Structure-Aware Neural Decoding for the X-Cube Fracton Code

A Learned Recovery Layer whose Logical Error Rate Improves with System Size

Brian Guarraci
Galatheus
brian@galatheus.com

June 2026

Abstract

The long-term storage of information in three dimensions can be cast as a decoding problem: lay down local, overlapping parity checks, and recover the stored state from the pattern of violated checks (the syndrome) with a prior. We argue the *right* checks are supplied by *fracton order*, whose immobile, sub-dimensional excitations give nonlocal conservation laws and a subextensive but growing logical count ($k = 6L - 3$ on the 3-torus); whether this becomes a passive, self-correcting memory is a Hamiltonian/lifetime question, while here we isolate the missing piece on the *decoding* side. The obstacle is decoding: surface-code-style MWPM does not directly apply — excitations are created four at a time with no string of defect pairs to match — and while code-specific decoders exist, generic BP+OSD struggles on the dense cube-check representation. We supply, to our knowledge, the first *neural* decoder for the X-cube fracton code: a weight-shared CNN with a structure-preserving line-aggregation head (AxisLineonHead) that reads the cube (Z) syndrome as an L^3 scalar field and predicts the $6L - 3$ logical-Z parities. Across $L \in \{3, 4, 6, 8, 10\}$ and $p \in \{0.03, 0.05\}$, over three seeds with disjoint test splits, it beats the trivial floor, BP+OSD-0, and our naive-weight Brown–Williamson implementation (BW-naive) at every operating point (paired McNemar $p < 10^{-12}$ vs. trivial and BP+OSD-0; we do not claim to benchmark the fully tuned Brown–Williamson decoder). Critically, its per-sector logical error rate (basis-relative) *improves* monotonically with code size at $p = 0.03$ ($0.042 \to 0.035 \to 0.018 \to 0.008 \to 0.003$ for $L = 3, 4, 6, 8, 10$), and its basis-invariant block-failure rate improves from $L = 4$ onward ($0.343 \to 0.118$, the lone $L = 3 \to 4$ step aside), while all three baselines degrade or saturate — the scaling signature a useful decoder must show and a fixed local lookup table cannot. An ablation isolates the mechanism: a global-pooling head ties trivial exactly while line-aggregation and coordinate-convolution heads win, so the advantage comes from the code's non-local line structure; a signal-existence oracle shows radius-1 local signal collapses by $L = 4$ and is negligible by $L = 6$, yet the learned decoder wins decisively. These results establish a scalable learned recovery primitive — a candidate active recovery layer for the broader self-correcting-memory program — and show that a geometry-matched neural decoder can extract nonlocal fracton syndrome structure increasingly well as the code grows.

1 Introduction

This paper is the quantum-fracton member of a four-paper program on learned recovery over constraint systems. The shared claim is not that one operational decoder applies unchanged across classical compression, surface-code QEC, and fracton QEC. It is narrower and more testable: useful learned decoders match their inductive bias to the geometry of the constraints they are asked to invert. The companion compression paper isolates when parity checks help a learned proposal; the gauge-structure paper proves the algebraic bridge and shows that exact algebraic enforcement can be the wrong architecture; the surface-code paper measures the capacity boundary when local stabilizer geometry is supplied by the code. This paper asks

whether the same geometry-aware principle can supply the missing learned recovery layer for the X-cube fracton code.

The resulting ladder is:

1. **Syndrome-based 3D storage.** Recover three-dimensional stored data from local, overlapping parity checks plus a learned prior. Empirically, *constraint locality* is what makes this work: local overlapping checks help, high-weight global projections hurt.
2. **The checks are a fracton theory.** The axial-parity codec's codespace is a discrete-$\mathbb{Z}_2$ scalar-charge fracton gauge theory with closed-form rank $3n^2 - 3n + 1$ (see §3); its parity checks *are* the fracton conservation laws. The classical storage scheme is the classical shadow of fracton physics.
3. **Quantum code instances.** The surface code and the X-cube fracton code [1] are quantum stabilizer instances in this constraint family. Fracton order is one proposed route toward *passive* quantum memory: its sub-dimensional excitations and conservation laws can obstruct error motion in ways unavailable to 2D codes, at a subextensive but growing logical count ($k = 6L - 3$ on the 3-torus). Whether a given model actually yields self-correction is a Hamiltonian/lifetime question, not settled here; this complements the broader push toward hardware-efficient high-rate qLDPC memories [11].
4. **The missing recovery layer (this paper).** Any active readout layer for such a memory needs a recovery rule, and X-cube decoding lacks a universally adopted surface-code-like standard decoder [12, 5]. We supply a learned one and show its logical error rate *improves* with system size (each L is trained separately — a scaling property, not zero-shot cross-size transfer).

The through-line across all four levels is one principle: *recovery is geometry-aware inference over a constraint system; a learned decoder wins when its architecture matches the error geometry.*

Why X-cube decoding is hard for generic decoders is structural. The X-error syndrome of the X-cube code is the set of violated cube (Z) stabilizers; the excitations are immobile fractons created four at a time at the corners of error membranes. There is no one-dimensional error string connecting pairs of defects, so a single surface-code-style MWPM over defect pairs — the workhorse of surface-code decoding — does not directly apply. General qLDPC decoders such as BP+OSD can be run, but the dense, highly-overlapping twelve-qubit cube checks frustrate belief propagation [12], and convergence collapses at moderate error rates (we observe BP convergence below 0.05 at $L = 6$, $p = 0.05$). Unlike surface codes, X-cube decoding has no universally adopted MWPM-like standard: code-specific decoders exist (notably Brown–Williamson's reduction to two-dimensional matching [5]), but generic BP+OSD struggles on the dense cube-check representation — leaving structural headroom for a learned decoder rather than only a marginal edge.

Scope and honesty. The self-correcting-memory payoff (level 3) is the *motivation*; we do not claim to demonstrate a self-correcting memory here. The concrete, defensible contribution of this paper is level 4: a learned decoder for the X-cube code that (i) beats the trivial baseline, BP+OSD-0, and our BW-naive implementation at every tested operating point over three seeds with disjoint test splits, (ii) *improves* per-sector with code size, and (iii) does so via a mechanism we isolate (non-local line structure). Level 2, the gauge bridge, is summarized in §3 as motivation for the program but is not used in the decoder benchmark.

2 Background: the X-cube code and the decoding task

The X-cube code [1] places its qubits on the $3L^3$ edges of a periodic $L \times L \times L$ cubic lattice. The stabilizer group is generated by (i) X-type "cross" operators — per vertex, the four-edge planar cross in each of the xy, yz, xz planes (two independent per vertex, since the third is their product), and (ii) Z-type "cube" operators — the product of the twelve edges of each elementary cube. The code is CSS ($H_X H_Z^\top = 0$ over $\mathbb{F}_2$) and we verify $k = N - \mathrm{rank}(H_X) - \mathrm{rank}(H_Z) = 6L-3$ for all simulated sizes $L \in \{2, 3, 4, 5, 6, 8, 10\}$ (giving $k = 9, 15, 21, 27, 33, 45, 57$), matching the known ground-state degeneracy on the 3-torus.

We study the code-capacity bit-flip (X) channel at rate p, detected by the cube (Z) stabilizers. The cube syndrome has exactly L^3 bits — one per cube center — so it is naturally an $L \times L \times L$ scalar field, a direct fit for a 3D convolutional decoder with no temporal axis. The decoding label is the vector of $6L - 3$ logical-Z parities of the error, $\ell_j = \langle \mathrm{logical}_Z^{(j)}, e \rangle \bmod 2$; the decoder predicts these per-sector classes from the syndrome field. We report *block failure* (any sector wrong) and the mean *per-sector logical error rate* (LER).

Why this is decoding, not classification. Predicting the $6L-3$ logical-Z parities is the full decoding output, not a proxy. Given a syndrome s, fix once a syndrome-consistent representative ("pure error") $R(s)$ derived from H_Z; the residual $e \oplus R(s)$ then lies in the logical+stabilizer group, so the only remaining freedom is which logical coset it occupies. In the experiments below the network is trained to predict the raw logical parities $\langle \mathrm{logical}_Z^{(j)}, e \rangle$ of the sampled error. Equivalently, an explicit recovery using an arbitrary $R(s)$ applies the logical correction with parity $\hat{\ell}_j \oplus \langle \mathrm{logical}_Z^{(j)}, R(s) \rangle$; if $R(s)$ is chosen with zero logical-Z parities this offset vanishes. A per-sector logical error occurs precisely when the predicted parity is wrong. This is amortized maximum-likelihood coset decoding, the same target used by neural surface-code decoders.

3 From classical compression to quantum memory: the gauge bridge

The link between the classical storage scheme (level 1) and the quantum code (level 3) is not an analogy but an identity at the level of the check algebra. The axial-parity codec's codespace equals $\mathrm{image}(\partial_x \partial_y \partial_z)$ over $\mathbb{F}_2$, with $\ker(\partial_x \partial_y \partial_z) = K_x + K_y + K_z$ and closed-form rank $3n^2 - 3n + 1$ (machine-verified in the accompanying code). This identifies the parity codec as a discrete-$\mathbb{Z}_2$ scalar-charge fracton gauge theory [3]: its line, plane, and global parity checks are exactly the fracton conservation laws (line charges, planar charges, total charge). The X-cube code is the adjacent $\mathbb{Z}_2$ stabilizer member of the same family [1]. So "syndromes recover 3D data" and "a fracton code stores a logical qubit" are two uses of the same constraint language. The shared object is not an identical operational decoder: in compression the task is reconstruction of a plausible constraint-consistent voxel field, while here it is logical-coset inference from a quantum syndrome. What transfers is the recovery paradigm: choose an architecture whose inductive bias matches the constraint geometry.

4 Decoder and baselines

AxisLineonHead. The decoder is a weight-shared 3D residual CNN (default width 64, depth 8) over the L^3 cube-syndrome field, followed by a line-aggregation head: features are collapsed along each logical-line axis, passed through a 2D transverse convolution, and gathered into the $6L - 3$ per-sector logits. The head is the only code-aware component; the body is a generic 3D CNN. Training uses 3×10^5 samples per cell, 30 epochs, a cosine learning-rate schedule, and batch size 2048 for $L \leq 6$, reduced to 768 at $L = 8$ and 384 at $L = 10$ for memory.

Baselines. (i) *Trivial*: predict the all-zero logical class (the maximum-likelihood guess in the absence of decoding), the relevant floor for this per-sector task. (ii) *BP+OSD-0*: belief propagation with order-0 ordered-statistics post-processing on H_Z, a standard general-purpose qLDPC decoder [12]; we report its block failure and BP convergence fraction. (iii) *BW-naive*: our naive-weight implementation of Brown–Williamson's X-cube matching framework, which exploits the rigid fracton structure to reduce decoding to two-dimensional matching problems [5]. We implement the cube-defect ("planeon") matching variant (per-plane MWPM + cluster neutralization; 100% syndrome-consistent, exact on isolated single-edge errors); BW's BP edge-reweighting that lifts its threshold $3.7\% \to 4.3\%$ is a documented extension we do not implement, so we do not claim to benchmark the fully tuned Brown–Williamson decoder. This planeon matching is syndrome-consistent but does not optimize over the logical wrapping-line degeneracies that distinguish syndrome-equivalent corrections; as L grows it produces many recoveries that fix the syndrome yet land in the wrong logical coset, which is why its per-sector LER can rise above the trivial floor even while its block failure stays at or below it (by $L = 10$ both are saturated near 1).

Signal-existence oracle (Gate-0). As a locality control, we evaluate a held-out local-pattern lookup over the syndrome in a *radius-1* transverse neighborhood of each logical line. This empirical maximum-likelihood estimate on the finite training split measures the usable signal in that window; it is a control for the mechanism analysis, not a formal Bayes-optimal bound and not a bound over local decoders of arbitrary receptive field. We run this control through $L = 6$; since the radius-1 advantage is already $\leq 0.3\%$ there, we use it as mechanism evidence rather than re-running it for the $L = 8, 10$ rows.

Protocol. Every cell is run over seeds $\{42, 43, 44\}$ with a disjoint test sampler seed offset from train and validation. We report seed-bootstrap 95% confidence intervals on block failure and paired McNemar tests (on identical shots) for learned-vs-trivial and learned-vs-BP+OSD-0. The predeclared win condition (fixed before evaluation) is: learned block-failure CI upper bound below the trivial CI lower bound at $L = 3$, $p = 0.05$. All numbers come from one reproducible pipeline built on `reproduce_xcube_results.py`: a single command for $L \leq 6$, and, for $L \geq 8$, per-cell shards of that same script merged with a parallel CPU BW-naive run. Exact commands and output paths are in `REPRODUCIBILITY.md`.

5 Results

The decoder wins at every operating point. Table 1 gives the full grid. At every (L, p) the learned decoder has strictly lower block failure than BW-naive, BP+OSD-0, and trivial, with non-overlapping seed CIs against trivial in the JSON and paired McNemar $p < 10^{-12}$ against trivial and BP+OSD-0 at every cell, and against BW-naive for $L \leq 6$ (where the paired test is tabulated; BW-naive is compared by aggregate failure rate at $L \geq 8$, see App. A) (e.g. at $L = 3, p = 0.03$, learned-vs-trivial b=53294, c=1794; learned-vs-BP+OSD-0 b=511, c=99; learned-vs-BW-naive b=12648, c=2499). BP+OSD-0 itself beats trivial at small codes but degrades with size and noise; its BP convergence fraction falls from ~ 0.8 ($L = 3, p = 0.03$) to ~ 0.03 ($L = 6$, $p = 0.05$). Our BW-naive implementation is stronger than the trivial floor on block failure (BP+OSD-0 is lower still), yet the learned decoder beats it at every cell on *both* metrics, and the margin *widens* with code size: at $L = 6, p = 0.03$ the learned per-sector LER (0.018) is 13× below BW-naive's (0.242). The contrast is sharp — the learned per-sector LER *improves* with L while BW-naive's *worsens* (the wrapping-line degeneracy it ignores grows with the code), so BW-naive's per-sector LER even exceeds the trivial floor by $L = 6$ despite still dominating trivial on block failure.

Table 1: Block failure and per-sector LER (3-seed means) for the learned decoder vs. our naive-weight Brown–Williamson decoder ("BW-naive"), BP+OSD-0, and the trivial floor, on the code-capacity X-channel. Lower is better. **The learned decoder wins at every cell on both metrics.** Learned/BW-naive block-failure 95% CIs and exact paired-McNemar counts are in App. A; McNemar $p < 10^{-12}$ ($b \gg c$) for learned-vs-trivial and learned-vs-BP+OSD-0 at every cell, and for learned-vs-BW-naive at $L \leq 6$ (at $L \geq 8$ BW-naive fails near-totally, block failure ≥ 0.99, and is compared by aggregate failure rate, not paired shot-by-shot). The $L = 8$ rows use a memory-reduced batch (768 vs 2048) and the $L = 10$ rows batch 384. The learned, trivial, and BW-naive decoders are evaluated on 40,000 test shots per seed for $L \leq 6$ and 30,000 per seed for $L \geq 8$; the much slower BP+OSD-0 baseline is evaluated on a subset (2,000 shots per seed through $L = 8$, 300 at $L = 10$), with BW-naive run on dedicated CPU nodes and merged. BW-naive (§4) dominates the trivial floor on block failure but, ignoring the wrapping-line degeneracy, its per-sector LER rises above trivial at higher p/L (by $L = 8$, so does BP+OSD-0).

		Block failure				Per-sector LER			
L	p	Learned	BW-naive	BP+OSD-0	Triv.	Learned	BW-naive	BP+OSD-0	Triv.
3	0.03	**0.306**	0.391	0.372	0.735	**0.042**	0.098	0.110	0.085
3	0.05	**0.626**	0.703	0.688	0.887	**0.104**	0.204	0.228	0.136
4	0.03	**0.343**	0.682	0.465	0.914	**0.035**	0.169	0.140	0.110
4	0.05	**0.724**	0.937	0.840	0.981	**0.110**	0.301	0.316	0.172
6	0.03	**0.282**	0.935	0.595	0.996	**0.018**	0.242	0.233	0.155
6	0.05	**0.800**	0.999	0.979	1.000	**0.104**	0.392	0.451	0.234
8	0.03	**0.194**	0.993	0.864	1.000	**0.008**	0.295	0.361	0.195
8	0.05	**0.850**	1.000	1.000	1.000	**0.095**	0.441	0.492	0.285
10	0.03	**0.118**	1.000	0.978	1.000	**0.003**	0.336	0.427	0.231
10	0.05	**0.863**	1.000	1.000	1.000	**0.080**	0.466	0.494	0.326

The advantage grows with code size. The scaling appears on both metrics (Fig. 1). On the *basis-invariant* block-failure rate (all $6L - 3$ sectors correct at once), the learned decoder improves with size at $p = 0.03$ from 0.343 at $L = 4$ to 0.118 at $L = 10$ (the $L = 3$ value 0.306 is the only non-monotone step), while every baseline saturates toward 1. Per logical qubit — the per-sector LER, reported in the frozen axis-aligned wrapping-line logical-Z basis the model is trained against, and thus *basis-relative* (block failure is the basis-invariant metric) — the learned decoder improves *monotonically* from $0.042 \to 0.035 \to 0.018 \to 0.008 \to 0.003$ as $L = 3 \to 4 \to 6 \to 8 \to 10$, while the trivial baseline *worsens* ($0.085 \to 0.110 \to 0.155 \to 0.195 \to 0.231$) and BP+OSD-0 worsens ($0.110 \to 0.140 \to 0.233 \to 0.361 \to 0.427$). At $L = 6$ this is a 9× per-sector advantage over trivial, widening to ∼24× at $L = 8$ and ∼66× at $L = 10$. Strikingly, by $L = 8$ both BP+OSD-0 and BW-naive have per-sector LER *above* the trivial floor, so the learned decoder is the only one of the four that improves on doing nothing *on the per-sector metric*; its learned-vs-trivial McNemar at $L = 8, p = 0.03$ has $c = 0$ — across 72,574 discordant shots the trivial decoder never once beat it. This persists at $L = 10$, where the learned decoder is again the only one of the four below the trivial floor (per-sector 0.003 vs. 0.336 BW-naive, 0.427 BP+OSD-0, 0.231 trivial) and again has $c = 0$ across 79,351 discordant shots. A decoder whose accuracy improves as the code grows is exhibiting the scaling behavior required of a useful decoder and that a fixed local lookup table cannot provide.

A threshold-crossing view. Plotting the learned decoder's error against the physical rate p, one curve per L, exposes where the size-scaling reverses (Fig. 2). On the basis-invariant *block-failure* metric the L-ordering inverts in a narrow band around $p^\star \approx 0.045$: for p below it, larger L gives *lower* block failure (the sub-threshold, size-improving regime our $p = 0.03$ results occupy), while for p above it the ordering flips and larger L is worse — which is precisely

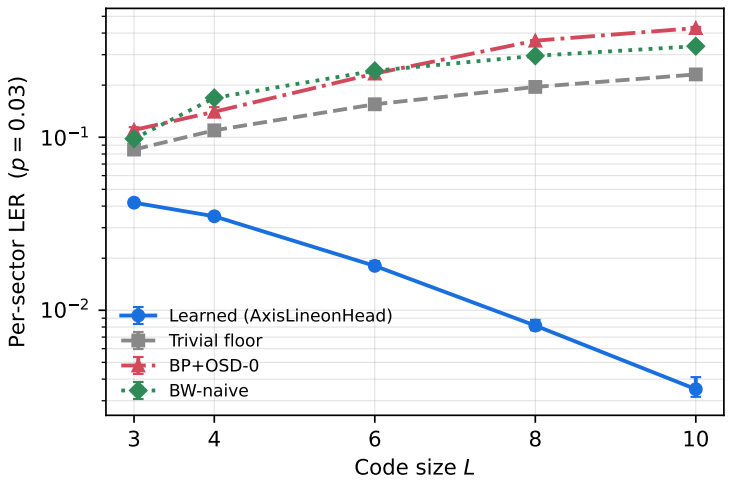

Figure 1: Per-sector LER versus code size L at $p = 0.03$ (log scale, 3-seed means, seed-bootstrap 95% CIs; data from Table 1). The learned decoder *improves* monotonically with L while the trivial floor, BP+OSD-0, and BW-naive all *degrade* — the monotone divergence is the paper's core claim. Figure emitted by `make_scaling_figure.py` directly from the `reproduce_xcube_results.py` output JSONs.

why the $p = 0.05$ block-failure rows in Table 1 climb with L. This estimate is deliberately coarse: away from $L = 4$ each curve is two operating points ($p \in \{0.03, 0.05\}$) joined by linear interpolation, so $p^\star$ should be read as ≈ 0.04–0.05 rather than a precisely resolved threshold, and it sits, reassuringly, near the tuned Brown–Williamson threshold ($\approx 4.3\%$ [5]) and below the optimal X-cube fracton-sector value ($\approx 7.5\%$ [4]). The *per-sector* metric, by contrast, shows no such crossing in this window: larger L stays at or below smaller L through $p = 0.05$ and only loses its advantage near $p \approx 0.11$–0.13 (the $L = 4$ sweep of §7), consistent with the monotone $p = 0.03$ improvement being a sub-threshold, basis-relative effect rather than a demonstrated code threshold.

Mechanism: the win is non-local. Two controls localize the source of the advantage. *(i) Ablation* (Table 2): replacing the line-aggregation head with global average pooling collapses performance *exactly* to the trivial baseline (block failure 0.7342 vs. trivial 0.7342), i.e. the global-pool decoder learns nothing useful; whereas both the line-aggregation head and a coordinate-convolution head recover the win (0.303 and 0.302). The structure-preserving head, not raw capacity, is what carries the result. *(ii) Signal-existence oracle*: the held-out radius-1 local-window lookup has per-sector advantage over trivial of 34.2% at $L = 3, p = 0.03$, falling to 5.7% at $L = 4, p = 0.03$, 0.1% at $L = 4, p = 0.05$, and $\leq 0.3\%$ for all $L = 6$ cells. The radius-1 local signal has largely collapsed by $L = 4$ and is negligible by $L = 6$, yet the learned decoder wins decisively throughout — so it is exploiting *non-local* correlations (the lineon/planon line and membrane structure of the code) outside this radius-1 oracle family. This also explains why the head matters: line aggregation is exactly the inductive bias that surfaces this non-local structure.

Predeclared win condition. The fixed-in-advance criterion (learned block-failure CI upper bound 0.6282 below trivial CI lower bound 0.8859 at $L = 3$, $p = 0.05$) is satisfied.

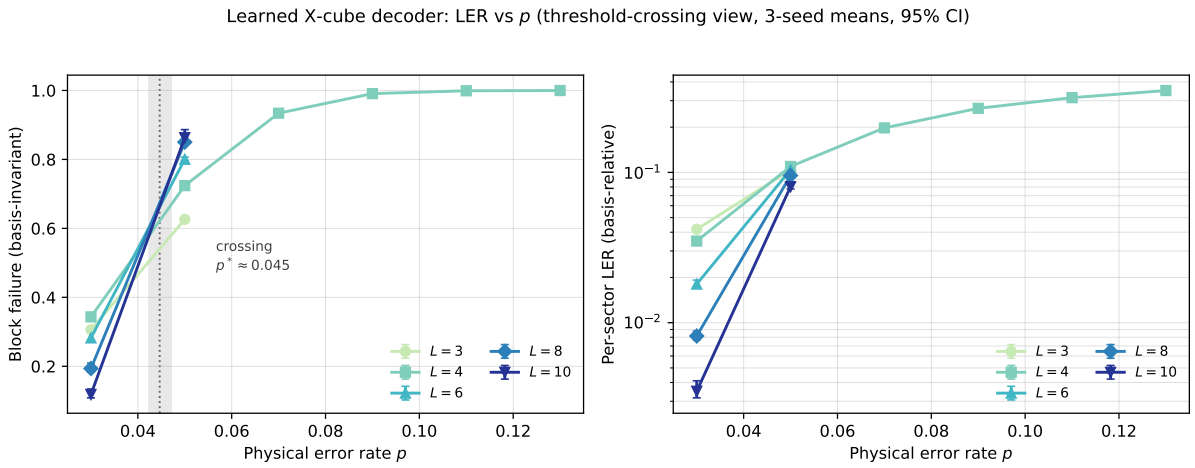

Figure 2: Learned-decoder LER versus physical error rate p, one curve per code size L (3-seed means, seed-bootstrap 95% CIs; same released JSONs as Table 1 plus the $L = 4$ high-p sweep). *Left*: block failure (basis-invariant); the L-curves cross in a band around $p^\star \approx 0.045$ (shaded), separating the sub-threshold regime (larger L better) from the super-threshold regime (larger L worse). *Right*: per-sector LER (log scale, basis-relative); the curves do not cross in this window. Away from $L = 4$ each curve is a two-point linear interpolation between $p = 0.03$ and $p = 0.05$, so the crossing is a coarse estimate (see Discussion). Figure emitted by `make_threshold_figure.py` directly from the `reproduce_xcube_results.py` output JSONs.

Table 2: Head ablation at $L = 3$, $p = 0.03$ (seed 42). Global pooling ties trivial; structure-preserving heads win.

Head	Block failure	Per-sector LER
Trivial (floor)	0.7342	0.0846
Global average pool	0.7342	0.0847
Coordinate conv	0.3022	0.0420
AxisLineonHead	0.3026	0.0421

6 Related work

Neural decoders have been developed extensively for the surface and toric codes, including equivariant architectures (The END [6]; the 3D-toric equivariant ML decoder [7], which shows weight-sharing is necessary but not sufficient) and scalable CNNs for 3D/4D toric codes [8]. For fracton codes specifically, prior decoding work is *algorithmic*: Brown–Williamson give a parallelized matching+BP fracton decoder [5] (threshold $\approx 4.3\%$), and the optimal X-cube fracton-sector threshold under X-noise is $\approx 7.5\%$ [4]. Neural-network quantum states have been applied to fracton *ground states* [9] but not to decoding. To our knowledge no *neural* decoder has been demonstrated for the X-cube fracton code; our contribution is scoped precisely as that — both *neural* and *X-cube-specific*. We benchmark it against the BW-naive implementation of Brown–Williamson's code-specific matching framework [5] described in §4, alongside the trivial floor and BP+OSD-0. We distinguish it from the parallel line of work that decodes high-rate qLDPC codes by *algebraic* means: Wills et al. [11] concatenate a quantum Reed–Solomon outer code over a bivariate-bicycle qLDPC inner code, decoded by list decoding and (Relay-)BP, with no learned component — a different code family *and* a non-neural method. Learned message-passing decoders for qLDPC and bivariate-bicycle codes also exist: Astra [10] uses a graph-neural/message-passing architecture on Tanner graphs and reports extrapolation across code distances; it is complementary to our X-cube-specific line-aggregation head rather than a prior

fracton decoder. Comprehensive surveys of the decoder landscape [12] confirm that machine-learning decoders and fracton-adjacent topics are each individually mainstream; the novelty here is their not-previously-combined intersection. The self-correcting-memory motivation follows the fracton literature (Haah's code [2] and the partial-self-correction line).

7 Discussion and limitations

The result establishes *existence*: a learned recovery layer for the X-cube code whose per-sector accuracy *scales* (improves) through $L = 10$ and that beats the trivial floor, BP+OSD-0, and our BW-naive implementation in the tested regime, with a mechanistic explanation. This is a candidate active recovery layer for the broader self-correcting-memory program; it does not by itself constitute a self-correcting memory, which is a statement about Hamiltonian dynamics and lifetime, not about decoding a code-capacity channel. On the baseline side, we use BP+OSD-0 as a scalable general-purpose qLDPC decoder; tuned higher-order OSD may narrow the gap at substantially greater cost, so we do not claim that all BP+OSD variants are dominated — only that a generic BP+OSD-0 decoder fails to capture the X-cube geometry in this regime. A common concern for learned decoders is that training cost scales unfavorably with code distance [12]. Our design speaks to the *model-size* side of this directly — the decoder is *weight-shared*, so its parameter count is independent of L. We do not measure training time or sample complexity versus L (each L is trained separately), and so make no training-cost-scaling claim; what we establish is an *accuracy*-scaling result — decode accuracy *improves* with code size ($0.042 \to 0.035 \to 0.018 \to 0.008 \to 0.003$ at $p = 0.03$ for $L = 3, 4, 6, 8, 10$) rather than degrading.

Two limitations bound the present scope. First, the win has a bounded operating range in p. A fine-grained $L = 4$ sweep (3 seeds) shows the per-sector advantage over trivial decaying smoothly with noise: a clear win through $p = 0.05$ (0.110 vs 0.172), narrowing to 0.198 vs 0.227 at $p = 0.07$ and 0.267 vs 0.274 at $p = 0.09$, and closing to a tie by $p \approx 0.11$–0.13 — so the useful range reaches the vicinity of the $\approx 7.5\%$ optimal threshold [4] (BP+OSD-0 is dominated throughout this range, per-sector LER 0.42–0.48). The block-failure threshold of Fig. 2 ($p^\star \approx 0.045$) is, correspondingly, only coarsely resolved: it rests on two operating points per L for $L \neq 4$. Sharpening it into a publication-grade threshold estimate requires retraining the learned decoder on a dense p grid (e.g. $p \in [0.030, 0.055]$ in steps of 0.005) at every L and locating the crossing with its finite-size scaling collapse; the released pipeline supports this directly (it is a wider `-ps` grid in `reproduce_xcube_results.py`), and we leave the dense sweep, together with an optimal-decoder reference curve, to future work. Second, the decoder is demonstrated on the X-cube code; transferring the line-aggregation inductive bias to other codes where matching also breaks down is the natural next step. We see a concrete, sequenced path: (i) the *3D toric code* first — it is genuinely three-dimensional and topological, so the line-aggregation head transfers most directly; then (ii) high-rate *bivariate-bicycle* qLDPC codes, the frontier near-term high-rate target [12, 11] (e.g. the gross $[[144, 12, 12]]$ code), where the baseline to beat is a tuned Relay-BP decoder [11] rather than vanilla BP+OSD, and where the head will likely need redesign toward the quasi-cyclic symmetry, since these codes are 2D and lack the lineon/planon structure the X-cube head exploits. Decoding such quasi-cyclic codes "remains a challenge" [12], which is exactly why a structure-aware learned decoder is worth testing there. Closing these gaps, on top of the gauge bridge of §3, is what would turn the arc of §1 from a through-line into a program.

Reproducibility

All numbers are emitted by reproducible scripts. The main grid, ablation, and signal-existence oracle come from `reproduce_xcube_results.py` (output `data/xcube_L346_gen.json`, with

the $L \geq 8$ scaling rows in `data/xcube_L810_gen.json`); the high-p sweep from the same script (`data/xcube_L4_psweep.json`); and the BW-naive baseline from the consolidated $L \leq 6$ run plus the parallel $L \geq 8$ CPU run (`reproduce_xcube_bw_hpc.py`), with the standalone `data/xcube_bw_baseline.json` retained for provenance. Full provenance, including the per-cell statistics of App. A, is in `REPRODUCIBILITY.md`. In the family release, these artifacts are grouped with the X-cube and gauge-theorem code under the fracton module of the companion repository.

A Per-cell confidence intervals and McNemar counts

Table 3 reports the seed-bootstrap 95% confidence intervals on block failure for the learned and BW-naive decoders, and the paired-McNemar discordant counts (b, c) on identical test shots (b = shots the learned decoder gets right while the comparison decoder gets them wrong; c = the reverse). All paired McNemar tests are overwhelmingly significant; the continuity-corrected χ^2 statistic (or an exact binomial test) gives $p < 10^{-12}$ in every tabulated paired comparison (learned-vs-BW-naive is tabulated for $L \leq 6$ only; at $L \geq 8$ BW-naive is compared by aggregate failure rate, not paired shot-by-shot). BP+OSD-0 is evaluated on the smaller cost-limited subsets noted in Table 1 (2,000 shots/seed through $L = 8$, 300 at $L = 10$); even at the smallest subset its block-failure Wilson 95% interval stays far from the learned value (e.g. at $L = 10, p = 0.03$, BP+OSD-0 0.978, Wilson $[0.966, 0.986]$ on 900 shots, vs. learned 0.118), so the reported gaps vastly exceed the baseline sampling error.

Table 3: Per-cell block-failure 95% CIs and paired-McNemar discordant counts (b, c) for the learned decoder against trivial, BP+OSD-0, and BW-naive (the last for $L \leq 6$; — at $L \geq 8$, where BW-naive fails near-totally and is compared by aggregate failure rate). $L \geq 8$ from task-parallel per-cell runs.

		Learned bf	BW-naive bf	McNemar (b, c)		
L	p	[95% CI]	[95% CI]	vs. trivial	vs. BP+OSD-0	vs. BW-naive
3	0.03	0.306 [0.303, 0.308]	0.391 [0.387, 0.394]	(53294, 1794)	(511, 99)	(12648, 2499)
3	0.05	0.626 [0.624, 0.628]	0.703 [0.699, 0.705]	(33405, 2079)	(499, 115)	(12663, 3470)
4	0.03	0.343 [0.343, 0.344]	0.682 [0.681, 0.685]	(69620, 1154)	(988, 257)	(43529, 2852)
4	0.05	0.724 [0.720, 0.728]	0.937 [0.936, 0.938]	(31555, 719)	(859, 146)	(26963, 1399)
6	0.03	0.282 [0.272, 0.296]	0.935 [0.934, 0.936]	(85691, 31)	(2067, 218)	(78726, 369)
6	0.05	0.800 [0.796, 0.807]	0.999 [0.999, 0.999]	(23950, 11)	(1108, 22)	(23854, 30)
8	0.03	0.194 [0.183, 0.210]	0.993 [0.992, 0.993]	(72574, 0)	(4067, 23)	—
8	0.05	0.850 [0.842, 0.856]	1.000 [1.000, 1.000]	(13485, 0)	(905, 0)	—
10	0.03	0.118 [0.108, 0.134]	1.000 [0.999, 1.000]	(79351, 0)	(792, 1)	—
10	0.05	0.863 [0.851, 0.887]	1.000 [1.000, 1.000]	(12337, 0)	(117, 0)	—

References

[1] S. Vijay, J. Haah, and L. Fu. Fracton topological order, generalized lattice gauge theory, and duality. *Phys. Rev. B*, 94:235157, 2016. arXiv:1603.04442.

[2] J. Haah. Local stabilizer codes in three dimensions without string logical operators. *Phys. Rev. A*, 83:042330, 2011. arXiv:1101.1962.

[3] M. Pretko. Subdimensional particle structure of higher rank $U(1)$ spin liquids. *Phys. Rev. B*, 95:115139, 2017. arXiv:1604.05329.

[4] H. Song et al. Optimal thresholds for fracton codes and random spin models with subsystem symmetry. arXiv:2112.05122, 2021. (Optimal X-cube fracton-sector threshold $\approx 7.5\%$.)

[5] B. J. Brown and D. J. Williamson. Parallelized quantum error correction with fracton topological codes. *Phys. Rev. Research*, 2:013303, 2020. arXiv:1901.08061.

[6] E. Egorov, R. Bondesan, and M. Welling. The END: An equivariant neural decoder for quantum error correction. arXiv:2304.07362, 2023.

[7] O. Weissl and E. Egorov. Equivariant machine learning decoder for 3D toric codes. arXiv:2409.04300, 2024. (Weight-sharing necessary but not sufficient.)

[8] N. P. Breuckmann and X. Ni. Scalable neural network decoders for higher dimensional quantum codes. *Quantum*, 2:68, 2018. arXiv:1710.09489.

[9] M. Machaczek, L. Pollet, and K. Liu. Neural quantum state study of fracton models. arXiv:2406.11677, 2024. (Ground states, not decoding.)

[10] A. S. Maan and A. Paler. Machine learning message-passing for the scalable decoding of QLDPC codes. arXiv:2408.07038, 2024.

[11] A. Wills, M. E. Beverland, L. S. Bishop, J. M. Gambetta, P. Rall, V. Siddhu, and A. W. Cross. Concatenating algebraic codes over high-rate quantum LDPC codes. arXiv:2605.21898, 2026.

[12] D. J. Spencer, S. P. Jain, A. Tanggara, Z. Sun, T. Haug, D. Khu, and K. Bharti. Quantum error correction and fault tolerance: A comprehensive tutorial. arXiv:2605.29137, 2026.