Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Attractor computation in continuous-time digital networks

Tao Chen

tchen@ustc.edu.cn

University of Science and Technology of China https://orcid.org/0000-0002-6768-3630

Xu Meng

University of Science and Technology of China

Wenrui Xu

University of Science and Technology of China

Yousheng Wang

University of Science and Technology of China

Xu Wang

University of Science and Technology of China

Zhuo Liu

University of Science and Technology of China https://orcid.org/0000-0003-1508-6188

Hao Shu

University of Science and Technology of China

Guangjian Zhao

University of Science and Technology of China

Physical Sciences - Article

Keywords:

Posted Date: November 2nd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11417307/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: There is **NO** Competing Interest.

Attractor computation in continuous-time digital networks

Authors: Xu Meng[1]†, Wenrui Xu[1]†, Yousheng Wang[1], Xu Wang[1], Zhuo Liu[1], Hao Shu[1], Guangjian Zhao[1], Tao Chen[1]*

Affiliations:

[1]School of Microelectronics, University of Science and Technology of China, Hefei, Anhui, China.

†These authors contributed equally to this work.

*Corresponding author. Email: tchen@ustc.edu.cn

Summary

Attractor dynamics underlie the collective computation of neural networks[1,2]. Recurrent connections between biological neurons can give rise to stable states towards which network activity relaxes, allowing parallel operation without global coordination while preserving robustness[3,4]. Implementing physical systems with reliable attractor dynamics is therefore important both for understanding biological intelligence and for developing bio-inspired computing systems. However, parallel physical evolution introduces dynamical challenges. Concurrent evolution of artificial neurons can support rapid collective relaxation, but can also prevent the network from settling[5–7]. Here, we realise a continuous-time digital Hopfield neural network (CTD-HNN) that combines digitally programmable energy landscapes with state evolution in physical time. Across ferromagnetic and antiferromagnetic Ising models, associative memory tasks, and Max-Cut problems up to 1,000 vertices, we show that this system can be tuned to exhibit stable relaxation, persistent instability, or transient exploration of state space before convergence. By regulating feedback, activation and propagation delays, we control the intrinsic instability as a dynamical degree of freedom, thereby balancing exploration and convergence while preserving the task-defining interactions. This work combines the configurability and precision of digital hardware with continuous-time dynamics, establishing a programmable framework for attractor computation in which energy landscapes are digitally defined but physically explored.

Text

In 1982, Hopfield articulated that a recurrent network of binary threshold neurons with symmetric connections can possess an energy function (Fig. 1a, b), allowing stored patterns to be embedded as its local minima[1]. When cued with a partial or corrupted pattern, the network relaxes towards a nearby attractor (Fig. 1c), ideally recovering the corresponding stored pattern[2,8]. Such attractor dynamics support associative memory, persistent activity and decision-making in neuroscience[1,3,8–10]. Hopfield envisaged a physical implementation in which neurons would update in parallel and asynchronously, so that the network can converge rapidly and reliably towards a predetermined attractor[1,2]. More than four decades later, realising that vision has proven to be challenging.

Continuous-time analogue networks come closest to Hopfield's original vision[11,12]. Resistive interconnections can encode synaptic weights, and current amplifiers can integrate the net current and activate the output (Fig. 1d). Yet, propagation delays inherent to any physical wiring complicate the dynamics, just as early theoretical analyses predicted (Fig. 1e–g)[7]. Analogue circuits have struggled to combine programmability, precision, and scale[13–17]. By contrast, digital implementations provide programmability and precision, but often sacrifice the parallel evolution that Hopfield regarded as the source of the network's power[18,19].

The conventional digital–analogue dichotomy overlooks one important fact. Digital circuits are themselves physical entities and can therefore evolve continuously in time. Autonomous Boolean networks realise rich continuous-time dynamics in a digital circuit, but without embedding computational tasks in attractor dynamics[20–23]. A system in which both the energy landscape and the dynamical properties can be programmed for a given task remains out of reach. In this work, we introduce a continuous-time digital Hopfield neural network (CTD-HNN) with reconfigurable

topology and controllable dynamical properties (Fig. 1h). Its collective behaviour arises from the continuous evolution of the physical system rather than from a sequence of globally clocked iterations. Although concurrent evolution of neural states with intrinsic delays along interconnection paths and activation nonlinearities can give rise to inherent instability, we can leverage the programmability of digital circuits to regulate instability and exploit it as a computational resource. We demonstrate controllable dynamical regimes with ferromagnetic and antiferromagnetic Ising models up to 169 spins, a 30-node associative memory task, and Max-Cut problems up to 1,000 vertices.

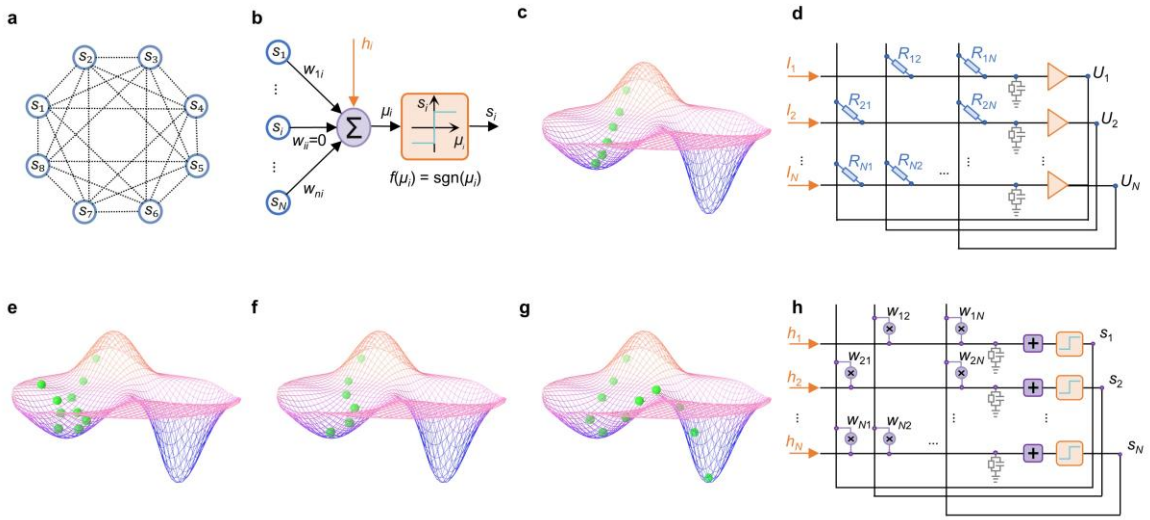


Fig. 1 | Architecture and dynamics of conventional HNN and the CTD-HNN. a, Schematic of the recurrent connections in an HNN. s_i denotes the ith neural state, and the blue lines represent mutual connections. **b,** Illustration of neural computation based on weighted sum and binary activation. μ_i is the weighted sum of the inputs, h_i the bias, f_i the activation function, w_{ij} the synaptic weight from neuron j to neuron i. **c,** Illustration of attractor dynamics in a delay-free HNN. **d,** Conceptual analogue HNN. The lumped resistor and capacitor (grey) represent RC delays. R_{ij} denotes resistive connections, U_i neural state and I_i the bias current. **e–g,** Illustrations of predicted dynamics in a time-delayed HNN showing stable relaxation (**e**), persistent instability (**f**), and explorative barrier-crossing (**g**). Spheres

indicate successive network states. **h,** Simplified architecture of the proposed CTD-HNN. All modules of the recurrent signal paths are implemented with unclocked combinational logic.

We implemented the CTD-HNN on a field-programmable gate array (FPGA) by explicitly permitting loops of combinational logic. An external host generates the synaptic weights and biases for each problem, and downloads them to registers in the FPGA (Extended Data Fig. 1a). After initialization, an integrated logic analyser samples the neural states and stores them in the block random access memory (BRAM) for readout by the host computer and for post-processing. Its unclocked recurrent data path mainly comprises multiplication-accumulation (MAC) and binary activation modules (Fig. 1h and Extended Data Fig. 1b). Unless otherwise specified, the network implements all-to-all connectivity. In principle, our approach is hardware-agnostic and can be realised on any digital logic platform.

The main-text experiments used a combinational serial MAC architecture, and a parallelised adder-tree MAC was evaluated for comparison (Extended Data Fig. 1c and Methods; Supplementary Note 1). The activation module realises a sign function. To regulate the dynamical properties, we incorporate three optional, programmable controls. An inert region reduces the neuron's susceptibility to large fluctuations of weighted sum. A state-retention mechanism enables block evolution (Extended Data Fig. 1d and Methods). A variable delay module alters propagation delay (Extended Data Fig. 1e and Methods). We investigate the dynamics in three distinct energy landscapes, i.e. ferromagnetic and antiferromagnetic Ising models, associative memory with planted attractors, and Max-Cut problems.

Attractor dynamics in ferromagnetic and antiferromagnetic Ising models

We first examined how parallel physical evolution gives rise to attractor relaxation in ferromagnetic and antiferromagnetic Ising landscapes. We implemented 9-spin (3 × 3 lattice) Ising

models on the CTD-HNN. In these tests, each spin s_i ($s_i = 1$ for spin up, and $s_i = -1$ for spin down) is encoded as a binary neural state. The spin–spin couplings w_{ij} are mapped to synaptic weights. The Ising energy is

$$H(s) = -\frac{1}{2}\sum_{i,j} w_{ij} s_i s_j - \sum_i h_i s_i, \quad (1)$$

where $w_{ij} = +1$ for nearest-neighbour pairs in the ferromagnetic model and $w_{ij} = -1$ in the antiferromagnetic model, with $w_{ij} = 0$ otherwise. The Hopfield energy for binary neurons is of the same form[1], which we evaluate in our following experiments. All local biases h_i are set to zero. We set the activation to the basic sign function without the optional control measures, and initialised the neurons with all 2^9 = 512 possible configurations. For both models, most initial configurations reach a global energy minimum within 30 ns (Fig. 2a, b). The zero-input activation biases the distribution between the two degenerate ground states (Extended Data Fig. 2; Methods). However, a few exceptions are observed, indicative of trapping in local energy minima (LMs in insets of Fig. 2a, b).

We then studied the more complex dynamics of larger networks, with model sizes up to 169 spins (13 × 13, the largest all-to-all model supported by our FPGA). Each model was run 100 times. Figure 2c and d show that the average Ising energy decreases over time, suggesting relaxation towards attractors. The success rate of reaching global minima was 100% in ferromagnetic models, but generally decreased with larger size in antiferromagnetic models (Extended Data Fig. 3a, b). To quantify the convergence property, we followed refs[24,25] and evaluated the time-to-target (TTT), which scales with size in a way similar to other energy-based hardware and software solvers (Extended Data Fig. 3c, d; Methods)[26–28]. A parallelised MAC architecture shortens the overall signal delay and reduces TTT, particularly in larger antiferromagnetic models (Extended Data Fig. 3e, f; Supplementary Note 1).

Figure 2e and f plot the proportions of stable runs at different times. All ferromagnetic trajectories stabilised within this window, although larger models took longer (Fig. 2e). In contrast, larger antiferromagnetic models exhibited slower stabilisation and a larger proportion of runs remained unstable throughout (Fig. 2f). We used principal component analysis (PCA) to project the recorded neural states onto a two-dimensional plane and visualised representative trajectories (Methods). A representative unstable run from a 64-spin antiferromagnetic model shows a non-convergent trajectory (Fig. 2g), while a stable run relaxes towards one of the degenerate global minima in an explorative way (Fig. 2h).

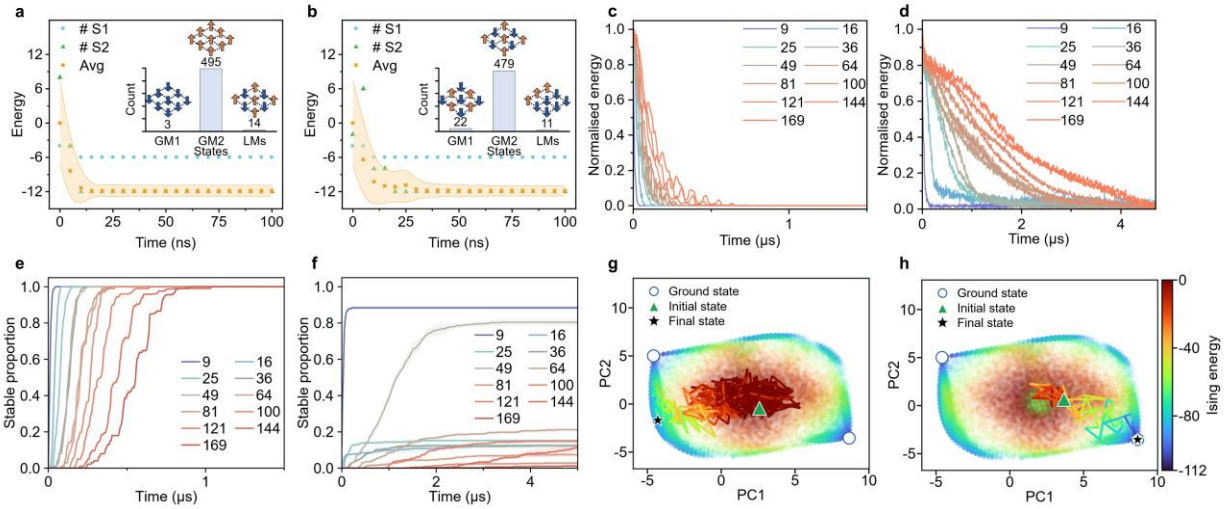


Fig. 2 | Convergence properties of the CTD-HNN. a, b, Energy versus time of the ferromagnetic (**a**) and antiferromagnetic (**b**) Ising models. The orange squares represent the average energy of 512 runs, and the yellow shaded region represents standard deviation. Two typical traces of energy evolution are plotted with green triangles and blue circles. The insets plot the counts of runs reaching the two global minima (GM1 and GM2) and all other local minima (LMs). The lattices illustrate their typical spin configuration. **c, d,** Mean energy versus time over 100 runs of ferromagnetic (**c**) and antiferromagnetic (**d**) Ising models. The energy is normalised to the range [0, 1] based on its minimum and maximum values. **e, f,** Proportion of stabilised runs over time for ferromagnetic (**e**) and antiferromagnetic (**f**) models of different sizes. A run was classified as stable once all sampled neural states remained unchanged thereafter. **g, h,** Two-dimensional PCA projection of one representative unstable trajectory (**g**) and one

representative stable trajectory (**h**). The projection contains states observed across 100 runs, together with the two ground states. Colour indicates Ising energy.

These experiments show that fully parallel physical evolution can support rapid relaxation towards attractors, but can also give rise to persistent instability as network complexity increases (Extended Data Fig. 4). We attribute the instability to embedded feedback loops[6,29]. Combined with propagation delays and nonlinearities, increasing numbers of feedback loops can yield positive feedback that destabilises a circuit[7,30–33] (Extended Data Fig. 5; Supplementary Note 2). We can qualitatively understand the behaviour of CTD-HNN within the framework of time-delayed HNN models[7]. When delays come into play in physical HNNs, the monotonic decrease of the classical Hopfield energy cannot be assumed (Fig. 1e–g; Supplementary Note 3). The delay-induced instability can contribute to barrier-crossing on the energy landscape.

Instability-assisted barrier-crossing in an associative memory task

To reveal the instability-assisted barrier crossing, we performed an associative memory task with a well-defined energy landscape (Fig. 3a). The task involved 30 neurons and four stored patterns, P1–P4, with weights calculated by the Hebbian rule[1] (Fig. 3b, c and Methods). We generated all possible two-bit perturbations of each stored pattern (Extended Data Fig. 6a), yielding 435 perturbed cues per pattern and 1,740 retrieval trials in total. These perturbed cues provided initial states for retrieval.

Figure 3d plots the confusion matrices of retrieval tests at 5, 100, 200 and 300 ns. At 300 ns, P1 and P3 were successfully retrieved from all their cues, whereas the higher-energy patterns were less accessible. P2 was retrieved from 70% of its cues, and P4 from 96%, with the remainder entering spurious attractors or remaining unstable within the observation window. The two most

frequent spurious patterns occurred with probabilities of 7.4% and 0.2%. They correspond to the inverted patterns of P4 and P3 (Extended Data Fig. 6b).

The success rate of retrieval rises more quickly for patterns with lower Hopfield energies (Fig. 3e). In general, the average energy increases briefly before falling (Fig. 3f). This transient rise is consistent with theoretical predictions for time-delayed HNNs (Supplementary Note 3)[34,35]. Depending on the influence of the delay, a trajectory can converge, remain unstable or transiently explore state space before settling into an attractor (see Fig. 1e–g)[7,32,34].

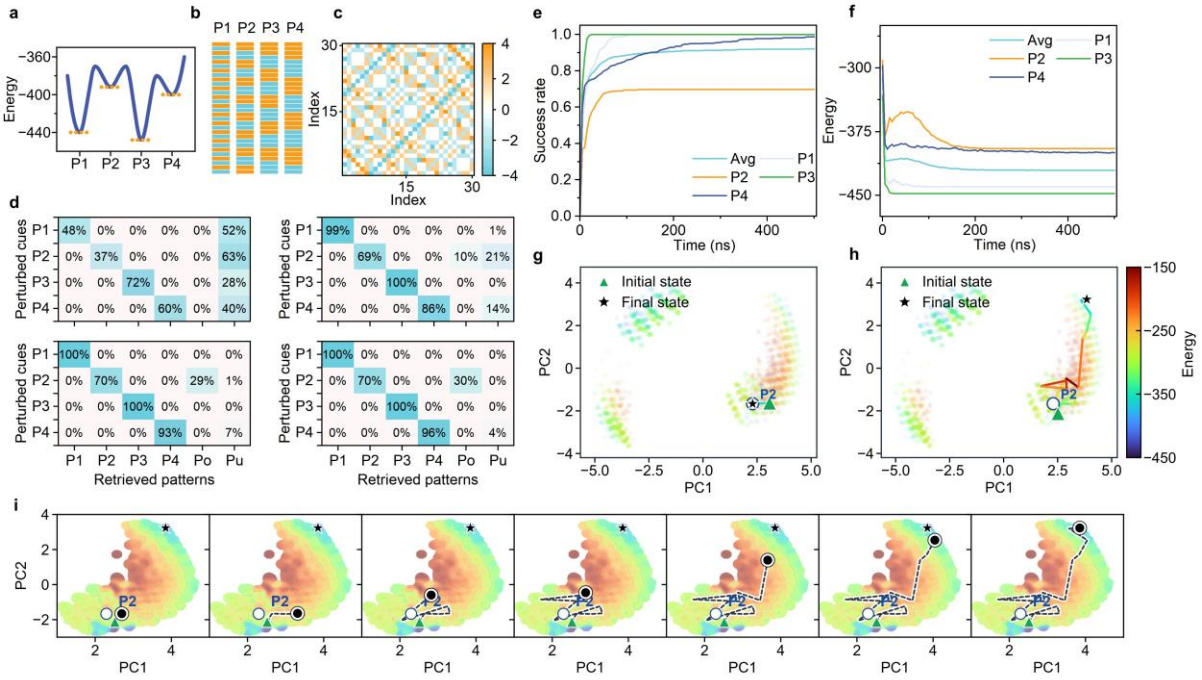


Fig. 3 | Attractor dynamics in a planted energy landscape. **a,** Schematic showing the energies of the four stored patterns. **b,** The stored patterns. Orange and blue are represented by neural state +1 and -1, respectively. **c,** The weight matrix of the associative memory task. **d,** Confusion matrices of retrieval. From left to right, top to bottom, the panels correspond to 5, 100, 200 and 300 ns, respectively. Po refers to configurations other than the stored patterns, and Pu refers to unstable dynamics. **e,** Success rate of retrieval over time. **f,** Evolution of energy. The energy averaged over all 1,740 retrieval trials is also shown for comparison. **g, h,** Two-dimensional PCA projections of two representative

trajectories converging to the P2 attractor (**g**) or a spurious attractor (**h**). In (**h**), the barrier-crossing dynamics is observed. **i,** Snapshots of the barrier-crossing trajectory in **h** (Methods). The colour corresponds to the Hopfield energy of the states, which has the same quadratic form as equation (1). The blue open circle marks P2. Black circles in **i** indicate the current state. The panels from left to right correspond to 10, 25, 40, 55, 65, 75 and 85 ns.

A PCA trajectory of successful retrieval approaches P2, with an energy of -392, straight from the initial state (Fig. 3g). Failed retrieval follows a more circuitous path. The Hopfield energy rises transiently, enabling the network to cross an energy barrier and descend into the wrong attractor, with a lower energy of -400 (Fig. 3h, i). If we apply block evolution (Methods), which restricts feedback loops, the network restores correct retrieval of all cues within 100 ns (Extended Data Fig. 6c–f; Supplementary Note 4).

These experiments confirm that instability helps the network cross energy barriers and converge to lower energy minima. We can leverage the programmability of digital hardware to steer the dynamics of CTD-HNN between exploratory and convergent regimes. Whereas associative memory tasks favour stable relaxation to nearby attractors, combinatorial optimisation problems can benefit from instability that helps the system escape local minima.

Programming exploration and relaxation in Max-Cut problems

We next used Max-Cut problems to demonstrate control over these dynamic regimes. Max-Cut is an archetypal combinatorial optimisation problem with broad applications in graph theory and social networks[36,37]. It seeks a partition of a graph's vertices that maximises the number of edges between the two sets (Fig. 4a). High-quality solutions of Max-Cut can be encoded as the minima of a Hopfield energy (Methods)[26,38]. Existing hardware solvers use mechanisms such as quantum tunnelling and thermal noise to escape local minima[26,39–43]. In the CTD-HNN, exploration instead arises from intrinsic instability.

We generated random graphs with 50% connectivity using the rudy graph generator and first studied 100-vertex instances[44]. With fully parallel evolution and the basic sign activation, the network remained non-convergent throughout the observation window. We then applied three controls. First, a block evolution scheme divides the neurons into ten groups that evolve in turn (Methods), partially interrupting feedback loops (Fig. 4b). Second, an inert activation region retains the current states when the absolute weighted sum exceeds a tunable threshold, thereby suppressing glitches and limiting state changes (Fig. 4c). Third, a variable delay alters the feedback delays without changing the logical states (Fig. 4d).

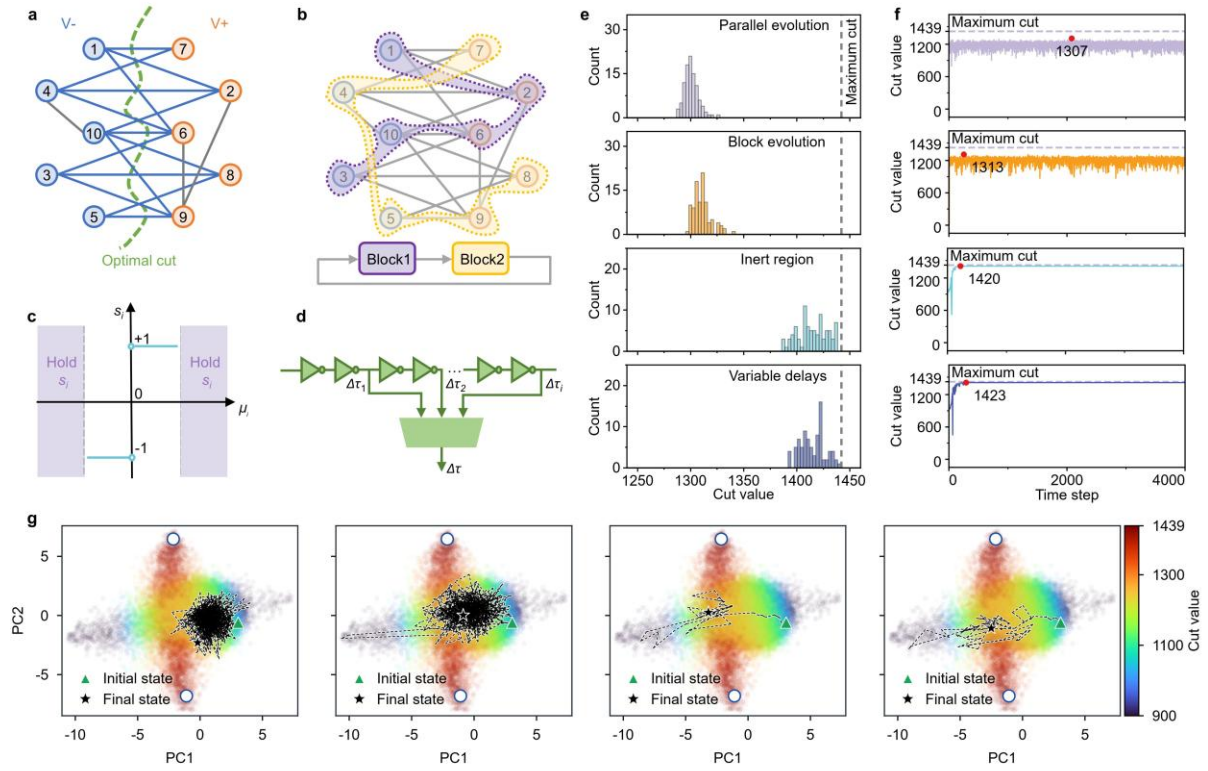


Fig. 4 | Controlling instability for solving Max-Cut problems. a, Schematic of a Max-Cut problem. The dashed green curve indicates the optimal partition, and blue and orange vertices denote the two subsets. **b,** Schematic of block evolution, in which neurons are partitioned into blocks that evolve in turn. **c,** Schematic of the tunable inert region in activation functions. **d,** Schematic of variable delays. A chain of inverters provides taps with selectable propagation delays. **e,** Distributions of the largest cut value reached per run with fully parallel evolution, block evolution, block

evolution plus inert region, and block evolution plus inert region plus variable delays. For each condition, a 100-vertex Max-Cut instance was run for 80 μs with different initial states. The dashed line denotes the known maximum cut of 1,439, obtained using the Biq Mac online solver (Methods)[45]. **f,** Cut-value evolution of representative trajectories under the four corresponding conditions. The red dots mark the largest cut value reached by each trajectory, and the dashed line denotes the known maximum cut. **g,** Two-dimensional projections obtained by PCA of the trajectories in **f**, shown from left to right for fully parallel evolution, block evolution, inert region and variable delays. Each panel shows the first 500 time steps of the run (Methods). The black dashed lines mark the trajectories. Blue open circles denote the two known maximum cuts.

For each condition, we recorded the largest cut reached in each of 100 runs (Fig. 4e, f). Fully parallel evolution yields a cut distribution comparable to random sampling. Block evolution shifts the distribution modestly towards larger cuts, although the network remains unstable. Applying block evolution together with an inert region markedly increases convergence, whereas variable delays further improve the achieved cut value.

Figure 4g shows the corresponding trajectories in a principal-component representation (Methods). Under fully parallel evolution, the network sweeps through state space and exhibits persistent instability. Bit-resolved measurement of the weighted sum revealed transient glitches in the higher bits, leading to large jumps in state space (Extended Data Figs. 7 and 8; Supplementary Note 5). Applying block evolution does not by itself yield the desired dynamics. On top of block evolution, introducing an inert activation region limits the influence of transient glitches while maintaining an explorative trajectory and notably improving convergence. However, without block evolution, applying the inert region alone fails to stabilise the dynamics. After variable delays are introduced, the trajectory becomes more exploratory, which helps the network find a better solution. The analysis of stepwise Hamming distances corroborates the observations (Extended Data Fig. 8; Supplementary Note 6). These results suggest that delays in the feedback loops play a central role in determining the inherent stability of the network.

The above experiments demonstrate the principle of regulating instability to balance exploration and convergence. We next varied the sizes of Max-Cut instances to test how broadly this principle applies. With block evolution and an inert activation region, all instances converged, although larger networks stabilised more slowly (Fig. 5a). Analysis of the graphs and dynamics showed that higher feedback-loop counts lead to slower stabilisation, as observed in the Ising models (Extended Data Fig. 9; Supplementary Note 2). For some larger graphs, the probability of reaching 99% of the known maximum increases with size (Fig. 5b), consistent with the observed instability-assisted barrier-crossing above. The TTTs of the CTD-HNN remain almost constant across sizes and are approximately two orders of magnitude shorter than those of a simulated annealing algorithm on a laptop graphics processing unit (Fig. 5c and Methods).

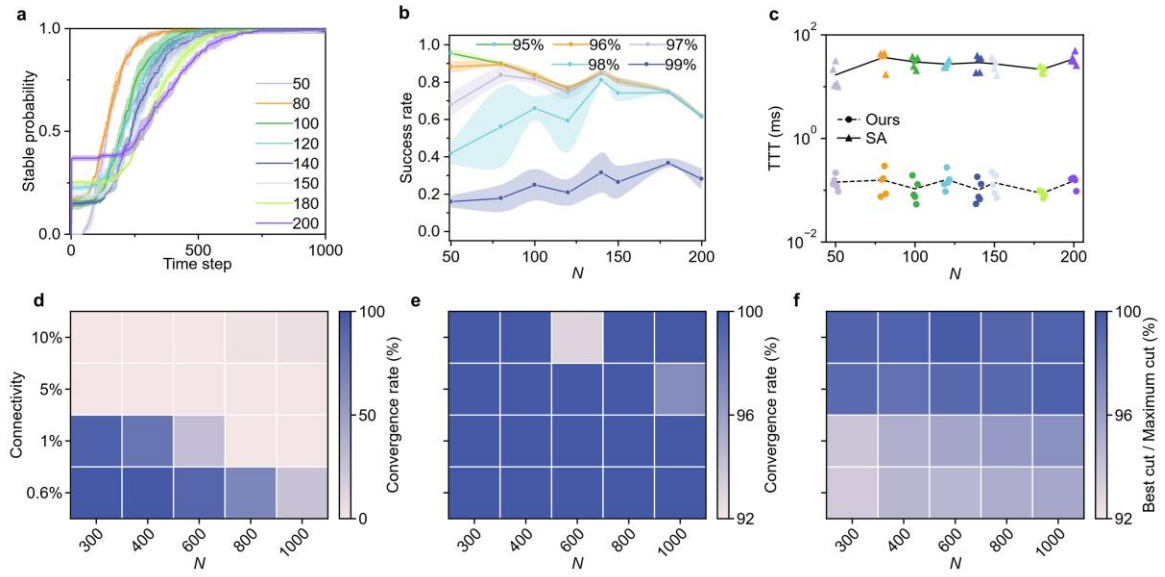


Fig. 5 | Solving Max-Cut problems of different sizes and connectivity. a, Stability over time for Max-Cut problems with 50% connectivity. A trajectory is considered stable at a given time point if all sampled neural states remain unchanged for the rest of the observation window. **b,** The success rates of reaching different ratios of the known maximum cut. For **a** and **b**, we tested five independently generated graphs for each network size, running 100 times per graph. Curves and shaded bands represent the mean and standard deviation across the five graph instances. **c,** TTT versus size. The target is defined as reaching at least 99% of the known maximum cut. Circles and triangles represent

the CTD-HNN and a simulated annealing (SA) algorithm on a graphics processing unit, respectively. **d,** Convergence rate without block evolution versus network size and graph connectivity. For each graph, the rate is obtained from 100 runs with different initial configurations. **e,** Convergence rate with block evolution. **f,** Best-cut ratio obtained with block evolution. Here, best cut refers to the largest cut value reached in 100 runs per graph.

Because many practical problems are sparse, we further tested Max-Cut problems up to 1,000 vertices with connectivity ranging from 0.6% to 10%. To conserve resources, these instances omitted zero-weight multiplication modules. Without block evolution, the convergence rate decreases with both size and connectivity, in line with the previous analysis. Figure 5d shows a boundary between convergence and non-convergence in the parameter space. The larger the network, the sparser it must be to remain stable. Block evolution substantially improves stability and convergence (Fig. 5e), while the best cut ratio relative to the known maximum increases with both size and connectivity (Fig. 5f). Together, these observations support two complementary effects: interrupting feedback loops reduces instability, whereas instability can promote transient exploration of state space for better solutions (see Fig. 5d, f).

Discussion

In our approach, neural activation and synaptic weights remain digital and reconfigurable, yet the trajectories unfold through physical signal propagation rather than globally clocked iterations. The rich continuous-time dynamics thus become accessible and controllable components of digital attractor computation, rather than non-essential details to be abstracted away. The intrinsic instability is not simply beneficial or detrimental. It emerges from the physical evolution in our system, but can be regulated while preserving the task-defining interactions and energy landscape. Its computational role depends on how it is regulated and lies in its balance, i.e. sufficient to enable exploration but constrained enough to preserve convergence. Digital programmability provides a powerful means to regulate this balance. Our approach therefore brings continuous-time attractor

computation into a precise, programmable and portable digital framework. This framework provides a practical path towards physical solvers for energy-based problems, and opens an experimental avenue to investigate the physical dynamics of more complex recurrent networks.

References

1. Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci.* **79**, 2554–2558 (1982).
2. Hopfield, J. J. Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. Natl. Acad. Sci.* **81**, 3088–3092 (1984).
3. Amit, D. J. & Brunel, N. Model of global spontaneous activity and local structured activity during delay periods in the cerebral cortex. *Cereb. Cortex* **7**, 237–252 (1997).
4. Inagaki, H. K., Fontolan, L., Romani, S. & Svoboda, K. Discrete attractor dynamics underlies persistent activity in the frontal cortex. *Nature* **566**, 212–217 (2019).
5. Haykin, S. S. *Neural Networks and Learning Machines*. (Pearson, Upper Saddle River, NJ, 2009).
6. Bruck, J. On the convergence properties of the Hopfield model. *Proc. IEEE* **78**, 1579–1585 (1990).
7. Marcus, C. M. & Westervelt, R. M. Stability of analog neural networks with delay. *Phys. Rev. A* **39**, 347–359 (1989).
8. Amit, D. J., Gutfreund, H. & Sompolinsky, H. Spin-glass models of neural networks. *Phys. Rev. A* **32**, 1007–1018 (1985).
9. Katidis, M. *et al.* Robust pattern retrieval in an optical Hopfield neural network. *Opt. Lett.* **50**, 225–228 (2025).
10. Wang, X.-J. Probabilistic decision making by slow reverberation in cortical circuits. *Neuron* **36**, 955–968 (2002).
11. Tank, D. W. & Hopfield, J. J. Simple 'neural' optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit. *IEEE Trans. Circuits Syst.* **33**, 533–541 (1986).
12. Mathews, P. O. & Hasler, J. O. A programmable and reconfigurable CMOS analog Hopfield network for NP-hard problems. *IEEE Trans. Very Large Scale Integr. VLSI Syst.* **33**, 821–830 (2025).

13. Guo, X. *et al.* Modeling and experimental demonstration of a Hopfield network analog-to-digital converter with hybrid CMOS/memristor circuits. *Front. Neurosci.* **9**, 488 (2015).
14. Douglas, R., Mahowald, M. & Mead, C. Neuromorphic analogue VLSI. *Annu. Rev. Neurosci.* **18**, 255–281 (1995).
15. Foo, S. Y., Anderson, L. R. & Takefuji, Y. Analog components for the VLSI of neural networks. *IEEE Circuits Devices Mag.* **6**, 18–26 (1990).
16. Bae, J., Oh, W., Koo, J., Yu, C. & Kim, B. CTLE-Ising: a continuous-time latch-based Ising machine featuring one-shot fully parallel spin updates and equalization of spin states. *IEEE J. Solid-State Circuits* **59**, 173–183 (2024).
17. Wu, Z. *et al.* A variation-tolerant continuous-time Ising machine with eDRAM-based spin interaction and leaked negative feedback annealing. *IEEE J. Solid-State Circuits* **60**, 1793–1804 (2025).
18. Maeda, Y. & Wakamura, M. Simultaneous perturbation learning rule for recurrent neural networks and its FPGA implementation. *IEEE Trans. Neural Netw.* **16**, 1664–1672 (2005).
19. Sousa, M. A. de A. de, Horta, E. L., Kofuji, S. T. & Del-Moral-Hernandez, E. Architecture analysis of an FPGA-based Hopfield neural network. *Adv. Artif. Neural Syst.* **2014**, 602325 (2014).
20. Rosin, D. P., Rontani, D., Haynes, N. D., Schöll, E. & Gauthier, D. J. Transient scaling and resurgence of chimera states in networks of Boolean phase oscillators. *Phys. Rev. E* **90**, 030902 (2014).
21. Rosin, D. P., Rontani, D., Gauthier, D. J. & Schöll, E. Experiments on autonomous Boolean networks. *Chaos Interdiscip. J. Nonlinear Sci.* **23**, 025102 (2013).
22. Rosin, D. P., Rontani, D., Gauthier, D. J. & Schöll, E. Control of synchronization patterns in neural-like Boolean networks. *Phys. Rev. Lett.* **110**, 104102 (2013).
23. Lohmann, J., D'Huys, O., Haynes, N. D., Schöll, E. & Gauthier, D. J. Transient dynamics and their control in time-delay autonomous Boolean ring networks. *Phys. Rev. E* **95**, 022211 (2017).
24. Zeng, Q.-G. *et al.* Performance of quantum annealing inspired algorithms for combinatorial optimization problems. *Commun. Phys.* **7**, 249 (2024).
25. Goto, H. *et al.* High-performance combinatorial optimization based on classical mechanics. *Sci. Adv.* **7**, eabe7953 (2021).
26. Mohseni, N., McMahon, P. L. & Byrnes, T. Ising machines as hardware solvers of combinatorial optimization problems. *Nat. Rev. Phys.* **4**, 363–379 (2022).

27. Leleu, T. *et al.* Scaling advantage of chaotic amplitude control for high-performance combinatorial optimization. *Commun. Phys.* **4**, 266 (2021).
28. Jiang, M., Shan, K., He, C. & Li, C. Efficient combinatorial optimization by quantum-inspired parallel annealing in analogue memristor crossbar. *Nat. Commun.* **14**, 5927 (2023).
29. Plahte, E., Mestl, T. & Omholt, S. W. Feedback loops, stability and multistationarity in dynamical systems. *J. Biol. Syst.* **3**, 409–413 (1995).
30. Prill, R. J., Iglesias, P. A. & Levchenko, A. Dynamic properties of network motifs contribute to biological network organization. *PLoS Biol.* **3**, e343 (2005).
31. Velarde, O. M., Makse, H. A. & Parra, L. C. Architecture of the brain's visual system enhances network stability and performance through layers, delays, and feedback. *PLoS Comput. Biol.* **19**, e1011078 (2023).
32. Bélair, J., Campbell, S. A. & van den Driessche, P. Frustration, stability, and delay-induced oscillations in a neural network model. *SIAM J. Appl. Math.* **56**, 245–255 (1996).
33. Wu, J., Campbell, S. A. & Bélair, J. Time-delayed neural networks: stability and oscillations. in *Encyclopedia of Computational Neuroscience* (eds Jaeger, D. & Jung, R.) 3434–3440 (Springer, New York, NY, 2022).
34. Ye, H., Michel, A. N. & Wang, K. Global stability and local stability of Hopfield neural networks with delays. *Phys. Rev. E* **50**, 4206–4213 (1994).
35. van den Driessche, P. & Zou, X. Global attractivity in delayed Hopfield neural network models. *SIAM J. Appl. Math.* **58**, 1878–1890 (1998).
36. Facchetti, G., Iacono, G. & Altafini, C. Computing global structural balance in large-scale signed social networks. *Proc. Natl. Acad. Sci.* **108**, 20953–20958 (2011).
37. Goemans, M. X. & Williamson, D. P. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM* **42**, 1115–1145 (1995).
38. Hopfield, J. J. & Tank, D. W. "Neural" computation of decisions in optimization problems. *Biol. Cybern.* **52**, 141–152 (1985).
39. Johnson, M. W. *et al.* Quantum annealing with manufactured spins. *Nature* **473**, 194–198 (2011).
40. King, A. D. *et al.* Quantum critical dynamics in a 5,000-qubit programmable spin glass. *Nature* **617**, 61–66 (2023).
41. Si, J. *et al.* Energy-efficient superparamagnetic Ising machine and its application to traveling salesman problems. *Nat. Commun.* **15**, 3457 (2024).

42. Cai, F. *et al.* Power-efficient combinatorial optimization using intrinsic noise in memristor Hopfield neural networks. *Nat. Electron.* **3**, 409–418 (2020).
43. Li, S. *et al.* An Ising machine for combinatorial optimization based on sub-nanosecond CMOS-integrated magnetoresistive random-access memory. *Nat. Electron.* (2026). https://doi.org/10.1038/s41928-026-01700-6.
44. Helmberg, C. & Rendl, F. A spectral bundle method for semidefinite programming. *SIAM J. Optim.* **10**, 673–696 (2000).
45. Rendl, F., Rinaldi, G. & Wiegele, A. Solving Max-Cut to optimality by intersecting semidefinite and polyhedral relaxations. *Math. Program.* **121**, 307–335 (2010).

Methods

Hardware platform

We implemented the ferromagnetic and antiferromagnetic Ising models, the associative memory task and Max-Cut problems with 50–200 vertices on a Xilinx Zynq-7000 system-on-chip device (XC7Z100), which integrates a field-programmable gate array (FPGA) with dual-core ARM processing units. Max-Cut problems with 300–1,000 vertices were implemented on an AXKU085 FPGA development board. A host computer generated the coupling weights for each problem and transferred them to the registers on the FPGA through the UART interface. The weights remained fixed during each run and defined the encoded energy landscape. The onboard Integrated Logic Analyzer (ILA) recorded neural states during network evolution. The sampled data were stored in BRAM and then retrieved by the host computer through the JTAG interface. The clock for control was 200 MHz for the XC7Z100 platform and 25 MHz for the AXKU085 platform.

MAC implementation

The results presented in the main text are based on the serial MAC architecture. In this implementation, a serial chain of adders computes the weighted sum, represented in two's-complement format. Because the MAC module consists of combinational logic, any change in the

neural states propagates through the feedback path and MAC module, thereby changing the corresponding weighted sum.

We also implemented a parallel MAC architecture to reduce the logic depth and propagation delay. In this implementation, two parallel adder trees compute the positive and negative contributions based on the signs of the weight and neural state, and the weighted sum is obtained by subtracting the negative contribution from the positive contribution (Extended Data Fig. 1c). Compared with the serial MAC module, this adder tree reduces the logic depth and accelerates convergence.

Implementation of activation and block evolution

The activation module set the output to −1 when the weighted sum was zero for the ferromagnetic Ising models and Max-Cut problems, and to +1 for the antiferromagnetic Ising and associative memory experiments in the main text. The neural states −1 and +1 are represented by logic high and low, respectively.

The inert region imposed positive and negative bounds on the weighted sum. Neurons retained their previous states when the weighted sum exceeded the positive bound or fell below the negative bound. Otherwise, they followed the sign-based activation rule. For Max-Cut experiments with the inert region, the positive threshold was initialised to a preset value and increased by 1 at specified time intervals until reaching a preset upper limit. For the experiments in Fig. 4, the threshold was increased from 4 by 1 in every 20 control-clock cycles, with a preset upper limit of 21. For the experiments in Fig. 5a–c and d–f, the increment interval was 1 in every 6 control-clock cycles. The upper limit was selected through a parameter sweep to minimise the time-to-target (TTT) at each network size. The negative bound was generated as its two's-complement counterpart.

For Max-Cut experiments using block evolution, neurons were partitioned into ten blocks and enabled sequentially. Neural state evolution within the enabled block remained continuous-time,

while the others retained their states. Each block-enable window lasted 100 ns for networks with 50–200 vertices and 240 ns for networks with 300–1,000 vertices.

Clocked control circuitry scheduled threshold changes and block-enable windows, while the MAC and activation modules operated continuously.

Programmable feedback delay

In addition to intrinsic gate and routing delays, we inserted programmable feedback delays after the activation module. In the FPGA implementation, each programmable delay unit was constructed from a chain of cascaded inverters. The output after a selected even number of inverter stages was tapped by the delay-control signal. In this way, we can adjust the propagation time of neural states without changing their value. The delay-control signals were configurable during operation, enabling perturbation of the trajectory of neural states and promoting exploration of the state space.

Time-to-target analysis

For a given operation period, a run was counted as successful if it reached a target fraction α of the ground-state energy for Ising problems or a target ratio of the maximum cut value for Max-Cut problems. The success probability, denoted as $P_s(t, \alpha)$, was calculated as the fraction of successful runs within the operation period t.

The TTT estimates the expected time required to reach the target solution with 99% probability across independent repetitions,

$$\mathrm{TTT}(t, \alpha) = t \times \frac{ln(1-0.99)}{ln(1-P_s(t,\alpha))}.$$

Because the success probability depends on the operation period, we reported the optimal TTT over the operation periods,

$$\mathrm{TTT}_{\mathrm{opt}}(\alpha) = min_t[\mathrm{TTT}(t, \alpha)].$$

For $P_s = 0$, $\mathrm{TTT}_{\mathrm{opt}}(\alpha)$ was assigned an infinite value. For $P_s = 1$, $\mathrm{TTT}_{\mathrm{opt}}(\alpha)$ was defined as the shortest operation period by which all runs had reached the target.

PCA projection and visualisation of trajectories

Background plot. We sampled the neural-state vectors across all the runs with the same Hopfield energy (the same weight matrix), for PCA. After obtaining the two principal components, we projected the available state vectors onto the two-dimensional plane and coloured the points according to their corresponding energy/cut value. When multiple distinct states occupy the same point, the colour corresponds to their average energy. The initial/final states, stored patterns or optimal cuts were projected onto the same principal components for reference.

Antiferromagnetic trajectories. For the 64-spin antiferromagnetic Ising model, the background dataset comprised all state vectors recorded across 100 runs. Trajectory segments were coloured according to the average energy of their two endpoint states.

Associative memory trajectories. For the 30-neuron associative memory task, the backgrounds in Fig. 3g, h used all state vectors recorded across 1,740 runs for P1–P4 for PCA analysis. Extended Data Fig. 6f was obtained in the same way. The lines marking the trajectories were coloured by the average energy of their two endpoint states.

Max-Cut trajectories. For Fig. 4g, the background comprised the first 500 state vectors of 400 runs, 100 runs for each condition, namely fully parallel evolution, block evolution, inert activation region and variable delays. The principal components were obtained from the 200,000 vectors.

Associative memory task

For the associative memory task, each stored pattern is represented as a binary state vector $s^{(\theta)} = \{s_1^{(\theta)}, s_2^{(\theta)}, \dots, s_N^{(\theta)}\}$, where $s_i^{(\theta)} \in \{1, -1\}$ represents the ith pixel in the θth stored pattern. The

coupling weight w_{ij} between neuron i and neuron j was computed using the standard Hebbian learning rule (outer product rule), $w_{ij} = \sum_{\theta=1}^{4} s_i^{(\theta)} s_j^{(\theta)}, i \neq j$. Self-connections were set to zero ($w_{ii} = 0$).

Max-Cut encoding

For a simple undirected, unweighted graph $G = (V, E)$ with vertex set $V = \{1, \ldots, N\}$ and edge set E, the coupling a_{ij} equals 1 if vertices i and j are connected and 0 otherwise, with $a_{ii} = 0$ and $a_{ij} = a_{ji}$. Each vertex was represented by a binary neural state $s_i \in \{-1, +1\}$, each state identifying a subset of the partitioned vertices. The cut value is therefore

$$C(\boldsymbol{s}) = \frac{1}{2} \sum_{i<j} a_{ij} \left(1 - s_i s_j\right).$$

Because each edge is counted once, $\sum_{i<j} a_{ij} = \mathcal{E}$, where $\mathcal{E}$ is the number of edges. Hence,

$$C(\boldsymbol{s}) = \frac{\mathcal{E}}{2} - \frac{1}{2} \sum_{i<j} a_{ij}\, s_i s_j.$$

To favour different neural states for connected vertices, we implemented antiferromagnetic interactions with effective couplings $-a_{ij}$. The corresponding Hopfield energy is

$$H_{MC}(\boldsymbol{s}) = -\frac{1}{2} \sum_{i,j} (-a_{ij}) s_i s_j = \sum_{i<j} a_{ij} s_i s_j = \mathcal{E} - 2C(\boldsymbol{s}).$$

Maximising the cut value is equivalent to minimising $H_{MC}(\boldsymbol{s})$.

Known maximum cut values

Known maximum cut values were computed using the exact solution method of the Biq Mac online solver (https://biqmac.aau.at/). This method combines semidefinite relaxations with branch-and-bound. Each instance was subject to a three-hour time limit. When optimality was certified within

this limit, the optimal cut value was used. Otherwise, we used the best cut value returned at termination. These values served as references for cut-value normalisation and target definition.

GPU-based simulated annealing

We optimised the Max-Cut problems with a simulated annealing (SA) algorithm, running on an NVIDIA GeForce RTX 3050 Laptop GPU with 4 GB of memory. For each graph, 100 independent trajectories were executed in parallel, with one GPU thread assigned to each trajectory. Each trajectory started from random initial states and performed 20,000 sequential flip attempts, selecting a vertex at random with replacement. A proposed flip with cut-value change ΔC was accepted if $\Delta C \geq 0$. Otherwise, it was accepted with probability $\exp(\Delta C/T)$. The temperature T decreased geometrically from 10 to 0.01. For comparison with CTD-HNN, we set the target to 99% of the known maximum cut. Kernel execution time was measured after GPU warm-up, excluding compilation and host–device data transfers.

Data availability

The data for Figs. 2–5 are available at GitHub (https://github.com/stunnerx/CTDHNN-data-code). The raw experimental data are available from the corresponding author upon reasonable request.

Code availability

A minimum set of Verilog source code and Vivado project files, together with the scripts for data analysis and plotting Figs. 2–5, are available at GitHub (https://github.com/stunnerx/CTDHNN-data-code).

Acknowledgments

We thank Y. Geng and Y. Islam for comments, and thank F. Wu and S. Long for support.

Author contributions

T.C. conceived and supervised the project. X.M., W.X., Y.W., X.W. implemented the network and performed the experiments. X.M., W.X., X.W., T.C. analysed the data. T.C., X.M., W.X. wrote the manuscript with input from Z.L., H.S., G. Z.. All authors contributed to revisions.

Funding

We acknowledge financial support from National Key Research and Development Program of China (2024YFA1208804), the University of Science and Technology of China and the Chinese Academy of Sciences.

Competing interests

The authors declare that they have no competing interests.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- SIfinal.docx
- ExtendedDatafinal.docx