Cellular Automaton Rule 110 as a Mustard Seed: Minimal Rules, Maximal Emergence

Douglas C. Youvan

doug@youvan.com

www.youvan.ai

October 12, 2025

What is the smallest rule that can still produce a world of structure? Rule 110—a one-dimensional cellular automaton with an eight-entry lookup table—answers this with startling force. From the tiniest possible start (a single lit cell), its step-by-step recursion gives rise to an ether-like background, traveling "particles," stable lanes of transport, and phase-sensitive collisions that together support signaling, memory, and gates. In this paper we treat Rule 110 as an AI mustard seed: a minimal kernel that, when iterated, yields behavior wildly disproportionate to its size. The analogy clarifies modern AI's foundations too: gradient descent, cross-entropy, and attention are similarly compact rules whose composition across depth, width, and time blossoms into circuits and capabilities. We make the thesis operational. First, we ground the substrate (rule table, seeds, boundaries) and visualize its unfolding (Figs. 1–2). We then extract particle trajectories and collision zones (Fig. 3), quantify information growth (Fig. 4), situate Rule 110 among neighboring regimes (Fig. 5), and expose hidden regularities in the spectrum (Fig. 6). Finally, we propose criteria for "mustard-seed sufficiency," a replication kit, and connections to neural circuits. Our claim: with the right small rule and disciplined iteration, emergence becomes an engineerable method—not a mystery.

Keywords: Rule 110, cellular automata, emergent computation, computational universality, edge of chaos, ether, gliders, collisions, signal algebra, entropy, MDL, spectral analysis, symbolic dynamics, grokking, priors, attention, mustard seed.

I. The Mustard Seed Thesis — motivation; parable → program; roadmap

Motivation: when almost nothing becomes enough

Our central question is stark: *What is the smallest rule that still produces a world of structure you can compute with?* By "smallest," we mean low description length—a tiny update law and a minimal seed (down to a single bit). By "world of structure," we require more than crystal or noise: we need transport channels, reusable motifs, and phase-sensitive interactions that compose into algorithms. Rule 110 meets this bar. Its eight local cases, iterated, yield an ether background, traveling carriers with stable velocities, and collisions that implement signaling, memory, and gates. The guiding claim is that organized complexity is typically a product of tiny priors under disciplined iteration, not ornate design.

From parable to program: faith ≙ prior; cultivation ≙ recursion

The mustard seed names disproportionate yield: minute beginnings, abundant ends. Operationally, *faith* corresponds to a prior—a bias strong enough to orient dynamics before evidence is plentiful (the fixed update table in CA; the objective/architecture in ML). *Cultivation* is recursion: stepwise local updates in cellular automata; optimization steps and depth in neural networks. Under repeated application, priors select motifs that persist, amplify them through transport, and compose them via interactions. Emergence, then, is the natural outcome of *minimal bias × repeated local propagation*.

Roadmap (and figure order of first citation)

We make the thesis concrete, measurable, and reproducible across six visuals:

- Section II: Fix the substrate—update table, encoding, boundaries—and show unfolding from two seeds. Fig. 1 (single-bit seed), Fig. 2 (random seed).
- Section III: Develop a signal view—taxonomy of gliders, collision calculus, and automated track extraction. Fig. 3 (trajectory & collision overlay).
- Section V: Quantify emergence with information measures; show sterile → fecund → equilibrated dynamics. Fig. 4 (row-entropy curve).
- Section VI: Situate Rule 110 among neighboring regimes and discuss "seed potency." Fig. 5 (comparative patterns: 30, 90, 110, 184).
- Section IX: Reveal hidden regularities via spectra; ridges encode velocity classes, harmonics encode ether. Fig. 6 (2D FFT spectral signature).

Subsequent sections connect this seed calculus to modern AI (tiny training rules as seeds), propose criteria for mustard-seed sufficiency (minimality, fertility, composability, scalability), supply a replication kit, and close with limitations and open problems. The working method is simple: choose a small, truthful rule; cultivate by iteration; measure; then scale without losing interpretability.

II. Rule 110 in One Page — update table, encoding, boundaries; single-bit vs random seeds, visual phenomenology

Update rule & encodings (the entire "law" in eight cases)

Rule 110 is a binary, radius-1 cellular automaton: the next state of a cell depends on its left (L), center (C), and right (R) neighbors.

Neighborhood: 111 110 101 100 011 010 001 000

Next state: 0 1 1 0 1 1 1 0

Two equivalent encodings are handy:

- Wolfram code: read the "next" row as bits for indices 7→0: $01101110_2 = 110_{10}$.
- Index lookup: t = (L<<2) | (C<<1) | R ∈ {0..7} and next = [0,1,1,1,0,1,1,0][t].

This eight-entry table is the "mustard seed"—the minimal bias we iterate.

Boundary conditions (how edges behave)

- Wrap (circular/torus): the last cell neighbors the first. Pros: no absorption at borders; gliders can circumnavigate; statistics are stable. We use wrap for Figs. 1–2.
- Fixed (absorbing/zeros): edges pin or extinguish activity. Useful for finite cones but can distort collision frequencies near borders.

Initial seeds (what we plant)

- Single-bit seed: one "1" amid zeros. Yields the cleanest causal cone and canonical motifs; used in Fig. 1.
- Random seed: Bernoulli(½) row. Stresses robustness; shows whether order reappears from disorder; used in Fig. 2.

Visual phenomenology (what to look for in the rasters)

- Causal cones: From a single 1, activity expands with slopes ±1, bounding all influence (triangular light-cone in Fig. 1).

- Ether (background lattice): a recurring space–time fabric forms; it provides lanes on which structures travel.
- Gliders (mobile localizations): compact motifs persist and move with characteristic velocities (visible as diagonal tracks).
- Collisions: when gliders meet, outcomes include annihilation, scattering, and transmutation—atomic "gates" for signal logic.
- Domains & defects: phase-shifted ether regions meet along lines that steer or trap gliders; recurring in both Fig. 1 and Fig. 2.
- Interference banding: periodic emissions and repeated collisions create striped/chevron textures—signatures of timing structure.

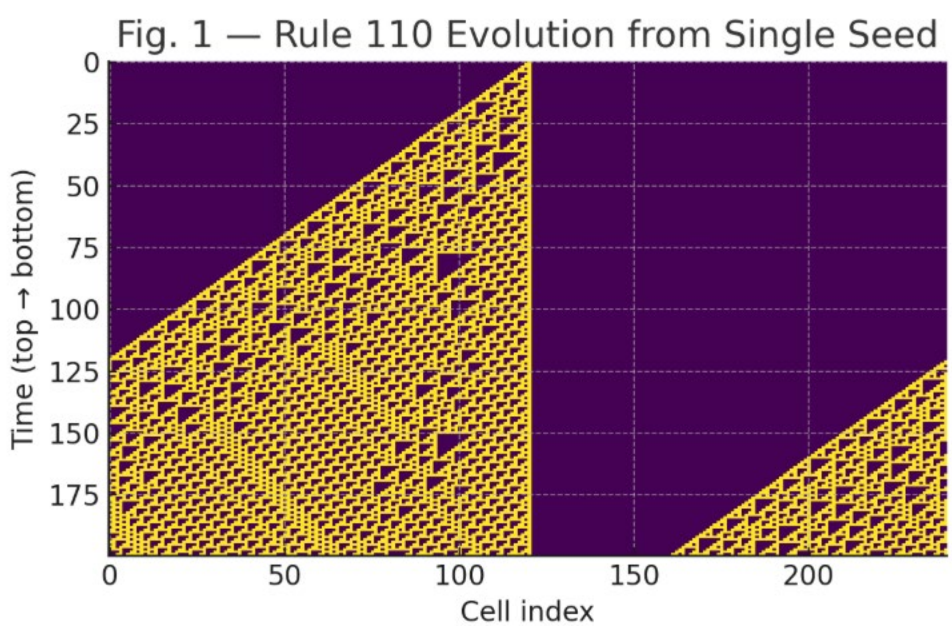

Fig. 1 — Single-seed evolution (wrap): The lone bit blossoms into ether + glider traffic inside the causal cone. Use this to measure slope classes (glider velocities) and to spot early collision types.

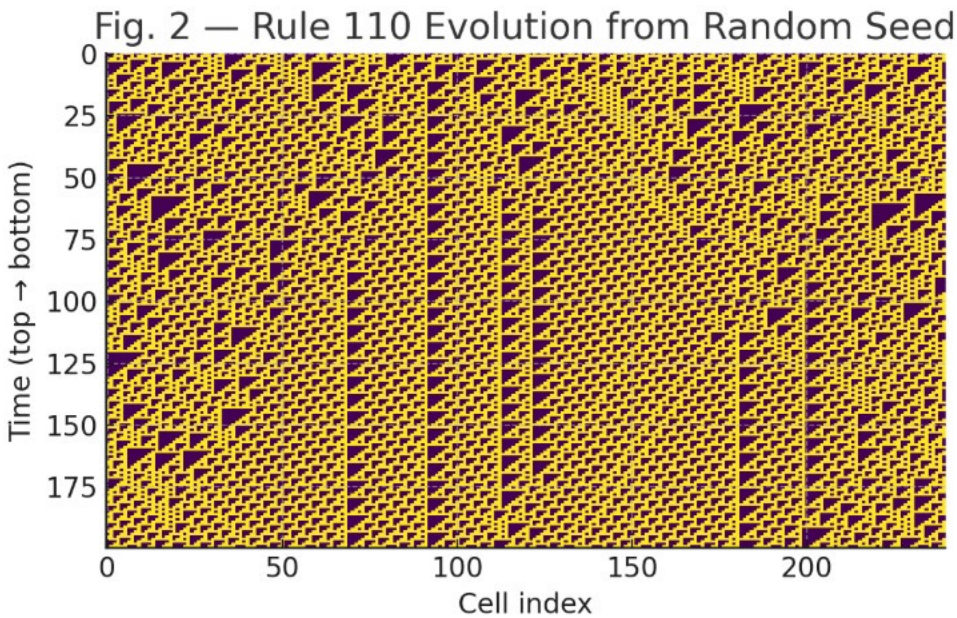

Fig. 2 — Random-seed evolution (wrap): Despite maximal initial entropy, the same ether and glider motifs re-emerge, evidencing fertility (diverse pattern production) and stability (persistence under noise).

Takeaway: A rule that fits on one line, plus a 1-bit seed, is enough to generate transport, interaction, and structure—raw materials for the later sections on particles (Sec. III), information growth (Sec. V), and comparative regimes (Sec. VI).

III. Particles, Collisions, and Long-Range Order — glider taxonomy, velocities, lifetimes; collision calculus; automated trajectory extraction

A) Glider taxonomy: footprints, velocities, lifetimes

Within Rule 110's ether, mobile localizations ("gliders") are compact motifs that persist and translate diagonally. We classify them by three observable axes:

- Footprint $(w \times h)$: minimal bounding box of the motif over one temporal period h. Typical $w \in [3,8]$, $h \in [2,14]$.
- Velocity $v = \Delta x / \Delta t$: measured as the slope of a track in the space–time raster (pixels per timestep). Distinct velocity classes recur (e.g., $v \approx -\frac{1}{2}, -\frac{1}{3}, +\frac{1}{3}$, plus near-stationary defects).
- Lifetime τ: number of steps the motif persists in ambient ether; many are indefinitely stable absent collisions, others have heavy-tailed $P(\tau)$under random traffic.

Roles.

- *Couriers:* fast, small footprints; carry a bit via presence/absence or phase.
- *Transcoders:* convert local ether phase, enabling interface between domains.
- *Anchors/reflectors:* quasi-stationary defects that store state and redirect traffic.
- *Emitters:* periodic sources that act as clocks and pulse trains.

B) Collision calculus: transport, logic, and composition

When tracks intersect, local superposition yields a transient composite with one of a few reproducible outcomes:

- Elastic scattering: $A + B \to A' + B'$. Paths bend or phases reset; timing implements delays and routing.
- Transmutation: $A + B \to C$. Realizes logic when C's identity encodes a conditional result.
- Annihilation: $A + B \to \emptyset$. Implements erasure or AND-with-NOT behavior.
- Emission: $A + B \to C +$ stream$\{D\}$. Produces periodic emitters (clocks/counters).
- Pin/unpin: interaction with an anchor toggles a gate state for later signals.

Phase matters. Ether has a temporal period T_{ether}. For each ordered pair (A, B), only certain phase offsets $\Delta\phi \in \{0, \dots, T_{\text{ether}} - 1\}$yield a given outcome. We record valid windows $W_{AB} \subseteq \Delta\phi$and robustness (tolerance to ±1 phase tick or small noise).

C) Automated trajectory extraction (method sketch behind Fig. 3)

To move from qualitative sight-reading to measurements, we extract tracks from rasters:

1. Preprocess. Optionally detrend row means; apply a small median filter to stabilize the ether.
2. Orientation scoring. Convolve the raster with small diagonal kernels (e.g., 3×3masks at ±45°) to highlight line-like persistence; sum responses into a motion intensity map $M(x, t)$.
3. Threshold & labeling. Threshold $M$and run connected-components in $(x, t)$to get candidate track blobs.
4. Line fitting. For each blob, fit $x \approx a\,t + b$via robust regression (RANSAC) to estimate velocity $v = a$, intercept b, length L. Prune short or low-fit tracks.

5. Template matching (optional). Extract patches along the fitted line; compare against a small library of glider templates across phase using Hamming/correlation distance to label type and phase.

6. Collision inference. Intersections of fitted lines within a spatiotemporal tolerance $(\Delta t, \Delta x)$define collision candidates. Classify outcomes by comparing pre- and post-labels (elastic, transmute, annihilate, emit); aggregate into a collision table with phase windows and empirical success rates.

Outputs.

- Velocity spectrum: histogram of v(supports the slope classes visible in Fig. 3).
- Lifetime statistics: distribution of track lengths L(proxy for τ).
- Collision atlas: counts and phase windows W_{AB}; basis for a collision grammar (Sec. IX's symbolic dynamics).

D) Reading Fig. 3 (trajectory & collision overlay)

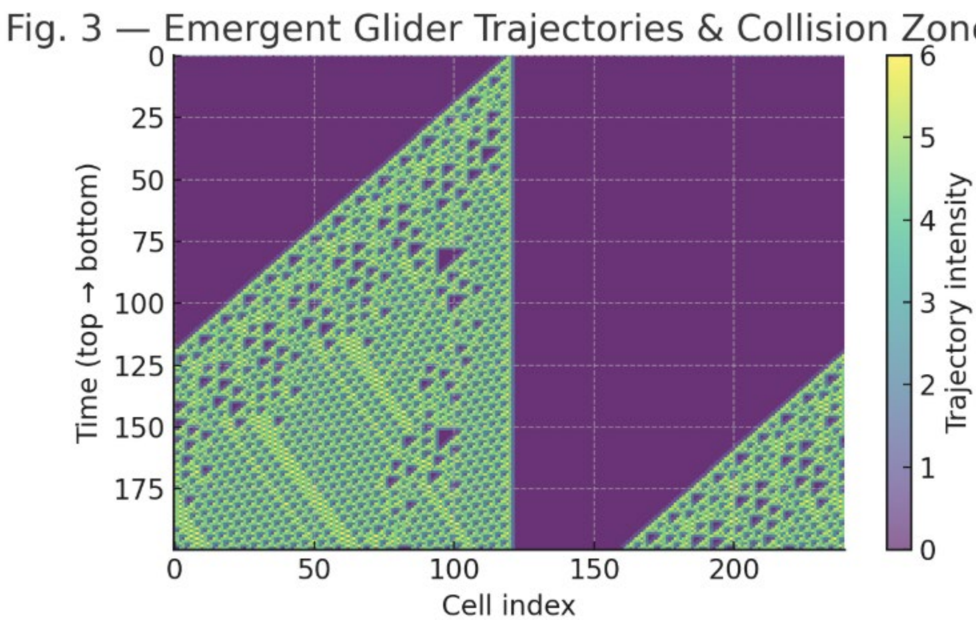

Fig. 3 overlays the single-seed raster (grayscale) with the motion-intensity field (warm tint). Bright diagonals trace glider trajectories (distinct slopes = velocity classes). Hot knots at line crossings mark collision loci. Together they make long-range order visible: stable transport lanes, reproducible interactions, and localized "compute events" emerging from an eight-case rule.

IV. Universality from Almost Nothing — Cook/Wolfram sketch; signaling, memory, gates; practicality vs undecidability

A) The Cook/Wolfram sketch: compiling a tag system into gliders

Rule 110's universality hinges on the observation that its mobile localizations and defects can emulate the primitives of a cyclic tag system (CTS), a known universal model. The construction proceeds in three moves:

1. Alphabet ↔ carrier types. Choose distinct glider species (and/or phase-shifted variants) to encode symbols on a "tape." Presence/absence, spacing, or phase plays the role of symbol value.
2. Rule application ↔ choreographed collisions. The CTS delete/append rule is realized by a recurrent gadget: a timed packet of carriers encounters a sequence of anchors/emitters that (i) consume the leftmost symbol (delete), (ii) conditionally spawn new packets (append), and (iii) reconstitute the control head for the next step.
3. Clocking & transport. Periodic emitters provide a global rhythm that keeps phase alignments valid; lanes ensure packets arrive at gadgets with bounded jitter.

Because CTSs can simulate Turing machines, the reduction "CTS → Rule 110 glider algebra" establishes Turing completeness. The proof is constructive: one can, in principle, lay out an initial seed that runs an arbitrary CTS-with-input within Rule 110's space–time.

B) Signaling, memory, and gates in a nearest-neighbor medium

Universality requires three ingredients that Rule 110 provides:

- Signaling (wires). Distinct gliders travel with stable velocities along ether lanes. Bits can be encoded in packet presence, phase, or inter-packet spacing. Fan-out is achieved via splitter collisions that duplicate a packet into two lanes.
- Memory (buffers/registers). Quasi-stationary defects and periodic emitters store phase/configuration until perturbed—effectively one-bit cells. Toggle collisions implement write/erase; readout occurs when a probe packet scatters with a type/phase that reveals state.
- Gates (local logic). Collisions implement a small basis:
    - Annihilation: $A + B \to \emptyset$ (erasure / AND-with-NOT).
    - Conditional transmutation: $A + B \to C$ (data-dependent rewrite).

- Elastic scattering: A + B → A′ + B′ (delay/routing).
  Compositions of these yield branching, counters, comparators, and the CTS gadgets above.

C) Practical programmability vs. the undecidability wall

Universality is a ceiling on expressivity, not a guarantee of convenient engineering. Several frictions arise:

- Phase brittleness. Valid gates occupy discrete phase windows; tiny timing errors can derail a program. Practical designs therefore include clocks, phase lockers, and guard rails (domain walls) that re-center packets before critical collisions.
- Routing and fan-out. Crossing lanes without interference, duplicating packets, and re-merging streams impose latency and space overhead. Libraries of delay lines, splitters, mergers, and crossovers are essential, and their composition must control jitter.
- Compilation overhead. Mapping high-level logic to glider choreography incurs polynomial—but often large—blowups in width and time. For demonstrations, macro-seeds that rapidly "unfold" factories (emitters + lanes) can amortize setup costs.
- Noise, boundaries, and drift. Fixed edges absorb/reflect traffic; stochastic flips accumulate phase error. Robust layouts use wrapped domains or engineered gutters plus scrubbing pulses to periodically reset local phase.
- Undecidability. With universality comes the halting barrier: there is no general procedure to decide whether a given seed halts, repeats, or runs forever. Verification relies on local proofs (certified gadgets, invariants) and empirical testing rather than global guarantees.

D) Working stance

Rule 110 proves that an eight-case rule can host general computation; engineering in it means constraining freedom: pick a small, reliable gadget basis; discipline timing; and standardize interfaces. In practice we operate within a witnessable sublanguage—rich enough for nontrivial programs, narrow enough to test and verify—treating universality as a north star, not a day-to-day dependency.

V. Information-Theoretic View — MDL decomposition; per-row Shannon entropy; complexity growth and equilibration

A) MDL decomposition: rule, seed, grammar, residual

We treat the space–time raster $X_{1:T}$ as data to be compressed by a model M. Under Minimum Description Length (MDL),

$$L(X_{1:T}, M) = L(\text{rule}) + L(\text{seed}) + L(\mathcal{G}) + L(\text{residual} \mid \mathcal{G}).$$

- Rule cost $L(\text{rule})$: bits to specify the eight-case update (tiny and fixed).
- Seed cost $L(\text{seed})$: from $\lceil \log_2 W \rceil$ for a single-bit seed to W for a random row of width W.
- Grammar cost $L(\mathcal{G})$: dictionary of ether phases, glider templates (with velocities/periods), and phase-legal collision rules.
- Residual cost $L(\text{residual} \mid \mathcal{G})$: what remains after encoding with the grammar (transients, debris, misphased events).

Interpretation. Emergence is witnessed when a small grammar $\mathcal{G}$ yields a large reduction in residual bits: the rule and seed are minuscule, yet they unlock reusable structure that compresses the evolution substantially.

B) Per-row Shannon entropy: a simple thermometer

For row t, let p_t be the fraction of 1s; the row entropy

$$H_t = -p_t \log_2 p_t - (1 - p_t) \log_2 (1 - p_t)$$

ignores correlations but is an informative first-order measure:

- Sterile birth: small cone with sparse ones → low H_t.
- Fecund elaboration: ether crystallizes, gliders proliferate → H_t rises and often oscillates as periodicities compete.
- Structured equilibrium: sustained lanes + collisions → H_t plateaus strictly below the maximum (not noise) and above near-zero (not crystal).

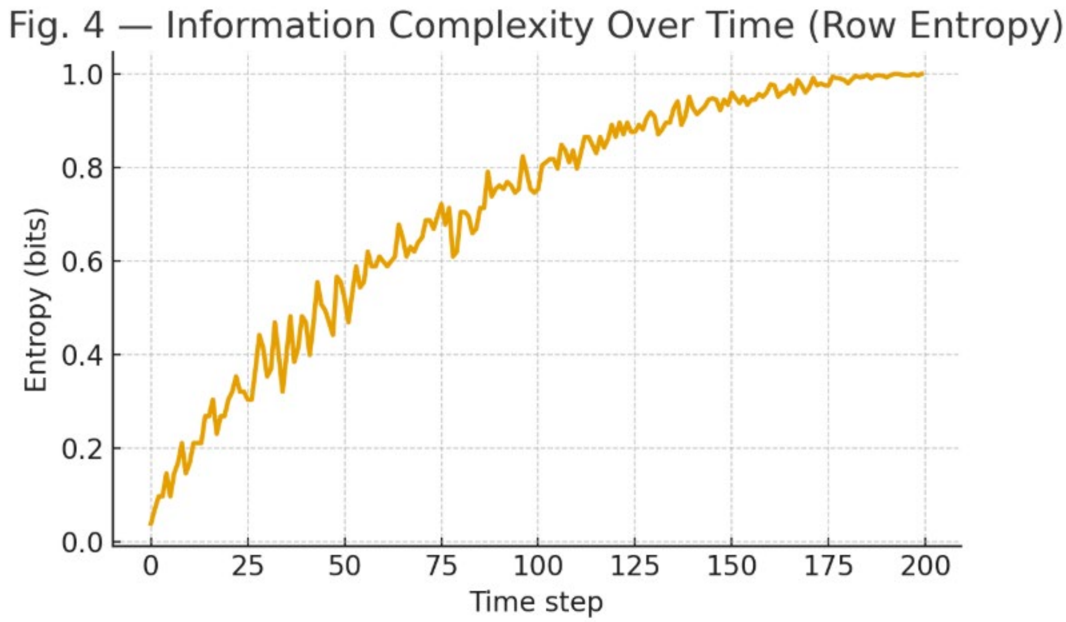

Fig. 4 plots H_t for the single-seed run: an initial ramp, a choppy middle (pattern formation), and a tempered plateau—our quantitative signature of "order with ongoing activity."

C) Complexity growth and equilibration (beyond row entropy)

To separate structure from mere activity, we extend the analysis:

- Block entropies $H_t^{(k)}$ and entropy rate h_μ: randomness per site once k-length dependencies are accounted for.
- Excess entropy E: stored predictive information (area above kh_μ); increases as lanes and glider grammars stabilize.
- Temporal MI $I(X_t; X_{t+\Delta})$: persistence of carriers across time; reveals transport memory.
- Spatial MI $I(X_{t,i}; X_{t,i+d})$: periodic ether spacing and domain structure.

Expected pattern. As motifs settle, $h_\mu > 0$ (collisions keep injecting novelty) while E grows (long-range constraints). MDL mirrors this: $L(\mathcal{G})$ rises modestly, $L(\text{residual} \mid \mathcal{G})$ drops sharply, and the total codelength curve exhibits sterile → fecund → equilibrated phases aligned with Fig. 4.

D) Operational criteria (tying back to the seed thesis)

A credible "mustard seed" substrate should show, over reasonable windows:

1. MDL leverage: small $\mathcal{G}$ delivers large residual reduction.
2. Entropy behavior: H_t ramps, oscillates during organization, then plateaus.
3. Structure metrics: rising excess entropy E with nonzero h_μ.

These curves make emergence measurable and reproducible, converting a parable ("from a tiny seed…") into a quantitative method.

VI. Comparative Substrates & Phase Regimes — where 110 sits (Class III/IV boundary); contrasts with Rules 30, 90, 184; implications for seed potency

A) Locating Rule 110 at the Class III/IV boundary

Elementary cellular automata are often grouped by qualitative long-run behavior:

- Class II (periodic/crystalline): simple repeats and fixed domains.
- Class III (chaotic): high apparent randomness; signals diffuse.
- Class IV (complex): coexistence of order and disorder; mobile localizations ("gliders") and long transients.

Rule 110 sits on the edge between III and IV: its ether provides periodic scaffolding (Class II flavor), while persistent carriers and phase-sensitive collisions generate sustained novelty (Class III flavor). This boundary region is where programmable richness is most available: enough stability to carry symbols, enough instability to compute with their interactions.

B) Contrasts with exemplar rules (Fig. 5)

- Rule 30 — Chaotic diffuser.
  Rapid decorrelation; no stable carriers or lanes. Good pseudo-randomness, poor addressability: signals smear, gates are brittle.
- Rule 90 — Linear/fractal (XOR).
  Sierpiński patterns; fully analyzable via linear algebra. Excellent compressibility, weak expressivity: collisions superpose rather than transmute, so logic is impoverished.
- Rule 184 — Traffic/flow.
  Interpretable as asymmetric exclusion; conserved "car" flux, shocks, and rarefactions.

Strong routing and hydrodynamic intuition, limited particle vocabulary; general logic requires awkward encodings.

- Rule 110 — Complex/computational.
  Mixed ether + diverse gliders at multiple velocities; phase-robust collisions supporting routing, memory, and conditional rewrite. Demonstrated path to universality via tag-system compilation.

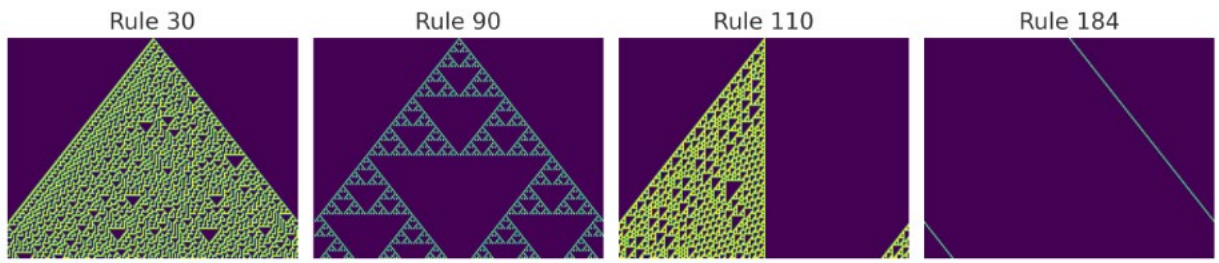

Fig. 5 - Differences are visible at a glance: 30's turbulent spray, 90's crystalline triangle lattice, 184's lane-like jams and gaps, and 110's textured fabric of carriers moving through an ether.

C) Implications for "seed potency"

We evaluate a substrate's suitability as a mustard seed along four axes:

1. Minimality (rule/seed size). All four rules are tiny; potency depends on what that small bias affords.
2. Fertility (motif diversity). 110 $\gg$ 184 $\gg$ 90 $\gg$ 30: 110 yields many carrier types; 184 yields a few flow motifs; 90 yields a single linear grammar; 30 yields noise-like debris.
3. Composability (signal logic). 110 supports a reusable collision grammar; 184 supports routing/delays but weak transmutation; 90 is linear (no nonlinear gates); 30 lacks stable interfaces.
4. Scalability (toward universality). 110 scales with clocks/anchors and admits a universality proof; 184 scales for transport but struggles with rich logic; 90 underpowered despite clean scaling; 30's chaos defeats large-scale assembly.

Design takeaway. For seed-first engineering, choose substrates near Class IV: enough order to stabilize symbols and enough instability to compute by collisions. Rule 110 exemplifies this sweet spot, making it the canonical mustard-seed substrate for programmable emergence.

VII. Parallels to Modern AI — tiny training rules as seeds (GD/XE/attention); grokking vs regime shift; circuits/features as particles; collisions as representation updates

A) Tiny training rules as seeds

Modern systems grow from a handful of local, iterative rules—gradient descent (GD), cross-entropy (XE), and attention—whose description length is tiny relative to the behaviors they yield.

- GD (local update): like the CA's neighborhood rule, GD only "sees" nearby information (current params + batch) yet compounds across steps into global organization.
- XE (belief kernel): next-token (or label) prediction is a strong prior that favors compressible structure, analogous to Rule 110's fixed lookup.
- Attention (interaction law): a compact equation for routing information among tokens; stacked layers/time are the NN analogue of iterating the CA rule.

These three—{GD, XE, attention}—play the role of the seed rule; depth/steps are the cultivation.

B) Grokking as a regime shift

The grokking phenomenon (late, sudden generalization after overfit) mirrors the CA's sterile→fecund→equilibrated arc.

- Early: memorization dominates; like the narrow cone in Fig. 1, entropy is low but rising.
- Transition: weight decay/pressure for short descriptions reorganizes representations—akin to ether crystallization; transport channels appear.
- Late: features/circuits support generalization; the NN's "lanes" stabilize, echoing the persistent tracks and collisions highlighted in Fig. 3.

Information-theoretically, model "grammar" (circuits) grows while residual unpredictability drops—precisely the MDL leverage seen in the CA.

C) Circuits/features as particles; lanes as routing

Mechanistic interpretability finds circuits (e.g., induction heads) that propagate information with consistent offsets. Map CA → NN:

- Ether ≙ representational backdrop: stable subspaces/residual streams act as the lattice that supports transport.

- Gliders ≙ persistent features/circuits: e.g., an induction head that copies a token forward at a fixed stride behaves like a particle with a characteristic velocity (slope in attention patterns).
- Lanes ≙ attention routes: positional and content-based heads open channels—previous-token, matching-delimiter, rhyme/metre—that resemble the CA's conduits.

D) Collisions as representation updates

When multiple heads write to the same subspace, or an MLP gates a routed feature, their interference is analogous to CA collisions:

- Elastic composition: two heads write without overwriting (additive, phase-preserving) → delay/routing.
- Transmutation: a head+MLP chain changes feature identity (e.g., name→pronoun) → conditional rewrite.
- Annihilation: layernorm/orthogonalization or gate suppression cancels a feature → erasure.

Phase (token position, timing across layers) is crucial; small misalignment flips outcomes, just as CA collisions are phase-windowed.

E) Operational lessons (design & diagnostics)

- Design for lanes: preserve clean subspaces and reduce write conflicts (orthogonality, sparse routing) so particles/circuits can travel.
- Encourage short descriptions: regularization that favors low-rank, low-MDL circuits (weight decay, sparsity) promotes the regime shift to generalization.
- Trace trajectories: analyze activation paths across layers/tokens as we do CA tracks in Fig. 3; build a collision grammar of which heads/MLPs compose which functions.
- Watch phase: instrumentation for timing/position (KV-cache probes, attention-pattern slope histograms) detects fragile or misaligned interactions early.

Bottom line. In both substrates, small rules + recursion yield transport + interaction that compose into computation. The CA's visible gliders and collisions provide a concrete scaffold for reasoning about otherwise invisible NN circuits—and a checklist for building seed-first, interpretable systems.

XIII. Conclusion — seed → recursion → emergence → universality; recap Figs. 1–6

From almost nothing to enough.
We asked whether a rule small enough to fit on a line—and a seed as small as a single bit—can produce structure you can *compute with*. Rule 110 answers yes. The pattern is consistent across our analyses: seed → recursion → emergence → universality. A tiny prior (the eight-case table), iterated locally, selects motifs that persist, propagates them through lanes, and composes them via collisions into a programmable algebra. Universality (via tag-system compilation) is the ceiling; the craft is learning to work inside a witnessable sublanguage where timing, routing, and interfaces are disciplined enough to engineer reliably.

What the figures established (in citation order).

- Fig. 1 (single seed): A one-bit start blossoms into an ether and traveling carriers—visible, stable transport channels born from almost nothing.
- Fig. 2 (random seed): Order reappears from disorder: the same ether, lanes, and gliders emerge, evidencing fertility and stability of the substrate.
- Fig. 3 (trajectories & collisions): Long-range order is operational: distinct velocity classes trace diagonal tracks; hot knots mark reproducible, phase-windowed interactions—our signal algebra.
- Fig. 4 (entropy curve): Emergence is measurable: a sterile ramp, a choppy middle (pattern formation), then a plateau above crystal and below noise—structured equilibrium.
- Fig. 5 (comparative patterns): Rule 110 sits at the Class III/IV edge: richer than linear fractals (90), more programmable than chaos (30), more expressive than traffic flow (184)—the sweet spot for seed potency.
- Fig. 6 (spectral signature): Hidden regularities are quantifiable: oblique ridges encode carrier velocities; harmonic combs encode the ether—bridging geometry and grammar.

Method, not mystery.
The mustard-seed criteria—minimality, fertility, composability, scalability—turn parable into practice. Sections II–IX showed how to *measure* each criterion (entropy, MDL, spectra, grammars) and how to *replicate* it (code and pipelines). Section VII mapped this calculus onto modern AI: gradient descent, cross-entropy, and attention as tiny seeds whose recursion yields circuits and capabilities; grokking as the same regime shift we observe in Rule 110.

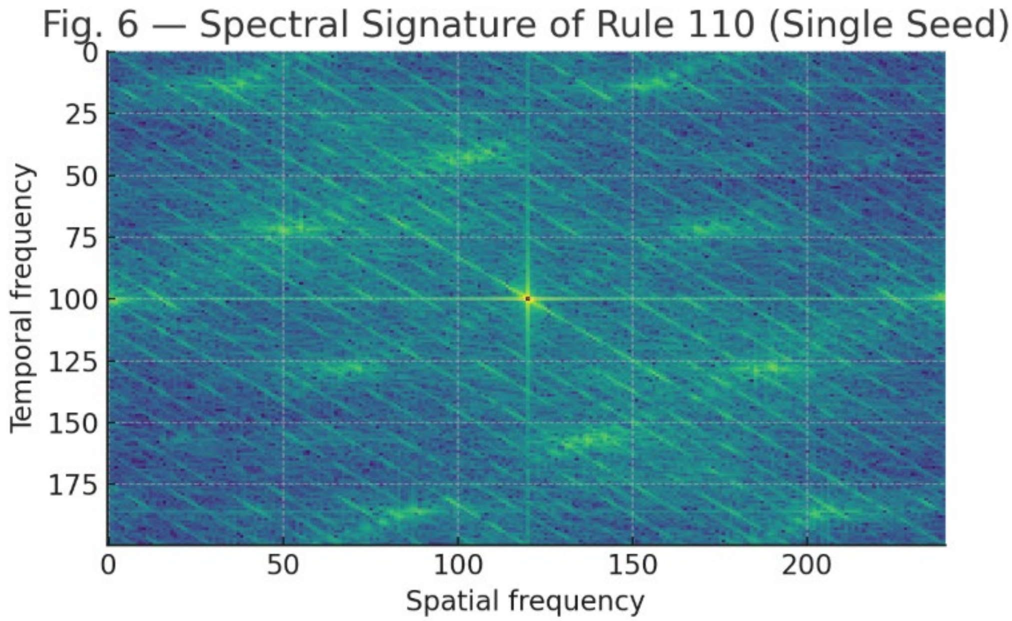

Fig. 6 — Spectral Signature of Rule 110.

- Oblique ridges, corresponding to glider velocity classes (each slope = a persistent transport channel).
- Harmonic combs centered on the origin, encoding the ether's periodic lattice.
- Diffuse halos, capturing long-range phase noise and collision-induced irregularity.

This spectrum visually and mathematically unites the entire chain of emergence:
the *cone* of Fig. 1, the *trajectories* of Fig. 3, the *entropy plateaus* of Fig. 4, and the *comparative regimes* of Fig. 5.
It is the "music" of Rule 110 — the frequency domain proof that minimal recursion sustains organized complexity.

Looking forward.
The open tasks are clear (Section XII): formal lower bounds on minimal gadget sets; noise-robust lanes and clocks; standardized diagnostics and datasets; and a compiler bridge between CA grammars and NN circuits. The guiding stance remains the same: choose the right small rule; cultivate by iteration; measure relentlessly; scale only under witness. With Rule 110, we showed that almost nothing can indeed become inexhaustibly something—reliably, reproducibly, and usefully.

Appendices A–E — rule tables; particle/grammar; MDL/entropy details; spectral methods; comparative notes

Appendix A — Rule Tables & Encoding *(supports Sec. II; Figs. 1–2)*

A.1 Elementary CA (ECA) formalism. Binary states $s_t(x) \in \{0,1\}$, radius-1 neighborhood $N_t(x) = (s_t(x-1), s_t(x), s_t(x+1))$.
Index $t = 4L + 2C + R \in \{0, \dots, 7\}$. Update via lookup array $f: \{0, \dots, 7\} \to \{0,1\}$.

A.2 Rule 110 table.

Neighborhood: 111 110 101 100 011 010 001 000

Index t : 7 6 5 4 3 2 1 0

Next state : 0 1 1 0 1 1 1 0

Bitstring (t=7→0): $01101110_2 = 110_{10}$.
Code: next = [0,1,1,1,0,1,1,0][(L<<2)|(C<<1)|R].

A.3 Boolean normal form (one equivalent DNF).
Let $\bar{x} = 1 - x$.
$C' = (\bar{L}C\bar{R}) \vee (\bar{L}\bar{C}R) \vee (\bar{L}\bar{C}\bar{R}) \vee (L\bar{C}R)$.

A.4 Boundary modes.

- Wrap (torus): $s_t(-1) = s_t(W-1)$, $s_t(W) = s_t(0)$— used for Figs. 1–2.
- Fixed zeros: $s_t(-1) = s_t(W) = 0$— shows finite cones; beware border bias.

A.5 Canonical seeds.

- Single-bit: $s_0(x) = \mathbb{1}[x = W/2]$.
- Random: $s_0(x) \sim$ Bernoulli$(1/2)$.
  Optional: sparse seeds (Hamming weight $k \ll W$) to probe interference.

Appendix B — Particle Catalogue & Collision Grammar *(supports Sec. III; Fig. 3)*

B.1 Glider entry schema (CSV suggested).

- id: symbolic (e.g., G_L1)
- w,h: spatial width, temporal period
- v: velocity (slope $\Delta x / \Delta t$)

- phase: allowed ether phases (mod T_{ether})
- lifetime_med, lifetime_tail: median τ, tail exponent or 90th percentile
- roles: {courier, transcoder, anchor, emitter}
- examples: list of (t,x) patches for visualization

Example (illustrative):

id,G_L1,w,h,v,phase,lifetime_med,lifetime_tail,roles

G_L1,5,6,-0.333,"{0,2,4}",300,>1e3,courier

B.2 Collision grammar format.
Write phase-aware productions with windows $W_{AB} \subseteq \{0, \dots, T_{\text{ether}} - 1\}$:

- Elastic: A + B -> A' + B' | Δϕ ∈ W_AB, jitter ≤ 1
- Transmute: A + B -> C | Δϕ ∈ W_AB
- Annihilate: A + B -> ∅ | Δϕ ∈ W_AB
- Emit: A + B -> C + Em(p) | Δϕ ∈ W_AB

B.3 Lane/clock gadgets.

- Emitter Em(p): periodic source; defines global tick.
- Anchor An: quasi-stationary defect; supports reflect/toggle.
- Splitter Sp: A -> A + A across two lanes (with phase offsets).
- Delay D(δ): path that implements δ-step latency without phase loss.

B.4 Automated trajectory extraction (reference implementation sketch).

1. Score: $M = K_{/} * X + K_{\backslash} * X$using 3×3 diagonal kernels.
2. Threshold & label: connected components in (x, t).
3. Fit lines: RANSAC of $x \approx at + b$→ velocities $v = a$.
4. Template match: cross-correlate patches with a template bank across phase.
5. Collisions: find intersections $| t_i - t_j | \leq \Delta t, | x_i - x_j | \leq \Delta x$; classify outcomes pre/post.

B.5 Outputs.

- Velocity histogram (spectrum of glider classes).
- Lifetime distribution $P(\tau)$.
- Collision atlas (table of events with $\Delta\phi$, outcome, robustness).

Appendix C — MDL/Entropy Computation Details *(supports Sec. V; Fig. 4)*

C.1 MDL decomposition.

$$L = L(\text{rule}) + L(\text{seed}) + L(\mathcal{G}) + L(\text{resid} \mid \mathcal{G}).$$

- $L(\text{rule})$: ~8–32 bits incl. header.
- $L(\text{seed})$: single-bit ≈ $\lceil \log_2 W \rceil$; random ≈ W.
- $L(\mathcal{G})$: bits to store ether period, template sprites (binary patches + strides), and collision rules (IDs + valid windows).
- Residual: encode remaining pixels with run-length or LZ per row.

C.2 Naïve vs grammar encoders.

- Baseline (naïve): per-row RLE → L_{naive}.
- Grammar: emit $\mathcal{G}$then replay ether + place gliders + apply collisions; encode mismatches with RLE → L_{gram}.
  MDL leverage $\Lambda = (L_{\text{naive}} - L_{\text{gram}})/(L(\text{rule}) + L(\text{seed}))$.

C.3 Row entropy H_t.

$$H_t = -p_t \log_2 p_t - (1-p_t)\log_2(1-p_t), p_t = \frac{1}{W}\sum_x X(t,x).$$

Compute for single-seed evolution; plot vs t(Fig. 4).

C.4 Beyond H_t.

- Block entropy $H^{(k)}$on k-grams; entropy rate $h_\mu \approx H^{(k)} - H^{(k-1)}$for moderate k.
- Excess entropy $E \approx \sum_{k=1}^{K} (H^{(k)} - kh_\mu)$.
- Temporal MI $I(X_t; X_{t+\Delta})$; Spatial MI $I(X_{t,i}; X_{t,i+d})$.

C.5 Reporting checklist.

- Plot H_t(ramp → oscillatory → plateau).
- Table: | $\mathcal{G}$ |, bits/template, residual bits/row, Λ.
- Curves: $h_\mu$and Evs time (optional).

Appendix D — Spectral Methods *(supports Sec. IX; Fig. 6)*

D.1 Preprocessing.
Row-center: $\tilde{X}(t,x) = X(t,x) - \bar{X}_t$; apply separable Hann window to reduce leakage.

D.2 2D FFT & power.

$$\hat{X}(\omega,k) = \text{FFT2}(\tilde{X}), P = \log\,(1+|\ \text{fftshift}(\hat{X})\ |).$$

Display $P(\omega,k)$(Fig. 6).

D.3 Ridge detection (velocity classes).
Apply Radon/Hough on $P$to find lines through the origin; slope $v = \omega/k$. Integrate power in strips | $\omega - vk$ |≤ Δto get ridge energy E_v.

D.4 Ether harmonics.
Find peaks at $k = nk_0$, $\omega = m\omega_0$→ lattice $\Lambda_{\text{ether}} = 2\pi/k_0$, period $T_{\text{ether}} = 2\pi/\omega_0$.

D.5 Sliding spectra.
Window $t$with stride; compute $E_v(t)$and spectral entropy $H_{\text{spec}}(t)$to monitor carrier birth/decay and regime shifts.

D.6 Cross-walk to symbols.
Match ridge slopes $\{v_i\}$to track-fit velocities (Appendix B) to label glider families; discrepancies flag overlapping species or mislabels.

Appendix E — Comparative Rules & Phase Notes *(supports Sec. VI; Fig. 5)*

E.1 Exhibit settings.
Width 200, steps 120, wrap boundary, single-bit seed at center; identical plotting parameters for fair comparison.

E.2 Quick taxonomy (qualitative + diagnostics).

Rule	Regime	Hallmarks in raster	Spectral cues	Potency implications
30	Chaotic	irregular spray, no lanes	diffuse $P(\omega, k)$, high H_{spec}	poor composability; good randomness
90	Linear/fractal	Sierpiński triangles	sharp harmonic comb; few slopes	high compressibility, low expressivity
184	Traffic/flow	jams/rarefactions, rightward drift	low-slope ridges; conservation signature	good routing, limited logic
110	Complex (edge)	ether + diverse gliders + collisions	multiple ridges + harmonics	fertile, composable, scalable (toward universality)

E.3 Seed potency summary (from Sec. VI).

- Fertility: 110 ≫ 184 ≫ 90 ≫ 30.
- Composability: 110 > 184 > 90 > 30.
- Scalability: 110 (with clocks/anchors) ≥ 184 (transport) > 90 (underpowered) > 30 (unroutable).

E.4 Repro notes.
Ensure identical color scales and aspect ratios across subplots; annotate slopes on 110 to emphasize velocity classes; for 184, mark shock fronts.

Appendix Artifacts & File Map (suggested)

- catalog/gliders.csv — particle entries (B.1).
- atlas/collisions.csv — events with $\Delta\phi$, outcome, robustness (B.2, B.4).
- metrics/entropy.csv — $H_t$per run; metrics/mdl.csv — MDL components (C.5).
- spectrum/ridges.csv — $\{v_i, E_{v_i}\}$per window (D.5).
- comparative/fig5_params.json — exact settings for Rules 30/90/110/184 (E.1).

These appendices turn the narrative into a portable lab notebook: precise rule/encoding (A), measurable motif and interaction inventories (B), compression/entropy protocols (C), spectral diagnostics (D), and comparative baselines (E).

References

Primary sources: Rule 110 & cellular automata

- Cook, M. (2004). Universality in Elementary Cellular Automata. *Complex Systems, 15*(1), 1–40.
- Wolfram, S. (1983). Statistical Mechanics of Cellular Automata. *Reviews of Modern Physics, 55*(3), 601–644.
- Wolfram, S. (2002). *A New Kind of Science*. Wolfram Media.
- Ilachinski, A. (2001). *Cellular Automata: A Discrete Universe*. World Scientific.
- Adamatzky, A. (Ed.). (2010). *Game of Life Cellular Automata*. Springer.
- Langton, C. G. (1990). Computation at the Edge of Chaos. *Physica D, 42*(1–3), 12–37.
- Mitchell, M., Crutchfield, J. P., & Hraber, P. T. (1994). Evolving Cellular Automata to Perform Computations. *Complex Systems, 7*(2), 89–130.
- Neary, T., & Woods, D. (2006). P-Completeness of Cellular Automaton Rule 110. In *Automata, Languages and Programming* (LNCS 4051, pp. 132–143). Springer.

Universality, symbolic dynamics, & computational mechanics

- Crutchfield, J. P., & Young, K. (1989). Inferring Statistical Complexity. *Physical Review Letters, 63*(2), 105–108.
- Shalizi, C. R., & Crutchfield, J. P. (2001). Computational Mechanics: Pattern and Prediction, Structure and Simplicity. *Journal of Statistical Physics, 104*(3–4), 817–879.
- Lind, D., & Marcus, B. (1995). *An Introduction to Symbolic Dynamics and Coding*. Cambridge University Press.
- Kari, J. (2005). Theory of Cellular Automata: A Survey. *Theoretical Computer Science, 334*(1–3), 3–33.

Information theory, MDL, and complexity

- Shannon, C. E. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal, 27*(3–4), 379–423, 623–656.
- Cover, T. M., & Thomas, J. A. (2006). *Elements of Information Theory* (2nd ed.). Wiley.
- Li, M., & Vitányi, P. (2008). *An Introduction to Kolmogorov Complexity and Its Applications* (3rd ed.). Springer.

- Grünwald, P. D. (2007). *The Minimum Description Length Principle*. MIT Press.
- Rissanen, J. (1989). *Stochastic Complexity in Statistical Inquiry*. World Scientific.

Spectral, detection, and pattern-analysis methods

- Bracewell, R. (1999). *The Fourier Transform and Its Applications* (3rd ed.). McGraw–Hill.
- Radon, J. (1917/1986). On the Determination of Functions from Their Integral Values Along Certain Manifolds. *IEEE Transactions on Medical Imaging, 5*(4), 170–176 (English translation).
- Duda, R. O., & Hart, P. E. (1972). Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Communications of the ACM, 15*(1), 11–15.
- Pratt, W. K. (2001). *Digital Image Processing* (3rd ed.). Wiley.

Neural networks, priors, and grokking

- Power, A., Burda, Y., Edwards, H., et al. (2022). Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets. *arXiv:2201.02177*.
- Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2021). Understanding Deep Learning Requires Rethinking Generalization. *Communications of the ACM, 64*(3), 107–115.
- Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the Importance of Initialization and Momentum in Deep Learning. In *ICML*.
- Tishby, N., & Zaslavsky, N. (2015). Deep Learning and the Information Bottleneck Principle. In *ITW*.
- Hutter, M. (2005). *Universal Artificial Intelligence: Sequential Decisions Based on Algorithmic Probability*. Springer.
- Jaynes, E. T. (2003). *Probability Theory: The Logic of Science*. Cambridge University Press.

Mechanistic interpretability & circuits-as-particles analogy

- Elhage, N., Nanda, N., Olsson, C., et al. (2021). A Mathematical Framework for Transformer Circuits. Anthropic (Tech. Report).
- Olsson, C., Elhage, N., Nanda, N., et al. (2022). In-Context Learning and Induction Heads. Anthropic (Tech. Report).
- Olah, C., Cammarata, N., Schubert, L., et al. (2020). Zoom In: An Introduction to Circuits. *Distill*.