on the idea that, instead of starting a new Markov chain for sampling at every training step, one can maintain one Markov chain throughout the entire training process. With a small learning rate, the RBM distribution only changes slightly at every training step. Therefore, if the Markov chain is sufficiently mixed at the previous training step, it will be close enough to equilibrium at the next training step. By maintaining one Markov chain throughout training, one is essentially using CD-k, with k very large.

PT is also known as replica exchange MCMC sampling, which maintains many copies of the system at different temperatures, and exchange of configurations at different temperatures is allowed according to some acceptance criteria. Since higher temperatures allow the system to explore the high energy configurations more efficiently, the resulting algorithm is less prone to getting stuck in local minima.

For training RBMs, PT maintains N copies of the same RBM, with the distributions

$$p_i(\mathbf{v}) = \frac{e^{-\beta_i E(\mathbf{v})}}{Z} \tag{3.26}$$

for a set of gradually increasing temperatures $\{T_1, T_2, \cdots, T_N\}$, with $\beta_i = 1/T_i$ denoting the inverse temperature. $T_1 = 1$ corresponds to the original RBM distribution, and T_N is usually chosen to be a very large number (e.g., $T_N = 100$) to ensure a proper exploration of the entire phase space. When running the Markov chain, each copy of the system evolves on its own using Gibbs sampling, and an additional cross-temperature state swap move is introduced. At each sampling step, two neighboring configurations $\mathbf{v}_i, \mathbf{v}_{i+1}$ are exchanged with probability

$$r = \frac{p_i(\mathbf{v}_{i+1})p_{i+1}(\mathbf{v}_i)}{p_i(\mathbf{v}_i)p_{i+1}(\mathbf{v}_{i+1})}. \tag{3.27}$$

Using Eq. (3.26), Eq. (3.27) becomes

$$r = \exp\big((\beta_i - \beta_{i+1})(E(\mathbf{v}_i) - E(\mathbf{v}_{i+1})\big). \tag{3.28}$$

While PCD and PT have already seen some success at training RBMs [146, 147, 148, 149], here, we show that they are not as effective in our case. Fig. 3.9 compares the performance of CD, PCD, PT and mode-assisted training, on the N-qubit W-state. With enough measurements, mode-assisted training consistently outperforms all other methods by at least one order of magnitude. PCD has the worst performance, and PT only outperforms CD on smaller systems.

PCD and PT are designed to improve on CD, but why are we seeing worse performance here? To understand this behavior, let us again focus on Fig. 3.6. While PCD and PT are designed to alleviate the slow mixing problem, their underlying proposal and acceptance steps are the same as CD. Therefore, they are still random walkers on Fig. 3.6, except with a new set of random walk rules, and they suffer from the same problem as CD: the random walker will get stuck on one mode configuration, unable to escape, and ergodicity is lost. Importantly, PCD and PT start from random initial conditions, and the distribution of the ending mode state they get stuck in is likely biased from the RBM distribution. While CD also gets stuck, it is initialized with configurations from the dataset, and the ending distribution will be much closer to the actual

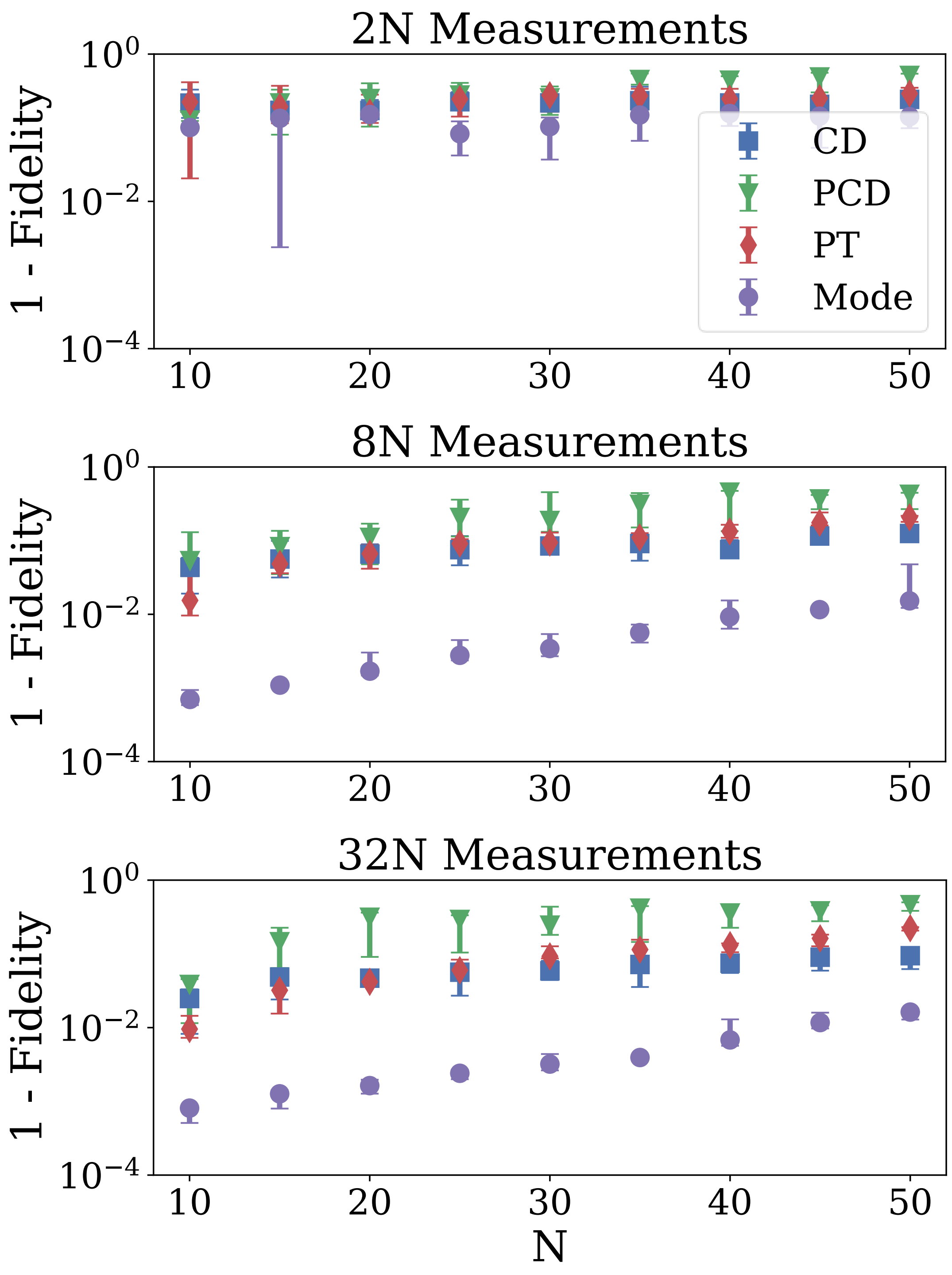


Figure 3.9. Comparison of 4 different training methods on the W-state. Data points are medians of 5 runs, and error bars represent the maximum and minimum values. Mode-assisted training consistently outperforms other methods.

RBM distribution.

Again, we see the superiority of mode-assisted training, and the usefulness of global information. While PCD and PT may offer improvements compared to CD in certain cases, they are still local samplers, in the sense that they only utilize local information, performing a random walk on a small region of the graph. One potential improvement for them could be explicitly providing global information at the proposal-acceptance step, like mode-hopping moves [150]. But again, doing so requires prior knowledge to the target distribution, while mode-assisted training achieves this automatically.

Further numerical experiments on entangled quantum systems

In this section, we further test the capabilities of mode-assisted training on two more highly-entangled quantum systems: the transverse-field frustrated Ising model (TFFIM) on the triangular lattice [165, 166], and the toric code model [167]. As a proof-of-concept demonstration, we focus on small systems where exact results are available using exact diagonalization, and compare the performance between mode-assisted training and CD-1.

Fig. 3.10 is an illustration of the TFFIM on the triangular lattice. The Hamiltonian reads:

$$H = J\sum_{\langle i,j\rangle}\sigma_i^z\sigma_j^z - h\sum_i \sigma_i^x, \tag{3.29}$$

where J is the nearest-neighbor antiferromagnetic Ising coupling and h is the transverse field. In this demonstration, we choose $h = J = 1$.

Without the transverse field, frustration would lead to a highly degenerate ground state [168]. Together with the transverse field, we arrive at a ground state wave function multi-modal in the σ^z basis.

In Fig. 3.11 (a), we plot the exact ground state of the 4×4 TFFIM on the triangular lattice. A synthetic dataset is generated by taking 10^4 projective measurements on this state, and we perform quantum state tomography on it using methods described in the main text.

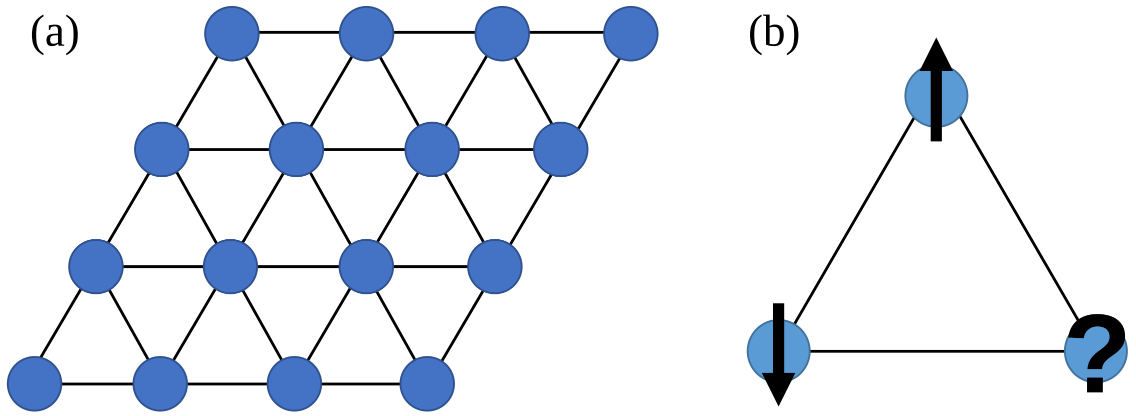


Figure 3.10. (a) An illustration of the triangular lattice. (b) A frustrated loop. With antiferromagnetic interactions, two neighboring spins tend to anti-align with each other, leaving the direction of the third spin undetermined.

The result is shown in Fig. 3.12: the performance of mode-assisted training and CD-1 are comparable. This is an example where CD already works very well, and we would not gain much from mode-assisted training. The reason is similar to the frustrated W-state we just showed: the background structure in the distribution in Fig. 3.11 (a) helps the sampler to jump between different modes, and the quality of the samples is already good using CD-1. In such cases, mode-assisted training would be an overkill.

Another example is the toric code model [167]. It was already demonstrated that an RBM can exactly and efficiently represent the ground state of the toric code, with analytically computable RBM weights [169, 170, 171]. However, the story is quite different if we actually train an RBM with data sampled from the toric code state, which is what it would happen in an actual experiment.

Fig. 3.11 (b) shows one ground state of the 2×2 toric code model: a multi-modal distribution with isolated modes, which, according to our analysis, is difficult for local samplers such as CD. The training curve in Fig. 3.13 confirms this prediction. Even for the 2×2 toric code state with 8 qubits, CD-1 performs rather poorly, while mode-assisted training can achieve

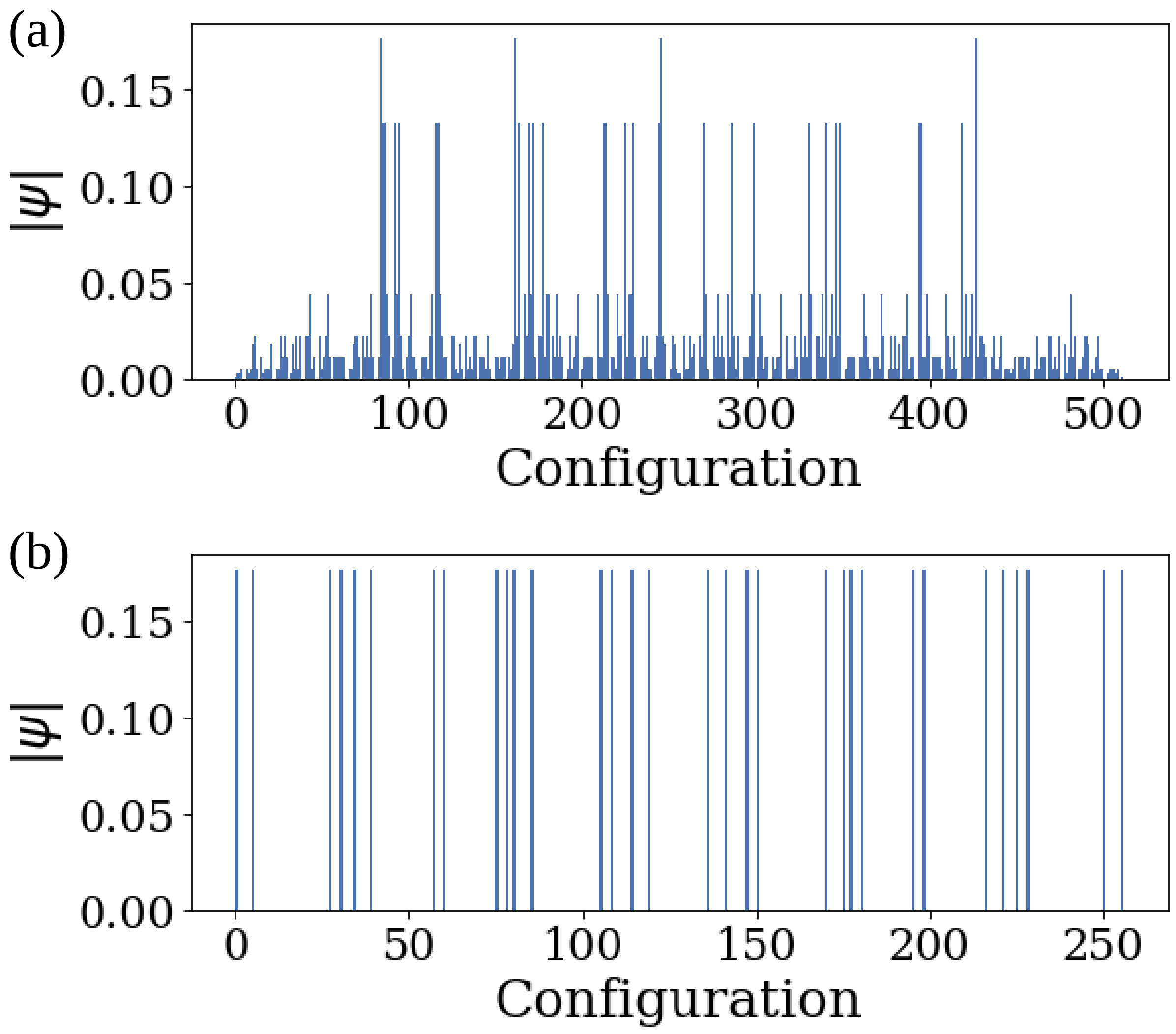


Figure 3.11. Visualization of the wave functions. (a) The ground state wave function of the 3×3 TFFIM. (b) One ground state of the 2×2 toric code model. While both distributions are multi-modal, the first distribution has a structured background, making jumps between modes easier. On the contrary, the second distribution has isolated modes, making it difficult for local samplers to capture the entire distribution.

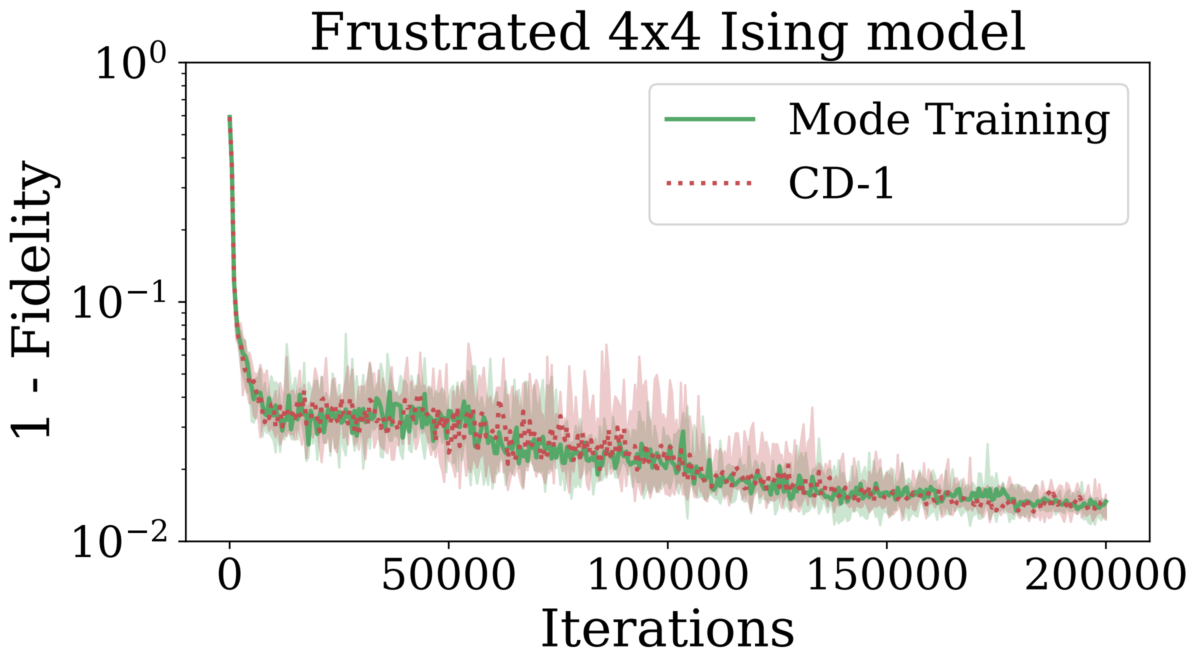


Figure 3.12. The training curve on the 4×4 TFFIM. Since sampling from this state is easy, mode-assisted training and CD-1 have comparable performance.

near perfect fidelity. And in the 3×3 case, CD-1 completely fails with final fidelity less than 0.1, but mode-assisted training can still reasonably reconstruct this state with fidelity near 0.9.

We can now clearly understand when the mode training is particularly advantageous. If a local sampler like CD performs well when training the RBM, then mode-assisted training would not be very useful. But for multi-modal states with strongly non-local features, mode-assisted training can offer significant advantages over traditional methods.

This section, in full, is a reprint of the material as it appears in Physical Review A [4]. Yuan-Hang Zhang, Massimiliano Di Ventra, 2022. The dissertation author was the primary investigator and author of this paper.

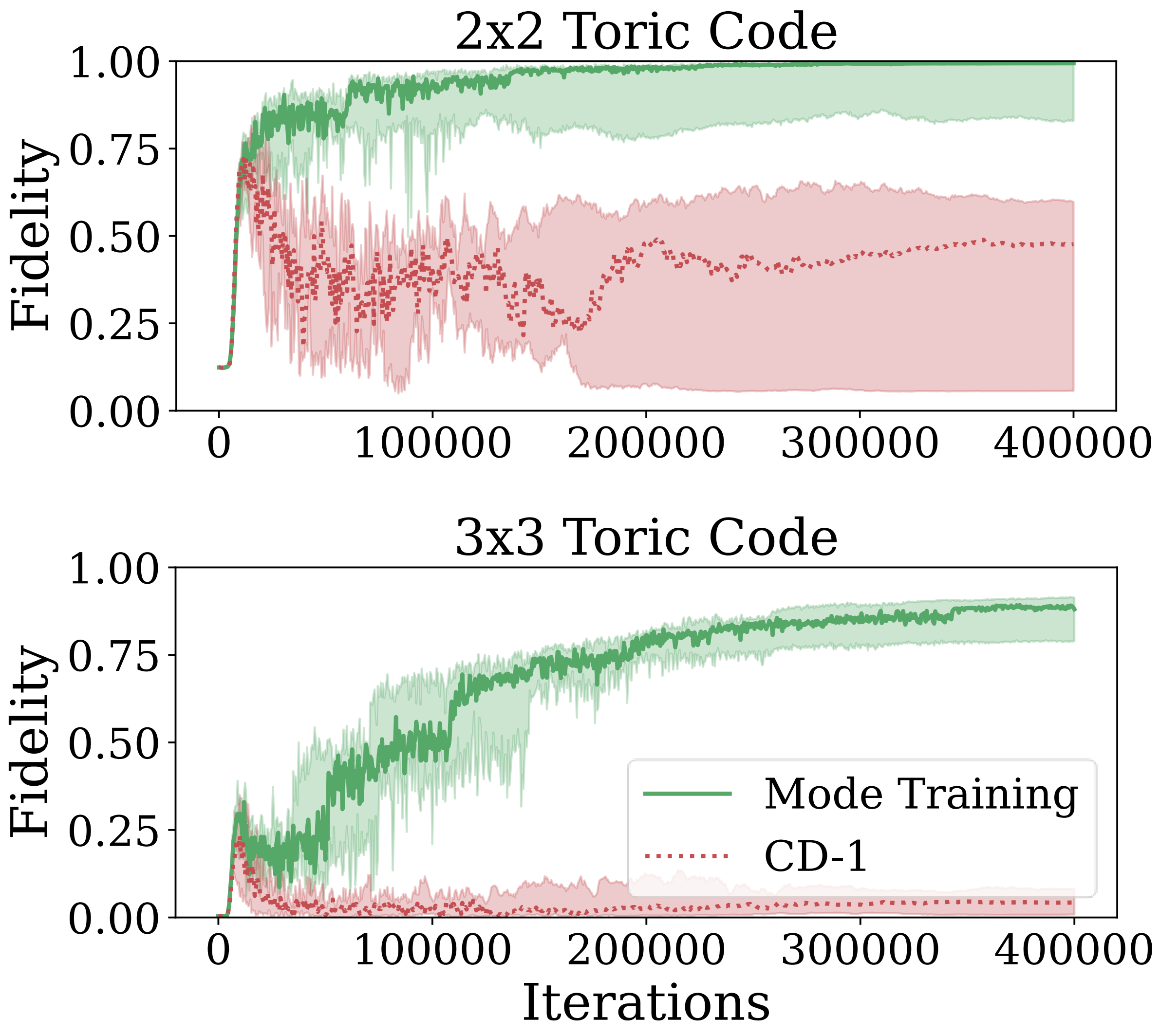


Figure 3.13. The training curve of the 2×2 and 3×3 toric code state. CD-1 completely fails, while mode-assisted training still performs reasonably well.

3.2 Solving quantum many-body problems with transformers

Inspired by the advancements in large language models based on transformers, we introduce the transformer quantum state (TQS): a versatile machine learning model for quantum many-body problems. In sharp contrast to Hamiltonian/task specific models, TQS can generate the entire phase diagram, predict field strengths with experimental measurements, and transfer such a knowledge to new systems it has never been trained on before, all within a single model. With specific tasks, fine-tuning the TQS produces accurate results with small computational cost. Versatile by design, TQS can be easily adapted to new tasks, thereby pointing towards a general-purpose model for various challenging quantum problems.

3.2.1 Introduction

Determining the state of a quantum many-body system is one of the fundamental problems in physics. While the exponential growth of the Hilbert space precludes brute-force calculations, computational methods such as quantum Monte Carlo [172] and tensor network-based methods [173] allow for efficient simulations of certain problems, each with their own strengths and weaknesses.

More recently, the advancements in machine learning techniques and models have influenced the physics community. In fact, the introduction of neural networks (NNs) as variational states for quantum many-body problems has greatly expanded the types and sizes of systems that can be efficiently tackled. For instance, the restricted Boltzmann machine [174, 156] was the first NN model applied to correlated quantum systems [136], followed by models with different architectures such as feed-forward [141, 175], convolutional [176, 177], recurrent [178], and autoregressive [179, 180, 181] ones. With the ability to encode area and volume-law entanglement [182], NNs are especially advantageous in dealing with high-dimensional systems. And with proper tricks, they can also greatly ease the fermion sign problem [141]. Yet, despite these successes, the previous approaches are limited to specific tasks.

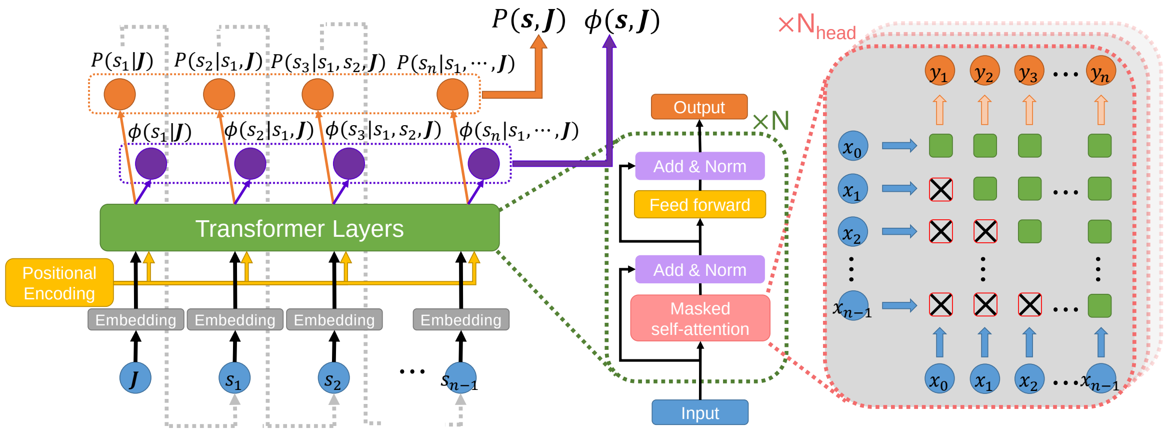
Recently, a new task-agnostic model has been put forward by the machine learning community: the transformer architecture [183]. Since its introduction, this model has dominated the field by achieving state-of-the-art results in almost every natural language processing task [184, 185, 186, 187], thus rendering the recurrent neural networks obsolete in merely a few years. Transformers have also been adapted to different tasks such as image recognition [188], audio processing [189] and graph classification [190], all achieving remarkable results.

This feat relies on an impressive aspect of transformer models: their ability to scale to very large sizes [187, 191]. When facing with a new task, few-shot learning [187] allows a general purpose model to easily adapt with merely a few examples in natural language. And when better performance is desired, fine-tuning on a small dataset produces satisfactory results within a short time [186].

These results give hope that such an architecture may be of great help in quantum physics as well. However, the application of transformers in this field is still rather limited, with a few results concerning quantum lattice models [181], open systems [192], quantum state tomography [143] and quantum circuit simulation [193], while the task-agnostic property is barely used. Therefore, the full potential of the transformer architecture has yet to be explored.

Contrary to the general-purpose models mentioned above, NN models in physics are usually highly specialized, serving a single purpose such as representing wave functions [136, 141, 175, 176, 177, 178, 179, 180], preparing and controlling quantum states [193, 194], recognizing phase transitions [195, 196], realizing quantum state tomography [140, 143, 197], etc. Such tasks share a lot of common knowledge, making it ideal to have a single, unified model that handles them all, with the possibility of discovering new physics at the intersection of different tasks.

As a first step, we may consider using NNs as variational wave functions. Traditionally, each NN can only represent a specific quantum state, and tasks such as generating a phase diagram requires retraining of the same NN from scratch for hundreds of times, even if nearby



$P(\boldsymbol{s},\boldsymbol{J})$
$\phi(\boldsymbol{s},\boldsymbol{J})$
$P(s_1|\boldsymbol{J})$
$P(s_2|s_1,\boldsymbol{J})$
$P(s_3|s_1,s_2,\boldsymbol{J})$
$P(s_n|s_1,\cdots,\boldsymbol{J})$
$\phi(s_1|\boldsymbol{J})$
$\phi(s_2|s_1,\boldsymbol{J})$
$\phi(s_3|s_1,s_2,\boldsymbol{J})$
$\phi(s_n|s_1,\cdots,\boldsymbol{J})$
Transformer Layers
Positional Encoding
Embedding
$\boldsymbol{J}$
s_1
s_2
s_{n-1}
Output
$\times$N
Add & Norm
Feed forward
Masked self-attention
Input
$\times$N$_{\text{head}}$
y_1 y_2 y_3 y_n
x_0 x_1 x_2 x_{n-1}

Figure 3.14. The structure of a TQS. Left: the overall architecture of our model. We use the standard encoder-only transformer architecture, utilizing an embedding layer to map different inputs into a single unified feature space, and pass them through N identical transformer encoder blocks, followed by two different output heads, parameterizing the amplitude P and phase ϕ, respectively. Middle: The structure of a transformer encoder block. Right: the mask structure in a masked self-attention operator. Squares with a cross represent the masks, blocking the flow of information, so that each site only has access to its predecessors. This ensures that the autoregressive property is satisfied.

data points have similar features.

In this paper, we consider a different perspective: instead of modeling a specific quantum state, we attempt to represent a *family of quantum states* within a single neural network. More precisely, we focus on the joint distribution of the wave function and relevant physical parameters such as interaction strength, external field, and/or system size. For the underlying NN, we choose the transformer architecture for its versatility and strong performance across different tasks.

We call this model a *transformer quantum state* (TQS), and show that it is capable of generating the entire phase diagram of a many-body system, predicting field strengths with as few as one experimental measurement, and transferring knowledge to new systems it has never seen before, all within a single model.

3.2.2 Transformer Quantum State

Consider the probability distribution $P(\mathbf{s},\mathbf{J}) \equiv P(s_1,\cdots,s_n,J_1,\cdots,J_m)$, where $s_i \in \{0, 1,\cdots,d-1\}$ are discrete variables representing the physical degrees of freedom such as spin or occupation number, and J_j correspond to other physical parameters, either continuous or discrete. Such a state space grows exponentially with the number of variables, and a compact representation is desired.

To represent $P(\mathbf{s},\mathbf{J})$, we adopt the transformer architecture, and autoregressively model the entire distribution as a product of conditional distributions,

$$P(\mathbf{s},\mathbf{J}) = P(\mathbf{J})\prod_{i=1}^{n} P(s_i|s_1,\cdots,s_{i-1},\mathbf{J}). \tag{3.30}$$

The structure of the transformer is shown in Fig. 3.14, with each output of the neural network representing one of the conditional distributions. For a detailed explanation of the transformer architecture, see Appendix 3.2.5.

Contrary to energy-based models such as restricted Boltzmann machines [136], the autoregressive structure allows for efficient sampling [179]. Since each conditional probability $P(s_i|s_1,\cdots,s_{i-1},\mathbf{J})$ does not depend on any variable s_j with $j > i$, starting from s_1, one can sequentially sample s_i according to the previously sampled configurations, using the i-th conditional distribution only. Using the idea developed in [180], efficiency of the sampling algorithm can be further improved by only sampling unique configuration strings, and the details are explained in Appendix 3.2.5.

We assume that $\mathbf{J}$ has a predefined prior distribution $P(\mathbf{J})$, which, in general, can be chosen as a uniform distribution over the range of interest (e.g., to study the transition in the Heisenberg J_1-J_2 model [198, 177], one can fix $J_1 = 1$ and make J_2 uniform over $[0,1]$).

Our aim is to model quantum states $|\psi(\mathbf{J})\rangle$, which are complex-valued quasiprobability distributions. To this end, we expand them in the computational basis and separate their amplitude A and phase ϕ,

$$\begin{aligned} |\psi(\mathbf{J})\rangle &= \sum_{\mathbf{s}} \psi(\mathbf{s},\mathbf{J})|\mathbf{s}\rangle \\ &= \sum_{\mathbf{s}} A(\mathbf{s},\mathbf{J})\exp(i\phi(\mathbf{s},\mathbf{J}))|\mathbf{s}\rangle. \end{aligned} \tag{3.31}$$

Since squared amplitude has the probability interpretation, we choose

$$A(\mathbf{s},\mathbf{J}) = \sqrt{P(\mathbf{s},\mathbf{J})}, \tag{3.32}$$

with $P(\mathbf{s},\mathbf{J})$ specified in Eq. (3.30). The phase ϕ has no restrictions and can be either positive or negative, and we represent it with a similar autoregressive structure:

$$\phi(\mathbf{s},\mathbf{J}) = \sum_i \phi(s_i|s_1,\cdots,s_{i-1},\mathbf{J}). \tag{3.33}$$

Ground state of a family of Hamiltonians

The first task we consider is finding the ground state $|\psi\rangle$ of many-body Hamiltonians. Per the standard procedure, this can be done by minimizing the variational energy estimation, $\langle\psi|\hat{H}|\psi\rangle$, over the target Hamiltonian $\hat{H}$. A minor complication is that, instead of a single Hamiltonian $\hat{H}$, we have now a *family* of Hamiltonians $\{\hat{H}(\mathbf{J})\}$. In Appendix 3.2.5, we show that the family of ground states $|\psi(\mathbf{J})\rangle$ corresponds to the ground state $|\Psi\rangle$ of the super-Hamiltonian $\mathscr{H} = \bigoplus_{\mathbf{J}} \frac{\hat{H}(\mathbf{J})}{|E_g(\mathbf{J})|}$ in the extended Hilbert space, and we can optimize the TQS by minimizing $\langle\Psi|\mathscr{H}|\Psi\rangle$, which follows the standard procedure.

Once we have the family of ground states $\psi(\mathbf{s},\mathbf{J})$, an immediate application is to estimate the physical parameters $\mathbf{J}$ using samples $\mathbf{s}$ from the wave function. This follows trivially from the conditional probability:

$$P(\mathbf{J}|\mathbf{s}) = \frac{P(\mathbf{s},\mathbf{J})}{P(\mathbf{s})}. \tag{3.34}$$

In practice, given a set of measurements $\{\mathbf{s}_k\}$, $\mathbf{J}$ can be predicted using standard maximum likelihood estimation [199], by maximizing the log-likelihood functional,

$$\mathscr{L}(\mathbf{J}) = \sum_k \log P(\mathbf{s}_k|\mathbf{J}). \tag{3.35}$$

In this way, we can efficiently determine physical properties of a quantum system with few measurements. Details of the implementation can be found in Appendix 3.2.5.

This task is somewhat similar to shadow tomography [200, 201], in the sense that we are predicting properties of a quantum system with a few measurements, but with more restrictions and with certain prior knowledge required. On the other hand, Ref. [195] considered another similar task of recognizing phases from measurements using machine learning, which is formulated as a classification task. In comparison, our task falls in the middle of the two mentioned above, and to the best of our knowledge, it has never been proposed. Under this setting, the TQS can handle this task extremely efficiently. In fact, with the prior knowledge that a quantum state $|\psi\rangle$ comes from a family of states $|\psi(\mathbf{J})\rangle$, we can efficiently determine the physical parameters $\mathbf{J}$, with as few as one measurement only.

Furthermore, we show that the TQS can transfer knowledge to new systems it has never seen before. This follows the pre-training plus fine-tuning methodology commonly adopted in natural language models [184, 185]. In the zero-shot setting [187], after training on the family of Hamiltonians $\hat{H}(\mathbf{J})$, TQS can generate the ground state of new Hamiltonians $\hat{H}(\mathbf{J}^*)$ with $\mathbf{J}^* \notin \{\mathbf{J}\}$, albeit with slightly larger error. When higher accuracy is desired, one can fine-tune the TQS on the specific Hamiltonian $\hat{H}(\mathbf{J}^*)$, to obtain accurate results within a much shorter time comparing to learning from scratch.

3.2.3 Results

As a prototypical test bed, we first examine the 1D transverse field Ising (TFI) model, whose Hamiltonian is

$$\hat{H} = -J\sum_{i=1}^{n-1} \sigma_i^z \sigma_{i+1}^z - h\sum_{i=1}^{n} \sigma_i^x, \tag{3.36}$$

where J is the coupling constant and σ^z and σ^x are Pauli matrices. In Appendix 3.2.5, we also provide numerical results on the 1D XYZ model.

Ground state calculations

To begin with, we pre-train the TQS on the family of TFI Hamiltonians $\hat{H}(n,h)$, specified in Eq. (3.36). We fix $J = 1$, and assume a uniform distribution of the transverse field $h \in [0.5, 1.5]$. The system size n can take any even integer value with equal probability in the range of $[10, 40]$. We explicitly enforced parity and spin flip symmetry on the TQS, with details elaborated in Appendix 3.2.5.

After pre-training for 10^5 iterations, we plot the ground state energy, E, and magnetization along the z direction, $m_z = \sum_i \langle \sigma_i^z \rangle / N$, for $n = 40$, $h \in [0, 2]$, in Fig. 3.15. Since we explicitly symmetrized the TQS with the $|0\rangle \leftrightarrow |1\rangle$ spin flip symmetry, we always have $\langle m_z \rangle = 0$, so $\langle |m_z| \rangle$ is plotted instead. Note that while the TQS is only trained in the range of $h \in [0.5, 1.5]$, it can infer the properties of the ground state when $h \in [0, 0.5)$ and $h \in (1.5, 2]$ with slightly larger error, without any additional inputs except the value of h.

Finite-size scaling can be easily carried out using TQS. With a variable input length, we can represent an arbitrary number of degrees of freedom within a single TQS model. Using the same model trained with $h \in [0.5, 1.5]$, in Fig. 3.16 we show that finite-size scaling analysis on the TFI model correctly identifies the phase transition point $h = 1$, and the predicted critical exponents satisfy $\beta/\nu = 0.130 \pm 0.010$, which match the theoretical predictions $\beta = 1/8$, $\nu = 1$. Details of the calculation can be found in Appendix 3.2.5.

Similar experiments are carried out where the TQS is trained in the range $h \in [0, 0.5] \cup [1.5, 2]$, and the results are shown in Fig. 3.17. Although training is only carried out either deep in the ferromagnetic phase or paramagnetic phase, TQS can still infer the ground state energy and magnetization of TFI near the phase transition with reasonable accuracy.

However, in Appendix 3.2.5 we show that, this interpolated state undergoes phase transition at $h = 1.24$ instead of $h = 1$, with critical exponents different from the usual Ising transition. Without access to training data near the phase boundary, TQS cannot accurately predict the phase transition. Rather, it generates a fictitious physical system with its own critical

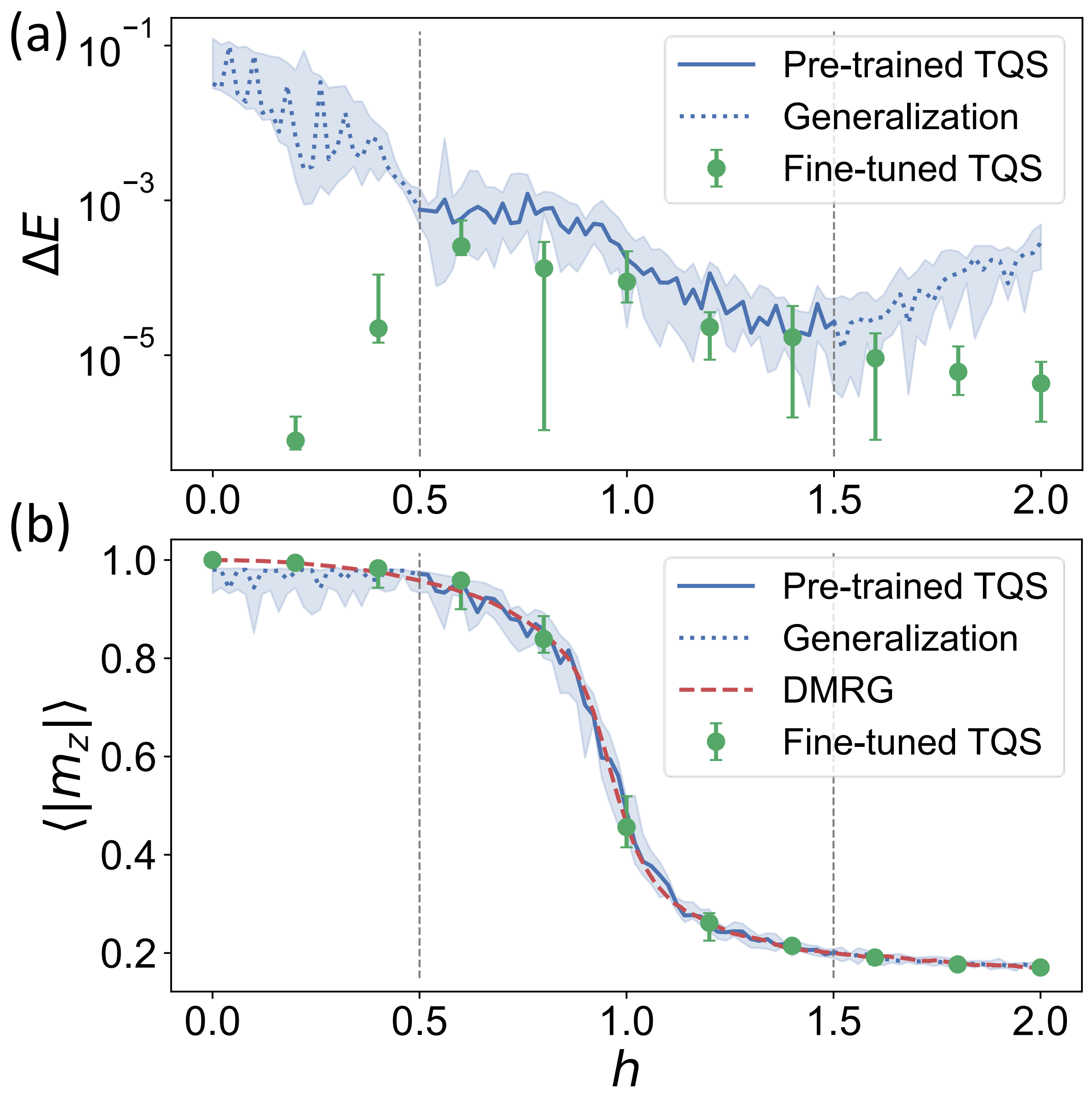


Figure 3.15. Results on the ground state of the TFI Hamiltonian, Eq. (3.36), with $n = 40$. Lines and data points are medians of 10 estimations, while shaded regions and error bars enclose 10th to 90th percentile. Dotted lines are generalizations to regions TQS has not been trained on. (a) The relative error of the ground state energy, $\Delta E = |(E - E_{\text{ground}})/E_{\text{ground}}|$. E_{ground} is estimated with DMRG, which is accurate up to 10^{-10}. (b) Absolute value of the magnetization along the z direction, $\langle|m_z|\rangle$. We can observe the transition near $h = 1$.

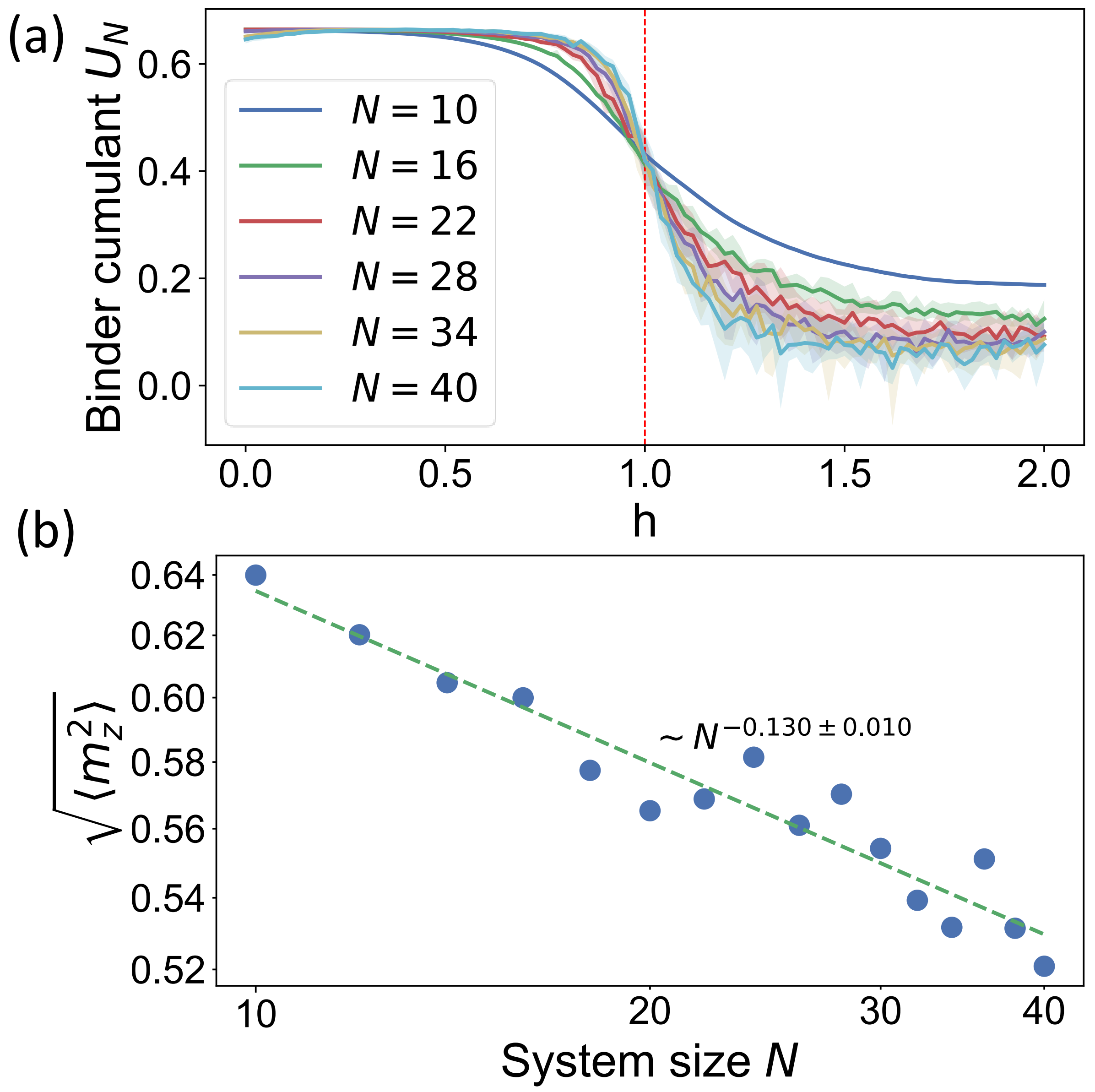


Figure 3.16. Finite-size scaling calculations on the TFI model, using the TQS trained with $h \in [0.5, 1.5]$. (a) Binder cumulant [202], $U_N = 1 - \frac{\langle m_z^4\rangle_N}{3\langle m_z^2\rangle_N^2}$, plotted for various system sizes N. At the critical point h_c, U_N is invariant with the system size N, and finding the crossing of various U_N curves can help us determine the critical point. In this figure, we identify $h_c = 1$, which agrees with the theoretical prediction. (b) Finite-size scaling of the mean-square-root magnetization [203] at the critical point $h = 1$. Using the finite-size scaling ansatz [204], at the critical point, $\sqrt{\langle m_z^2\rangle}|_{h_c} \sim N^{-\beta/\nu}$. A linear fit on the log-log scale gives $\beta/\nu = 0.130 \pm 0.010$, which matches the theoretical values $\beta = 1/8$ and $\nu = 1$.

behaviors.

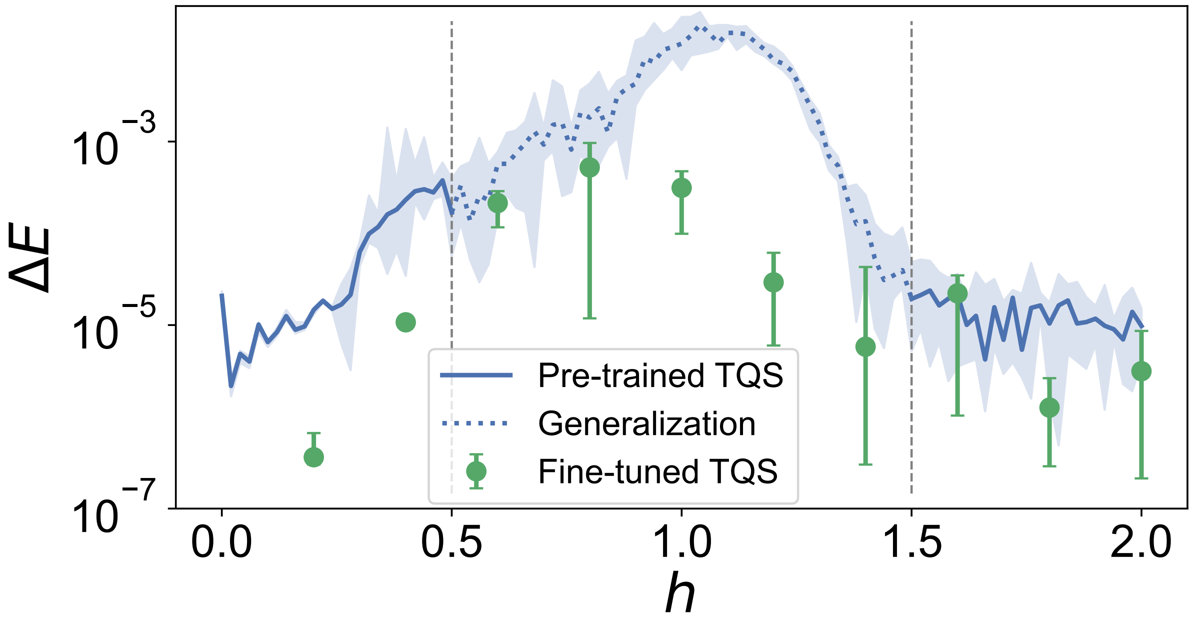


Figure 3.17. Relative error of the ground state energy of the TFI Hamiltonian, Eq. (3.36), with $n = 40$. Lines and data points are medians of 10 estimations, while shaded regions and error bars enclose 10^{th} to 90^{th} percentile. Dotted lines are generalizations to regions TQS has not been trained on. Data points below 10^{-7} are not shown for a clearer illustration.

At this point, we further fine-tune the TQS on specific points $\hat{H}(n^*, h^*)$ for an additional 2×10^3 iterations, and the results are also shown in Figs. 3.15, 3.17. Outside of the pre-trained region, the accuracy improved dramatically up to a few orders of magnitudes. Within the pre-trained region, there is also a small improvement in accuracy, but not as much since the pre-trained model already works well.

As a further test, we fix $h = 1$, and compute the ground state energy of systems with different sizes $n \in [10, 80]$ (using the model trained in $h \in [0.5, 1.5]$). The result is plotted in Fig. 3.18. Again, even if the pre-trained model has never seen any system with more than 40 spins, it can generalize to much larger systems, and their energy estimations can be greatly improved by fine-tuning for an additional 2×10^3 iterations.

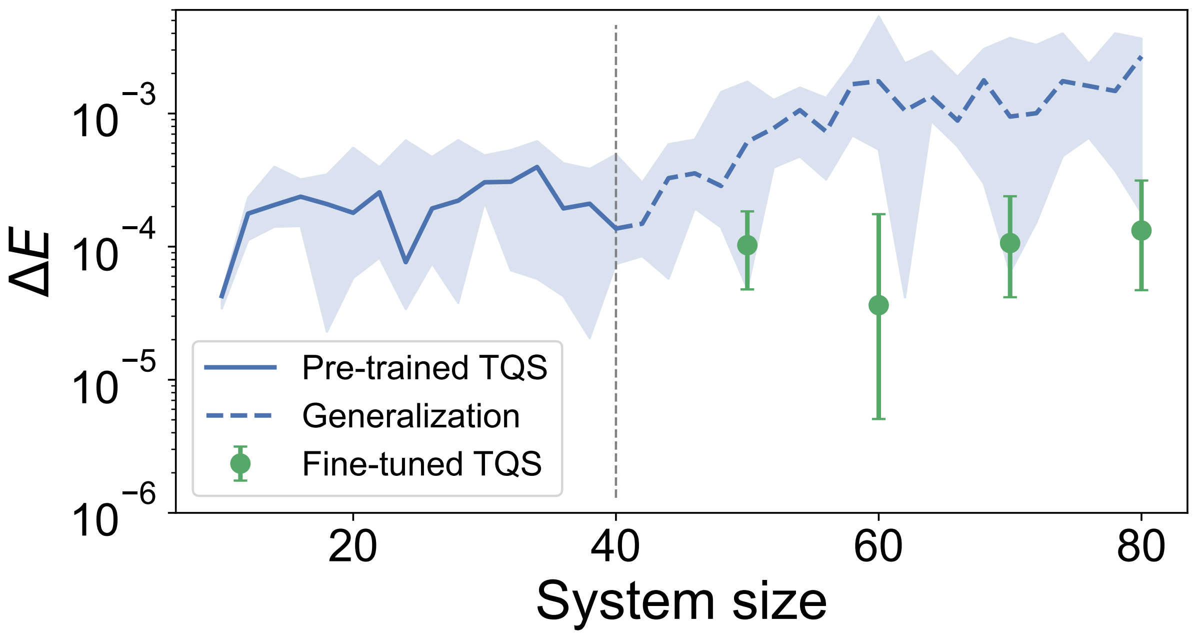


Figure 3.18. Relative error of the ground state energy of the TFI Hamiltonian, Eq. (3.36), with $h = 1$. Lines and data points are medians of 10 estimations, while shaded regions and error bars enclose 10th to 90th percentile. Dashed lines are generalizations to regions TQS has not been trained on. The pre-trained TQS can infer the ground state energy of much larger systems than what it is trained on, without any additional input except the system size n, albeit with slightly larger error. By fine-tuning with an additional 2×10^3 iterations, the accuracy improves by an order of magnitude.

Predicting parameters

Next, with the learned distribution $P(\mathbf{s},n,h)$, we want to predict the transverse field h using experimentally available measurements. To this end, we simulate the experiment by computing the ground state of the TFI model using the density matrix renormalization group (DMRG) [173, 205], and generate a synthetic dataset with projective measurements in the computational basis. Details of DMRG calculations can be found in Appendix 3.2.5.

We fix $n = 40$, and predict h by maximizing Eq. (3.35), the log-likelihood functional, with varying number of measurements. The results are shown in Fig. 3.19. Surprisingly, with as few as one measurement, TQS gives reasonable estimations of h. Increasing the number of measurements improves the quality of prediction, and an empirical power law scaling of the prediction error versus the number of measurements is observed.

3.2.4 Discussion

In summary, our results demonstrate how the TQS learns various ground state properties of a physical system, and appropriately uses the acquired knowledge to solve new problems. TQS marks the first step towards a general purpose model for quantum physics. Although we only explored here the ground states of many-body Hamiltonians, it is possible to encode many additional operations and information into the TQS, such as unitary transformations, time evolution, positive operator-valued measurements, etc. Thanks to the flexibility of neural sequence models and the transformer architecture, all the additional information can be formulated as new tokens to be passed into the embedding layer, thus maintaining the model structure simple and unified.

Limited by available computational resources, we were unable to train larger models for a wider range of tasks. But we believe that, with the advancements in the development of new computing paradigms such as MemComputing [7], such models can be pushed even further. This would help researchers understand various challenging quantum phenomena, and assist

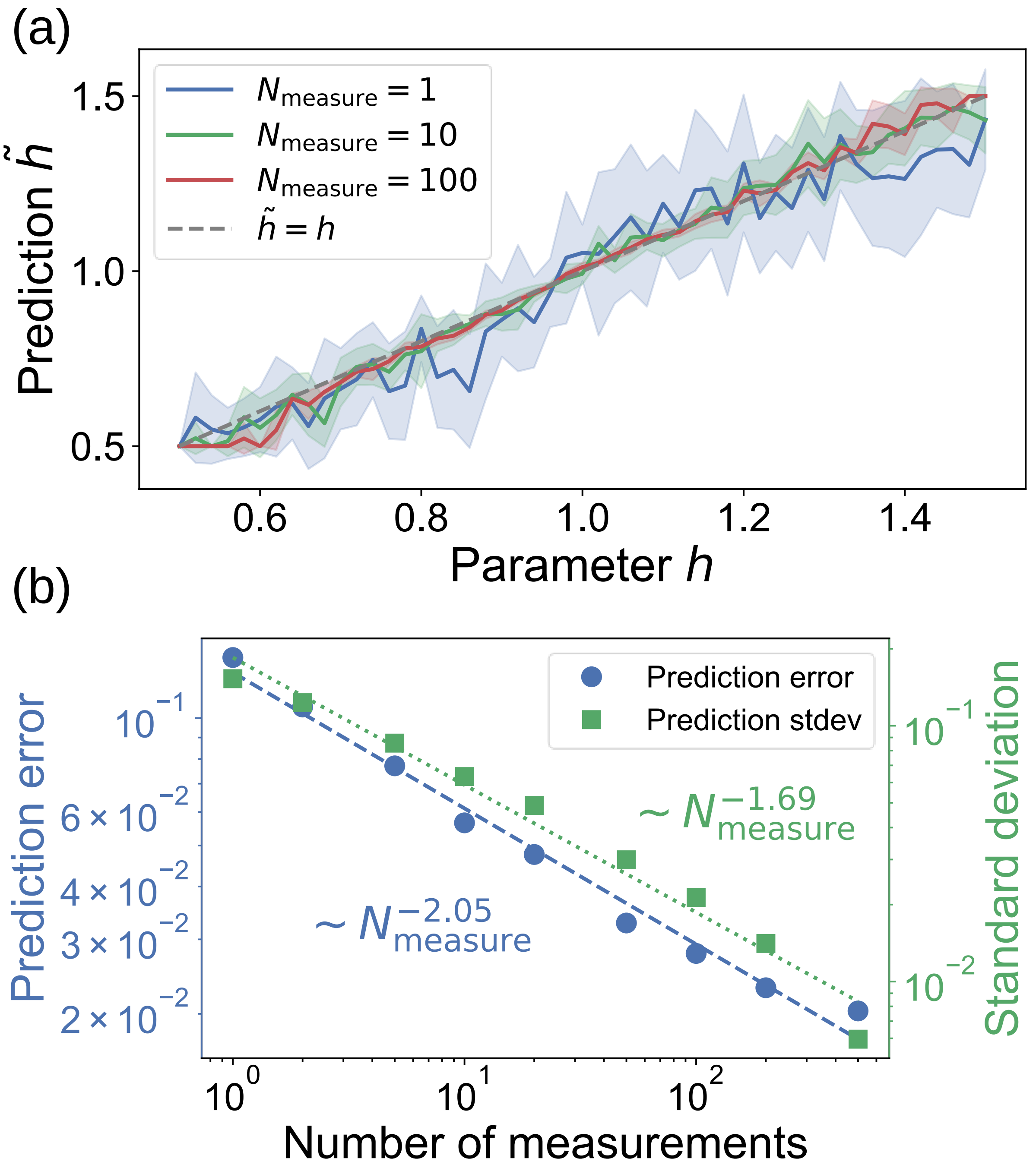


Figure 3.19. (a) The predicted field strength $\tilde{h}$ vs. the actual field strength h, with varying number of measurements. Solid lines are mean values of 10 predictions, and shaded regions enclose one standard deviation. The dashed line represents the expected result, $\tilde{h} = h$. (b) Scaling of the prediction error, $|\tilde{h} - h|$, and standard deviation, $\sigma_{\tilde{h}}$, vs. the number of measurements. Each data point is computed with 10 predictions. We observe an empirical power law scaling, with $|\tilde{h} - h| \sim N_{\text{measure}}^{-2.05}$ and $\sigma_{\tilde{h}} \sim N_{\text{measure}}^{-1.69}$.

them in the design and characterization of near-term quantum devices.

3.2.5 Appendix

Transformer implementation details

As illustrated in Fig. 3.14, we adopt the standard encoder-only transformer structure [183]. The discrete spin variables s_i are first one-hot encoded [206], and the parameters J_j are represented with a scaled one-hot vector. To input interaction strengths and external fields, the scale is the value of the interaction itself. To input the system size n, we choose the scale to be $\ln n$, and append another parity dimension to the input vector, indicating whether n is even or odd.

Since the input does not entirely consist of one-hot vectors, the embedding layer performs a linear transformation, mapping the input vectors into a d_e dimensional embedding space.

We use a mixed-style positional encoding. The spin variables s_i have a well-defined position, and we use the D-dimensional sinusoidal positional encoding [183, 207] on them, where D is the spatial dimension of the physical system. This ensures that the neural network can correctly generalize to larger system sizes it has never been trained on before. On the other hand, the parameters J_j do not have a position, and we use a learnable positional encoding [188] instead.

After embedding and positional encoding, we pass the embedded inputs through N identical transformer encoder layers, with structures defined in [183]. The feed-forward sublayer consists of two linear layers, with the hidden dimension in the middle also being d_e. We use multi-head self-attention [183] with 8 heads for the larger model, and 2 heads for the smaller model. ReLU activation [208] is used throughout the neural network.

After N transformer encoder layers, we use two output heads to model the amplitude and phase of the target wave function. The amplitude head is a linear layer followed by a softmax activation [209], and the phase head is a linear layer followed by a softsign activation, which is defined in [178] and computes the function $(-\infty < x < +\infty)$

$$\text{softsign}(x) = \frac{x}{1+|x|}. \tag{3.37}$$

We scale the softsign output by π, to output a phase in the range of $(-\pi, \pi)$.

The TQS mentioned in the main text has $N = 8$ transformer encoder layers with embedding size $d_e = 32$, and the number of parameters is about 7.7×10^4. The smaller model in Appendix 3.2.5 has $N = 2$ transformer encoder layers with embedding size $d_e = 16$, resulting in 5.2×10^3 parameters. The implementation of TQS is carried out using the PyTorch library [210].

Variational optimization of the ground state energy

TQS is trained by minimizing the ground state energies of a family of Hamiltonians, $\{\hat{H}(\mathbf{J})\}$.

For a single Hamiltonian $\hat{H}$, the energy derivative reads [136]:

$$\frac{\partial E}{\partial \theta_k} = 2\text{Re}\left(\left\langle E_{\text{loc}}(\mathbf{s}) \frac{\partial \log \psi(\mathbf{s})^*}{\partial \theta_k} \right\rangle_{P(\mathbf{s})}\right) \tag{3.38}$$

where $\langle \cdot \rangle_{P(\mathbf{s})}$ denotes expectation over the distribution $P(\mathbf{s})$, and

$$E_{\text{loc}}(\mathbf{s}) = \sum_{\mathbf{s}'} \hat{H}(\mathbf{s}, \mathbf{s}') \frac{\psi(\mathbf{s}')}{\psi(\mathbf{s})} \tag{3.39}$$

is the local energy estimator.

Since the autoregressive wave function is explicitly normalized, it is shown in [178] that the variance of the gradient can be reduced by subtracting a baseline energy,

$$\frac{\partial E}{\partial \theta_k} = 2\text{Re}\left(\left\langle \left(E_{\text{loc}}(\mathbf{s}) - \langle E_{\text{loc}}(\mathbf{s}') \rangle_{P(\mathbf{s}')}\right) \frac{\partial \log \psi(\mathbf{s})^*}{\partial \theta_k} \right\rangle_{P(\mathbf{s})}\right) \tag{3.40}$$

without introducing bias. This follows from

$$\begin{aligned}&\mathrm{Re}\left\langle \langle E_{\mathrm{loc}}(\mathbf{s}')\rangle_{P(\mathbf{s}')} \frac{\partial \log \psi(\mathbf{s})^*}{\partial \theta_k} \right\rangle_{P(\mathbf{s})} \\ =&\langle E_{\mathrm{loc}}(\mathbf{s}')\rangle_{P(\mathbf{s}')} \sum_{\mathbf{s}} P(\mathbf{s}) \frac{1}{2} \frac{1}{P(\mathbf{s})} \frac{\partial P(\mathbf{s})}{\partial \theta_k} \\ =&\frac{E}{2} \frac{\partial}{\partial \theta_k} \sum_{\mathbf{s}} P(\mathbf{s}) = \frac{E}{2} \frac{\partial}{\partial \theta_k} 1 = 0.\end{aligned} \tag{3.41}$$

In our problem setting, we have a family of Hamiltonians $\hat{H}(\mathbf{J})$ parameterized by $\mathbf{J}$, with ground state energies $E_g(\mathbf{J})$. Without loss of generality, we suppose all $E_g < 0$; otherwise we can simply shift the energy levels by adding a constant. Then, we define the super-Hamiltonian,

$$\hat{\mathscr{H}} = \bigoplus_{\mathbf{J}} \frac{\hat{H}(\mathbf{J})}{|E_g(\mathbf{J})|}, \tag{3.42}$$

to be the direct sum of all (possibly infinite) Hamiltonians $H(\mathbf{J})$, weighted by their ground state energies, $\frac{1}{|E_g(\mathbf{J})|}$. Note that $\hat{\mathscr{H}}$ is block diagonal, with no interaction across different $\mathbf{J}$. One can easily show that, the ground state of $\hat{\mathscr{H}}$ is the direct sum of all ground states, $|\Psi\rangle = \bigoplus_{\mathbf{J}} |\psi(\mathbf{J})\rangle$,

$$\begin{aligned}\hat{\mathscr{H}}|\Psi\rangle =&\left(\bigoplus_{\mathbf{J}} \frac{\hat{H}(\mathbf{J})}{|E_g(\mathbf{J})|} \right) \left(\bigoplus_{\mathbf{J}} |\psi(\mathbf{J})\rangle \right) \\ =&\bigoplus_{\mathbf{J}} \frac{\hat{H}(\mathbf{J})|\psi(\mathbf{J})\rangle}{|E_g(\mathbf{J})|} \\ =&-\bigoplus_{\mathbf{J}} |\psi(\mathbf{J})\rangle = -|\Psi\rangle\end{aligned} \tag{3.43}$$

with eigenvalue -1. Therefore, we can follow the standard procedure and minimize

$$\langle \Psi | \hat{\mathscr{H}} | \Psi \rangle = \sum_{\mathbf{J}} \frac{\langle \psi(\mathbf{J}) | \hat{H}(\mathbf{J}) | \psi(\mathbf{J}) \rangle}{|E_g(\mathbf{J})|}. \tag{3.44}$$

We don't have access to the exact ground state energies $E_g(\mathbf{J})$, so we instead approximate them with variationally approximated ground state energies, $\tilde{E}_g(\mathbf{J}) = \langle E_{\mathrm{loc}}(\mathbf{s}, \mathbf{J}) \rangle$, which become increasingly more accurate as optimization goes on.

In practice, at each optimization iteration, we sample a random $\mathbf{J}$ according to $P(\mathbf{J})$, and compute the energy derivative Eq. (3.40), scaled by $\frac{1}{|\tilde{E}_g(\mathbf{J})|}$. We set an upper limit of 5 to the scaling factor, to avoid divergences when $\tilde{E}_g(\mathbf{J}) \to 0$ during optimization.

The entire training procedure is carried out using the Adam optimizer [59], with $\beta_1 = 0.9$ and $\beta_2 = 0.98$. We varied the learning rate during training according to the formula,

$$\mathrm{lr}(i_{\mathrm{step}}) = 5d_e^{-0.5}\min(i_{\mathrm{step}}^{-0.75}, i_{\mathrm{step}} i_{\mathrm{warmup}}^{-1.75}) \tag{3.45}$$

where d_e is the embedding size of the model, i_{step} is the current number of training steps, and i_{warmup} is the number of warm up steps. We used $i_{\mathrm{warmup}} = 4000$. This corresponds to linearly increasing the learning rate during the first 4000 iterations, and polynomially decreasing it during the rest of the training. This learning rate schedule is inspired from [183]. During fine-tuning, we use a different learning rate schedule:

$$\mathrm{lr}(i_{\mathrm{step}}) = 5d_e^{-0.5}(i_{\mathrm{step}} + 10^5)^{-0.75}. \tag{3.46}$$

Sampling algorithm

The autoregressive structure of TQS already makes sampling efficient, and the efficiency is further improved by adopting the sampling algorithm in [180], which only samples unique configuration strings.

During sampling, we first fix a large batch size, N_{batch}, and autoregressively sample the spins to form partial strings, $\mathbf{s}^k = s_1 s_2 \cdots s_i$, with associated number of occurrences, n_k. At the $(i+1)$-th sampling step, s_{i+1} is sampled from the conditional distribution $P(s_{i+1}|s_1, \cdots, s_i, \mathbf{J})$, resulting in n_{k0} occurrences of $s_{i+1} = 0$ and n_{k1} occurrences of $s_{i+1} = 1$, with $n_{k0} + n_{k1} = n_k$. After this step, we obtain two unique partial strings, $\mathbf{s}^{k0} = s_1 s_2 \cdots s_i 0$ and $\mathbf{s}^{k1} = s_1 s_2 \cdots s_i 1$, with occurrences n_{k0} and n_{k1}, respectively. This procedure starts from an empty set and is repeated until the number of unique strings reaches a maximum, N_{unique}, after which no new partial string branches are generated, and the remaining spins are sampled in the regular way.

The complexity of this sampling algorithm is approximately proportional to N_{unique} and does not depend on N_{batch}. Therefore, we can choose extremely large batch sizes to greatly improve on the accuracy of estimated expectation values, with negligible increase in computation time. For all the experiments mentioned in this paper, we choose $N_{\text{batch}} = 10^6$, $N_{\text{unique}} = 10^2$ during training, and $N_{\text{unique}} = 10^3$ during evaluations.

Implementing symmetries

The transformer architecture itself does not observe any symmetry, but most Hamiltonians do. To impose symmetries without spoiling the autoregressive structure, we follow the approaches in previous works [175, 178, 179] and explicitly symmetrize the wave function in a similar way.

Suppose $\hat{\mathscr{T}}$ is a discrete symmetry of $\hat{H}$, with $\hat{\mathscr{T}}^m = \mathbb{1}$ ($m \in \mathbb{N}$). By definition, we have $[\hat{H}, \hat{\mathscr{T}}] = 0$, and one can simultaneously diagonalize both operators within the same eigenbasis. Under this basis, the ground state $|\psi\rangle$ is also an eigenstate of $\hat{\mathscr{T}}$,

$$\hat{\mathscr{T}}|\psi\rangle = \omega_{\hat{\mathscr{T}}}|\psi\rangle \tag{3.47}$$

where $\omega_{\hat{\mathscr{T}}} = e^{2\pi i k/m}$, $k \in \mathbb{N}$. Expanding Eq. (3.47) in the computational basis, we get

$$\psi(\hat{\mathscr{T}}^{-1}\mathbf{s}) = \omega_{\hat{\mathscr{T}}}\psi(\mathbf{s}). \tag{3.48}$$

In terms of amplitude and phase, Eq. (3.48) becomes

$$\begin{aligned} A(\hat{\mathscr{T}}\mathbf{s}) &= A(\mathbf{s}), \\ \phi(\hat{\mathscr{T}}\mathbf{s}) &= \phi(\mathbf{s}) - \frac{2\pi k}{m}. \end{aligned} \tag{3.49}$$

The output wave function from TQS clearly does not satisfy Eq. (3.49). To explicitly enforce the symmetry $\hat{\mathscr{T}}$, we define

$$\begin{aligned}\tilde{P}(\mathbf{s}) &= \frac{1}{m}\sum_{n=0}^{m-1} P(\hat{\mathscr{T}}^n \mathbf{s}) \\ \tilde{\phi}(\mathbf{s}_0) &= \mathrm{Arg}\left(\sum_{n=0}^{m-1} \psi(\hat{\mathscr{T}}^n \mathbf{s}_0)\right) \\ \tilde{\phi}(\hat{\mathscr{T}}^n \mathbf{s}_0) &= \tilde{\phi}(\mathbf{s}_0) - \frac{2\pi k n}{m}\end{aligned} \tag{3.50}$$

where $\psi(\mathbf{s}) = \sqrt{P(\mathbf{s})}e^{i\phi(\mathbf{s})}$, P, ϕ are outputs from the TQS, and $\tilde{P}$, $\tilde{\phi}$ are symmetrized probability and phase, respectively. $\mathbf{s}_0$ is an arbitrary initial configuration in each symmetry sector, predefined so that the phases within the symmetry sector can be assigned consistently. We choose $\mathbf{s}_0$ to be the configuration with the smallest decimal value, converted from its binary bitstring, within each symmetry sector.

Sampling from the symmetrized wave function has almost no additional computational cost. We follow the same procedure detailed in the previous section, and apply a random symmetry operation $\hat{\mathscr{T}}^n$ to the sampled configuration $\mathbf{s}$ in the end [179, 178]. However, to compute the exact value of $\psi(\mathbf{s})$, one needs to evaluate all configurations within the symmetry sector and explicitly calculate Eq. (3.50), which is m times more expensive.

Another symmetry worth mentioning is the $U(1)$ symmetry of the Heisenberg model, which leads to zero magnetization. This symmetry is particularly easy to implement, and we follow the same method developed in [178], by setting the probability of a partial string to 0 whenever the number of up spins or down spins exceeds half of the system size.

Note that, while the Hamiltonian $\hat{H}$ may satisfy several symmetries $\hat{\mathscr{T}}_1, \hat{\mathscr{T}}_2, \cdots$, it is possible that $[\hat{\mathscr{T}}_1, \hat{\mathscr{T}}_2] \neq 0$, making it impossible to diagonalize all symmetries at the same time. However, this does not pose a problem for us. Although $\hat{\mathscr{T}}_1$ and $\hat{\mathscr{T}}_2$ do not commute in general, they do commute in certain symmetry sectors (for example, $\omega_{\hat{\mathscr{T}}_1} = \omega_{\hat{\mathscr{T}}_2} = 1$). To implement symmetries, we need to know $\omega_{\hat{\mathscr{T}}}$ as a prior knowledge, and this information then helps us determine all compatible symmetries.

As another remark, in our implementation of TQS, we only enforced the symmetries $\hat{\mathscr{T}}$ with $\omega_{\hat{\mathscr{T}}} = 1$. We noticed that, any symmetry with $\omega_{\hat{\mathscr{T}}} \neq 1$ would impose a non-trivial phase structure to the wave function, which is somewhat arbitrary and could significantly slow down the training.

Predicting parameters

With the learned distribution $P(\mathbf{s}, \mathbf{J})$, we can predict the parameters $\mathbf{J}$ from a batch of measurements $\{\mathbf{s}_i\}$. As illustrated in the main text, this is achieved through maximizing the log-likelihood functional Eq. (3.35).

We carried out the maximization using the Nelder-Mead method [211], a heuristic searching algorithm based on a moving simplex, implemented in the SciPy library [212], with a tolerance of 10^{-9}.

An alternative method to predict the parameters would be supervised fine-tuning, which adds a parameter prediction head as an additional output of TQS. This would have the advantage of reducing the computational cost to one forward pass, at the expense of fine-tuning cost. We leave this as a future work.

Note that we didn't use any phase information during the prediction. To make use of the phase structure, one needs to perform measurements in different bases, and compute a generalized likelihood function that takes all bases into account. For this, we refer the readers to [140]. Alternatively, it is possible to use informationally-complete positive operator-valued measurements (IC-POVM) to encode the complete information of a quantum state, which is developed in [142]. We can adapt the TQS structure to be compatible with IC-POVM, which we also leave as a future work.

DMRG calculations

For the 1D transverse field Ising model in the main text and the 1D XYZ model in Appendix 3.2.5, we use density matrix renormalization group (DMRG) as a benchmark to evaluate the performance of our algorithm. DMRG can be extremely accurate for 1D systems, yet performs rather poorly in 2 or more dimensions [173].

We used the TeNPy library [205] to perform DMRG calculations. For all DMRG results mentioned in the paper, we use a maximum bond dimension of 100, and terminate when the energy tolerance 10^{-10} is achieved.

Additional numerical results

Performance benchmarking In this section, we compare the accuracy and training cost of TQS with restricted Boltzmann machine (RBM) [136], another widely adopted framework for neural network quantum states.

To ensure a fair comparison, we trained a smaller TQS with 5.2×10^3 parameters (embedding size $d_e = 16$, two transformer encoder blocks), to compare with an RBM with approximately the same number of parameters (hidden-to-variable ratio $\alpha = 3$). The TQS is trained using the setting described in the previous sections, with $N_{\text{unique}} = 2000$, while the RBM is trained using stochastic reconfiguration (SR) [213, 136], contrastive divergence with 10 sampling steps (CD-10) [214] and batch size 24800. Under this setting, the computational cost for each training iteration is approximately the same. To model continuous physical parameters in RBMs, we use continuous visible neurons normalized to $[-2,2]$, together with regular binary neurons with values ± 1 for the spin variables.

At this point, we try to reproduce the experiment described in the main text using RBMs. We focus on the transverse field Ising (TFI) model, with the transverse field h as an additional input to the neural network. The RBM is trained for 10^5 iterations, and the learning rate decreases according to the formula

$$\mathrm{lr}(i_{\text{step}}) = \mathrm{lr}_{\max} i_{\text{step}}^{-0.5} \tag{3.51}$$

where i_{step} is the current number of training steps, and $\text{lr}_{\text{max}} = 0.02$. TQS is also trained for 10^5 iterations, and the results are shown in Figs. 3.20, 3.21.

In Fig. 3.20, the training range is $h \in [0.5, 1.5]$, and the RBM learned an energy curve that is almost linear in h. And in Fig. 3.21 the training range is $h \in [0, 0.5] \cup [1.5, 2]$, but the RBM only learned the properties in the $[1.5, 2]$ range. In comparison, TQS did an almost perfect job in both cases.

This result is expected, since TQS is designed for flexibility and is able to learn different quantum states at the same time, even with a tiny model size. On the other hand, RBM can accurately represent a single quantum state, but it is much less flexible when it comes to a family of quantum states.

As another test, we train both models at a single data point $h = 1$ for 10^5 iterations, and the result is shown in Fig. 3.22. RBM works very well in this case, converging in about 10^3 iterations, and does not improve much afterwards. On the other hand, TQS converges much slower, but continues to see improvements up to 10^5 iterations.

Again, this result is expected. With a simple structure, RBM can be easily trained using the SR algorithm, leading to a fast convergence. However, TQS has a much more sophisticated structure, and needs a warm-up period in the learning rate schedule for a smoother convergence. This makes the training slower, but with a potentially higher final accuracy.

Finite-size scaling The idea of finite-size scaling [50] can help us understand divergent behaviors in the thermodynamic limit using only numerical results in finite systems. Assume we have some physical quantity Ω that diverges in the thermodynamic limit at a critical value h_c,

$$\Omega(h) \sim |\Delta h|^{-\omega}, \tag{3.52}$$

where $\Delta h = (h - h_c)/h_c \to 0$. The correlation length, $\xi(h) \sim |\Delta h|^{-\nu}$, also diverges with critical

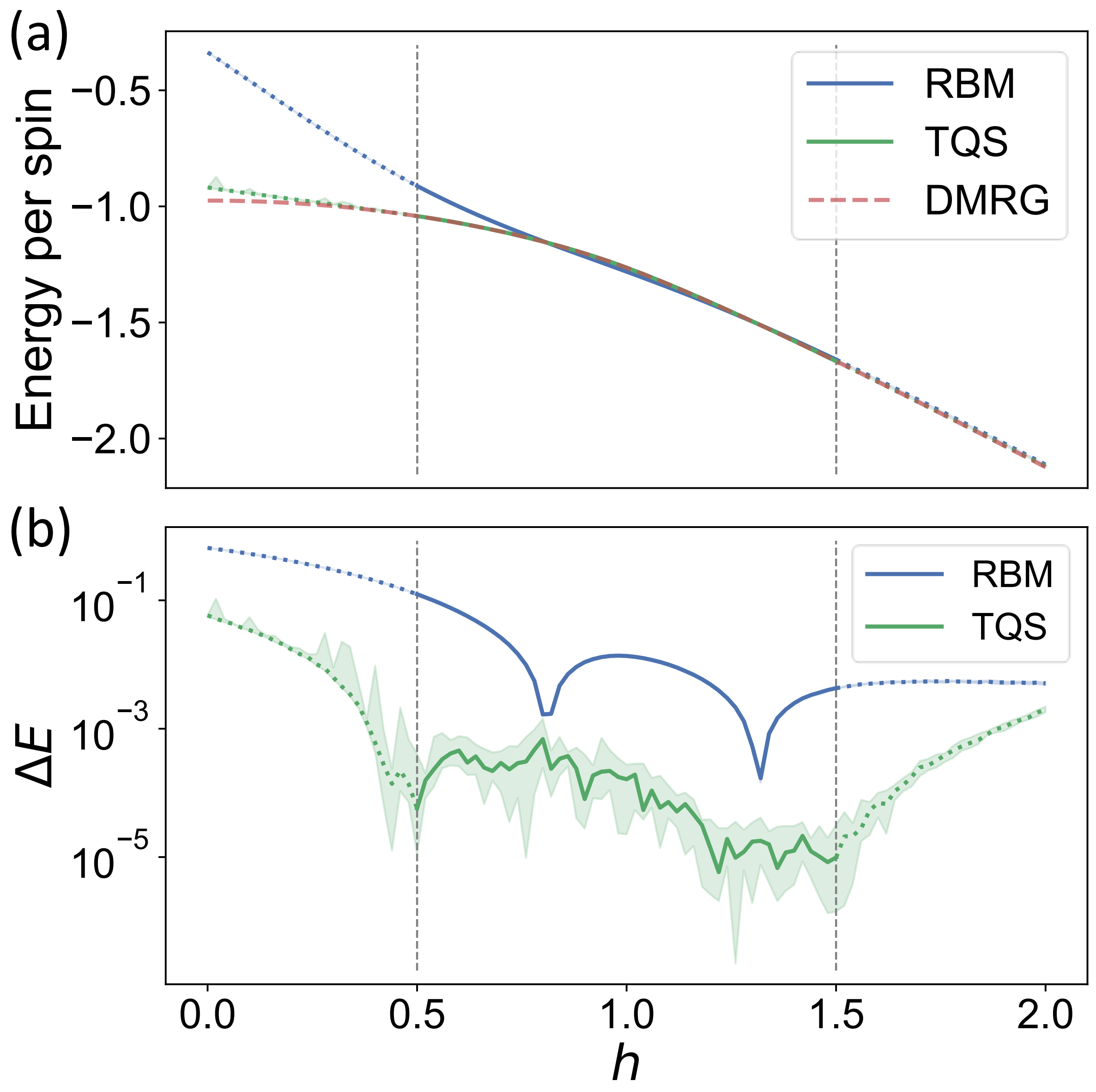


Figure 3.20. Comparison of TQS and RBM on the ground state of the TFI model, with a variable transverse field h. (a) Energy per spin and (b) Relative error of the ground state energy, $\Delta E = |(E - E_{\text{ground}})/E_{\text{ground}}|$. Both models are trained in the range $h \in [0.5, 1.5]$.

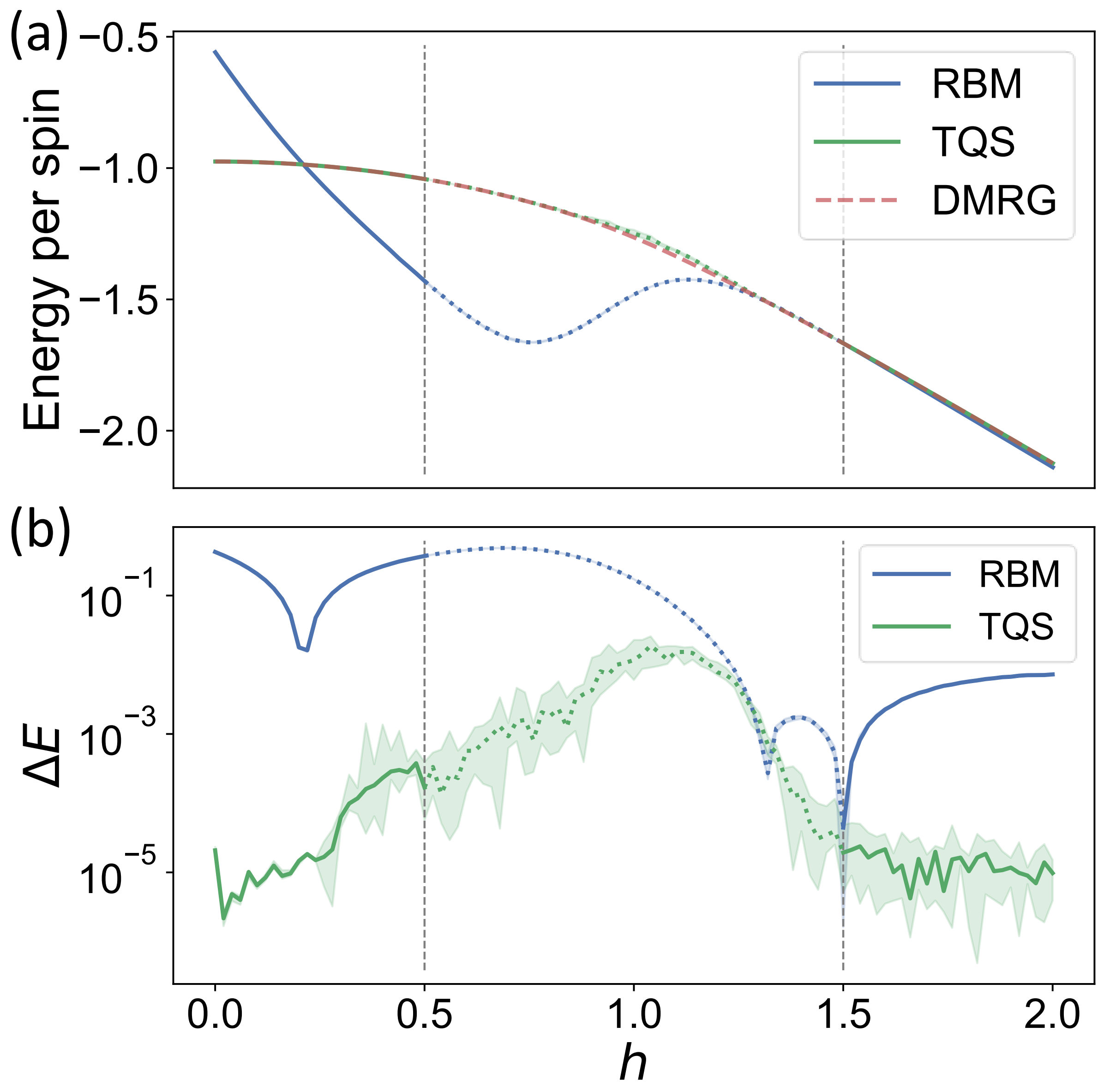


Figure 3.21. Comparison of TQS and RBM on the ground state of the TFI model, with a variable transverse field h. (a) Energy per spin and (b) Relative error of the ground state energy. Both models are trained in the range $h \in [0, 0.5] \cup [1.5, 2]$.

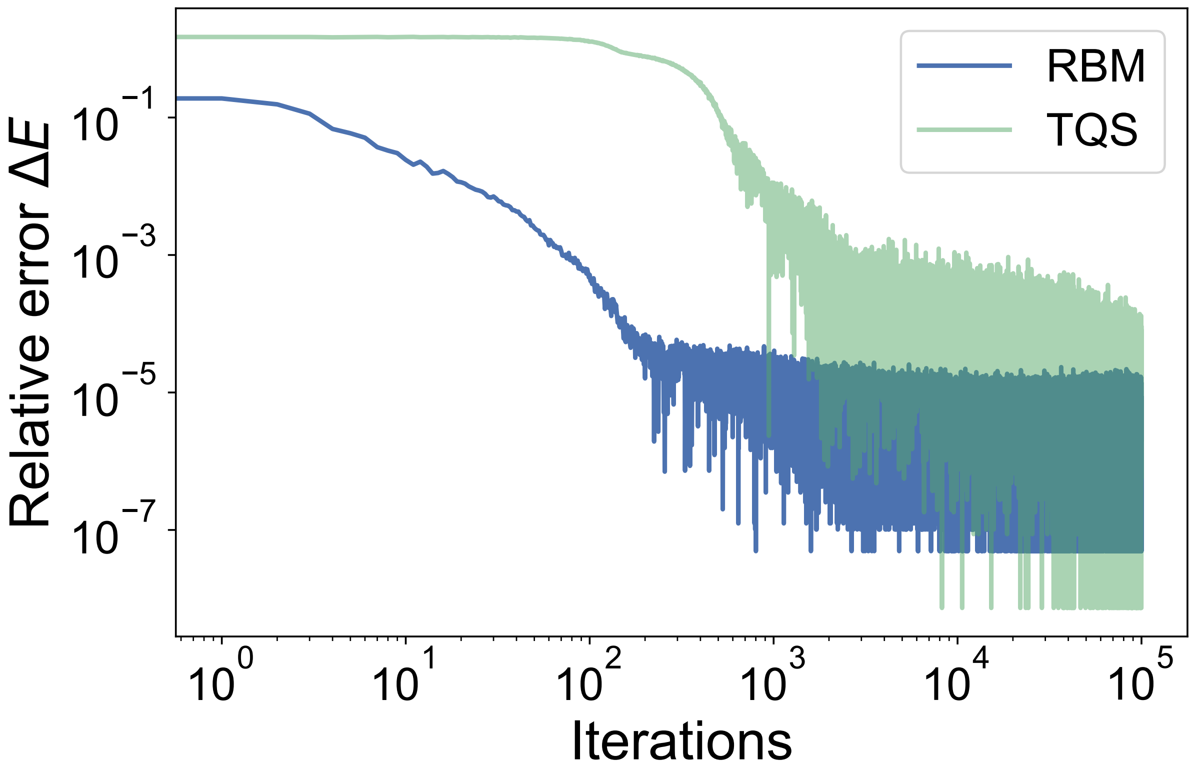


Figure 3.22. The training curve of TQS and RBM on the ground state of the TFI model with transverse field $h = 1$. On a single data point, RBM converges faster than TQS.

exponent ν. Therefore, Ω correlates with ξ as

$$\Omega \sim \xi^{\omega/\nu}. \tag{3.53}$$

For a finite system of linear size N, the behavior of $\Omega(h,N)$ deviates according to the ratio ξ/N. When $\xi \ll N$, finite-size effects are negligible, and Eq. (3.53) is preserved. However, since the correlation length cannot exceed the system size in finite systems, if $\xi \gg N$, Ω has to scale with N instead. This leads to the finite-size scaling ansatz

$$\Omega(h,N) \sim \xi^{\omega/\nu} f(N/\xi), \tag{3.54}$$

where $f(x)$ is a scaling function that satisfies

$$f(x) \sim \begin{cases} \text{const}, & x \to \infty, \\ x^{\omega/\nu}, & x \to 0. \end{cases} \tag{3.55}$$

By defining $g(x) = x^{-\omega} f(x^{\nu})$, we can rewrite Eq. (3.54) as

$$\Omega(h,N) \sim N^{\omega/\nu} g(N^{1/\nu}|\Delta h|) \tag{3.56}$$

Therefore, at the critical point h_c, Ω scales as

$$\Omega(h_c,N) \sim N^{\omega/\nu}. \tag{3.57}$$

Determining h_c is another task on its own. A common method is to compute the Binder cumulant [202],

$$U_N = 1 - \frac{\langle m_z^4 \rangle_N}{3\langle m_z^2 \rangle_N^2}, \tag{3.58}$$

which is invariant with system size N at the critical point [202]. Therefore, the crossing point of

$U_N - h$ curves for different N gives the critical point h_c.

In Fig. 3.16(a) in the main text, we used the Binder cumulant to show that TQS can correctly identify the TFI phase transition at $h = 1$. And in Fig. 3.16(b), we computed the ratio β/ν by fitting the scaling of magnetization m_z to Eq. (3.57). On a side note, since TQS is explicitly symmetrized to have $\langle m_z \rangle = 0$, we followed the method in [203] and used the mean-square-root magnetization $\sqrt{\langle m_z^2 \rangle}$ instead.

The results in Fig. 3.16 are obtained using a TQS model trained in $h \in [0.5, 1.5]$ near the phase boundary. What if the TQS has never been trained on any data near the critical point? To test this, we performed the same analysis using TQS trained in $h \in [0, 0.5] \cup [1.5, 2]$, either deep in the paramagnetic or ferromagnetic regime. The results are shown in Fig. 3.23. This time, TQS failed to find the correct critical point $h_c = 1$. However, quite surprisingly, TQS managed to find a plausible interpolation between the two phases, with a new critical point $h = 1.24$, and critical exponents $\beta/\nu = 0.277 \pm 0.006$.

Of course, this interpolated phase transition is not physical, and the computed critical exponents seem to suggest that this fictitious system has a fractal dimension between 1 and 2. It would be an interesting future work to look into the neural network and analyze what actually happened here.

Heisenberg XYZ model In this section, we further benchmark the performance of TQS with additional numerical experiments. We focus on the 1D Heisenberg XYZ model in a longitudinal field [215], whose Hamiltonian is given by

$$\begin{aligned}\hat{H} = J \sum_{i=1}^{n-1} \Big[&(1+\gamma)\sigma_i^x \sigma_{i+1}^x + (1-\gamma)\sigma_i^y \sigma_{i+1}^y \\ &+\Delta \sigma_i^z \sigma_{i+1}^z \Big] + h \sum_{i=1}^{n} \sigma_i^z \end{aligned} \tag{3.59}$$

We fix $J = 1, \gamma = 0.2$, and consider the parameter range $h \in [0, 1]$, $\Delta \in [-1, 1]$. The system size n can take any even integer value with equal probability in the range of $[10, 40]$. The TQS has the same structure as the one described in the main text, with 8 layers and embedding

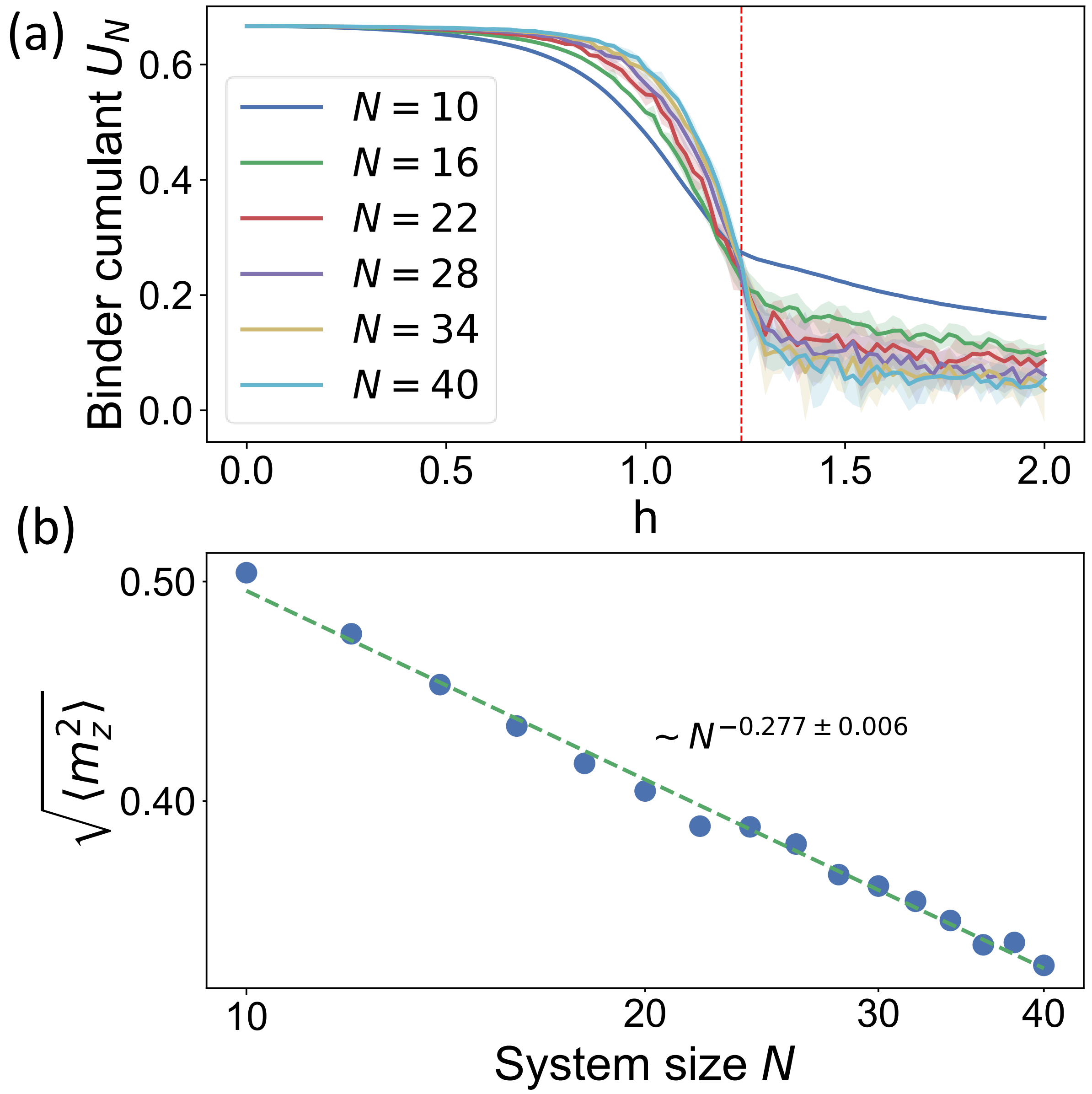


Figure 3.23. Finite-size scaling calculations on the TFI model, using the TQS trained in $h \in [0, 0.5] \cup [1.5, 2]$. (a) Binder cumulant [202], $U_N = 1 - \frac{\langle m_z^4 \rangle_N}{3\langle m_z^2 \rangle_N^2}$, plotted for various system sizes N. The curves cross at $h = 1.24$. (b) Finite-size scaling of the mean-square-root magnetization at the critical point $h = 1.24$. A linear fit on the log-log scale gives $\beta/\nu = 0.277 \pm 0.006$.

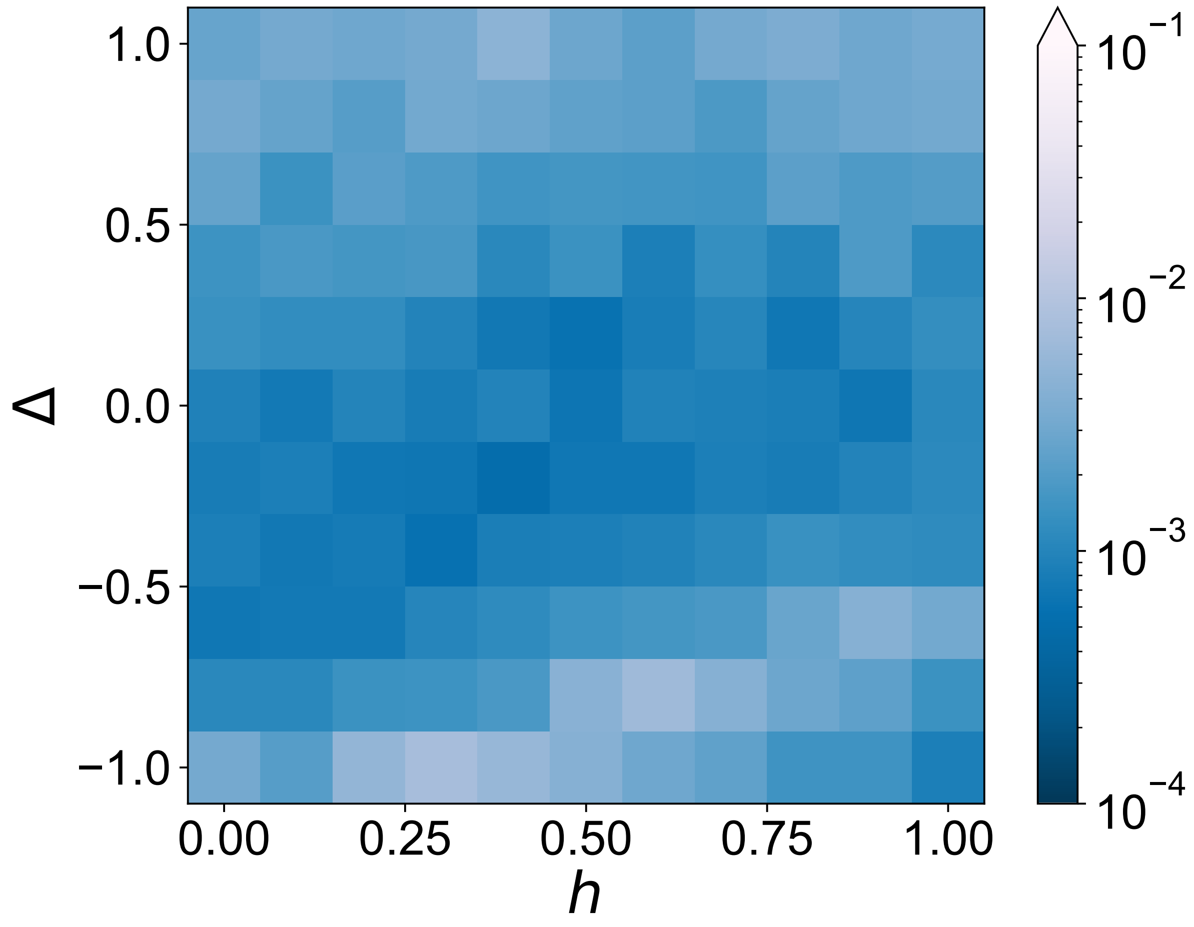


Figure 3.24. The relative error of the ground state energy, $|(E - E_{\text{ground}})/E_{\text{ground}}|$, plotted against the external field h and longitudinal interaction strength Δ, with $n = 40$. In this figure, h and Δ are in the pre-training range.

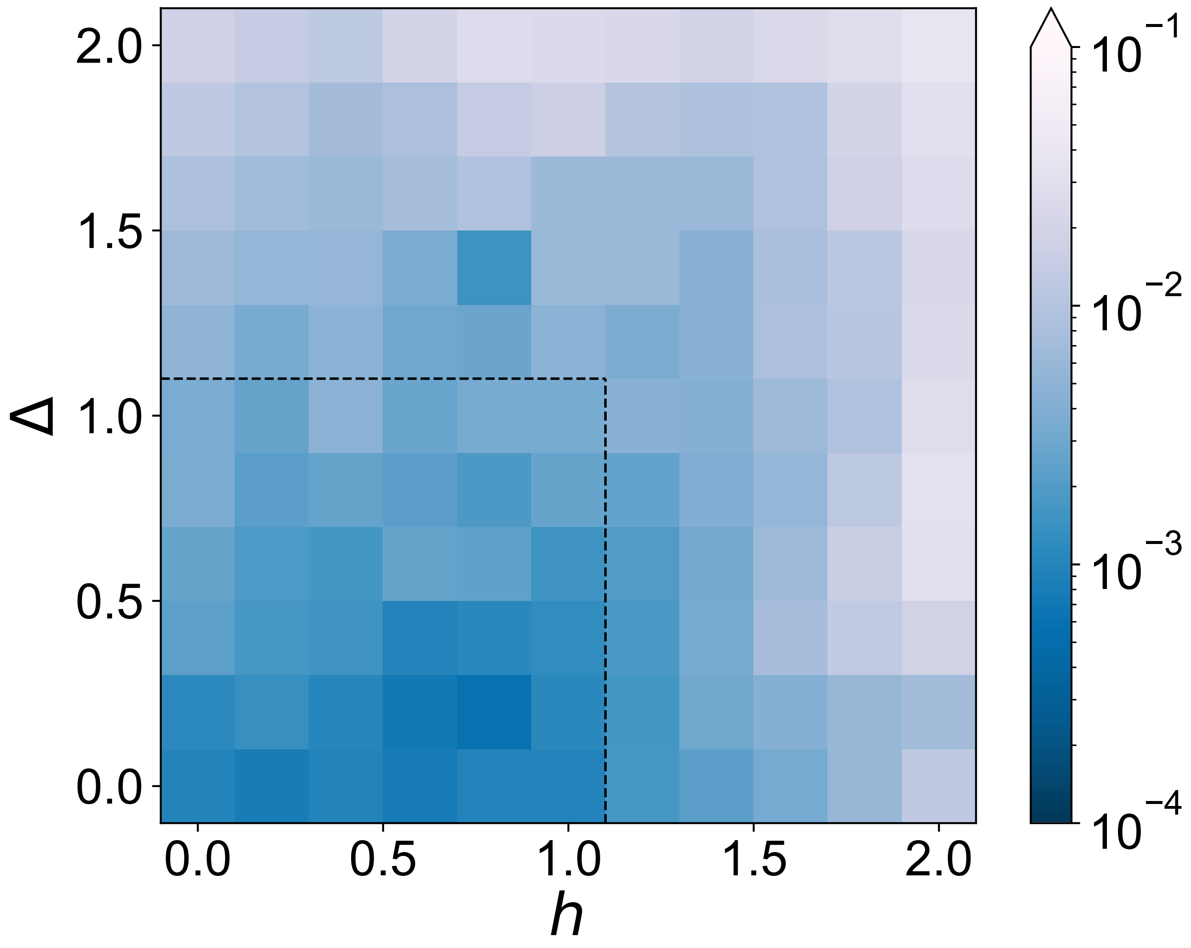


Figure 3.25. The relative error of the ground state energy, $|(E - E_{\text{ground}})/E_{\text{ground}}|$, in an extended parameter range. The bottom left corner is part of the pre-training range, separated with black dashed lines for visual clarity.

size 32. We trained the TQS for 10^5 iterations without implementing any symmetry, and the relative errors of the ground state energy, $|(E - E_{\text{ground}})/E_{\text{ground}}|$, for system size $n = 40$, are plotted in Figs. 3.24, 3.25.

Fig. 3.24 shows the results in the pre-trained range, $h \in [0,1], \Delta \in [-1,1]$, and the accuracy is at the order of 10^{-3}. In Fig. 3.25, we extended the parameter range to $h \in [0,2], \Delta \in [0,2]$, with pre-trained and extended parameter ranges separated by black dashed lines. TQS can still reasonably infer the ground state properties outside of the pre-trained range, but its accuracy gradually decreases as we move further away.

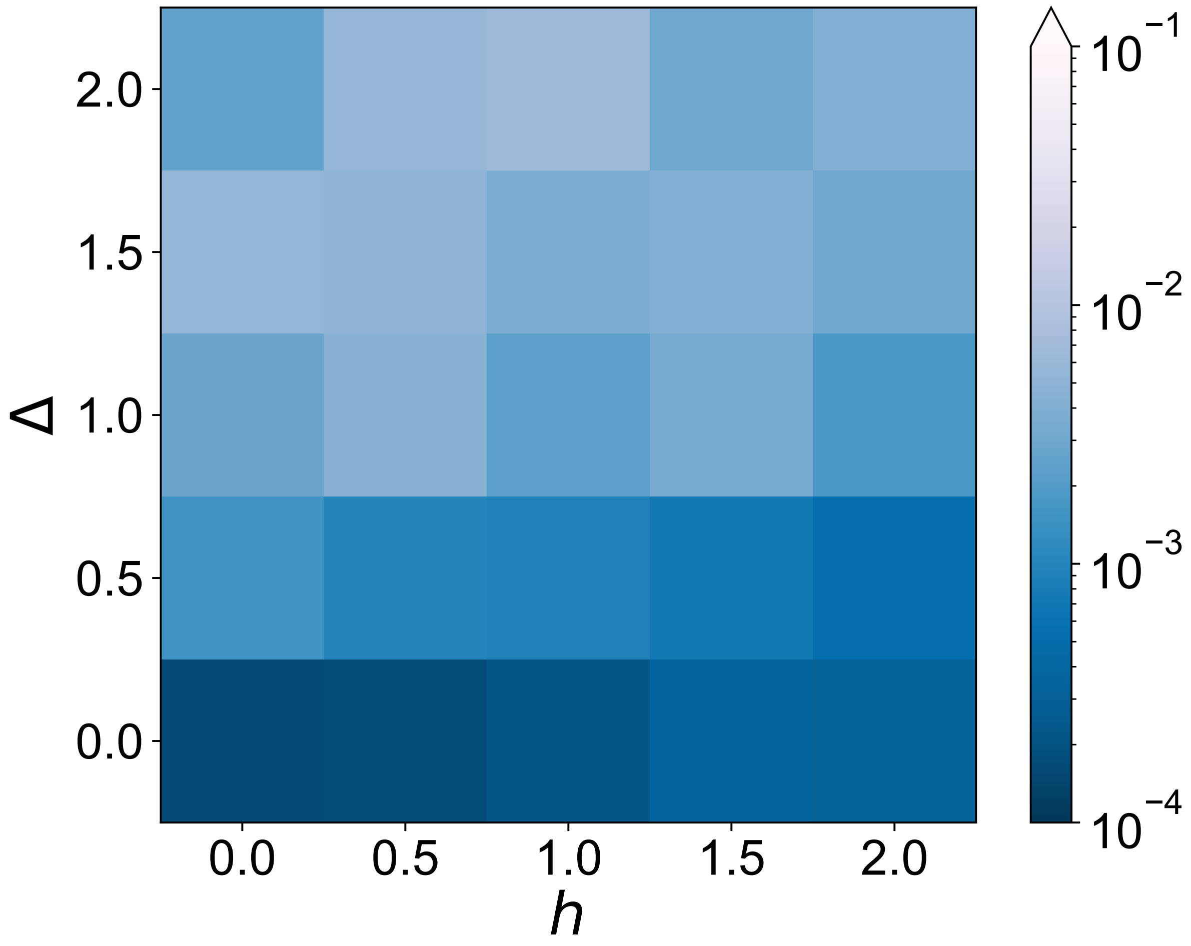


Figure 3.26. The relative error of the ground state energy, $|(E - E_{\text{ground}})/E_{\text{ground}}|$, after fine-tuning the TQS on specific parameter points for 2×10^3 iterations.

To improve on the energy estimations, we fine-tune the pre-trained TQS at selected parameters for 2×10^3 iterations. The results are plotted in Fig. 3.26. With fine-tuning, the ground state energy accuracy improved by another order of magnitude, allowing us to more accurately estimate the ground state properties on a much wider parameter range, with minimal computational cost.

This section, in full, is a reprint of the material as it appears in Physical Review B [5]. Yuan-Hang Zhang, Massimiliano Di Ventra, 2023. The dissertation author was the primary investigator and author of this paper.

Bibliography

[1] Yuan-Hang Zhang, Chesson Sipling, Erbin Qiu, Ivan K Schuller, and Massimiliano Di Ventra. Collective dynamics and long-range order in thermal neuristor networks. *Nature Communications*, 15(1):6986, 2024.

[2] Yuan-Hang Zhang and Massimiliano Di Ventra. Directed percolation and numerical stability of simulations of digital memcomputing machines. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 31:063127, 2021.

[3] Yuan-Hang Zhang and Massimiliano Di Ventra. Implementation of digital memcomputing using standard electronic components. *International Journal of Circuit Theory and Applications*, 2024.

[4] Yuan-Hang Zhang and Massimiliano Di Ventra. Efficient quantum state tomography with mode-assisted training. *Physical Review A*, 106(4):042420, 2022.

[5] Yuan-Hang Zhang and Massimiliano Di Ventra. Transformer quantum state: A multi-purpose model for quantum many-body problems. *Physical Review B*, 107(7):075147, 2023.

[6] Chesson Sipling and Massimiliano Di Ventra. Memory-induced long-range order in dynamical systems. *arXiv preprint arXiv:2405.06834*, 2024.

[7] Massimiliano Di Ventra. *MemComputing: fundamentals and applications*. Oxford University Press, 2022.

[8] Jay K.-C. Sun, Chesson Sipling, Yuan-Hang Zhang, and Massimiliano Di Ventra. Memory in neural activity: long-range order without criticality. *in preparation*, 2024.

[9] Carver Mead. Neuromorphic electronic systems. *Proceedings of the IEEE*, 78(10):1629–1636, 1990.

[10] Danijela Marković, Alice Mizrahi, Damien Querlioz, and Julie Grollier. Physics for neuromorphic computing. *Nature Reviews Physics*, 2(9):499–510, 2020.

[11] Catherine D Schuman, Shruti R Kulkarni, Maryam Parsa, J Parker Mitchell, Prasanna Date, and Bill Kay. Opportunities for neuromorphic computing algorithms and applications. *Nature Computational Science*, 2(1):10–19, 2022.

[12] Bhavin J Shastri, Alexander N Tait, Thomas Ferreira de Lima, Wolfram HP Pernice, Harish Bhaskaran, C David Wright, and Paul R Prucnal. Photonics for artificial intelligence and neuromorphic computing. *Nature Photonics*, 15(2):102–114, 2021.

[13] Julie Grollier, Damien Querlioz, KY Camsari, Karin Everschor-Sitte, Shunsuke Fukami, and Mark D Stiles. Neuromorphic spintronics. *Nature electronics*, 3(7):360–370, 2020.

[14] Javier Del Valle, Juan Gabriel Ramírez, Marcelo J Rozenberg, and Ivan K Schuller. Challenges in materials and devices for resistive-switching-based neuromorphic computing. *Journal of Applied Physics*, 124(21), 2018.

[15] Erbin Qiu, Yuan-Hang Zhang, Massimiliano Di Ventra, and Ivan K Schuller. Reconfigurable cascaded thermal neuristors for neuromorphic computing. *Advanced Materials*, 36(6):2306818, 2024.

[16] Yoeri Van De Burgt, Ewout Lubberman, Elliot J Fuller, Scott T Keene, Grégorio C Faria, Sapan Agarwal, Matthew J Marinella, A Alec Talin, and Alberto Salleo. A non-volatile organic electrochemical device as a low-voltage artificial synapse for neuromorphic computing. *Nature materials*, 16(4):414–418, 2017.

[17] Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9):1659–1671, 1997.

[18] Natalia Caporale and Yang Dan. Spike timing–dependent plasticity: a hebbian learning rule. *Annu. Rev. Neurosci.*, 31:25–46, 2008.

[19] NG Pavlidis, OK Tasoulis, Vassilis P Plagianakos, G Nikiforidis, and MN Vrahatis. Spiking neural network training using evolutionary algorithms. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 4, pages 2190–2194. IEEE, 2005.

[20] Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. Deep learning in spiking neural networks. *Neural networks*, 111:47–63, 2019.

[21] Dante R Chialvo. Emergent complex neural dynamics. *Nature physics*, 6(10):744–750, 2010.

[22] Janina Hesse and Thilo Gross. Self-organized criticality as a fundamental property of neural systems. *Frontiers in systems neuroscience*, 8:166, 2014.

[23] John M Beggs and Nicholas Timme. Being critical of criticality in the brain. *Frontiers in physiology*, 3:163, 2012.

[24] Serena Di Santo, Pablo Villegas, Raffaella Burioni, and Miguel A Muñoz. Landau–ginzburg theory of cortex dynamics: Scale-free avalanches emerge at the edge of synchronization. *Proceedings of the National Academy of Sciences*, 115(7):E1356–E1365, 2018.

[25] Jordan O'Byrne and Karim Jerbi. How critical is brain criticality? *Trends in Neurosciences*, 45(11):820–837, 2022.

[26] Viola Priesemann and Oren Shriki. Can a time varying external drive give rise to apparent criticality in neural systems? *PLoS computational biology*, 14(5):e1006081, 2018.

[27] Jens Wilting and Viola Priesemann. 25 years of criticality in neuroscience—established results, open controversies, novel concepts. *Current opinion in neurobiology*, 58:105–111, 2019.

[28] John M Beggs. Addressing skepticism of the critical brain hypothesis. *Frontiers in computational neuroscience*, 16:703865, 2022.

[29] Ludmila Brochini, Ariadne de Andrade Costa, Miguel Abadi, Antônio C Roque, Jorge Stolfi, and Osame Kinouchi. Phase transitions and self-organized criticality in networks of stochastic spiking neurons. *Scientific reports*, 6(1):35831, 2016.

[30] Silvia Scarpetta, Ilenia Apicella, Ludovico Minati, and Antonio De Candia. Hysteresis, neural avalanches, and critical behavior near a first-order transition of a spiking neural network. *Physical Review E*, 97(6):062305, 2018.

[31] Benjamin Cramer, David Stöckel, Markus Kreft, Michael Wibral, Johannes Schemmel, Karlheinz Meier, and Viola Priesemann. Control of criticality and computation in spiking neuromorphic networks with plasticity. *Nature communications*, 11(1):2853, 2020.

[32] Erbin Qiu, Pavel Salev, Felipe Torres, Henry Navarro, Robert C Dynes, and Ivan K Schuller. Stochastic transition in synchronized spiking nanooscillators. *Proceedings of the National Academy of Sciences*, 120(38):e2303765120, 2023.

[33] Gouhei Tanaka, Toshiyuki Yamane, Jean Benoit Héroux, Ryosho Nakane, Naoki Kanazawa, Seiji Takeda, Hidetoshi Numata, Daiju Nakano, and Akira Hirose. Recent advances in physical reservoir computing: A review. *Neural Networks*, 115:100–123, 2019.

[34] Wei Yi, Kenneth K Tsang, Stephen K Lam, Xiwei Bai, Jack A Crowell, and Elias A Flores. Biological plausibility and stochasticity in scalable vo2 active memristor neurons. *Nature communications*, 9(1):4661, 2018.

[35] Maksim Belyaev and Andrei Velichko. A spiking neural network based on the model of vo2–neuron. *Electronics*, 8(10):1065, 2019.

[36] Andrei Velichko, Maksim Belyaev, and Petr Boriskov. A model of an oscillatory neural network with multilevel neurons for pattern recognition and computing. *Electronics*, 8(1):75, 2019.

[37] AA Velichko, DV Ryabokon, SD Khanin, AV Sidorenko, and AG Rikkiev. Reservoir computing using high order synchronization of coupled oscillators. In *IOP Conference Series: Materials Science and Engineering*, volume 862, page 052062. IOP Publishing, 2020.

[38] Rui Yuan, Qingxi Duan, Pek Jun Tiw, Ge Li, Zhuojian Xiao, Zhaokun Jing, Ke Yang, Chang Liu, Chen Ge, Ru Huang, et al. A calibratable sensory neuron based on epitaxial vo2 for spike-based neuromorphic multisensory system. *Nature communications*, 13(1):3973, 2022.

[39] AMNF Zylbersztejn and Nevill Francis Mott. Metal-insulator transition in vanadium dioxide. *Physical Review B*, 11(11):4383, 1975.

[40] Wulfram Gerstner and Werner M Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[41] Luiz Alberto Luz de Almeida, Gurdip Singh Deep, Antonio Marcus Nogueira Lima, and Helmut Neff. Modeling of the hysteretic metal-insulator transition in a vanadium dioxide infrared detector. *Optical Engineering*, 41(10):2582–2588, 2002.

[42] Per Bak, Chao Tang, and Kurt Wiesenfeld. Self-organized criticality: An explanation of the 1/f noise. *Physical review letters*, 59(4):381, 1987.

[43] Zeev Olami, Hans Jacob S Feder, and Kim Christensen. Self-organized criticality in a continuous, nonconservative cellular automaton modeling earthquakes. *Physical review letters*, 68(8):1244, 1992.

[44] Barbara Drossel and Franz Schwabl. Self-organized critical forest-fire model. *Physical review letters*, 69(11):1629, 1992.

[45] John M Beggs and Dietmar Plenz. Neuronal avalanches in neocortical circuits. *Journal of neuroscience*, 23(35):11167–11177, 2003.

[46] L Gil and D Sornette. Landau-ginzburg theory of self-organized criticality. *Physical review letters*, 76(21):3991, 1996.

[47] Juan A Bonachela and Miguel A Munoz. Self-organization without conservation: true or just apparent scale-invariance? *Journal of Statistical Mechanics: Theory and Experiment*, 2009(09):P09009, 2009.

[48] Josué X De Carvalho and Carmen PC Prado. Self-organized criticality in the olami-feder-christensen model. *Physical review letters*, 84(17):4006, 2000.

[49] S.R.B. Bearden, F. Sheldon, and M. Di Ventra. Critical branching processes in digital memcomputing machines. *Europhysics Letters*, 127(3):30005, 2019.

[50] Michael E Fisher and Michael N Barber. Scaling theory for finite-size effects in the critical region. *Physical Review Letters*, 28(23):1516, 1972.

[51] Yann LeCun. The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*, 1998.

[52] Anna Kalogirou, Eric E Keaveny, and Demetrios T Papageorgiou. An in-depth numerical study of the two-dimensional kuramoto–sivashinsky equation. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2179):20140932, 2015.

[53] Joschka Boedecker, Oliver Obst, Joseph T Lizier, N Michael Mayer, and Minoru Asada. Information processing in echo state networks at the edge of chaos. *Theory in Biosciences*, 131:205–213, 2012.

[54] Bruno Del Papa, Viola Priesemann, and Jochen Triesch. Criticality meets learning: Criticality signatures in a self-organizing recurrent neural network. *PloS one*, 12(5):e0178683, 2017.

[55] Gisiro Maruyama. On the transition probability functions of the markov process. *Nat. Sci. Rep. Ochanomizu Univ*, 5(1):10–20, 1954.

[56] Joseph Hoshen and Raoul Kopelman. Percolation and cluster distribution. I. Cluster multiple labeling technique and critical concentration algorithm. *Physical Review B*, 14(8):3438, 1976.

[57] Gunnar Pruessner. *Self-organised criticality: theory, models and characterisation*. Cambridge University Press, 2012.

[58] David W Scott. On optimal and data-based histograms. *Biometrika*, 66(3):605–610, 1979.

[59] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[60] N. W. Ashcroft and N. D. Mermin. *Solid State Physics*. Saunders College Publishing, 1976.

[61] H. R. Wilson. Excitatory and inhibitory interactions in localized populations of model neurons. *Biophysics Journal*, 12:1–24, 1972.

[62] John G. Kirkwood. The Statistical Mechanical Theory of Transport Processes I. General Theory. *The Journal of Chemical Physics*, 14(3):180–201, 3 1946.

[63] James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pages 115–123. PMLR, 2013.

[64] Massimiliano Di Ventra and Yuriy V Pershin. The parallel approach. *Nature Physics*, 9(4):200–202, 2013.

[65] M. Di Ventra, F. L. Traversa, and I.V. Ovchinnikov. Topological field theory and computing with instantons. *Ann. Phys. (Berlin)*, 529:1700123, 2017.

[66] M. Di Ventra and I. V. Ovchinnikov. Digital memcomputing: from logic to dynamics to topology. *Annals of Physics*, 409:167935, 2019.

[67] F. L. Traversa and M. Di Ventra. Polynomial-time solution of prime factorization and np-complete problems with digital memcomputing machines. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27:023107, 2017.

[68] M. Di Ventra and F.L. Traversa. Memcomputing: Leveraging memory and physics to compute efficiently. *J. Appl. Phys.*, 123:180901, 2018.

[69] F. L. Traversa and M. Di Ventra. Memcomputing integer linear programming. *arXiv:1808.09999*, 2018.

[70] M. Di Ventra and F. L. Traversa. Absence of chaos in digital memcomputing machines with solutions. *Phys. Lett. A*, 381:3255, 2017.

[71] Sean RB Bearden, Yan Ru Pei, and Massimiliano Di Ventra. Efficient solution of boolean satisfiability problems with digital memcomputing. *Scientific Reports*, 10(1):1–8, 2020.

[72] F.L. Traversa and M. Di Ventra. Universal memcomputing machines. *IEEE Trans. Neural Netw. Learn. Syst.*, 26(11):2702, 2015.

[73] O. Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2001.

[74] T. Sauer. *Numerical Analysis*. Pearson, 2018.

[75] F. L. Traversa, P. Cicotti, F. Sheldon, and M. Di Ventra. Evidence of exponential speed-up in the solution of hard optimization problems. *Complexity*, 2018:7982851, 2018.

[76] F. Sheldon, P. Cicotti, F. L. Traversa, and M. Di Ventra. Stress-testing memcomputing on hard combinatorial optimization problems. *IEEE Trans. Neural Netw. Learn. Syst.*, 31:2222, 2020.

[77] Forrest Sheldon, Fabio L Traversa, and Massimiliano Di Ventra. Taming a nonconvex landscape with dynamical long-range order: Memcomputing ising benchmarks. *Physical Review E*, 100(5):053311, 2019.

[78] H. Manukian, F. L. Traversa, and M. Di Ventra. Accelerating deep learning with memcomputing. *Neural Networks*, 110:1, 2019.

[79] H. Manukian, Y. R. Pei, S. R. B. Bearden, and M. Di Ventra. Mode-assisted unsupervised learning of restricted boltzmann machines. *Comm. Phys.*, 3:1, 2020.

[80] Bart Selman and Henry Kautz. Domain-independent extensions to gsat: Solving large structured satisfiability problems. In *IJCAI*, volume 93, pages 290–295. Citeseer, 1993.

[81] Marc Mézard, Giorgio Parisi, and Riccardo Zecchina. Analytic and algorithmic solution of random satisfiability problems. *Science*, 297(5582):812–815, 2002.

[82] Armin Biere. Cadical, lingeling, plingelingg, treengeling and yalsat entering the sat competition 2017. *Proceedings of SAT Competition 2017: Solver and Benchmarks Descriptions*, pages 14–15, 2017.

[83] R. Rajaraman. *Solitons and Instantons: An Introduction to Solitons and Instantons in Quantum Field Theory*. North Holland, 1982.

[84] S. Coleman. *Aspects of Symmetry, Chapter 7*. Cambridge University Press, 1977.

[85] Julyan HE Cartwright and Oreste Piro. The dynamics of runge–kutta methods. *International Journal of Bifurcation and Chaos*, 2(03):427–449, 1992.

[86] Wolfgang Barthel, Alexander K Hartmann, Michele Leone, Federico Ricci-Tersenghi, Martin Weigt, and Riccardo Zecchina. Hiding solutions in random satisfiability problems: A statistical mechanics approach. *Physical review letters*, 88(18):188701, 2002.

[87] Andrew M Stuart and AR Humphries. Numerical analysis of dynamical systems. *Acta numerica*, 3(1):467–572, 1994.

[88] Danny Birmingham, Matthias Blau, Mark Rakowski, and George Thompson. Topological field theory. *Physics Reports*, 209(4-5):129–340, 1991.

[89] A. T. Fomenko. *Visual Geometry and Topology*. Springer-Verlag, 1994.

[90] I. V. Ovchinnikov. Topological field theory of dynamical systems. ii. *Chaos*, 23:013108, 2013.

[91] I. V. Ovchinnikov. Introduction to supersymmetric theory of stochastics. *Entropy*, 18:108, 2016.

[92] Malte Henkel, Haye Hinrichsen, Sven Lübeck, and Michel Pleimling. *Non-equilibrium phase transitions*, volume 1. Springer, 2008.

[93] N.G. van Kampen. *Stochastic Processes in Physics and Chemistry*. Elsevier, 1992.

[94] Hans-Karl Janssen. On the nonequilibrium phase transition in reaction-diffusion systems with an absorbing stationary state. *Zeitschrift für Physik B Condensed Matter*, 42(2):151–154, 1981.

[95] Peter Grassberger. On phase transitions in schlögl's second model. *Zeitschrift für Physik B Condensed Matter*, 47(4):365–374, 1982.

[96] GJO Jameson. The incomplete gamma functions. *The Mathematical Gazette*, 100(548):298, 2016.

[97] Robert R Schaller. Moore's law: past, present and future. *IEEE spectrum*, 34(6):52–59, 1997.

[98] John Shalf. The future of computing beyond moore's law. *Philosophical Transactions of the Royal Society A*, 378(2166):20190061, 2020.

[99] Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*. Cambridge university press, Cambridge, United Kingdom, 2010.

[100] Raisul Islam, Haitong Li, Pai-Yu Chen, Weier Wan, Hong-Yu Chen, Bin Gao, Huaqiang Wu, Shimeng Yu, Krishna Saraswat, and H-S Philip Wong. Device and materials requirements for neuromorphic computing. *Journal of Physics D: Applied Physics*, 52:113001, 2019.

[101] Daniel R Solli and Bahram Jalali. Analog optical computing. *Nature Photonics*, 9(11):704–706, 2015.

[102] A Prasanna De Silva and Seiichi Uchiyama. Molecular logic and computing. *Nature nanotechnology*, 2(7):399–410, 2007.

[103] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[104] M. Di Ventra and Y. V. Pershin. The parallel approach. *Nature Physics*, 9:200, 2013.

[105] Fabio Lorenzo Traversa and Massimiliano Di Ventra. Universal memcomputing machines. *IEEE transactions on neural networks and learning systems*, 26(11):2702–2715, 2015.

[106] Fabio L Traversa and Massimiliano Di Ventra. Polynomial-time solution of prime factorization and np-complete problems with digital memcomputing machines. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(2), 2017.

[107] Massimiliano Di Ventra and Yuriy V. Pershin. *Memristors and Memelements: Mathematics, Physics, and Fiction*. Springer, Berlin, Germany, 2023.

[108] Massimiliano Di Ventra, Fabio L Traversa, and Igor V Ovchinnikov. Topological field theory and computing with instantons. *Annalen der Physik*, 529(12):1700123, 2017.

[109] Massimiliano Di Ventra and Igor V Ovchinnikov. Digital memcomputing: from logic to dynamics to topology. *Annals of Physics*, 409:167935, 2019.

[110] Sean RB Bearden, Yan Ru Pei, and Massimiliano Di Ventra. Efficient solution of boolean satisfiability problems with digital memcomputing. *Scientific reports*, 10(1):19741, 2020.

[111] Massimiliano Di Ventra and Fabio L Traversa. Absence of chaos in digital memcomputing machines with solutions. *Physics Letters A*, 381(38):3255–3257, 2017.

[112] Massimiliano Di Ventra and Fabio L Traversa. Absence of periodic orbits in digital memcomputing machines with solutions. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(10), 2017.

[113] Fabio L Traversa, Pietro Cicotti, Forrest Sheldon, and Massimiliano Di Ventra. Evidence of exponential speed-up in the solution of hard optimization problems. *Complexity*, 2018, 2018.

[114] Fabio L Traversa and Massimiliano Di Ventra. Memcomputing integer linear programming. *arXiv preprint arXiv:1808.09999*, 2018.

[115] Forrest Sheldon, Pietro Cicotti, Fabio L Traversa, and Massimiliano Di Ventra. Stress-testing memcomputing on hard combinatorial optimization problems. *IEEE transactions on neural networks and learning systems*, 31(6):2222–2226, 2019.

[116] https://www.memcpu.com/.

[117] Tristan Sharp, Rishabh Khare, Erick Pederson, and Fabio Lorenzo Traversa. Scaling up prime factorization with self-organizing gates: A memcomputing approach. *arXiv preprint arXiv:2309.08198*, 2023.

[118] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[119] P. Gypens, J. Leliaert, M. Di Ventra, B. Van Waeyenberge, and D. Pinna. Nanomagnetic self-organizing logic gates. *Physical Review Applied*, 16:024055, 2021.

[120] Dyk Chung Nguyen, Yuan-Hang Zhang, Massimiliano Di Ventra, and Yuriy V Pershin. Hardware implementation of digital memcomputing on small-size fpgas. *arXiv preprint arXiv:2305.01061*, 2023.

[121] Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, 1971.

[122] Daniel Primosch, Yuan-Hang Zhang, and Massimiliano Di Ventra. Self-averaging of digital memcomputing machines. *Phys. Rev. E*, 108:034306, 2023.

[123] Stephan Eggersglüß and Rolf Drechsler. Robust algorithms for high quality test pattern generation using boolean satisfiability. In *2010 IEEE International Test Conference*, pages 1–10, Austin, TX, USA, 2010.

[124] Carla P Gomes, Bart Selman, Nuno Crato, and Henry Kautz. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *Journal of automated reasoning*, 24(1-2):67–100, 2000.

[125] Andrei Vladimirescu. *The SPICE book*. Wiley, New York, 1994.

[126] Ken Kundert. *The Designer's Guide to SPICE and SPECTRE®*. Springer Science & Business Media, Berlin, Germany, 2006.

[127] Yuan-Hang Zhang. Github repository: Collective dynamics and long-range order in thermal neuristor networks. https://github.com/yuanhangzhang98/collective_dynamics_neuristor, 2024.

[128] Wolfgang Barthel, Alexander K Hartmann, Michele Leone, Federico Ricci-Tersenghi, Martin Weigt, and Riccardo Zecchina. Hiding solutions in random satisfiability problems: A statistical mechanics approach. *Physical review letters*, 88(18):188701, 2002.

[129] International Electrotechnical Commission. *IEC 60384-4, Fixed capacitors for use in electronic equipment - Part 4: Sectional specification - Fixed aluminium electrolytic capacitors with solid (MnO2) and non-solid electrolyte*, 2016.

[130] Analog Devices. *AD834, 500 MHz Four-Quadrant Multiplier*, 2012. Rev. F.

[131] Texas Instruments. *Single-Supply, High-Speed, Precision Logarithmic Amplifier datasheet*, 2004. Rev. A.

[132] Ibrahim M Elfadel and John L Wyatt Jr. The" softmax" nonlinearity: Derivation using statistical mechanics and useful properties as a multiterminal analog circuit element. *Advances in neural information processing systems*, 6, 1993.

[133] Jacob Sillman. Analog implementation of the softmax function. *arXiv preprint arXiv:2305.13649*, 2023.

[134] Haik Manukian, Yan Ru Pei, Sean RB Bearden, and Massimiliano Di Ventra. Mode-assisted unsupervised learning of restricted boltzmann machines. *Communications Physics*, 3(1):1–8, 2020.

[135] Haik Manukian and Massimiliano Di Ventra. Mode-assisted joint training of deep boltzmann machines. *Scientific Reports*, 11(1), 2021.

[136] Giuseppe Carleo and Matthias Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606, 2017.

[137] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*, volume 1. MIT Press, Cambridge, 2016.

[138] Giacomo Torlai and Roger G Melko. Latent space purification via neural density operators. *Physical review letters*, 120(24):240503, 2018.

[139] Xun Gao and Lu-Ming Duan. Efficient representation of quantum many-body states with deep neural networks. *Nature communications*, 8(1):1–6, 2017.

[140] Giacomo Torlai, Guglielmo Mazzola, Juan Carrasquilla, Matthias Troyer, Roger Melko, and Giuseppe Carleo. Neural-network quantum state tomography. *Nature Physics*, 14(5):447–450, 2018.

[141] Zi Cai and Jinguo Liu. Approximating quantum many-body wave functions using artificial neural networks. *Physical Review B*, 97(3):035116, 2018.

[142] Juan Carrasquilla, Giacomo Torlai, Roger G Melko, and Leandro Aolita. Reconstructing quantum states with generative models. *Nature Machine Intelligence*, 1(3):155–161, 2019.

[143] Peter Cha, Paul Ginsparg, Felix Wu, Juan Carrasquilla, Peter L McMahon, and Eun-Ah Kim. Attention-based quantum tomography. *Machine Learning: Science and Technology*, 3(1):01LT01, 2021.

[144] Tobias Schmale, Moritz Reh, and Martin Gärttner. Scalable quantum state tomography with artificial neural networks. *arXiv preprint arXiv:2109.13776*, 2021.

[145] Shahnawaz Ahmed, Carlos Sánchez Muñoz, Franco Nori, and Anton Frisk Kockum. Quantum state tomography with conditional generative adversarial networks. *Physical Review Letters*, 127(14):140502, 2021.

[146] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071, 2008.

[147] David J Earl and Michael W Deem. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, 7(23):3910–3916, 2005.

[148] Guillaume Desjardins, Aaron Courville, Yoshua Bengio, Pascal Vincent, Olivier Delalleau, et al. Parallel tempering for training of restricted boltzmann machines. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 145–152. MIT Press Cambridge, MA, 2010.

[149] KyungHyun Cho, Tapani Raiko, and Alexander Ilin. Parallel tempering is efficient for learning restricted boltzmann machines. In *The 2010 international joint conference on neural networks (ijcnn)*, pages 1–8. IEEE, 2010.

[150] Cristian Sminchisescu, Max Welling, and Geoffrey Hinton. A mode-hopping mcmc sampler. Technical report, Technical Report CSRG-478, University of Toronto, 2003.

[151] Yongtao Guan and Stephen M Krone. Small-world mcmc and convergence to multi-modal distributions: From slow mixing to fast mixing. *The Annals of Applied Probability*, 17(1):284–304, 2007.

[152] Ugo Fano. Description of states in quantum mechanics by density matrix and operator techniques. *Reviews of modern physics*, 29(1):74, 1957.

[153] Zdenek Hradil. Quantum-state estimation. *Physical Review A*, 55(3):R1561, 1997.

[154] Hartmut Häffner, Wolfgang Hänsel, CF Roos, Jan Benhelm, Michael Chwalla, Timo Körber, UD Rapol, Mark Riebe, PO Schmidt, Christoph Becher, et al. Scalable multiparticle entanglement of trapped ions. *Nature*, 438(7068):643–646, 2005.

[155] Yihui Quek, Stanislav Fort, and Hui Khoon Ng. Adaptive quantum state tomography with neural networks. *npj Quantum Information*, 7(1):1–7, 2021.

[156] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[157] Asja Fischer and Christian Igel. An introduction to restricted boltzmann machines. In *Iberoamerican congress on pattern recognition*, pages 14–36. Springer, 2012.

[158] Daniel M Greenberger, Michael A Horne, and Anton Zeilinger. Going beyond bell's theorem. In *Bell's theorem, quantum theory and conceptions of the universe*, pages 69–72. Springer, 1989.

[159] Gary Kochenberger, Jin-Kao Hao, Fred Glover, Mark Lewis, Zhipeng Lü, Haibo Wang, and Yang Wang. The unconstrained binary quadratic programming problem: a survey. *Journal of combinatorial optimization*, 28(1):58–81, 2014.

[160] Haik Manukian, Fabio L Traversa, and Massimiliano Di Ventra. Accelerating deep learning with memcomputing. *Neural Networks*, 110:1–7, 2019.

[161] Wolfgang Dür, Guifre Vidal, and J Ignacio Cirac. Three qubits can be entangled in two inequivalent ways. *Physical Review A*, 62(6):062314, 2000.

[162] Yan Ru Pei, Haik Manukian, and Massimiliano Di Ventra. Generating weighted max-2-sat instances with frustrated loops: an rbm case study. *J. Mach. Learn. Res.*, 21:159–1, 2020.

[163] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

[164] Thomas MJ Fruchterman and Edward M Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.

[165] R Moessner and Shivaji Lal Sondhi. Ising models of quantum frustration. *Physical Review B*, 63(22):224401, 2001.

[166] Yan-Cheng Wang, Yang Qi, Shu Chen, and Zi Yang Meng. Caution on emergent continuous symmetry: a monte carlo investigation of the transverse-field frustrated ising model on the triangular and honeycomb lattices. *Physical Review B*, 96(11):115160, 2017.

[167] A Yu Kitaev. Fault-tolerant quantum computation by anyons. *Annals of Physics*, 303(1):2–30, 2003.

[168] Scott Kirkpatrick. Frustration and ground-state degeneracy in spin glasses. *Physical Review B*, 16(10):4630, 1977.

[169] Dong-Ling Deng, Xiaopeng Li, and S. Das Sarma. Machine learning topological states. *Physical Review B*, 96(19):195145, 2017.

[170] Zhih-Ahn Jia, Yuan-Hang Zhang, Yu-Chun Wu, Liang Kong, Guang-Can Guo, and Guo-Ping Guo. Efficient machine-learning representations of a surface code with boundaries, defects, domain walls, and twists. *Physical Review A*, 99(1):012307, 2019.

[171] Yuan-Hang Zhang, Zhih-Ahn Jia, Yu-Chun Wu, and Guang-Can Guo. An efficient algorithmic way to construct boltzmann machine representations for arbitrary stabilizer code. *arXiv preprint arXiv:1809.08631*, 2018.

[172] WMC Foulkes, Lubos Mitas, RJ Needs, and Guna Rajagopal. Quantum monte carlo simulations of solids. *Reviews of Modern Physics*, 73(1):33, 2001.

[173] Ulrich Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of physics*, 326(1):96–192, 2011.

[174] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[175] Kenny Choo, Giuseppe Carleo, Nicolas Regnault, and Titus Neupert. Symmetries and many-body excitations with neural-network quantum states. *Physical review letters*, 121(16):167204, 2018.

[176] Xiao Liang, Wen-Yuan Liu, Pei-Ze Lin, Guang-Can Guo, Yong-Sheng Zhang, and Lixin He. Solving frustrated quantum many-particle models with convolutional neural networks. *Physical Review B*, 98(10):104426, 2018.

[177] Kenny Choo, Titus Neupert, and Giuseppe Carleo. Two-dimensional frustrated j 1- j 2 model studied with neural network quantum states. *Physical Review B*, 100(12):125124, 2019.

[178] Mohamed Hibat-Allah, Martin Ganahl, Lauren E Hayward, Roger G Melko, and Juan Carrasquilla. Recurrent neural network wave functions. *Physical Review Research*, 2(2):023358, 2020.

[179] Or Sharir, Yoav Levine, Noam Wies, Giuseppe Carleo, and Amnon Shashua. Deep autoregressive models for the efficient variational simulation of many-body quantum systems. *Physical review letters*, 124(2):020503, 2020.

[180] Thomas D Barrett, Aleksei Malyshev, and AI Lvovsky. Autoregressive neural-network wavefunctions for ab initio quantum chemistry. *Nature Machine Intelligence*, 4(4):351–358, 2022.

[181] Di Luo, Zhuo Chen, Kaiwen Hu, Zhizhen Zhao, Vera Mikyoung Hur, and Bryan K Clark. Gauge invariant autoregressive neural networks for quantum lattice models. *arXiv preprint arXiv:2101.07243*, 2021.

[182] Dong-Ling Deng, Xiaopeng Li, and Sankar Das Sarma. Quantum entanglement in neural network states. *Physical Review X*, 7(2):021021, 2017.

[183] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[184] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational*

Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[185] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. *OpenAI blog*, 2018.

[186] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[187] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[188] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[189] Linhao Dong, Shuang Xu, and Bo Xu. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5884–5888. IEEE, 2018.

[190] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[191] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

[192] Di Luo, Zhuo Chen, Juan Carrasquilla, and Bryan K Clark. Autoregressive neural network for simulating open quantum systems via a probabilistic formulation. *Physical review letters*, 128(9):090501, 2022.

[193] Juan Carrasquilla, Di Luo, Felipe Pérez, Ashley Milsted, Bryan K Clark, Maksims Volkovs, and Leandro Aolita. Probabilistic simulation of quantum circuits using a deep-learning architecture. *Physical Review A*, 104(3):032610, 2021.

[194] Yuan-Hang Zhang, Pei-Lin Zheng, Yi Zhang, and Dong-Ling Deng. Topological quantum compiling with reinforcement learning. *Physical Review Letters*, 125(17):170501, 2020.

[195] Juan Carrasquilla and Roger G Melko. Machine learning phases of matter. *Nature Physics*, 13(5):431–434, 2017.

[196] Evert PL Van Nieuwenburg, Ye-Hua Liu, and Sebastian D Huber. Learning phase transitions by confusion. *Nature Physics*, 13(5):435–439, 2017.

[197] Yuan-Hang Zhang and Massimiliano Di Ventra. Efficient quantum state tomography with mode-assisted training. *Physical Review A*, 106:042420, 2022.

[198] HJ Schulz, TAL Ziman, and Didier Poilblanc. Magnetic order and disorder in the frustrated quantum heisenberg antiferromagnet in two dimensions. *Journal de Physique I*, 6(5):675–703, 1996.

[199] In Jae Myung. Tutorial on maximum likelihood estimation. *Journal of mathematical Psychology*, 47(1):90–100, 2003.

[200] Scott Aaronson. Shadow tomography of quantum states. *SIAM Journal on Computing*, 49(5):STOC18–368, 2019.

[201] Hsin-Yuan Huang, Richard Kueng, and John Preskill. Predicting many properties of a quantum system from very few measurements. *Nature Physics*, 16(10):1050–1057, 2020.

[202] Kurt Binder. Finite size scaling analysis of ising model block distribution functions. *Zeitschrift für Physik B Condensed Matter*, 43:119–140, 1981.

[203] Sin Yang Pang, Sithi Vinayakam Muniandy, and Mohd Zahurin Mohamed Kamali. Critical dynamics of transverse-field quantum ising model using finite-size scaling and matrix product states. *International Journal of Theoretical Physics*, 58:4139–4151, 2019.

[204] Sei Suzuki, Jun-ichi Inoue, and Bikas K Chakrabarti. *Quantum Ising phases and transitions in transverse Ising models*, volume 862. Springer, 2012.

[205] Johannes Hauschild and Frank Pollmann. Efficient numerical simulations with Tensor Networks: Tensor Network Python (TeNPy). *SciPost Phys. Lect. Notes*, page 5, 2018. Code available from https://github.com/tenpy/tenpy.

[206] Alice Zheng and Amanda Casari. *Feature engineering for machine learning: principles and techniques for data scientists*. " O'Reilly Media, Inc.", 2018.

[207] Zelun Wang and Jyh-Charn Liu. Translating math formula images to latex sequences using deep neural networks with sequence-level training, 2019.

[208] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[209] John S Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing*, pages 227–236. Springer, 1990.

[210] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[211] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[212] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[213] Sandro Sorella, Michele Casula, and Dario Rocca. Weak binding between two aromatic rings: Feeling the van der waals attraction by quantum monte carlo methods. *The Journal of chemical physics*, 127(1):014105, 2007.

[214] Miguel A Carreira-Perpinan and Geoffrey Hinton. On contrastive divergence learning. In *International workshop on artificial intelligence and statistics*, pages 33–40. PMLR, 2005.

[215] Ulrich Schollwöck, Johannes Richter, Damian JJ Farnell, and Raymond F Bishop. *Quantum magnetism*, volume 645. Springer, 2008.