

Boolean hypercubes as a neural substrate

HypercubeAI — topology-native intelligence on Q_n

David Charles Liptak

Independent

dliptak001@gmail.com

4 September 2026

Copyright 2026 David Charles Liptak. Licensed under the Apache License, Version 2.0.

Abstract

Classical nets sit on Euclidean grids or on stored random graphs. This note puts the same architectures on the Boolean hypercube Q_n so that locality, addressing, and symmetry are one object. The geometry is stated as fact. The seven HypercubeAI members are different readings of that geometry. Task results live in the implementations of Appendix A.

Contents

1	The object Q_n	2
2	The bare cube	3
2.1	Vertex-transitive symmetry	3
2.2	Hamming geometry	3
2.3	Bitwise addressing	3
3	Change of origin	4
4	Multi-scale soft modularity	5
5	Pooling and packing	6
6	Architectures on the same substrate	6
7	Related work	8
8	Limitations and non-claims	8
9	What remains open	8
A	Implementations	9

1 The object Q_n

Q_n is a graph with 2^n vertices, one for each binary string of length n . Two vertices are neighbors when their strings differ in a single bit.

The same object has two other names. As a group, Q_n is the Cayley graph of $(\mathbb{Z}/2\mathbb{Z})^n$ generated by the standard basis $e_0, \dots, e_{n-1}$. As a poset, it is the covering graph of the Boolean lattice B_n , ranked by Hamming weight from a chosen origin. Every vertex has degree n . There are $n 2^{n-1}$ edges. The automorphism group is

$$\text{Aut}(Q_n) \cong (\mathbb{Z}/2\mathbb{Z})^n \rtimes S_n$$

of order $2^n n!$: XOR-translate by any mask, or permute the n axes. Weight sharing in HypercubeCNN uses only the translation factor $(\mathbb{Z}/2\mathbb{Z})^n$. The S_n factor relabels bit axes; the architectures below do not share weights across that action.

A network on Q_n is a field: a value, or a short vector of values, at each vertex,

$$x: V(Q_n) \rightarrow \mathbb{R}^c.$$

Input, hidden state, and output can occupy the same 2^n addresses. The legal wires are the bit-flip edges.

Bit order. Throughout, bit 0 is the rightmost bit, matching the implementations: $e_i = 1 \ll i$ and the neighbor of v along axis i is $v \oplus (1 \ll i)$.

Worked neighbor, $n = 3$. Vertex 101 has neighbors 100, 111, and 001 (flip bits 0, 1, and 2). The Hamming-1 gather at 101, center first, is

$$(x(101), x(100), x(111), x(001)).$$

The same rule at 000 yields

$$(x(000), x(001), x(010), x(100)).$$

Nothing about the rule changed but the origin.

Q_n is bipartite by Hamming-weight parity. The map

$$\alpha(v) = v \oplus (2^n - 1)$$

sends each vertex to its unique antipode, at distance n . The involution α partitions $V(Q_n)$ into 2^{n-1} pairs $\{v, \alpha(v)\}$. For $n > 1$ the pair $\{v, \alpha(v)\}$ is *not* an edge. Diameter is n : the graph distance between any two of the 2^n sites is at most n bit-flips.

Spectrum. The adjacency eigenvalues of Q_n are $n - 2k$ with multiplicity $\binom{n}{k}$, $k = 0, \dots, n$. The eigenfunctions are the Walsh characters of $(\mathbb{Z}/2\mathbb{Z})^n$. Convolution by a translation-invariant kernel is pointwise multiplication in that basis.

The first three properties below belong to this bare cube — addresses and legal wires, nothing on the edges yet. Soft modularity appears once the edges carry weights.

2 The bare cube

2.1 Vertex-transitive symmetry

There is an automorphism sending any vertex to any other. Every site has the same degree, the same neighborhood type, and the same role. Local rules therefore mean the same thing at every address. Random graphs bake in hubs and dead ends; grids bake in corners and boundaries. Q_n has neither.

Weight sharing *can* be exact under the $(\mathbb{Z}/2\mathbb{Z})^n$ action: a shared kernel written at v is the same kernel at $v \oplus g$. That is a choice. A CNN shares. An LCN trains a private table at each vertex. What always copies is the rule. Write a gather, a recurrent step, or an attention support once; vertex-transitivity instantiates it at every address, with no boundary case and no privileged site. A private weight field does not copy.

2.2 Hamming geometry

Distance is bit count, not Euclidean. The closed ball of radius r about v is

$$B_r(v) = \{u : \text{wt}(u \oplus v) \leq r\},$$

of size $\sum_{k=0}^r \binom{n}{k}$. Neighborhoods used by convolutions, lookback, and sparse attention are combinatorial objects, not pixel patches. Degree is $\log_2 N$: each of 2^n sites has exactly n neighbors.

Lemma 1 (Diameter). $\text{diam}(Q_n) = n$. A $\sqrt{N} \times \sqrt{N}$ four-neighbor grid on the same $N = 2^n$ vertices has diameter $\Theta(2^{n/2})$. Graph distance on the cube is therefore at most n bit-flips, independent of the 2^n population.

That bound is path length. It is not a mixing time for leaky or contractive dynamics.

Among subsets of size $\sum_{k=0}^r \binom{n}{k}$, a Hamming ball minimizes vertex boundary [4]. For other cardinalities the minimizers are initial segments of simplicial order. The Hamming-1 stencil is compact in that sense; compactness does not by itself select an architecture.

2.3 Bitwise addressing

The neighbor of vertex v along axis i is $v \oplus e_i$. Walks, kernels, delay lines, and Hamming-ball masks are arithmetic on the index. There is no CSR matrix and no neighbor table that grows with $|E| = n 2^{n-1}$. Given n and a seed for any frozen weights, the graph reconstructs. Scaling $n \mapsto n + 1$ doubles N and adds one generator; no remesh, no new pointer table. Memory holds fields and weights, not an edge list.

3 Change of origin

Write $\tau_g(v) = v \oplus g$. Translation by a fixed mask moves every covering edge and every Hamming ball rigidly. The diagrams below are Q_n ranked by distance from a chosen origin: columns are $\text{wt}(v \oplus \text{origin})$. Covering edges remain single-bit flips.

Each of B_3 and B_4 is drawn from $00\dots 0$. B_3 is drawn a second time from 101, which is the 000 view with labels XOR-relabeled by 101. Vertex-transitivity is that statement, not an abstract group fact sitting off to the side.

Once the change of origin is visible, three consequences follow:

- The wiring copies. A Hamming-1 gather, a ball $B_r(v)$, a delay-line tap, or a lookback window written at one vertex is the same geometric object at every vertex, at the cost of one XOR.
- A shared kernel copies. Under weight sharing the taps at v are the taps at $v \oplus g$. That is the CNN case, and the ESN / WTF / Etalon update *rule*.
- A private weight field does not copy. An LCN table at v and a stored Hopfield pattern are data on the cube, not part of the cube. τ_g moves addresses; it does not duplicate trained numbers.

Hasse diagram of B_3 (Q_3) from origin 000
columns = Hamming distance from 000; covering edges = single-bit flips

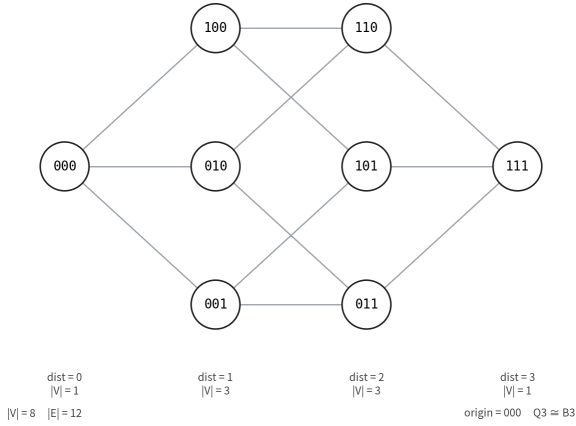


Figure 1: Q_3 ranked by distance from 000. Rank sizes 1, 3, 3, 1. $|V| = 8$, $|E| = 12$.

Hasse diagram of B_3 (Q_3) from origin 101
affine translate of the 000 view by XOR 101; columns = Hamming distance from 101

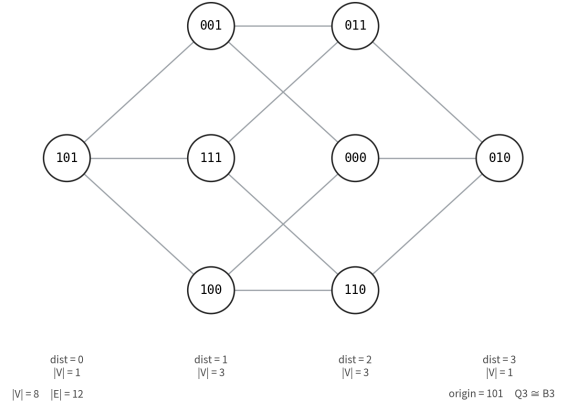


Figure 2: The same Q_3 , ranked by distance from 101 = $000 \oplus 101$. Every wire is still a single-bit flip.

Odd n has two central ranks, each of size $\binom{n}{\lfloor n/2 \rfloor}$. Even n has a single middle rank of size $\binom{n}{n/2}$ (the Sperner level). That is the only reason to draw Q_4 after Q_3 : the middle of the even lattice is one crowded column of $\binom{4}{2} = 6$, not two columns of 3. Change of origin is already visible on the Q_3 pair.

Hasse diagram of B4 (Q4) from origin 0000
columns = Hamming distance from 0000; covering edges = single-bit flips

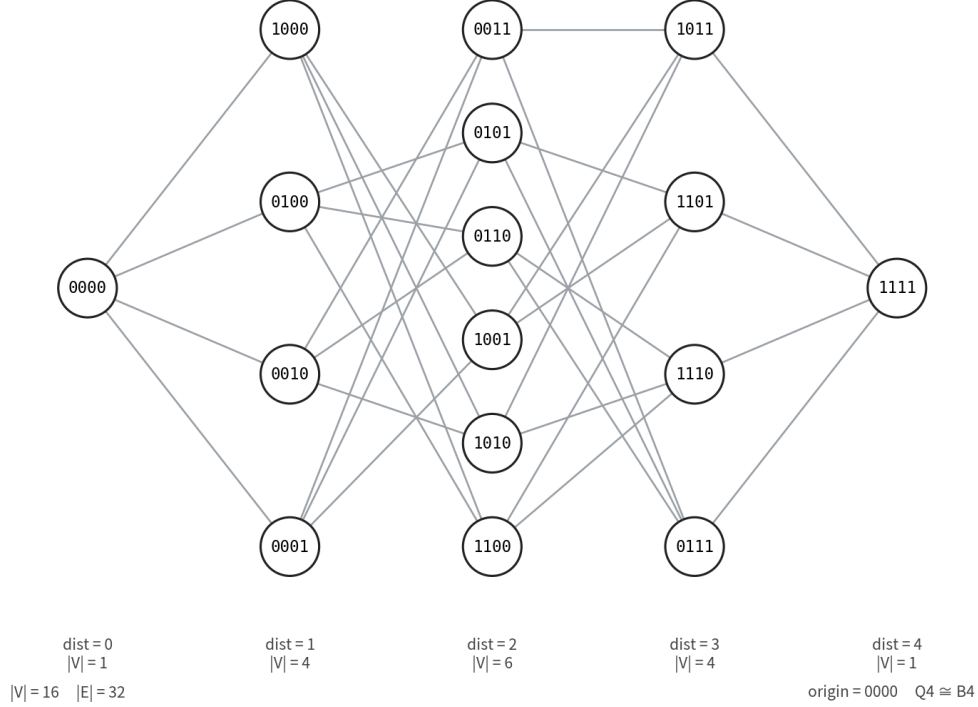


Figure 3: Q_4 ranked by distance from 0000. Rank sizes 1, 4, 6, 4, 1. $|V| = 16$, $|E| = 32$. The middle rank is the unique largest level.

4 Multi-scale soft modularity

The bare cube is uniform. Weights break that uniformity without touching the edge set. Place random, frozen, or trained magnitudes on the n bit-flip edges. Implementations store a number on each directed edge: $W(v, v \oplus e_i)$ need not equal $W(v \oplus e_i, v)$. A near-zero magnitude is a soft cut. The wire remains legal; it carries almost no signal.

Sweep a cutoff on those magnitudes and the surviving directed graph is a nested filtration of one weighted cube. A high cutoff leaves tight clusters; lowering it merges those clusters into larger regions; cutoff zero restores every legal wire. The subgraphs nest, so each vertex has a component at every scale of one weight field. The hierarchy is a reading of the weights, not a second graph.

When the magnitudes are i.i.d., the undirected version of that cutoff is bond percolation on Q_n , with critical probability

$$p_c = \frac{1}{n} + \frac{1}{n^2} + O(n^{-3})$$

[1, 7]; the coefficient of n^{-3} is $7/2$. A trained or frozen field is not an i.i.d. sample. For those fields the filtration is deterministic.

A hard cut cannot produce the nested family. Drop one generator everywhere and Q_n splits into

two disjoint copies of Q_{n-1} with nothing in between. Soft cuts keep every axis and let magnitude decide which wires conduct.

The same thresholding is available on a grid or a random graph. Grid modules inherit corners and boundaries. Random-graph modules have no Hamming address and no cheap translate. On Q_n a module is a set of addresses; its image under τ_g is that set at a new origin. The geometry of possible modules translates even when a particular trained weight configuration does not.

5 Pooling and packing

Two maps used by the family are not covering edges of Q_n .

Antipodal pooling. For $v = 0, \dots, 2^{n-1} - 1$, write

$$\text{out}(v) = \max\{x(v), x(v \oplus (2^n - 1))\}$$

or the mean of the same pair. The output index v is a vertex of Q_{n-1} (the low $n-1$ bits). Population halves; dimension drops by one; the image is an exact cube. This is geometric multi-scale structure: a hierarchy of subcubes. Soft modularity is the other multi-scale story — weights, edge set fixed. Pooling changes the graph; magnitudes do not.

Packing. The cube accepts only $N = 2^n$ sites. Domain data that is not already a field on V must be packed:

$$P: \text{data} \rightarrow \mathbb{R}^{2^n}.$$

A 28×28 image into dimension 10 ($N = 1024$), a length-2048 spectrum onto dimension 11, a scalar stream broadcast onto selected vertices — those are packing choices, and they are part of the model. Row-major placement of 28×28 pixels into 1024 addresses does *not* send pixel neighbors to Hamming-1 neighbors. Hamming geometry begins after P . The cube does not invent a metric for the source domain.

6 Architectures on the same substrate

On a grid, locality is Euclidean and boundary-laden. On a random graph, locality is a stored sparse matrix with no global isometry. On Q_n , locality, addressing, and symmetry are the same object. Each member below is a classical architecture rewritten on that object — or, for Etalon, a walk that exists because the antipode does.

Two bets: frozen geometry plus a thin trained head, or trained geometry on the same addresses.

Product	Ancestry	Time	What trains	Primitive
HypercubeESN	echo-state	stream	HCNN readout	gather + delay line
HypercubeWTF	ESN on a static field	synthetic orbit	HCNN readout	gather; drive $x(v \oplus t)$
HypercubeEtalon	not a reservoir	one transit	HCNN readout	a geodesic $r \rightarrow \alpha(r) \rightarrow r$
HypercubeCascade	Etalon then WTF	transit + orbit	HCNN readout	both, one address space
HypercubeCNN	conv net	static field	shared taps	Hamming-1 kernel; antipodal pool
HypercubeLCN	locally connected	static, depth	private taps	gather + lookback
HypercubeHopfield	modern Hopfield	retrieval	patterns stored	softmax inside $B_r(v)$

HypercubeCNN. The shared Hamming-1 stencil, self tap plus one weight per axis, $K = n + 1$:

$$y(v) = w_{\bullet} x(v) + \sum_{i=0}^{n-1} w_i x(v \oplus e_i).$$

The same $(w_{\bullet}, w_0, \dots, w_{n-1})$ is used at every v . Optional antipodal pooling as above.

HypercubeESN. A leaky-integrator reservoir on the same gather. Each vertex taps M delay slices of its n neighbors. The recurrent block is frozen and rescaled to a target spectral radius. The readout is HypercubeCNN.

HypercubeWTF. The same frozen reservoir, aimed at a field with no clock. Pass $t = 0, \dots, T - 1$ drives vertex v with the packed field at $v \oplus t$. A frozen initial condition is reloaded every sample. The trained piece is the HCNN on the end state.

HypercubeEtalon. Not a reservoir. For each start r , a causal neighbor-tanh walk follows one geodesic $r \rightarrow \alpha(r)$ and returns. There are $n!$ geodesics from r to $\alpha(r)$; the implementation selects one. Path length on each leg is n . The trained piece is a thin HCNN on the resulting field. “Interferometer” is a metaphor for that walk.

HypercubeCascade. One etalon transit, then one WTF orbit, then a small HypercubeCNN. All three stages share V . Only the readout trains.

HypercubeLCN. The same gather, private weights at every vertex and depth. Lookback of width ℓ is a skip along the depth axis. Input, hidden field, and output occupy the same vertices. Private tables do not copy under τ_g ; the gather does.

HypercubeHopfield. Patterns are stored explicitly. Each vertex performs softmax attention over the restriction of those patterns to $B_r(v)$, not Hebbian all-to-all and not a two-dimensional window. Cost scales with ball size, not with 2^n . Capacity of the ball-restricted net is not a theorem of this note.

7 Related work

Harper’s vertex-isoperimetric theorem on Q_n is due to Harper [4]. Bond percolation on Q_n and its critical window are Borgs et al. [1]; the expansion $p_c = n^{-1} + n^{-2} + \frac{7}{2}n^{-3} + O(n^{-4})$ is van der Hofstad and Slade [7]. Modern Hopfield retrieval as softmax attention over stored patterns is Ramsauer et al. [6]; the construction here restricts that attention to a Hamming ball. Reservoirs whose graph is an explicit stored object include GraphESN [3]. Group-equivariant CNNs in the sense of Cohen and Welling [2] share under a group action; HypercubeCNN shares under translations on $(\mathbb{Z}/2\mathbb{Z})^n$, not under the full hyperoctahedral group. Katori et al. [5] use the unit cube $[0, 1]^N$ as the space in which a continuous state moves. This family puts activations on the vertices of the graph Q_n . Same word, different object.

8 Limitations and non-claims

N is a power of two. There is no Q_n on 1000 sites. Growing the net means $n \mapsto n + 1$: population doubles and one generator appears.

Packing is a design choice. A field on V is native. An image, a spectrum, or a token sequence is not. A poor packing is not a defect of Q_n .

Cost follows 2^n . A Hamming-1 gather at every vertex is $\Theta(n 2^n)$ arithmetic and $\Theta(2^n)$ storage for one field. Ball- r attention is $\Theta(2^n \sum_{k \leq r} \binom{n}{k})$. The practical window in the current codes is roughly $n = 5 \dots 16$ (32 to 65,536 vertices). The algebra does not stop at 16; the memory does.

Soft modularity is a reading of one weight field by cutoff. It is not a theorem that those partitions improve a task.

This note does not claim that Q_n is the only good neural graph. It does not prove capacity for ball-restricted Hopfield retrieval. Empirical numbers live in the repositories, with the data and the protocol that produced them.

9 What remains open

The Boolean hypercube is not a new object. What is thin is the public record of using it as the domain of a field that a network reads and writes — same addresses, several architectures, measurements that travel with the protocol that produced them. The seven implementations exist so that record can be built in the open.

Six questions the geometry makes well-posed, and that this note does not answer:

Packing. Packing is how data that does not already live on the cube gets an address. A map P sends each source point to a vertex of Q_n . After that map, nearby means Hamming-near. If two pixels, bins, or tokens that the source treated as neighbors land far apart on the cube, a Hamming-1 gather will not see that they were neighbors. Row-major placement of a 28×28 image into 1024 addresses does that: adjacent pixels need not be Hamming-1. A layout that respects the source’s own neighborhood might not. Images, spectra, token sequences, and graphs each need their own P . A poor packing is a fact about P , not a verdict on Q_n .

Frozen versus trained geometry. When does a shared Hamming-1 kernel, a private LCN table, or a frozen reservoir plus a thin head actually use the balls of Q_n , and when is the cube only an addressing scheme? Using the balls means the field varies across Hamming neighbors in a way the gather, the kernel, or the reservoir exploits. An addressing scheme is a cube-shaped index with no such variation. Vertex-transitivity does not decide which of those two a given member is. An experiment does.

Ball-restricted retrieval. Softmax attention inside $B_r(v)$ costs what the ball costs, not what the whole cube costs. Ramsauer et al. [6] count patterns against the full ambient space. Here each query only sees $B_r(v)$. How many patterns that ball can hold, as a function of n and r , is open.

Walks. There are $n!$ shortest paths from r to $\alpha(r)$. HypercubeEtalon walks one of them. A different geodesic visits a different sequence of vertices and writes a different field. Whether the readout cares is untested in this note.

Scale. The codes stop near $n = 16$ because memory follows 2^n . The algebra does not. Whether the same XOR gathers and translations remain useful at $n = 20$ is both an engineering problem and a mathematical one.

Filtration. Soft modularity is a reading of one weight field by cutoff. Raise the cutoff and weak edges drop; the cube can fall into pieces. Nobody has checked whether those pieces are the parts a task uses.

These six questions are open. Appendix A is the working copy: change P , change n , change the cutoff, pick a geodesic, or store a pattern set, and publish the protocol with the result.

A Implementations

C++ is consumed from the repository (`FetchContent / find_package`). Python wheels are on PyPI. The pip slug uses hyphens; the import name uses underscores (`pip install hypercube-esn`, then `import hypercube_esn`).

Product	Source	pip install
HypercubeESN	https://github.com/dliptak001/HypercubeESN	hypercube-esn
HypercubeCNN	https://github.com/dliptak001/HypercubeCNN	hypercube-cnn
HypercubeHopfield	https://github.com/dliptak001/HypercubeHopfield	hypercube-hopfield
HypercubeLCN	https://github.com/dliptak001/HypercubeLCN	hypercube-lcn
HypercubeEtalon	https://github.com/dliptak001/HypercubeEtalon	hypercube-etalon
HypercubeWTF	https://github.com/dliptak001/HypercubeWTF	hypercube-wtf
HypercubeCascade	https://github.com/dliptak001/HypercubeCascade	hypercube-cascade

References

- [1] Christian Borgs, Jennifer T. Chayes, Remco van der Hofstad, Gordon Slade, and Joel Spencer. Random subgraphs of finite graphs: III. the phase transition for the n -cube. *Combinatorica*, 26(4):395–410, 2006.
- [2] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2990–2999, 2016.
- [3] Claudio Gallicchio and Alessio Micheli. Graph echo state networks. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2010.
- [4] L. H. Harper. Optimal numberings and isoperimetric problems on graphs. *Journal of Combinatorial Theory*, 1(3):385–393, 1966.
- [5] Yuichi Katori, Hakaru Tamukoh, and Takashi Morie. Reservoir computing based on dynamics of pseudo-billiard system in hypercube. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.
- [6] Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Lukas Gruber, Markus Holzleitner, Thomas Adler, David P. Kreil, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. Hopfield networks is all you need. In *International Conference on Learning Representations*, 2021. arXiv:2008.02217.
- [7] Remco van der Hofstad and Gordon Slade. Expansion in n^{-1} for percolation critical values on the n -cube and $\mathbb{Z}^n$: the first three terms. *Combinatorics, Probability and Computing*, 15(5): 695–713, 2006.