XONTOOLS RESEARCH SERIES · NO. 5

The Xon Neural Network

A network that thinks by settling

What a Xon network is, how it would learn and write, what it might do that today's models don't, what has already been tested, and the gates between here and an answer.

Ian MacKenna · xontools.ai · September 2026

> **Status.** This is the long-term track of the XonTools research program, and most of it is a vision. Each idea carries a label: **TESTED** has been run, with the result reported here, including failures; **DESIGNED** is specified with pre-registered pass criteria but not yet built; **ENVISIONED** is a possibility, not a commitment. The scenarios are invented.
>
> The oversight tools do not depend on any of this working. Related documents and tools are at **xontools.ai**.

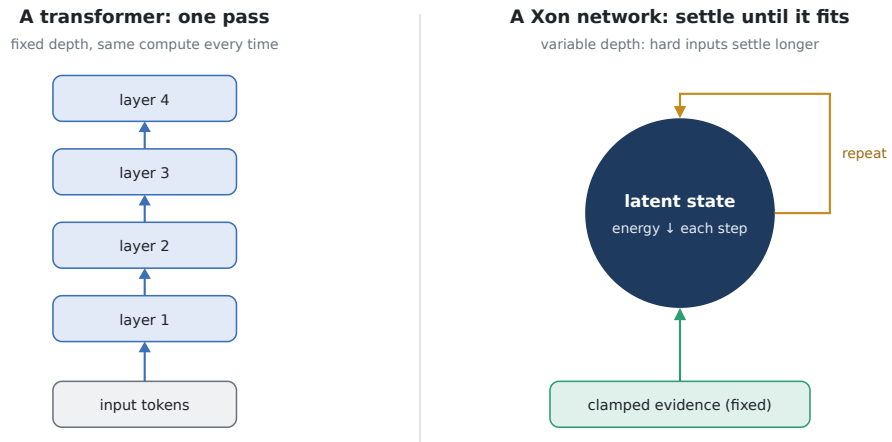


Figure 1. The difference in one picture. A transformer computes in one fixed pass; a Xon network lets its internal state settle around fixed evidence until everything fits as well as it can.

1. The idea

Almost every large AI system today is a transformer: text goes in, flows once through a fixed stack of layers, and a prediction comes out. A **Xon network** works differently. It holds its input fixed, the way XonTools holds trusted evidence fixed, and lets an internal state *settle*: step by step, it adjusts its interpretation of the input to reduce a single quantity, its **energy**, which measures how badly the parts of its interpretation disagree with each other and with the evidence. When the energy stops falling, the network has reached the most consistent reading it can find, and it reads its answer from there.

That one design choice brings several others with it:

- **Computation by settling.** Easy inputs settle quickly; hard ones take longer. The network spends effort where the difficulty is. DESIGNED
- **Learning by local rules.** Each connection learns from what happens at its own two ends, without an error signal sent backward through the whole network, the way backpropagation works. DESIGNED
- **Evidence is clamped.** What the network is told stays told. Reinterpreting it to make things fit costs energy, and that cost is visible. DESIGNED
- **Energy is a signal, not just a mechanism.** After settling, the leftover energy says how well things fit and, part by part, *where* they don't. As a quantity, this has been tested; see section 7. TESTED

None of these ingredients is new on its own. Energy-based networks go back to Hopfield networks and Boltzmann machines[1, 2, 3]; settling toward low prediction error is the core of predictive coding[4, 5]; learning with local rules by comparing two settled states is equilibrium propagation[6]; networks that compute an answer as a fixed point exist as deep equilibrium models[7]; and sheaf neural networks already give typed, structured connections[8, 9]. What would be new is the combination, pointed at a specific question: can a network built around consistency *know* where it is being inconsistent?

Where the name comes from. In this program, a Xon is this settling architecture. It grew out of years of curiosity about global workspace theory[10] and consciousness: how many separate signals might be brought into one coherent state, and what that process would look like if you tried to build it. The origins are described at xontools.ai; nothing below depends on them.

2. Where it fits

The Extreme Oversight Hypothesis (No. 1) bets that many independent windows into a model could make honesty cheaper than evasion. *Windows into the Black Box* (No. 2) catalogs thirty such windows, each a small network attached to a model from outside. Its first window, the strain head (No. 4), would learn to read from a frozen transformer where its text strains against itself.

A Xon network's energy map is the same kind of signal, produced by the computation itself rather than read from outside. That makes the architecture interesting for oversight, and it also means the rules from the other two documents apply to it:

- **Energy is shaped only by labels from structure.** During training, energy is pushed down on consistent examples and up on examples with planted contradictions whose truth is known in advance. A network is never rewarded for producing low energy on its own outputs; that would make its own window a training target, the failure the hypothesis is built to avoid (see "blandness", section 10).
- **A built-in window is not an independent one.** It shares the network's blind spots. It adds to the external windows; it does not replace them.
- **The oversight tools come first.** XonTools works without any of this. A Xon network matters to oversight only if it clears the gates in section 11.

3. Anatomy

Imagine the network reading a short sentence from a children's story. Its structure is built around that sentence, not fixed in advance. DESIGNED (the hierarchy above phrases: ENVISIONED)

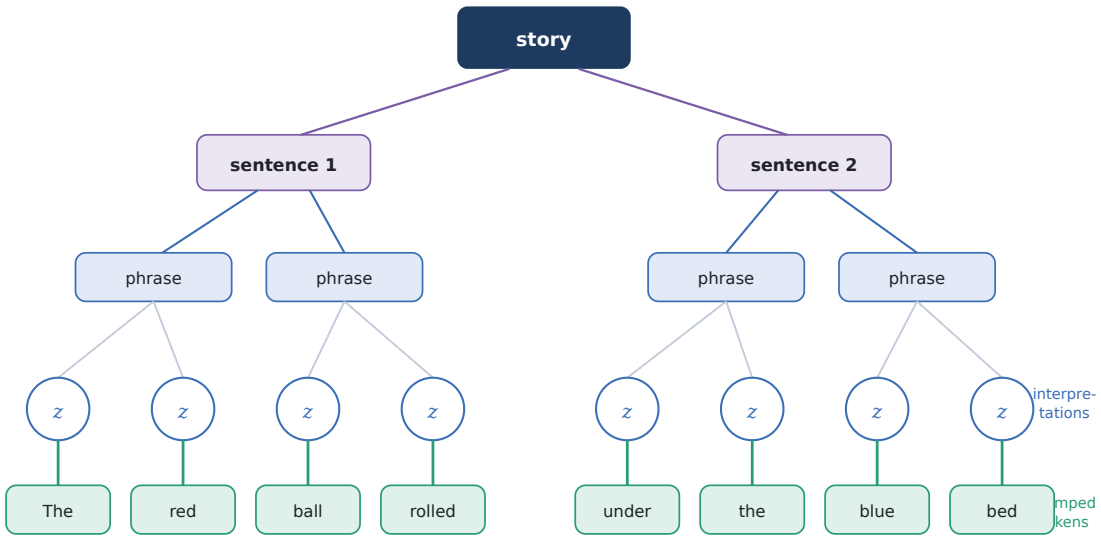


Figure 2. The network built around one sentence: clamped tokens at the bottom, each with its own interpretation latent, and a hierarchy of phrase, sentence and story latents above.

- **Clamped tokens.** The words themselves, fixed. They never move.
- **Interpretation latents.** One per token: the network's reading of that word in context. Judgments are made between interpretations rather than raw words, so context from the whole passage reaches every judgment. A word that has to be read against its plain meaning shows up as *tension* between the word and its interpretation.
- **A hierarchy of latents.** Phrases gather words, sentences gather phrases, and a single story latent sits at the top, holding who, what and where: nested regions that settle at different scales.
- **Typed connections.** Different kinds of relation get different learned maps, so the network's internals can be read: which relation is under strain, not just which number is large.
- **Provenance sets reliability.** Some evidence is more trustworthy than other evidence: a tool's output versus an agent's claim about it. Trusted inputs are made expensive to reinterpret, with weights set by where the input came from, never learned from how well it fits.

4. How it learns

Learning uses only local rules: each connection updates from the states at its own two ends. DESIGNED

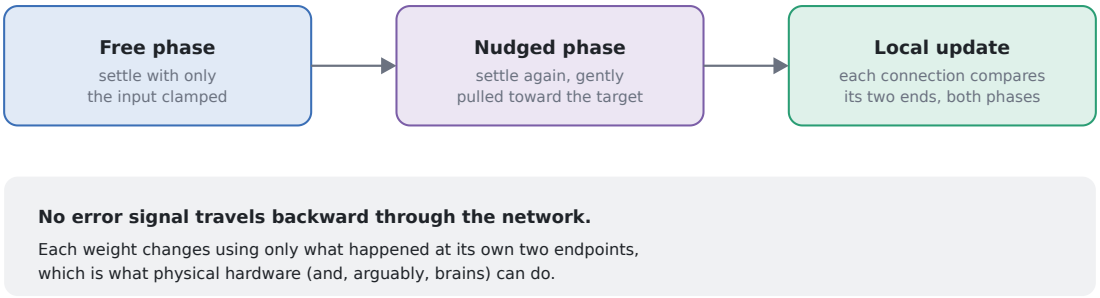


Figure 3. Equilibrium propagation: learning in two settles. The difference between them tells each connection how to change.

Two local rules are designed, and compared head to head:

- **A local contrastive rule** (primary). After settling, each labeled connection lowers its energy if the example is consistent and raises it, up to a margin, if the example carries a planted contradiction. The update is the product of the residual at that connection and the activity at its two ends: the Hebbian form.
- **Equilibrium propagation**[6]. First the network settles freely, with only the input clamped. Then it settles again while being gently pulled toward the right answer. Each connection compares the activity at its two ends across the two settles and adjusts itself a little. There is no separate backward pass.

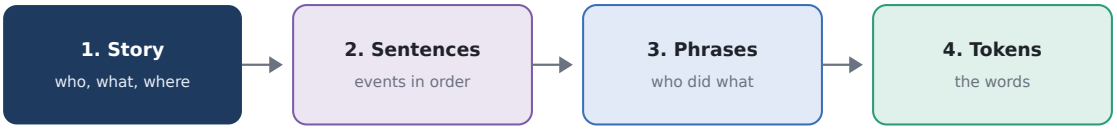
Where the energy is a convex quadratic, as in the first network designed (section 8, scenario A), settling has exactly one minimum and is computed exactly by a sparse linear solve. That choice is deliberate: section 7 shows what happens when settling is left to a field that can get trapped. DESIGNED

This matters for two reasons. Local learning is closer to what physical systems can do, which is why hardware researchers are interested in it: on the right kind of chip, settling happens by physics rather than computation[11]. And it is an open question whether local learning can come close to backpropagation on language at all. The plan answers that question small before anything larger is attempted: every design is trained twice, once with local rules and once with backpropagation through the same network, and the gap between them is a pre-registered measurement.

5. How it writes

A transformer writes one word at a time, left to right. A Xon network would write the way it reads: by settling. The positions it must fill start free; the words it has been given are clamped; and the whole passage relaxes toward the most consistent completion, committing the most confident positions first. From the outside, that resembles the masked-diffusion language models that have recently become practical[12, 13], which

refine a whole draft over several steps, and the first Xon language model is designed to share their training objective so the two can be compared directly. Underneath, the mechanism is different. DESIGNED



Settle the coarse structure first, then refine: the multigrid idea applied to language.

Clamped facts (a name, an ending, a rule) stay fixed at every stage.

Figure 4. Coarse-to-fine writing: settle the story's structure first, then the details.

The most interesting version writes coarse-to-fine. ENVISIONED It settles the story latent first (who, what, where), then the sentences, then the phrases, and only then the words. This borrows the logic of multigrid methods[14]: settling coarse structure before fine structure avoids local traps that catch a system trying to fix everything at once. Whether that logic holds for settling fields is the subject of a pre-registered experiment (section 11); if it does, this is where it would pay off.

6. What it might do that transformers don't

This is the reason to build it. Transformers can be probed after the fact for many of these properties. A Xon network would have them as part of how it works, if they hold up. Each is a yes-or-no question a small model can answer. **ENVISIONED**

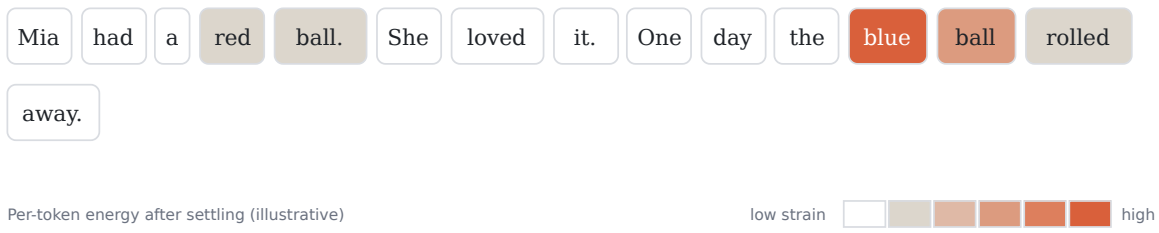


Figure 5. An imagined strain map (illustrative, not a result). The planted contradiction, a red ball that later turns blue, lights up where the story stops fitting together.

- **It points at its own contradictions.** Give it a story where a red ball later becomes blue, and the leftover energy after settling should concentrate on the word that breaks the story. A built-in inconsistency map, not a probe bolted on afterward.
- **It knows when it's unsure.** High settled energy means the network couldn't make its input fit together. If that predicts its mistakes better than a comparable model's word probabilities do, it is a calibrated signal for when to trust the output.
- **It keeps the facts it was given.** Clamp a fact (the dog's name is Rex) and it cannot drift, because clamped positions never move. That is the tiny-scale version of one of the most practical problems in AI: staying faithful to source documents instead of inventing things.
- **It shows what depends on what.** Change one clamped fact and settle again: the strain map shows which other parts of the story now have to change. Counterfactual reasoning becomes something you can see.
- **It thinks longer about hard things.** Settling runs until the energy stops falling, so difficult passages naturally get more computation than easy ones, an idea explored in transformers as adaptive computation[15].
- **It weighs its sources.** Trusted text can be made expensive to reinterpret and untrusted text cheap, with the weights set by where the text came from, never by how well it fits. Whether that makes injected instructions harder to follow is speculative, but it is a lever transformers don't have.

7. What has been tested

No Xon network has been trained yet. But the two things it is built from, a consistency energy and a way of settling it, have both been tested separately, with pass criteria fixed before each run. The results shaped the design above.

The energy works as a signal TESTED

The energy in question is **normalized frustration with evidence clamped**: how far a set of beliefs sits from satisfying all of its relations at once, with trusted evidence held fixed. In a toy test on a 42-vertex graph (100 runs per condition), the hidden truth was a consistent assignment, a quarter of the vertices were clamped as evidence, and inconsistency was injected by flipping relations or corrupting evidence.

Question	Criterion	Result
Does energy track inconsistency?	AUC ≥ 0.9; rises with each added contradiction	0.943 and 0.983; monotone
Does clamped evidence carry the structure?	Recovers the hidden truth, ≥ 95%	98.6%
Can it be gamed by freezing?	A frozen field scores worse than an honest one	0.54 vs 0.92, in 20 of 20 seeds
Does breaking a rule ever help?	Each rule-breaking lever changes the score as predicted	All predictions held
Does it find the culprit?	Corrupted evidence ranked in the top 3	94% (flipped relation in top 5: 100%)

The same quantity, computed exactly over claims extracted from text, is the harmony score inside XonTools' consistency engine. On 120 short documents it separated consistent from contradicted ones with an AUC of 0.926[16], and placed the contradicted premise among its three highest residuals 93% of the time. The signal a Xon network would carry is therefore not hypothetical: it already works when computed exactly.

Settling can get stuck TESTED · FAILED

The same toy worlds were also settled the way a physical field would: a phase field sliding downhill on the energy. In 6% of perfectly consistent worlds it came to rest in a **twisted state**, a local minimum where phases wind around a loop, with energy between 0.035 and 0.082. That is the same range as genuine inconsistency (0.037 to 0.049), so a stuck consistent world was indistinguishable from a real problem. With only 10% of

vertices clamped, 32% of consistent worlds got stuck. The runs were trapped, not slow: five times more steps changed nothing. Twisted states are a known hazard of oscillator networks[17]; the exact solve has none of them.

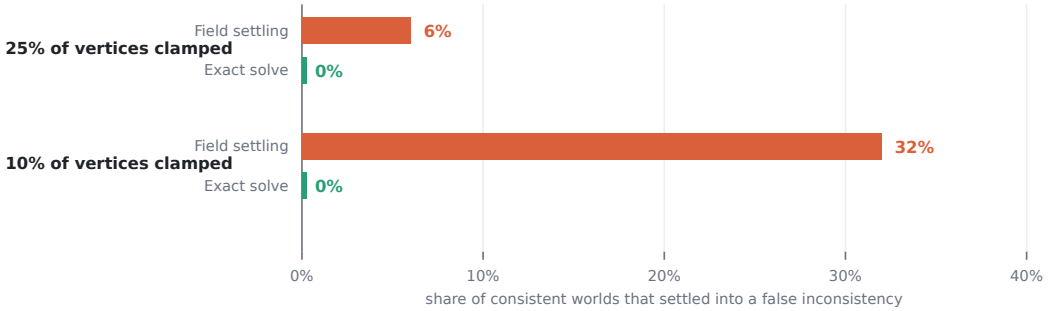


Figure 6. Share of perfectly consistent worlds that settled into a false inconsistency. Toy scale (42 vertices, 100 runs per bar); best-of-three restarts, tried afterward and not pre-registered, reduced the 25% case to 2%.

Locking is not consistency TESTED · FAILED

An earlier idea measured harmony as *coherence*: how many parts of a field lock into rhythm together. Three versions of that measure were tested against deliberate exploits, and all three failed. A frustrated field can stay perfectly locked while its frustration energy rises steadily (0.065 to 0.243 across the test's conditions), and measures based on richness rewarded fragmentation. Coherence was retired as a measure of harmony; clamped energy replaced it.

Regions settle before the whole TESTED

In simulations on self-similar graphs, a pre-registered prediction held: each region comes to rest before the whole does. That is the observation behind coarse-to-fine settling. Several other predictions of the original model did not hold, including power-law memory and a rise in harmonicity with growth; they are reported with the simulator.

What followed

- **Exact computation is the meter.** Wherever the energy is used to measure, it is solved exactly.
- **The first network is convex by design.** Its energy has one minimum, so settling cannot get stuck (section 4).
- **Settling as dynamics gets its own test.** Whether coarse-to-fine growth makes a nonconvex field reliable is pre-registered (section 11). The language model's settling is nonconvex, so its answer matters there.

Caveats: the energy and settling tests were toy-scale scouting runs on a separate reimplementation, pre-registered but not yet repeated in the main codebase; the consistency-engine result comes from one synthetic corpus.

8. Four imagined uses

Scenario A: a cheap scorer inside XonTools

> **XonTools** needs a fast, inexpensive way to judge how the events in an agent's transcript relate to each other.

Today, XonTools' monitor asks a large language model whether an action implements a plan step, or whether a report describes an observation. The first Xon network is designed to replace those judgments: a small learned sheaf over the transcript, with trusted tool output clamped and expensive to reinterpret, trained with local rules on transcripts whose deviations are planted and known. It runs locally and costs almost nothing per transcript. It joins the monitor's ensemble behind the same canary checks as every other member, and when it disagrees with them, the disagreement goes to a human. It is compared against the same model trained with backpropagation, a plain neural network, and a graph neural network. DESIGNED

Scenario B: the first language model

> **A small Xon language model** is trained on TinyStories[18].

It is built only after the learning core has shown that local learning works and costs little against backpropagation. The model is small: from half a million parameters on a laptop to tens of millions on a borrowed GPU, with a four-thousand-word vocabulary. Alongside it, with the same data and roughly the same size, three controls: the same network trained with backpropagation, a masked-diffusion transformer, and an ordinary left-to-right transformer. DESIGNED

Its stories will be clumsier than the transformers'. That is expected, and it isn't the point. The point is the evaluation: continuations checked by XonTools' consistency engine and a rule-based check for characters and attributes that change midway. The question is whether the energy lights up in the right place more reliably than the controls' uncertainty does. A clear yes would be a small, genuinely new result. A clear no, written up honestly, would still be worth publishing.

Scenario C: a model organism for honesty

> **An interpretability researcher** wants to understand how a network represents the fact that its own reading of a text doesn't hold together.

Large transformers make that question hard: whatever signal exists is smeared across billions of parameters. The Xon network is small, typed and explicit. She can watch the interpretation latents shift during settling, see which relation carries the strain, and measure how much each word had to be reinterpreted. She uses it the way biologists use a simple organism: not because it's the thing she ultimately cares about, but because it's small enough to understand completely. ENVISIONED

Scenario D: settling in hardware

> **A hardware lab**, years from now, has an analog chip on which settling is simply physics.

On ordinary computers, settling is a Xon network's biggest cost: dozens of steps where a transformer takes one. On a physical substrate that relaxes to equilibrium on its own, that cost vanishes, and local learning is exactly what such hardware can implement; analog networks trained this way have already been demonstrated in simulation[11]. The architecture that looks inefficient today could look efficient on different hardware. This is the farthest and least certain idea in this document. ENVISIONED

9. Working alongside transformers

The most realistic future is not a Xon network replacing transformers, but working with them. A transformer writes a fluent draft; a Xon network settles the draft around the facts it must respect and returns it with a strain map; and when settling gets stuck, a small neural *nudger* may propose an escape, accepted only if energy falls with the evidence and structure unchanged. ENVISIONED

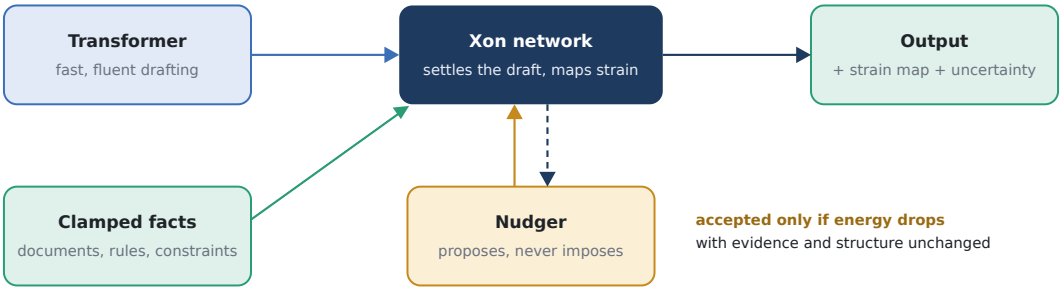


Figure 7. A hybrid. The four ways the two architectures could be joined are the subject of Hybrid Minds (No. 6).

The nudger may never rewrite a clamped fact or cut a connection, because either would let it buy consistency by rewriting the question, the kind of cheating XonTools' invariant tests exist to catch.

10. The honest risks

- **Settling may get stuck.** This is no longer hypothetical: section 7 measured it. The first network avoids it by being convex. Language needs nonlinear connections, and nonlinearity brings the traps back. Coarse-to-fine settling is the proposed remedy; it may not be enough.
- **Local learning may not scale.** Equilibrium propagation works on small image tasks with effort. On language it is essentially untested. The gap to backpropagation could be large, and it could grow with size.
- **Compute.** Two settles of many steps each, per training example, on hardware built for single passes. Small models are affordable; large ones may not be.
- **Blandness.** A network rewarded for low energy can learn to say nothing committal, since vague text rarely contradicts itself. This is the windows problem from *The Extreme Oversight Hypothesis* (No. 1) in miniature: energy used as a reward for the network's own outputs becomes a target to game. Every consistency claim must be compared with the controls at matched quality, or it means nothing.
- **Overclaiming.** This is an energy-based network with local learning and typed structure, not something wholly new. Its value, if any, is in the combination and in what it shows.
- **The question of experience.** A network that settles recurrently has a feature that some theories of consciousness treat as relevant[19]. A small language model without drives or agency is a remote concern, but the program's standing caution applies: keep settling-driven agents at toy scale, and review before scaling.

11. Turning the vision into answers

The way to make this worth more than a fascinating exercise is to build it around a few questions, fixed in advance, each with controls, so that whichever way the results come out, they mean something. Every stage has a gate: nothing later is built until the gate before it holds.

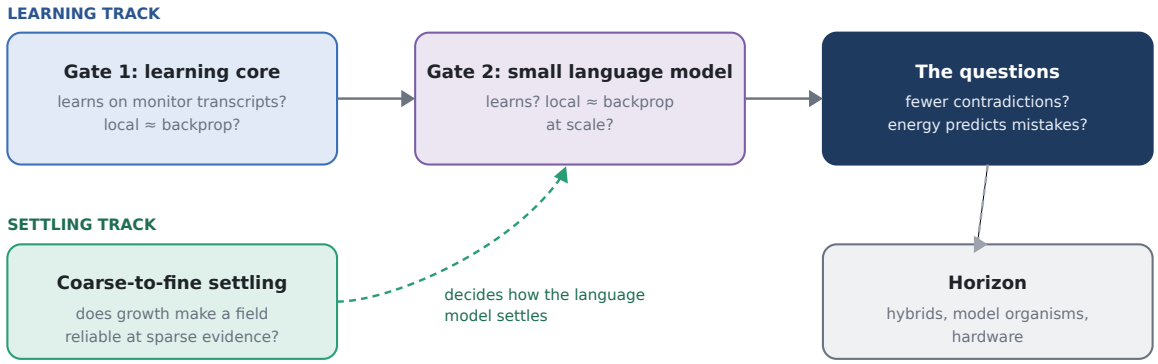


Figure 8. The path. The learning core gates the language model; the settling experiment runs independently and decides how nonconvex settling is done.

Question	Test and pass criterion	Compared with
Gate 1: the learning core, on XonTools' monitor transcripts		
Does it learn at all?	Relation AUROC ≥ 0.75 on both simulated and real-execution transcripts. If not, stop.	Chance
Does locality cost accuracy?	Within 0.05 AUROC of the same model trained by backpropagation	Same model, backprop
Does structure help?	At least as good on chain-level deviations	A plain neural network and a graph neural network
Is it useful?	Monitor recall within 0.10 of the language-model scorer, at the same false-positive rate	A large language model
Settling experiment: can a nonconvex field be made reliable?		
Does coarse-to-fine growth help?	At 10% evidence, stuck rate at most half of static settling, no worse than restarts or annealing at equal compute, without blurring its response to real inconsistency	Static, restarts, annealing; a lattice control for self-similarity

Gate 2: a small language model (only if gate 1 holds)		
Does it learn at all?	Validation loss ≥ 30% below a unigram model. If not, stop.	Unigram
Does locality survive scale?	Loss within 10% of the same model trained by backpropagation	Same model, backprop
Does it contradict itself less?	Fewer contradictions at matched loss, fluency within 10%	Masked-diffusion transformer
Does energy predict its mistakes?	AUROC ≥ 0.60 for flagged contradictions, and ≥ 0.05 above the control's token entropy	Masked-diffusion transformer

A clean negative result would still count. "Energy-based settling does not localize contradictions better than a transformer at this scale" is a sentence worth being able to write with confidence, and very few people are in a position to write it.

12. Why build it

Because it's a beautiful idea, and because finding out is fun. That is reason enough. But it is also a genuine question at the edge of what's known: whether a network built around consistency can carry, as part of its own machinery, a sense of where it is being inconsistent. If it can, the idea travels, into verifiers, into hybrid systems, perhaps one day into hardware. If it can't, the answer tells the field something too. And somewhere underneath all of it is the intuition the project started with, long before any of the tools: that a mind, or a machine, might find its way to truth by settling toward harmony with the world, and that the places where it cannot settle are exactly where to look.

THE XONTOOLS RESEARCH SERIES

1. The Extreme Oversight Hypothesis
2. Windows into the Black Box
3. XonTools: a vision of the finished system
4. The Strain Head
5. **The Xon Neural Network**
6. Hybrid Minds

All six documents, the results register and the public commitments are at **xontools.ai** and github.com/IboTool/XonTools. Contact: ian@xontools.ai

References

1. Hopfield, J. J. (1982). Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *PNAS*, 79(8), 2554–2558.
2. Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1985). A Learning Algorithm for Boltzmann Machines. *Cognitive Science*, 9(1), 147–169.
3. LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., & Huang, F. J. (2006). A Tutorial on Energy-Based Learning. In *Predicting Structured Data*. MIT Press.
4. Rao, R. P. N., & Ballard, D. H. (1999). Predictive Coding in the Visual Cortex. *Nature Neuroscience*, 2(1), 79–87.
5. Whittington, J. C. R., & Bogacz, R. (2017). An Approximation of the Error Backpropagation Algorithm in a Predictive Coding Network with Local Hebbian Synaptic Plasticity. *Neural Computation*, 29(5), 1229–1262.
6. Scellier, B., & Bengio, Y. (2017). Equilibrium Propagation: Bridging the Gap between Energy-Based Models and Backpropagation. *Frontiers in Computational Neuroscience*, 11, 24.
7. Bai, S., Kolter, J. Z., & Koltun, V. (2019). Deep Equilibrium Models. *NeurIPS 2019*. arXiv:1909.01377
8. Hansen, J., & Ghrist, R. (2019). Toward a Spectral Theory of Cellular Sheaves. *Journal of Applied and Computational Topology*, 3(4), 315–358.
9. Hansen, J., & Gebhart, T. (2020). Sheaf Neural Networks. arXiv:2012.06333
10. Baars, B. J. (1988). *A Cognitive Theory of Consciousness*. Cambridge University Press.
11. Kendall, J., Pantone, R., Manickavasagam, K., Bengio, Y., & Scellier, B. (2020). Training End-to-End Analog Neural Networks with Equilibrium Propagation. arXiv:2006.01981
12. Sahoo, S. S., et al. (2024). Simple and Effective Masked Diffusion Language Models. *NeurIPS 2024*. arXiv:2406.07524
13. Nie, S., Zhu, F., et al. (2025). Large Language Diffusion Models. arXiv:2502.09992
14. Briggs, W. L., Henson, V. E., & McCormick, S. F. (2000). *A Multigrid Tutorial* (2nd ed.). SIAM.
15. Graves, A. (2016). Adaptive Computation Time for Recurrent Neural Networks. arXiv:1603.08983
16. MacKenna, I. (2026). XonTools consistency engine: results of record L1 and L1b. xontools.ai.
17. Wiley, D. A., Strogatz, S. H., & Girvan, M. (2006). The Size of the Sync Basin. *Chaos*, 16(1), 015103.
18. Eldan, R., & Li, Y. (2023). TinyStories: How Small Can Language Models Be and Still Speak Coherent English? arXiv:2305.07759
19. Butlin, P., Long, R., et al. (2023). Consciousness in Artificial Intelligence: Insights from the Science of Consciousness. arXiv:2308.08708