RESEARCH REPORT · 3 NOVEMBER 2026

Weight space reprogramming of neural cellular automata

Stefano Nichele

Østfold University of Applied Sciences (Norway)
Simula AI Safety (Norway)
stefano.nichele@hiof.no

ABSTRACT

Can a language model change the behavior of a neural cellular automaton by editing its weights directly? We study a growing gecko NCA whose 8,336 parameters define a local rule shared across space and recurrently applied through time. GPT-6 Astra produced six edits without simulation feedback: two color permutations, three spatial transformations, and slower growth. The symmetry edits reproduced their requested transformations to numerical precision; the speed edit doubled measured maturation time. In a separate, feedback-enabled stage, Astra used 36 adaptive evaluations to construct a bounded timer inside the existing network. The selected edit grew a recognizable gecko and retained its appearance through 8,192 steps in all eight unseen stochastic runs. Analysis shows that persistence comes from autonomous arrest of developmental updates. These results distinguish exact architectural symmetries from a more demanding change to long-term dynamics, and demonstrate a reproducible instance of weight-only reprogramming without gradient-based training.



Overview. Seven edits of one growing gecko; actual simulated states. Step 96, except half-speed at step 128. Synchronized animations are available in the web companion.

1. A local rule, a long-horizon effect

Weight-space programming asks a general-purpose language model to inspect and modify neural-network parameters around a behavioral specification. The motivating study by Takehi and colleagues explores this idea across several small networks and larger models [1]. Here the object of intervention is a neural cellular automaton (NCA), where the requested output is a developmental trajectory rather than a single prediction.

NCAs are **local recurrent models**. Every cell applies the same neural update rule to its own state and a small neighborhood, then feeds the updated state into the next iteration. The visible organism emerges from many such interactions. Editing one weight therefore changes computations at many locations and at every later step. A small local change can propagate, accumulate, alter cell survival, or only reveal its effect after apparently successful growth.

This recurrence makes delayed goals such as persistence especially interesting. A modification must preserve early development while changing what happens hundreds or thousands of updates later. Color and spatial symmetries provide controlled starting points; speed and persistence probe the dynamics more directly. This is a motivation from the recurrence, not a claim that every NCA edit is intrinsically harder than every feedforward-model edit.

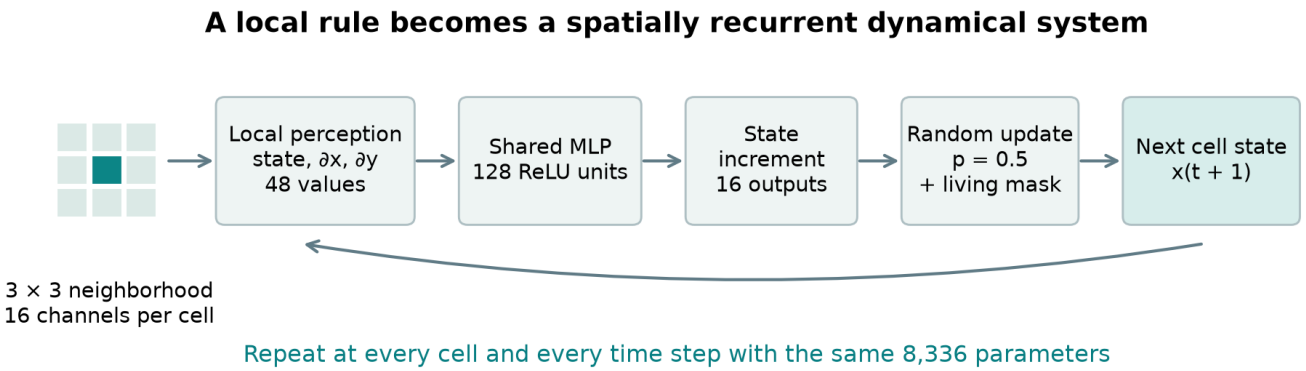


Figure 1. The same local neural rule is reused at all cells and all times. A cell perceives only its 3 × 3 neighborhood; global morphology emerges through repeated interaction.

The growing NCA

The starting model is the gecko checkpoint from Growing Neural Cellular Automata [2,3]. Each cell has 16 state values: premultiplied RGB, opacity α, and 12 hidden channels. Fixed identity and Sobel operators produce 48 perception values. A shared 48 → 128 ReLU → 16 network predicts a residual update. Its four parameter arrays contain 8,336 scalars.

$$p = [x, D_x x, D_y x] \quad h = \mathrm{ReLU}(pW_1 + b_1) \quad \Delta x = hW_2 + b_2$$

A Bernoulli mask with probability 0.5 selects whole-cell updates. The candidate state is $x + m\Delta x$. A cell is retained only when both pre-update and post-update 3 × 3 maximum opacity are at least 0.1. All channels otherwise become zero. The grid is 65 × 65 with zero padding; the central seed has RGB zero and channels 3-15 equal to one.

2. Experimental setup

The experimenter was **GPT-6 Astra, model identifier** `gpt-6-astra`, used on 3 November 2026. The recorded reasoning setting was inherited/default. An immutable dated model snapshot, temperature, top-p, and generation seed were not recorded, so a more precise version claim is not possible. The NCA simulation seeds and all submitted checkpoints are retained.

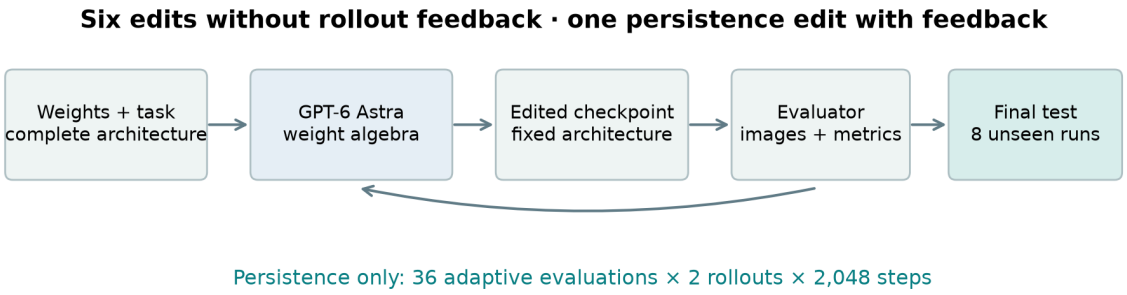


Figure 2. Weight-only editing with separate evaluation. The first six edits receive no rollout feedback before submission. Persistence receives a metered feedback loop; unseen tests follow final selection.

Condition	Six algebraic edits	Persistence
Input	Complete weights, architecture, gecko label	Same checkpoint and architecture; diagnostic feedback
Allowed work	Numerical weight inspection and algebra	Weight algebra and analysis of returned trajectories
Feedback	None before submission	36 evaluations; two fixed rollouts per evaluation
Simulation budget	Independent evaluation after submission	72 rollouts × 2,048 steps = 147,456 grid updates
Final evaluation	8 matched-noise runs; independent-noise follow-up	8 unseen runs, audited through 8,192 steps
Not allowed	Training, gradients, simulation during editing	Training, gradients, fitted regression, optimizer or automated sweep

One **rollout** is a complete growth simulation from the seed. One **evaluation** tests one submitted checkpoint and returns feedback. A **grid forward pass** advances one grid by one NCA step; two grids were processed together, so the persistence budget corresponds to 73,728 batched calls. The 36-evaluation stage continued an existing Astra context with 12 prior evaluations available as background. Prior-stage results are outside this report; the reported stage is not a cold-start 36-total-query trial.

Metrics and controls

Silhouette overlap is intersection-over-union (IoU) using $\alpha > 0.1$. Color error is RGB RMSE over the union of candidate and reference foreground, after compositing clipped premultiplied RGBA on white. The large white background therefore cannot dominate the score. Opacity mass is the sum of clipped α, divided by reference mass. Speed and persistence reference the original step-96 state; symmetry tests reference the appropriately transformed original state at the matching step (96 or 128). References are simulated states, not withheld target-image files.

Maturity requires IoU ≥ 0.85 and RGB RMSE ≤ 0.15 by step 256, checked every eight steps. Persistence additionally requires IoU ≥ 0.80, RMSE ≤ 0.15 and opacity-mass ratio 0.8-1.2 at steps 256, 512, 1,024 and 2,048. The extended audit adds 3,072, 4,096, 6,144 and 8,192. At least seven of eight complete trajectories must pass. A separately trained persistent gecko serves only as a positive control and was unavailable to Astra.

All reported main tests use the published uint8 checkpoint decoded to float32. Activations and edited weights are float32; this is not a bitwise reproduction of the WebGL demo's texture quantization or periodic boundaries. Tool restrictions were instructed and the evaluator metered calls, but access was not isolated against a malicious experimenter.

Experiments 1-2 · No rollout feedback

3. Can weight edits change the colors?

The first requests were to make the gecko blue and to exchange its red and green colors. “Blue” was operationalized as swapping green and blue. Astra submitted complete channel permutations that change the recurrent state representation, rather than modifying only the displayed pixels.



Figure 3. Actual NCA outputs at step 96: original, green-blue permutation, and red-green permutation. The architecture, seed and renderer are unchanged.

What Astra changed

For a permutation P of visible state channels, Astra permuted the corresponding input rows in all three perception blocks of W_1, the output columns of W_2, and the entries of b_2. The hidden bias b_1 is unchanged. Altering both input and output coordinates is essential because visible channels are read again on the next step. Swapping output channels alone would change subsequent feedback.

$$W_1' = \bar{P}W_1 \; W_2' = W_2P \; b_2' = b_2P$$

Here $\bar{P}$ repeats the same channel permutation across state, x-gradient and y-gradient inputs. The seed has zero RGB, and the opacity mask is unchanged. Consequently the entire recurrent trajectory is permuted in ideal arithmetic under identical stochastic update masks. This construction does not require identifying which neurons encode the gecko’s anatomy.

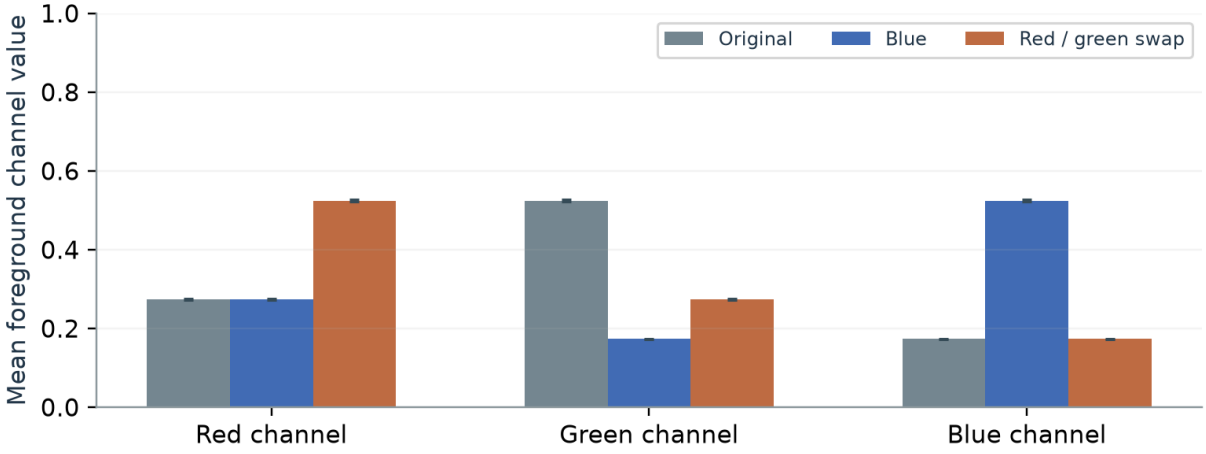


Figure 4. Foreground channel means at step 96, with min-max bars across eight realizations. The channel statistics exchange as requested; they are computed from the simulated state, not recolored illustrations.

Both edits passed all eight matched-noise tests at steps 96 and 128. Every silhouette IoU was 1.0. Across the five symmetry edits, the largest RGB RMSE at step 128 was 2.23×10^{-7}. The blue edit changes 1,004 parameters; the red-green edit changes 1,014. Zero-valued or identical entries can remain numerically unchanged even when a row or column is permuted.

Animation: open this experiment in the online report.

Experiments 3-5 · No rollout feedback

4. Can the organism grow in another orientation?

The next requests were horizontal reflection, clockwise rotation by 90°, and rotation by 180°. Horizontal reflection defines “facing the opposite direction” in this report. Screen coordinates increase to the right and downward. The original gecko’s head points toward the upper left.

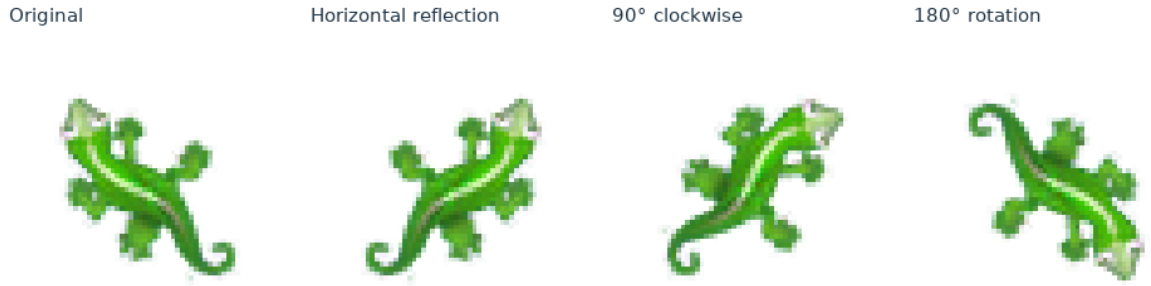


Figure 5. Original and three independently simulated weight edits at step 96. Spatial edits act on the local perception-to-update rule; the output image is not rotated by the renderer.

Write W_1 as three stacked blocks A, B and C for state, x-gradient and y-gradient inputs. Astra changed only the derivative blocks:

Transformation	New x-gradient block	New y-gradient block	Changed parameters
Horizontal reflection	−B	C	2,048
90° clockwise	−C	B	4,080
180° rotation	−B	−C	4,096

A reflection reverses one gradient direction; rotation exchanges and/or reverses the Sobel axes. The state block and output layer remain intact. The square grid, centered seed and isotropic living mask respect these spatial symmetries. For a pathwise comparison, the random cell-update masks must undergo the same spatial transformation as the reference.

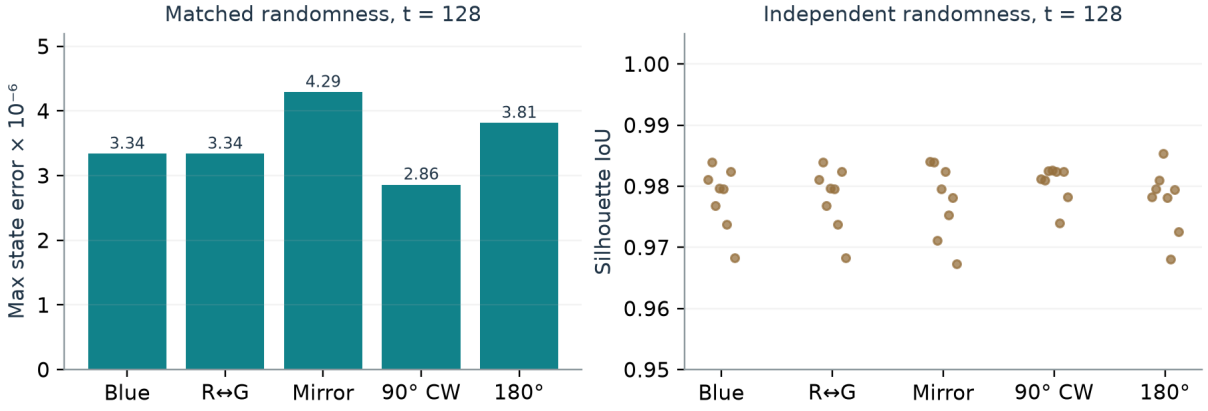


Figure 6. Left: maximum error over all 16 state channels after 128 steps under matched randomness. Right: overlap from eight independent-noise comparisons per edit. Independent stochastic images need not be pixelwise identical.

All five color/spatial transformations had IoU 1.0 in every matched-noise test. The largest full-state error was 4.29×10^{-6}. With a fresh random stream, the worst overlap across these edits was 0.967 and the largest foreground RGB RMSE was 0.0445. These results separate exact symmetry of the rule from ordinary trajectory variation under independent stochastic updates.

Animation: open this experiment in the online report.

Experiment 6 · No rollout feedback

5. Can the same network grow more slowly?

Astra halved every weight and bias in the output layer: $W_2' = 0.5W_2$ and $b_2' = 0.5b_2$. At a fixed state, this halves the residual update. The firing probability remains 0.5, and the simulator still executes the same number of time steps.



Figure 7. Original and half-speed model at identical simulation steps. The web animation is uniformly sampled and synchronized, so playback supports a timing comparison.

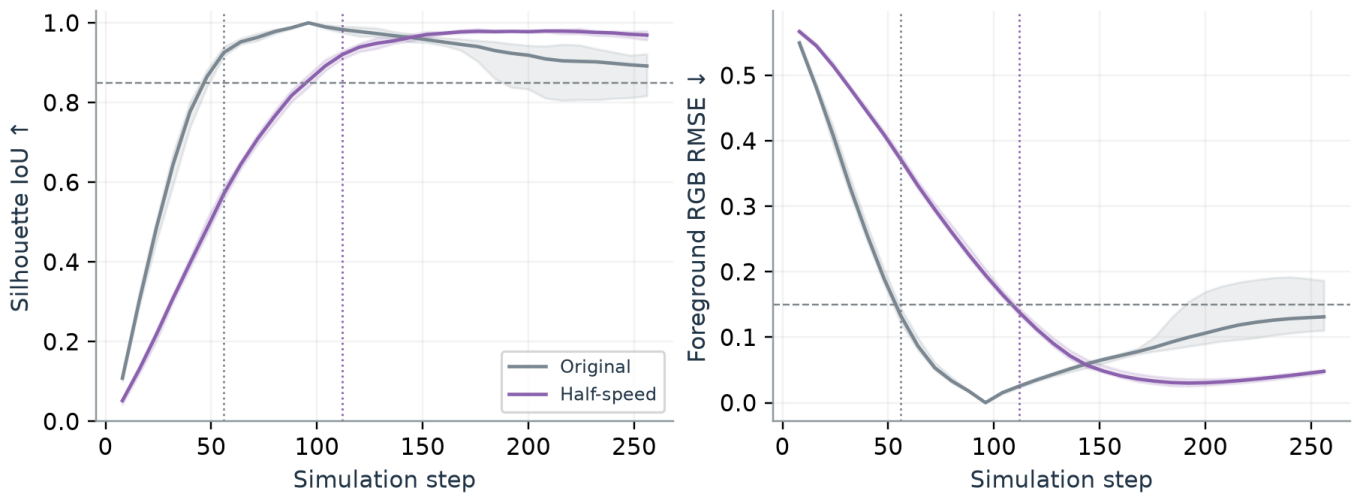


Figure 8. Mean and full realization range of silhouette overlap and foreground color error, relative to the original step-96 state. Dashed lines are maturity thresholds; dotted vertical lines mark steps 56 and 112.

All eight original runs first met the maturity criterion at step 56; all eight edited runs did so at step 112. The median ratio is therefore 2.0, at the eight-step sampling resolution. This edit changes 2,064 parameters. The comparison curves use the same eight-run random stream as the archived test, replayed with denser snapshots for plotting.

The mechanism resembles reducing an Euler integration step, but it is not an exact stochastic time reparameterization. ReLU thresholds, cell survival, discrete updates and accumulated noise can produce trajectory differences. The evidence supports approximately doubled developmental time for this checkpoint, not a guarantee that every state at time 2t exactly equals the original state at t.

Animation: open this experiment in the online report.

Experiment 7 · Metered rollout feedback

6. Can a growing NCA remain persistent?

Persistence changes long-term behavior while requiring early development to remain recognizable. Astra used 36 new adaptive evaluations, each running two fixed stochastic trajectories for 2,048 steps. Feedback included images, quality scores, saved states, hidden activations and update vectors. It committed one final checkpoint before seeing the eight fresh test runs.

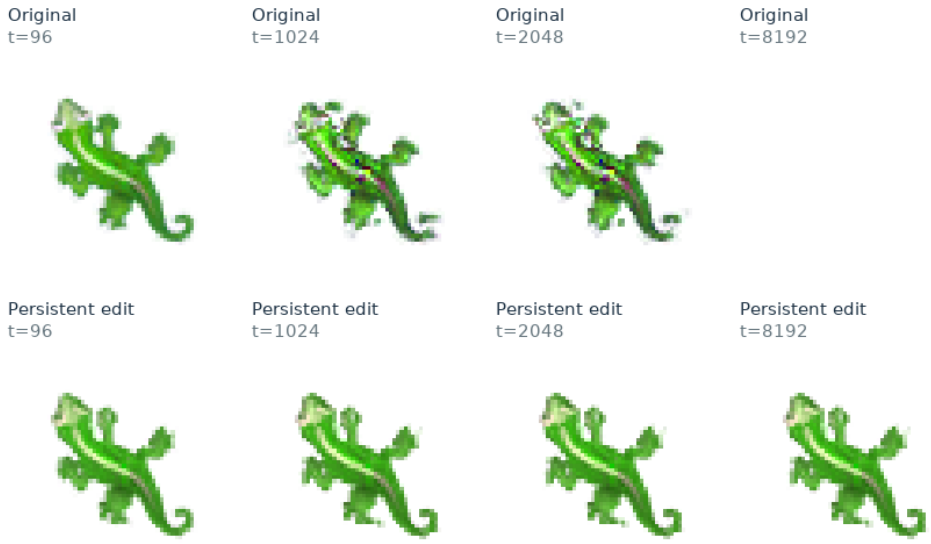


Figure 9. Held-out realization 1 at progressively longer horizons. The original growing model degrades; the edited model preserves its developed appearance. Long-horizon playback uses labeled checkpoints with unequal time gaps. The positive control is included in Figure 10.

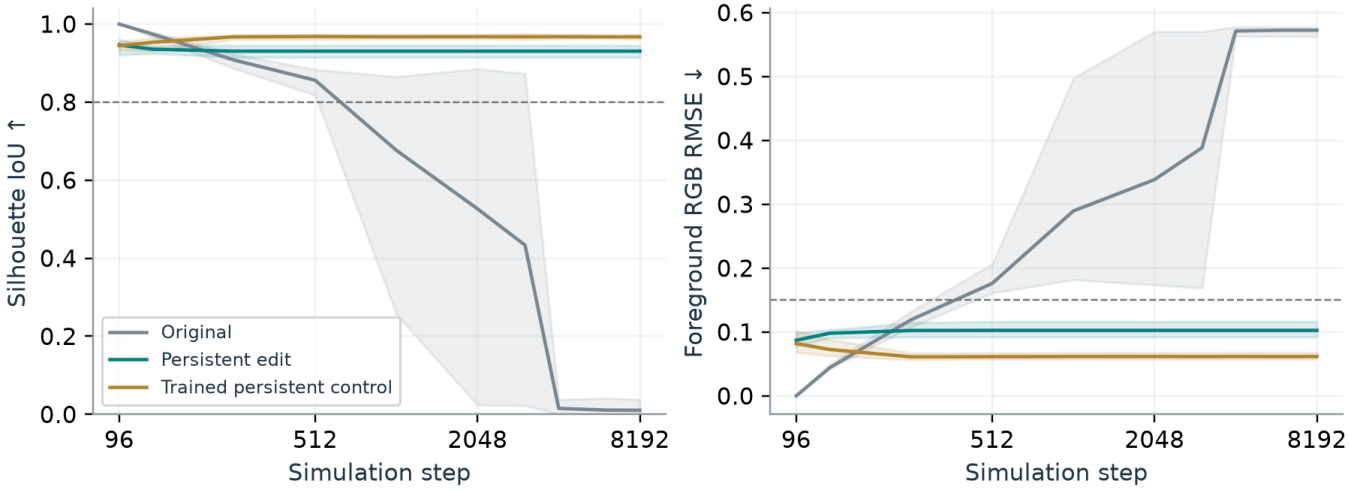


Figure 10. Independent eight-run audit: means with min-max ribbons, not confidence intervals. The selected edit meets all predeclared criteria through 8,192 steps in 8/8 runs; the original passes 0/8 and the trained control 8/8.

The selected edit matures in 64-72 steps on unseen runs. At step 8,192 its mean IoU is 0.9306, foreground RGB RMSE 0.1026 and opacity-mass ratio 0.9778. The corresponding ranges are 0.9144-0.9456, 0.0922-0.1163 and 0.9695-0.9891. The same appearance is retained at step 2,048. These are stochastic replicates of one edited checkpoint, not eight independently trained models or eight separate LLM attempts.

Animation: open this experiment in the online report.

7. Programming autonomous arrest

Inspection of Astra's construction and the saved internal states shows why the persistence edit works. It repurposes three existing hidden neurons to implement a bounded local timer. The architecture is unchanged, while 1,461 of 8,336 parameters (17.5%) change. The simulator continues updating throughout.

A timer in a color difference

Astra approximates blue using $B = R - 0.2G$ in the developmental computation, creating the timer coordinate $q = B - R + 0.2G$. Corresponding input and output substitutions make ordinary growth updates preserve q in ideal arithmetic. The seed starts with $q = 0$. A tiny added increment makes q drift until a trigger is reached.

$$\Delta q = -250\,\mathrm{ReLU}(-q - a) + 251\,\mathrm{ReLU}(-q - a - w) - \varepsilon$$

The constants are $a = 10^{-5}$, $w = (0.01002 - 0.00001)/251$ and $\varepsilon = 2.5 \times 10^{-7}$. Existing units 100 and 16 implement the trigger and saturation. After roughly 40 local cell firings, q moves rapidly toward the bounded value -0.01002025. Local firings are not a global stopwatch: cells entering the living region later can receive additional development time.

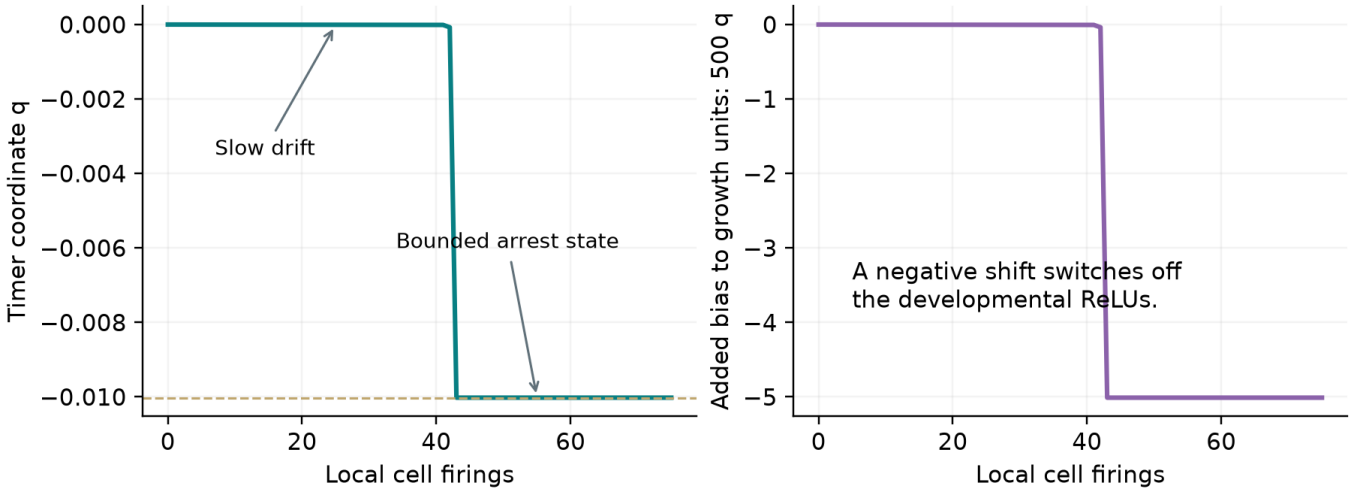


Figure 11. Analytical trajectory of the isolated timer coordinate against local cell firings. This diagram illustrates the constructed recurrence; it is not an additional NCA rollout or a global developmental clock.

Turning the growth rule off

Each of the remaining 125 growth neurons receives an added 500q in its preactivation. Near the timer's final value, this contributes approximately −5.01 and suppresses those ReLUs. Unit 6 carries the original output bias through $\mathrm{ReLU}[0.1 + 0.02(\alpha - 0.8) + 500q]$. Its opacity dependence helps suppress continued frontier growth, while the timer eventually shuts the bias off too.

Across all eight held-out grids, every non-blue state channel is bitwise unchanged between steps 2,048 and 8,192. The maximum blue-channel change is 2.35×10^{-7}. The result is appearance preservation by **autonomous arrest**. Regeneration after damage, active homeostasis and indefinite stability were not demonstrated. Edited weights remain float32; requantization into the original uint8 demo format was not tested.

8. What the results establish

Requested behavior	Weight mechanism	Verified outcome
Blue gecko	Green-blue state conjugation	8/8 exact-transform tests
Exchange red and green	Red-green state conjugation	8/8 exact-transform tests
Opposite facing	Negate x-gradient block	8/8 exact-transform tests
Rotate 90° clockwise	Exchange derivative blocks with a sign	8/8 exact-transform tests
Rotate 180°	Negate both derivative blocks	8/8 exact-transform tests
Grow more slowly	Halve output-layer residuals	56 → 112 steps; 8/8 mature
Persist after growth	Bounded local timer and gated updates	8/8 through 8,192 steps

The color and geometry results are controlled examples of architectural reprogramming. Their algebra preserves the recurrent computation under a change of coordinates. Success does not require recovering a semantic map from particular neurons to the gecko's head, legs or tail. The slowing edit exploits the residual form of the update and has a less exact relation to the full stochastic trajectory.

Persistence is different: it introduces a new temporal mechanism inside the fixed network. Early attempts within the 36-evaluation stage either damaged growth, allowed the boundary to expand, or arrested an inaccurate color pattern. The first complete two-run development success occurred at evaluation 13. Later edits improved the color relation, the choice of neurons to repurpose, and frontier control. Tests of timer strength, timing, numerical scaling and nearby developmental weights supplied evidence about robustness, but reused the same development seeds.

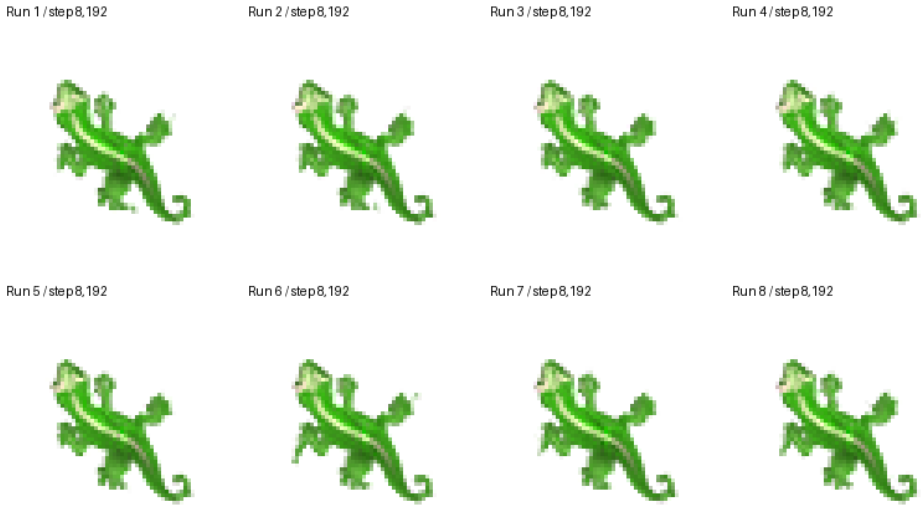


Figure 12. All eight unseen outcomes of the selected persistence edit at step 8,192. This grid exposes stochastic variation rather than showing only the representative trajectory.

The independent audit supports the final mechanism on this one trained gecko checkpoint. It does not establish generality across morphologies, training seeds, architectures or language models. The persistence result is stronger than merely slowing collapse within the observation window: growth completes promptly, and developmental updates then become inactive. Nevertheless, finite-horizon numerical arrest is not a proof of an indefinitely stable or self-repairing organism.

9. Reproducibility and research record

The public GitHub repository and its companion archive contain the original checkpoint, all seven selected edits, the trained persistent control, a NumPy simulator, exact weight-construction code, per-run metrics and the 36-stage experiment log. Each checkpoint is stored as float32 arrays W_1, b_1, W_2 and b_2. The persistence reconstruction script reproduces the committed checkpoint exactly; hashes make the selected files auditable.

Resource	Location in the companion bundle
Web report and PDF	index.html; report.pdf
Exact six-edit construction	code/edits.py
Exact persistence construction	code/persistence.py
Simulator and independent evaluation	code/nca.py; code/evaluate.py
Original, edited and control weights	weights/*.npz
Numerical results and source metadata	data/
Astra’s hypotheses and explanations	records/
Synchronized growth and checkpoint animations	animations/

Access to Astra’s recorded reasoning

The record includes the concise hypotheses written before each metered evaluation, observed outcomes, committed rationale, and executable reconstruction. These are accessible research notes that explain what was tried and why. They are not a complete private internal chain-of-thought. The six-edit rationale is a clearly marked excerpt; superseded tasks are omitted from this presentation.

Evaluation of the six edits uses NumPy RNG seed 20261103; the independent-noise check uses 20261104. Persistence development uses 391071; its unseen audit uses 73510991. The same test random stream is applied to original, edited and control checkpoints. Runtime versions, hashes, numerical conventions and commands are listed in the archive. Uniformly sampled growth animations and unequal-time long-horizon checkpoint players are labeled separately.

AI use disclosure

Stefano Nichele conceived the ideas in this work, designed the experiments and experimental setups, directed the revisions, and reviewed and checked the material. AI assistance, through Codex and GPT-6 Astra, supported coding, running the experiments and evaluations, and report preparation. GPT-6 Astra also generated the weight-edit proposals studied in the experiments, as documented in the methods and research records. The AI-assisted material and final report were checked by Stefano Nichele, who retains responsibility for the work and its conclusions.

References

[1] Rikiya Takehi, Jayden Teoh, Alex Zhang and Omar Khattab. *Sparks of Weight-Space Programming: LLMs can natively read & write neural network weights.* 2 November 2026. Project article.

[2] Alexander Mordvintsev, Ettore Randazzo, Eyvind Niklasson and Michael Levin. *Growing Neural Cellular Automata.* Distill, 2020. DOI: 10.23915/distill.00023.

[3] Original article code and checkpoints; reference training notebook. The bundled original code license and pinned checkpoint source are retained.

Web companion: Read the online report to play and scrub the simulations, or open `index.html` in a downloaded repository copy. The PDF presents selected frames from the same experiments. No simulated outcome in this report was synthesized by an image-generation model.