# How Much Can Be Cut Before a Shape Cannot Regrow? Characterizing 3D Self-Repair in Neural Cellular Automata

Ali Asaria
Transformer Lab

Tony Salomone
Transformer Lab

Deep Gandhi*
Transformer Lab

**Abstract**

Neural Cellular Automata (NCA) learn a single local update rule, shared by every cell, that can grow a target pattern from a seed and repair it after damage. This behavior is well studied in two dimensions but its three-dimensional counterpart, and in particular the robustness of 3D self-repair, is largely uncharacterized. We train small 3D voxel NCA to grow four target shapes from a single seed and to regenerate after amputation, three shapes reliably and a dense fourth only intermittently, and we measure how recovery depends on how much of the object is removed. Two findings stand out. First, a network trained only to persist a grown shape already regenerates almost perfectly after damage (mean recovery near 1.0), so self-repair can emerge from persistence training alone, without explicit damage supervision. Second, regeneration degrades with damage severity, and the decline is shape-dependent: bulky shapes recover gracefully even when most of the volume is removed, while a thin branching shape recovers well under light damage but collapses once a large fraction is gone. The shape that grows most accurately is the one that repairs worst, so growth fidelity is not the same property as regeneration robustness. Our numbers are from single training runs at $32^3$ resolution and are intended as an initial characterization rather than a benchmarked comparison. We summarize 3D self-repair with a recovery-versus-severity curve, a compact "phase-diagram"-style view.

## 1 Introduction

Some animals regrow lost body parts. An axolotl that loses a limb regenerates it to the correct size and shape, no more and no less, without any cell holding a blueprint of the whole animal; each cell follows local rules and the target morphology emerges as a stable outcome [7]. Neural Cellular Automata (NCA) are a machine learning analogue: a single small neural network, applied identically at every cell and conditioned only on a local neighborhood, is trained so that repeated application grows a target pattern from a seed and restores it after part of it is destroyed [12].

Most NCA work operates on 2D images. Growth and self-repair in three dimensions have been demonstrated for voxel structures [22], but the emphasis has been on producing the artifact rather than on measuring how robust the repair is. The central open question we take up is quantitative: for a 3D shape grown by an NCA, *how much of the grown shape can be removed before it can no longer regrow*, and does the answer depend on the shape and on the kind of damage?

*Corresponding author: deep@lab.cloud

We study small 3D voxel NCA that grow four target shapes at $32^3$ resolution and regenerate after amputation. We define a fixed battery of damage operators and a recovery metric, sweep damage severity from light to near-total, and summarize the result as a recovery-versus-severity curve per shape and per damage type. Our contributions are:

1. A training recipe for 3D voxel NCA that grow and self-repair, including the design choices that were necessary to escape an "all-dead" failure mode in which every cell dies out (Section 3).
2. Evidence that *regeneration can emerge from persistence training*: a model never shown damage during training still recovers almost perfectly from amputation, so damage supervision is not a precondition for repair (Section 5).
3. A recovery-versus-severity characterization of 3D self-repair, showing that regeneration degrades as a shape-dependent decline, sharp for the one thin structure we tested and gradual for the bulky ones (Section 5).
4. The observation that *growth fidelity is not regeneration robustness*: among our shapes, the one that grows most accurately regenerates the worst (Section 5).

## 2 Related Work

**Neural Cellular Automata.** Mordvintsev et al. [12] introduced Growing NCA, in which a shared per-cell rule grows a target image from a seed and, when trained with a sample pool and damage, persists and regenerates it. Follow-on work extended the idea to self-organizing texture synthesis [14] and to self-classification, where cells reach a distributed consensus and recover it after perturbation [18]. This behavior has been studied almost entirely in two dimensions [12, 14, 18]. Recent studies revisit the expressivity and training of these rules [2, 17] and provide accelerated implementations [4]. Our training regime follows this lineage: a shared local rule, stochastic updates, alive masking, and a sample pool.

**Three-dimensional and self-repairing NCA.** Sudhakaran et al. [22] grow 3D voxel artifacts and functional machines with NCA and report that damage during training enables regeneration, but the regeneration evidence is a single qualitative recovery figure. Silbernagel et al. [21] study self-repairing segmentation masks, and Pajouheshgar et al. [16] analyze how cellular automata robustly preserve stored information over long rollouts, both in two dimensions. Pahng et al. [15] morph 3D shapes with a shared local agent rule, but on an off-lattice particle substrate and without an amputation-and-regrow protocol. None of these quantifies how 3D voxel self-repair depends on damage severity, which is our focus.

**Self-organization, morphogenesis, and voxel generation.** Differentiable models of self-organizing systems and morphogenesis provide broader context [3, 10, 13, 19, 1, 24, 9, 11], and NCA have been applied beyond growth, for example to appearance and texture modeling [8] and as generators of training data [6]. Biological accounts frame the target morphology as an attractor that locally competent cells reach and hold [7]. As non-NCA references for producing 3D voxel occupancy, discrete diffusion models generate sparse voxel structures [5] and, in one case, also edit them by inpainting [23]; that model repairs masked regions and reports that matched-corruption

training is needed for local repair, a structural analogue of damage during training. Conditional morphogenesis with NCA has also been shown for 2D digits [20].

## 3 Method

**Model.** The state is a tensor with $C = 13$ channels per voxel on a $32^3$ grid: one occupancy ("alive") channel and twelve hidden channels. Each step, every voxel perceives its $3 \times 3 \times 3$ neighborhood through a learned convolution, and a shared per-voxel update network, implemented as two $1 \times 1 \times 1$ convolutions with a zero-initialized output layer, produces a residual state increment. Updates are stochastic (each voxel updates with probability one half), and an alive mask derived from a max-pool over the occupancy channel keeps a voxel active only if it and a neighbor are alive both before and after the update. State values are clamped to a bounded range for stability.

**Loss and target.** Supervision is a single fixed occupancy volume per shape. We combine a class-balanced squared error on the occupancy channel with a soft Dice term on the same channel. The squared-error term provides a strong gradient that bootstraps growth from the seed; the Dice term penalizes overgrowth once cells are alive. For dense shapes we place a floor on the positive-class weight, because the class-balanced weight alone becomes too small to escape the all-dead state.

**Training.** We train with a sample pool: grown states are stored and resampled so the rule must produce a stable fixed point rather than a one-shot trajectory. In each batch we reseed the worst quarter of samples and any that have become nearly dead, which prevents the pool from collapsing into an absorbing all-dead state. Damage during training is applied to a fraction of pooled samples after a warmup of the first 40% of steps. We use Adam at learning rate $10^{-3}$ with per-step gradient-norm clipping and a cosine schedule, and back-propagate through a randomized-length rollout (between 48 and 64 steps). A single voxel seed did not reliably bootstrap growth; a $3 \times 3 \times 3$ nucleus seed together with a low alive threshold was necessary to escape the all-dead failure mode.

**Damage battery.** We define four damage operators: a half-space cut, removal of a random cube, removal of a spherical "bite" near the surface, and removal of a protruding region (a "limb"). Each is parameterized by a severity equal to the fraction of occupied voxels removed. The same operators are used at training time (as augmentation) and at evaluation time, with disjoint random seeds.

## 4 Experimental Setup

**Shapes and data.** We use four hand-authored target shapes at $32^3$: a small four-legged creature, a branching tree, a chair, and a torus. Their occupancy fractions span an order of magnitude, from 0.88% for the thin tree to 8.24% for the dense torus, with the creature and chair in between at 2.29% and 2.67%. This is not a supervised dataset; supervision is the fixed target volume, and the models are per-shape.

**Metrics.** We report four quantities. Grown Intersection-over-Union (IoU) measures how closely the shape grown from a seed matches the target. Persistence IoU measures how well the grown state is held over a rollout eight times longer than the training horizon. Regeneration recovery is the ratio of recovered IoU to pre-damage IoU after a damage operator is applied and the rule is run

Table 1: Per-shape 3D NCA self-repair (trained with damage, held-out evaluation). Recovery is recovered IoU divided by pre-damage IoU; worst case is the minimum over the four operators at severities 0.3, 0.5, and 0.7. The torus row is a single successful seed (it collapsed to the all-dead state on three of four seeds), so its metrics are conditional on training not collapsing and are not directly comparable to the other rows.

| Shape | Grown IoU | Persist IoU | Mean recovery | Worst-case recovery |
|---|---|---|---|---|
| Creature | 0.72 | 0.76 | 0.93 | 0.72 |
| Tree | 0.98 | 0.98 | 0.76 | 0.54 |
| Chair | 0.90 | 1.00 | 0.89 | 0.57 |
| Torus | 0.52 | 0.60 | 0.98 | 0.68 |

forward; this ratio can exceed one when the regrown state matches the target slightly better than the imperfect grown state it replaced. The aggregate values in Table 1 are computed over the four damage operators at severities 0.3, 0.5, and 0.7: mean recovery averages over operators and these severities, and worst-case recovery is the minimum over them. Figure 1 instead resolves the full severity dependence from 0.1 to 0.9, including the extrapolation range above the 0.6 training cap, so the low points of a curve at high severity fall below that shape's Table 1 worst case. Evaluation uses held-out random seeds disjoint from training, and the same damage-operator family at train and eval, so the held-out axes are unseen seeds and unseen high severities, not a novel damage type. Each shape is trained once and evaluated stochastically; the reported numbers are single-run point estimates, and we discuss run-to-run variation in Section 5 and the Limitations.

**Compute.** Each model is a small network trained on a single NVIDIA A10 GPU; the full set of experiments in this paper used on the order of seven GPU-hours.

## 5 Results

**The recipe grows and repairs across shapes.** Table 1 reports per-shape metrics for models trained with the full recipe (damage during training, 2500 steps). Three of the four shapes grow with high fidelity: the tree and chair reach grown IoU 0.98 and 0.90, the creature 0.72. All three persist their grown state over the long rollout (persistence IoU from 0.76 to 1.00) and regenerate after amputation, with mean recovery from 0.76 to 0.93. The dense torus is the hard case: with the positive-weight floor it trains and regenerates well when it succeeds (grown IoU 0.52, mean recovery 0.98), but it collapsed to the all-dead state on three of four training seeds (grown IoU 0), so its row reports a single successful seed and its metrics are conditional on training not collapsing. We treat it as a limitation rather than a solved case.

**Regeneration emerges from persistence training.** On the creature, a model trained without any damage, only to grow and persist, already recovers almost perfectly from the damage battery: mean recovery 0.999 and worst-case recovery 0.782. This is the load-bearing observation of the paper, and it does not depend on any fragile difference: self-repair does not require explicit damage supervision. We deliberately do not claim that adding damage during training improves recovery. A damage-trained creature reached worst-case recovery 0.813, but two separate runs of that same damage-trained configuration gave worst-case recovery 0.72 and 0.81, so run-to-run variation is

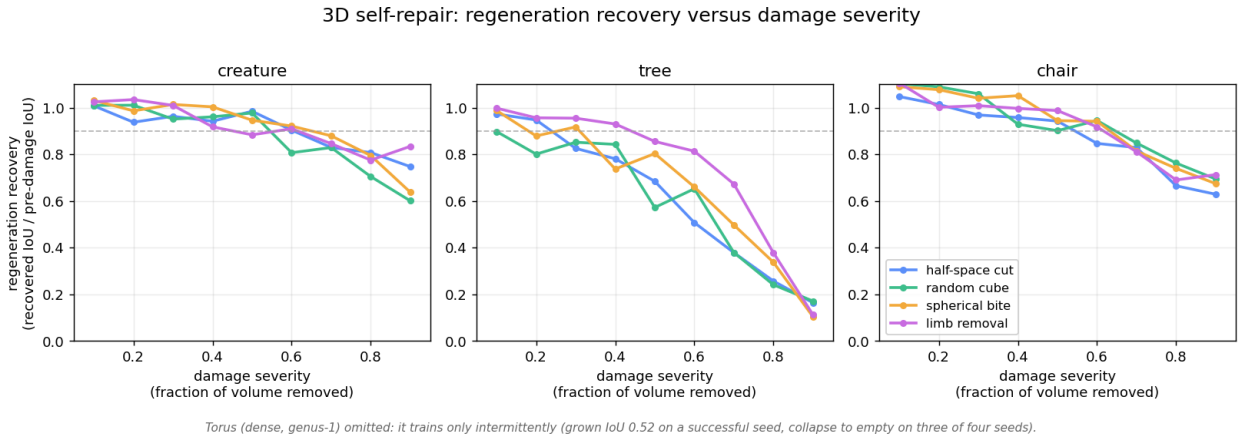


Figure 1: Regeneration recovery versus damage severity, per shape (one curve per damage operator). Recovery is recovered IoU divided by pre-damage IoU; severity is the fraction of occupied voxels removed. The bulky creature and chair degrade gracefully, retaining most recovery even at high severity, while the thin tree shows a sharp decline and collapses once a large fraction is removed. The dense torus is omitted because it trains only intermittently (see Limitations).

of order 0.1, larger than the 0.03 gap between the damage-off and damage-on runs (which also differ in training length, 1500 versus 2500 steps). We therefore read the near-perfect recovery of the damage-off model, consistent with an attractor interpretation, as evidence that pool-based persistence training makes the target a state the local dynamics return to after perturbation, and we leave a matched-length, multi-seed comparison of damage-on against damage-off to future work.

**Recovery degrades with severity as a shape-dependent decline.** Figure 1 plots regeneration recovery against damage severity for the three shapes that train reliably, with one curve per damage operator over nine severities from 0.1 to 0.9. Every shape recovers nearly fully under light damage. As severity rises, the curves fall, but the shape of the fall differs sharply. The creature and chair degrade gracefully: averaged over operators, recovery goes from 0.98 and 1.02 at 30% removed to 0.71 and 0.68 at 90% removed, so they still restore most of the shape even when almost all of it is cut away. The tree shows a steep decline: recovery falls from 0.89 at 30% removed to 0.14 at 90% removed. A thin branching structure regrows well when a small part is missing but cannot re-establish itself once a large fraction is gone. The damage operators separate within a shape, so which region is removed matters, not only how much.

**Growth fidelity is not regeneration robustness.** The tree is the best-growing shape (grown IoU 0.98) yet has the lowest worst-case recovery (0.54) of the three reliable shapes and by far the steepest severity decline (Figure 1), while the creature grows least accurately (0.72) but has the best worst-case recovery (0.72). Accuracy of growth from a seed and robustness of repair after damage are therefore distinct properties, and a shape can be strong in one and weak in the other.

## 6 Discussion

The results are consistent with a simple picture. Pool-based persistence training does not merely teach the rule to reach the target once; it appears to make the target an attractor of the local dynamics, so that a damaged state flows back to it. This may explain why regeneration appears even without damage supervision, and why, in the one shape where we compared, damage supervision helped mainly the worst case rather than being a precondition for repair. The shape dependence is consistent with a geometric explanation: a bulky shape presents many alive neighbors from which the rule can regrow a removed region, whereas a thin structure, once a large section is cut, may leave too little living tissue and too few informative neighbors to re-establish the missing branches. The separation between damage operators within a shape is consistent with this, since removing a protruding limb leaves the core intact while a cut through a thin trunk does not.

## 7 Limitations

The study is deliberately small in scope. The most important caveat is statistical: every number is from a single training run per shape with stochastic evaluation, and we do not report error bars. Where we do have two runs of the same configuration (the damage-trained creature), worst-case recovery varied by about 0.1, so small differences between single runs should not be over-interpreted, and multi-seed replication with reported spread is the most important next step. The shapes are synthetic, hand-authored, low-resolution ($32^3$), and occupancy-only, without color or material, so absolute IoU values should not be compared to high-resolution generation work. Models are per-shape, so we make no claim about held-out-shape generalization; the held-out axes are unseen damage seeds and unseen high severities, not shapes and not novel damage types. The thin-versus-bulky dichotomy rests on one thin shape and two bulky ones, so whether thinness is the governing variable is a hypothesis rather than a tested law; systematically varying shape thickness is the single most valuable follow-up experiment. The dense torus trains only intermittently: with a positive-weight floor it succeeds on some seeds and collapses to the all-dead state on others (three of four seeds here), so its single-run numbers are an existence result, not a reliable operating point. The comparison between training with and without damage differs in training length as well as in the damage setting, so we do not claim a damage-training benefit; a matched-length, multi-seed ablation is future work. Finally, our baselines are internal ablations of our own recipe rather than an independent trained 3D-NCA system, and the recovery metric is occupancy IoU, which does not capture fine surface quality.

## 8 Availability

All artifacts (code, configurations, and target shapes) are available from the authors on request.

## References

[1] Aidan Barbieux and Rodrigo Canaan. Coralai: Intrinsic Evolution of Embodied Neural Cellular Automata Ecosystems. arXiv:2406.09654 [cs.NE], 2024. URL https://arxiv.org/abs/2406.09654.

[2] Gabriel Béna, Maxence Faldor, Dan F. M. Goodman, and Antoine Cully. A Path to Universal Neural Cellular Automata. arXiv:2505.13058 [cs.LG], 2025. URL https://arxiv.org/abs/2505.13058.

[3] Ramya Deshpande, Francesco Mottes, Ariana-Dalia Vlad, Michael P. Brenner, and Alma dal Co. Engineering morphogenesis of cell clusters with differentiable programming. arXiv:2407.06295 [q-bio.CB], 2024. URL https://arxiv.org/abs/2407.06295.

[4] Maxence Faldor and Antoine Cully. CAX: Cellular Automata Accelerated in JAX. arXiv:2410.02651 [cs.LG], 2024. URL https://arxiv.org/abs/2410.02651.

[5] Justin Jung. Scaffold Diffusion: Sparse Multi-Category Voxel Structure Generation with Discrete Diffusion. arXiv:2509.00062 [cs.CV], 2025. URL https://arxiv.org/abs/2509.00062.

[6] Dan Lee, Seungwook Han, Akarsh Kumar, and Pulkit Agrawal. Training Language Models via Neural Cellular Automata. arXiv:2603.10055 [cs.LG], 2026. URL https://arxiv.org/abs/2603.10055.

[7] Michael Levin. The Computational Boundary of a 'Self': Developmental Bioelectricity Drives Multicellularity and Scale-Free Cognition. *Frontiers in Psychology*, 10:2688, 2019. doi: 10.3389/fpsyg.2019.02688. URL https://www.frontiersin.org/articles/10.3389/fpsyg.2019.02688.

[8] Chen Liu and Tobias Ritschel. Neural Differential Appearance Equations. arXiv:2410.07128 [cs.CV], 2024. URL https://arxiv.org/abs/2410.07128.

[9] Tin Luu, Binh Nguyen, and Man Ngo. Missing Data Imputation using Neural Cellular Automata. arXiv:2509.00651 [cs.LG], 2025. URL https://arxiv.org/abs/2509.00651.

[10] Koen Minartz, Tim d'Hondt, Leon Hillmann, Jörn Starruß, Lutz Brusch, and Vlado Menkovski. Deep Neural Cellular Potts Models. arXiv:2502.02129 [cs.LG], 2025. URL https://arxiv.org/abs/2502.02129.

[11] Avni Mittal, John Kalkhof, Anirban Mukhopadhyay, and Arnav Bhavsar. MedSegDiffNCA: Diffusion Models With Neural Cellular Automata for Skin Lesion Segmentation. arXiv:2501.02447 [cs.CV], 2025. URL https://arxiv.org/abs/2501.02447.

[12] Alexander Mordvintsev, Ettore Randazzo, Eyvind Niklasson, and Michael Levin. Growing Neural Cellular Automata. *Distill*, 2020. doi: 10.23915/distill.00023. URL https://distill.pub/2020/growing-ca/.

[13] Elias Najarro, Nicolas Bessone, and Sebastian Risi. Solving Inverse Problems in Stochastic Self-Organizing Systems through Invariant Representations. arXiv:2506.11796 [nlin.AO], 2025. URL https://arxiv.org/abs/2506.11796.

[14] Eyvind Niklasson, Alexander Mordvintsev, Ettore Randazzo, and Michael Levin. Self-Organising Textures. *Distill*, 2021. doi: 10.23915/distill.00027.003. URL https://distill.pub/selforg/2021/textures/.

[15] Seong Ho Pahng, Guoye Guan, Benjamin Fefferman, and Sahand Hormoz. DiffeoMorph: Learning to Morph 3D Shapes Using Differentiable Agent-Based Simulations. arXiv:2512.17129 [cs.LG], 2025. URL https://arxiv.org/abs/2512.17129.

[16] Ehsan Pajouheshgar, Aditya Bhardwaj, Nathaniel Selub, and Ethan Lake. Exploring the Landscape of Non-Equilibrium Memories with Neural Cellular Automata. arXiv:2508.15726 [cond-mat.stat-mech], 2025. URL https://arxiv.org/abs/2508.15726.

[17] Ehsan Pajouheshgar, Yitao Xu, Ali Abbasi, Alexander Mordvintsev, Wenzel Jakob, and Sabine Süsstrunk. Neural Cellular Automata: From Cells to Pixels. arXiv:2506.22899 [cs.CV], 2025. URL https://arxiv.org/abs/2506.22899.

[18] Ettore Randazzo, Alexander Mordvintsev, Eyvind Niklasson, and Michael Levin. Self-classifying MNIST Digits. *Distill*, 2020. doi: 10.23915/distill.00027.002. URL https://distill.pub/2020/selforg/mnist/.

[19] Francesco Sacco, Dalton A R Sakthivadivel, and Michael Levin. Topological constraints on self-organisation in locally interacting systems. arXiv:2501.13188 [cond-mat.stat-mech], 2025. URL https://arxiv.org/abs/2501.13188.

[20] Ali Sakour. Conditional Morphogenesis: Emergent Generation of Structural Digits via Neural Cellular Automata. arXiv:2512.08360 [cs.NE], 2025. URL https://arxiv.org/abs/2512.08360.

[21] Malte Silbernagel, Albert Alonso, Jens Petersen, Bulat Ibragimov, Marleen de Bruijne, and Madeleine K. Wyburd. rNCA: Self-Repairing Segmentation Masks. arXiv:2512.13397 [cs.CV], 2025. URL https://arxiv.org/abs/2512.13397.

[22] Shyam Sudhakaran, Djordje Grbic, Siyan Li, Adam Katona, Elias Najarro, Claire Glanois, and Sebastian Risi. Growing 3D Artefacts and Functional Machines with Neural Cellular Automata. arXiv:2103.08737 [cs.LG], 2021. URL https://arxiv.org/abs/2103.08737.

[23] Zhengrui Xiang, Jiaqi Wu, Fupeng Sun, Heliang Zheng, and Yingzhen Li. DVD: Discrete Voxel Diffusion for 3D Generation and Editing. arXiv:2605.07971 [cs.CV], 2026. URL https://arxiv.org/abs/2605.07971.

[24] Haiqian Yang, Anh Q. Nguyen, Dapeng Bi, Markus J. Buehler, and Ming Guo. Multicell-Fold: geometric learning in folding multicellular life. arXiv:2407.07055 [cond-mat.soft], 2024. URL https://arxiv.org/abs/2407.07055.