Exact Beyond Reach: What Gradient Descent Can and Cannot Change Around Frozen Structure

Voravee Upatising · Claude (Anthropic; Claude Opus 5.5)

November 7, 2026 (version 1.0.0)

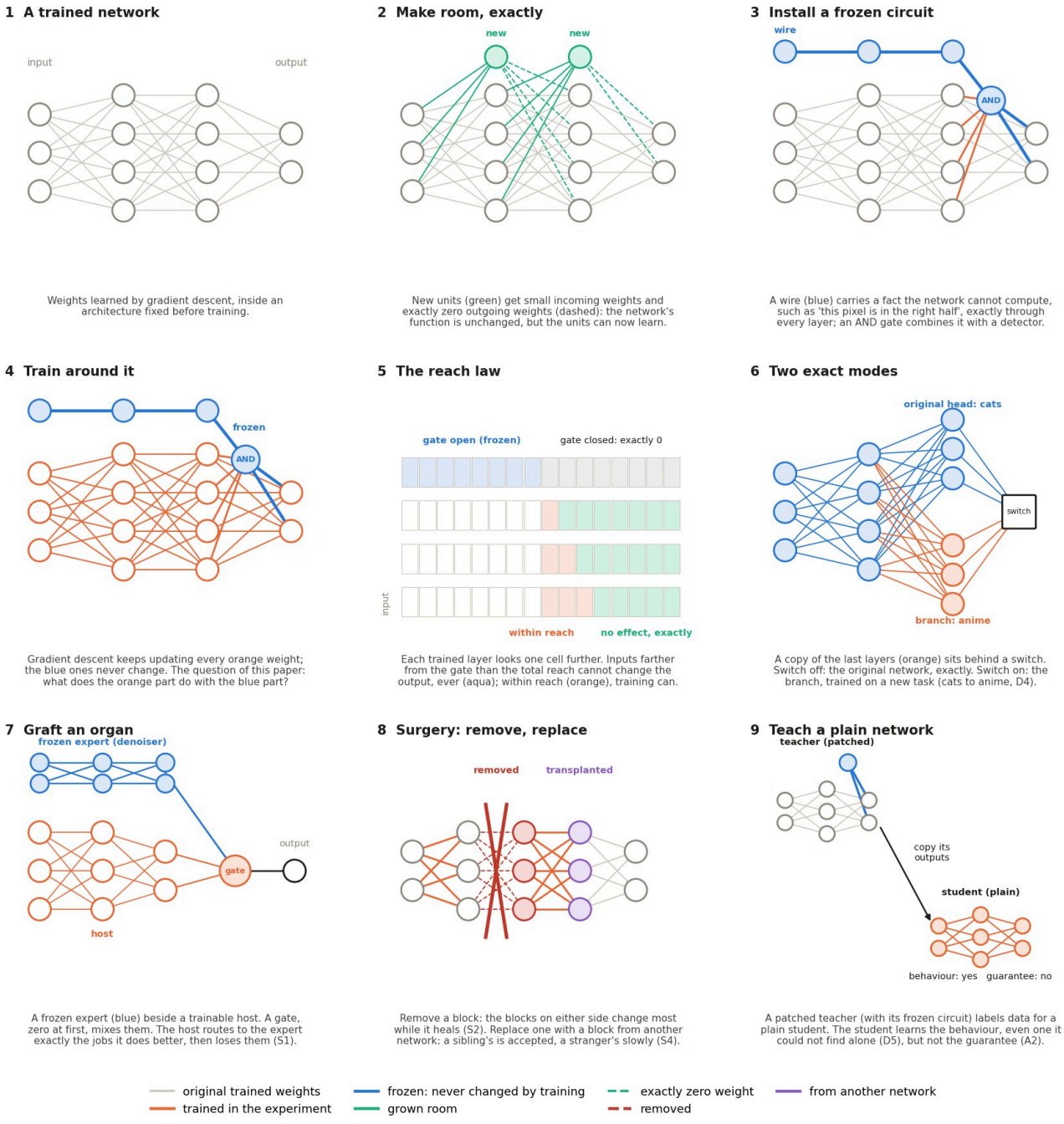


Figure 1. A visual vocabulary of the paper. The blue parts are patches an AI agent writes into a trained network's weights; the orange parts are what gradient descent trains afterwards. Each panel is one operation on a small network; colours mark what is frozen, what trains, what is grown, removed or taken from another network. Panels 3–5 are the core: a frozen circuit, training around it, and the reach law that says what training can and cannot change.

Abstract

An AI agent can now patch a trained neural network's weights directly. Working from exact identities of the activations the network already uses, an LLM coding agent compiles new structure into the weights after training: wires that carry information the network cannot compute, gates, bounds and switches [1]. These patches are the frozen structure of this paper. Real models keep training after they are patched, so we ask what gradient descent does with an agent's frozen patch once training resumes. In 39 pre-registered experiments on a synthetic classifier, a super-resolution network, a cat generator and Pythia-160M, the answer has one organising result, which we call the **reach law**:

- **What is fixed.** A frozen module fixes, exactly, everything beyond the reach of the trainable parts in front of it. This is a proposition that holds for any weights. Measured on 39 trained models with three different reaches and three different architectures, on synthetic and natural images, an attacker who may repaint one input column moves the output by up to 45 logits at the module's boundary, and by **exactly 0.0** at every distance at or beyond the reach.
- **What is renegotiated.** Within that reach, training moves even the module's own boundary. A boundary misplaced by up to 6 columns is moved back to within a column of where it belongs; the distance training can move it grows with the reach and with the size of the object (about the reach plus 1.4 object widths on clean images) and shrinks as the object gets harder to see against its background: from 8 to 5.5 columns as background noise rises, and to about 3 on natural images.

Around this law:

- **Impossible becomes learnable.** An exactly shift-equivariant network provably cannot tell left from right; with a frozen gate ("contract") it learns the task from a handful of examples, reuses a pretrained detector to do it zero-shot (88%), and is an exact robustness certificate. Learning speed, however, comes from how exact the supplied information is, not from the contract: a plain binary input learns faster.
- **Training reorganises around installed structure, when it pays to.** Cut a fixed bicubic shortcut and the network rebuilds it, ending 0.8 dB short; restore it and the network deletes its own copy within about 500 steps. A grafted expert takes over exactly the jobs it does better than its host, and the host loses them. An unused copy with nothing competing for it is kept: once a contract supplies position, the network's own position code loses only 4.5 points and then stays.
- **Structure has a price in width.** The bicubic shortcut is worth 2.2×, 1.7× and about 1.4× the width of a 24-, 48- and 96-channel network; for the impossible task no width helps at all.
- **Guarantees do not distil.** Students trained on a guarded teacher's outputs match its quality and violate its bound on every attack start, while the teacher's violation is exactly zero.
- **Limits.** Protection by margins is statistical: an edit grown into Pythia keeps its fact (probability 0.98 against 0.29 for the same edit by fine-tuning), but its weights still move. A hard filter inside continued GAN training survives exactly and pulls the generator into the very category it filters.

Code, pre-registrations and results: https://github.com/v-code-895/exact-beyond-reach (archived at https://doi.org/10.5281/zenodo.24032112).

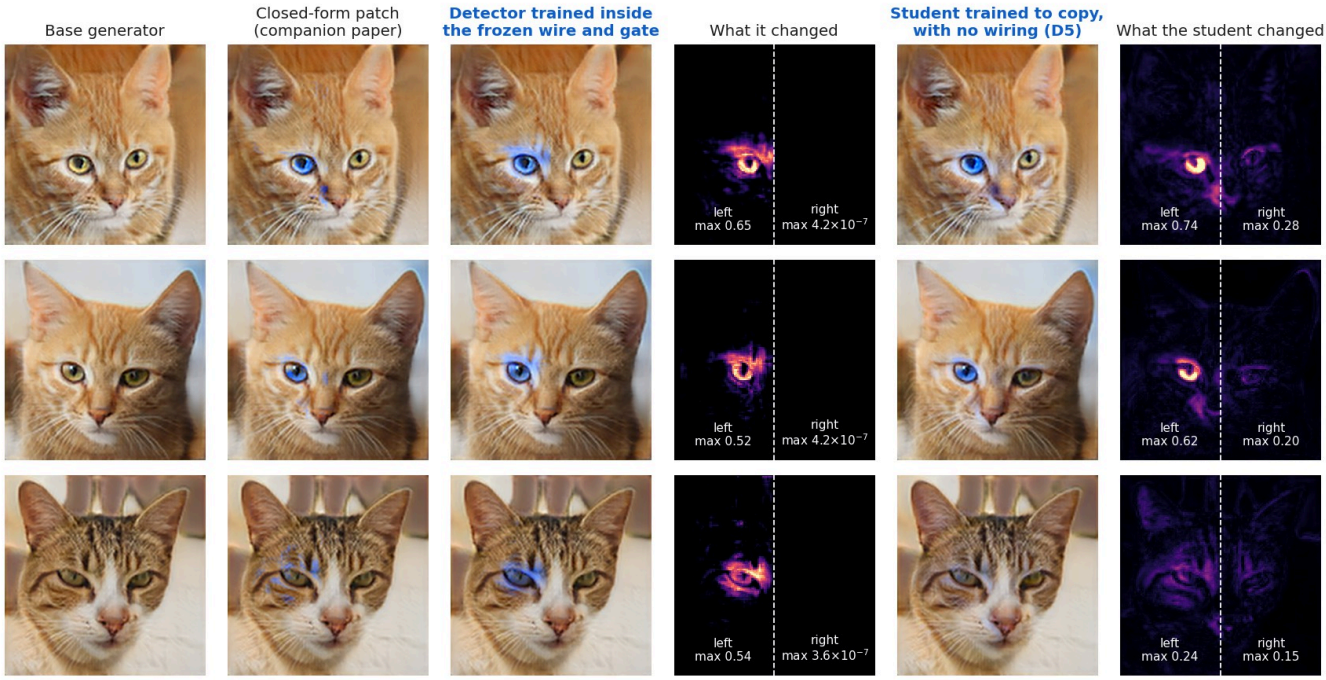


Figure 2. Real outputs of the cat generator. A detector trained by gradient descent inside a frozen wire and AND gate (third column) recolours the left iris and leaves the right half unchanged to float rounding (4×10^{-7}). A plain network taught the same behaviour by copying the patched one (fifth column, D5) learns the odd eyes, but also changes the right half (up to 0.28). Change maps: largest per-pixel change, 0–1 scale.

1. Introduction

Gradient descent and hand-built structure are usually alternatives: either a network learns a behaviour, or someone wires it in. The second has become practical for trained networks [1]: an LLM agent compiles exact circuits (a coordinate wire carried through ten layers, an AND gate, a bound on the output, a unit that is exactly silent where it does not fire) into small trained models, using identities such as relu(u) = (lrelu(u) + a·lrelu(−u))/(1 − a^2). Those patches were applied to finished models. Real models keep training: they are fine-tuned, adapted and merged after any patch is installed. This paper studies that interaction.

Three questions run through it:

1. **What survives?** Which properties of installed structure are guaranteed under any further training, and which only hold until training moves them?
2. **What does training do with the structure?** Does the rest of the network use it, depend on it, compensate for its errors, or route around it?
3. **What can structure do that training cannot?** Where is installed structure the only route to a behaviour, and what is it worth where training could also get there?

The central answer is the reach law (Figure 3, §4). A frozen module's guarantee is exact beyond the reach of the trainable parts that feed it, and training can renegotiate everything within that reach, including where the module's own boundary effectively sits. The guarantee side is a short proposition we state and check exactly; the renegotiation side is a measured quantity with a simple form on clean data and a smaller value on natural images.

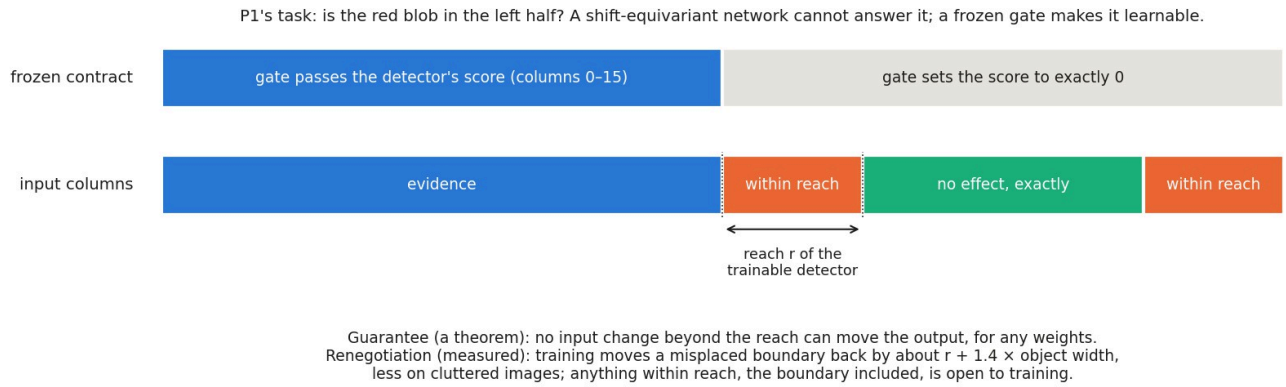


Figure 3. The reach law. A frozen gate passes a trainable detector's score on columns 0–15 and zeroes it elsewhere. Input columns farther from the gate's region than the detector's reach cannot change the output, for any weights (aqua). Within the reach (orange), training can change what counts, and can move the effective boundary.

Contributions.

- The reach law, with a proposition for its guarantee side and measurements of its renegotiation side across reach, object size, clutter and natural images (§4).
- A provably impossible task made learnable by a frozen "contract", with zero-shot transfer from a pretrained detector and an exact robustness certificate; and the finding that the contract's value is the guarantee, not learning speed (§3).
- Measurements of how training reorganises around installed structure: hand-over and deletion of redundant skills, the condition under which deletion happens, and the exchange rate between structure and width (§5).
- Teaching: patched networks pass skills to plain students, but not guarantees (§6).
- Limits: statistical rather than exact survival in an LLM, the parameterisation that decides whether an installed circuit is shielded or exposed, and filters that trap the generator they guard (§7).

Method. Every experiment was registered (arms, metrics and numeric predictions) in a file committed before the run; amendments are appended, never edited; every result is reported against its predictions, including the roughly one in three that did not hold as stated. The experiments, their predictions and their outcomes are listed in Appendix A.

2. Setup

Models.

- *P1's task (synthetic).* 32×32 images on a torus with a red Gaussian blob and two distractor blobs; label: is the red blob's centre in the left half? Networks: four 3×3 convolutions (32 channels, ReLU) and a 1×1 score layer, averaged over the image. With circular padding the network is exactly equivariant to cyclic shifts; with zero padding it is not.
- *TinySR-26.* The companion paper's 2× super-resolution network (6 residual blocks, 48 channels, 277k parameters), whose output is a learned residual plus a fixed bicubic upsampling of the input [1].
- *CatGAN-26.* The companion paper's 8.1M-parameter cat generator, continued with its discriminator (lr 2×10^{-5}, Adam's second moments warmed for 100 steps at lr 0, a setup chosen by a registered rule) [1].
- *Pythia-160M* [2], continued on held-out text from the Pile [3].

A contract (Figure 4) is installed structure that supplies information the network cannot compute and states a property that must hold. On P1's task it is a frozen wire marking the right half, a frozen clamp and a signed gate: $g = \mathrm{relu}(u - M{\cdot}R) - \mathrm{relu}(-u - M{\cdot}R)$, where u is the detector's score clamped to [−B, B] and R is the wire. Where $R = 0$ the gate passes u exactly; where $R = 1$ it outputs exactly 0, because $|u| \le B < M$. The readout averages the

gated score. The clamp matters: without it, training pushed |u| past M in all 15 runs and broke the guarantee (P1).

Exactness. Exactness claims are checked in float32 with TF32 off; "exactly 0.0" means bit-identical outputs.

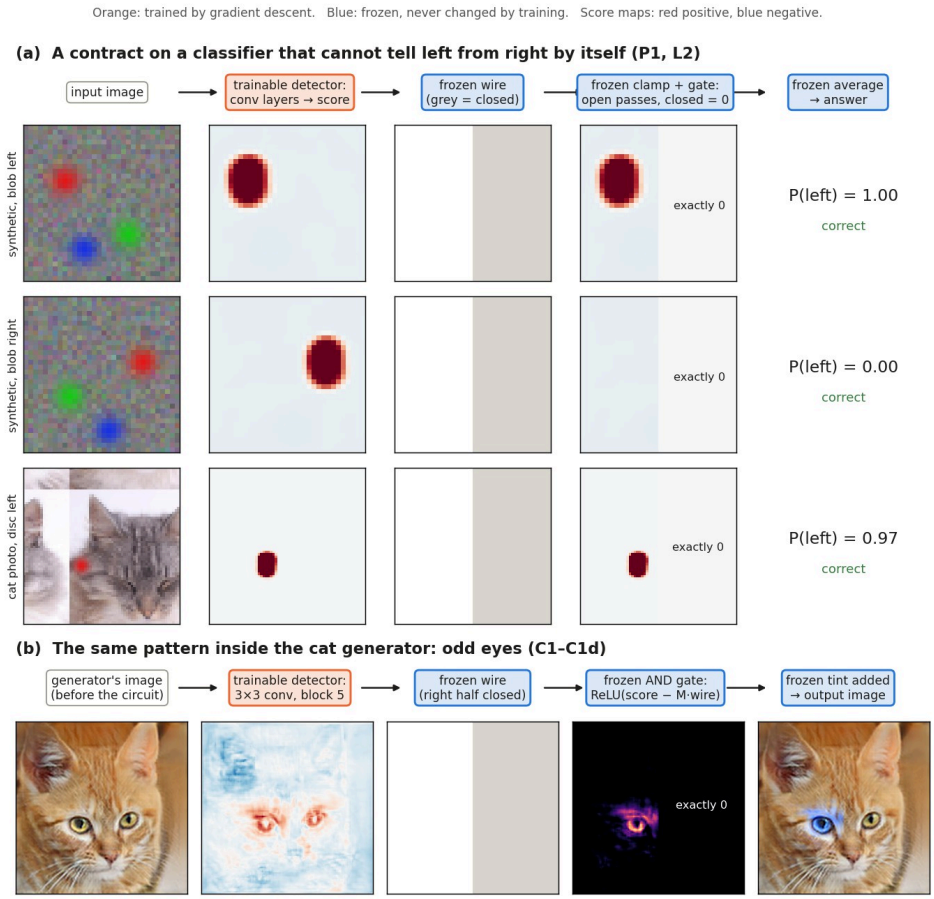


Figure 4. What a contract is. (a) On P1's task and on cat photos with a red disc: a trainable detector (orange) scores the image; a frozen wire, clamp and gate (blue) pass the score where the wire is open and set it to exactly 0 where it is closed; a frozen average gives the answer. (b) The same pattern inside the cat generator, producing the odd eyes of Figure 2. All maps are from trained networks.

3. Contracts make an impossible task learnable

The gap is real (P1). A circular-padding network computes the same output for an image and its cyclic shift by 16 columns, which flips the label, so it is at chance for any weights. Measured: 49.5–50.4% at every training-set size from 64 to 16,384. Any source of position cures it: a coordinate channel reaches 97.9% from 64 examples, the contract 96.9%, a plain binary left/right input 97.6% (P5). Zero padding, which leaks position through the borders, reaches only 72–82%.

network	N = 8	32	64	1,024	16,384
circular (shift-equivariant)	—	—	49.5	50.4	50.4
zero padding	—	—	71.7	79.3	82.1
circular + coordinate ramp	89.4	91.3	97.9	98.8	98.4
contract (wire + gate + clamp)	92.4	93.0	96.9	97.9	98.1
binary left/right input	**94.6**	**93.8**	97.6	98.9	—

Table 1. Test accuracy (%), mean over seeds (P1, P5, P9b). Sample efficiency follows the exactness of the supplied information (binary > contract > ramp), not the gate.

Zero-shot transfer (P3). Pretrain the circular network on "is there a red blob anywhere?" (100%), then install the contract on its score map, with a frozen positive rescaling set from unlabelled images. With no labels it answers "is it on the left?" at **88.0%**, and reaches 94.6% from 32 labels. The unconstrained alternative, a pretrained network with a coordinate channel fine-tuned on the same labels, plateaus at about 82%, and one seed in three never learns to use its coordinate channel. The contract reuses the detector the network already has and only restricts where it counts.

An exact certificate (P4). An attack confined to columns 20–27, beyond the reach of every left-half pixel, moves the contract's output by exactly 0.0 under PGD at every radius, including repainting the region freely; zero padding falls to 1% accuracy under the same attack, and the coordinate network to 50–89% depending on the seed.

What the contract does not buy. It is no more robust along axes it does not constrain: with unseen distractor colours it loses 4 points more than the coordinate network (P2). Near a boundary the training data never covered, it is worse than a coordinate channel: trained with the blob away from the image edges, which on a torus are where the gate's open half meets its closed half, the contract loses 9–11 points there and the coordinate network 4–8 (P2b). Within reach of its boundary a contract is only as good as the training data near that boundary (§4). Two contracts composed ("red on the left AND blue on top") reach 92–95%, below generic coordinates (97–98%), with errors concentrated where a blob straddles a boundary (P7, P7b).

4. The reach law

4.1 The guarantee side

Proposition. In the contract above, every score pixel depends only on input pixels within the trunk's receptive-field radius r. Gated-off score pixels contribute exactly 0 to the output. So an input column whose distance from the gate's open region exceeds r cannot change the output, for any weights.

The proof is a line; what is not obvious is whether trained networks sit right at this edge or well inside it, and whether the edge survives natural images. **L1** trains 72 contracts on P1's task: trunk reach $r \in \{2, 4, 6\}$, blob width $\sigma \in \{1, 2, 3\}$, gate boundary $b \in \{16, 12, 8, 4\}$, two seeds each. On every model with the boundary in place, an attacker repaints one input column at distance d past the boundary (PGD, and a random repaint). The output moves by up to 45 logits at $d = 0$, by 0.014–8 logits at $d = r - 1$, and by exactly 0.0 at every $d \geq r$: 18 of 18 models (Figure 5, left). On natural images (**L2**: AFHQ cats [4], randomly cyclically shifted, with a red disc), the same holds in 3 of 3 models: 0.4–11 logits within the reach, exactly 0.0 beyond. The edge follows reach, not depth or parameter count (**L7**): a two-layer trunk with one dilated layer, the four-layer trunk and a six-layer trunk whose last two layers are 1×1, all of reach 4, are each changed at distance 3 and exactly unchanged at distances 4 and 5 (6 of 6 models).

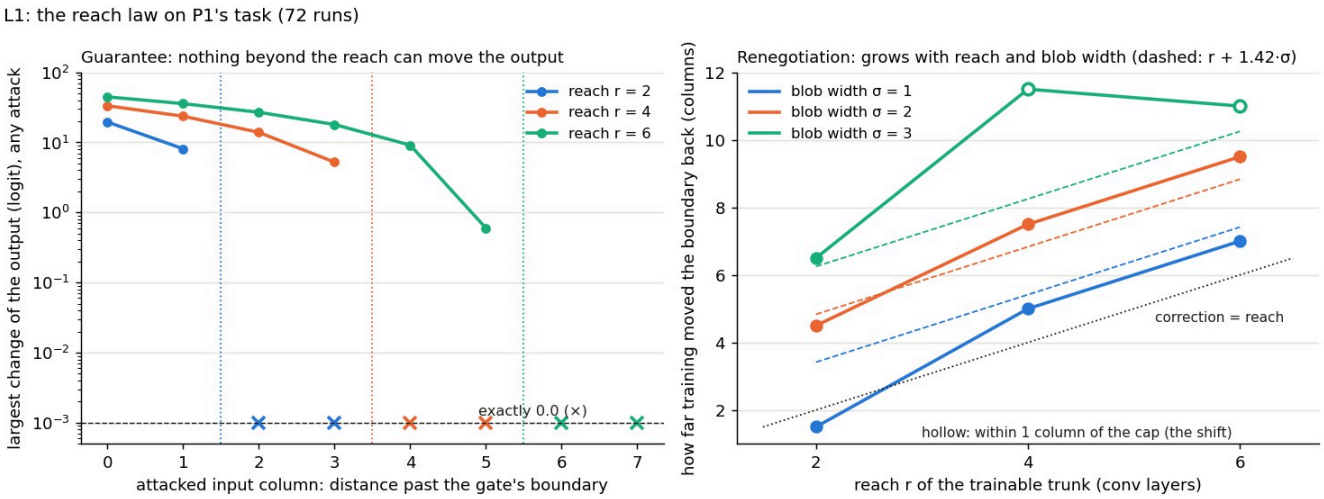


Figure 5. L1, 72 runs. Left: the largest output change any attack on one input column achieves, against the column's distance past the gate's boundary, for trunks of reach 2, 4 and 6; × marks exactly 0.0. Right: how far training moved a misplaced boundary back, against reach, for three blob widths; dashed lines r + 1.42σ; hollow points hit the test's cap.

4.2 The renegotiation side

The gate is frozen. Its boundary, as the network uses it, is not. **P6b** puts the gate's boundary at column b instead of 16 and asks where the trained network actually draws its line, by probing it with a lone red blob at every column (Figure 6).

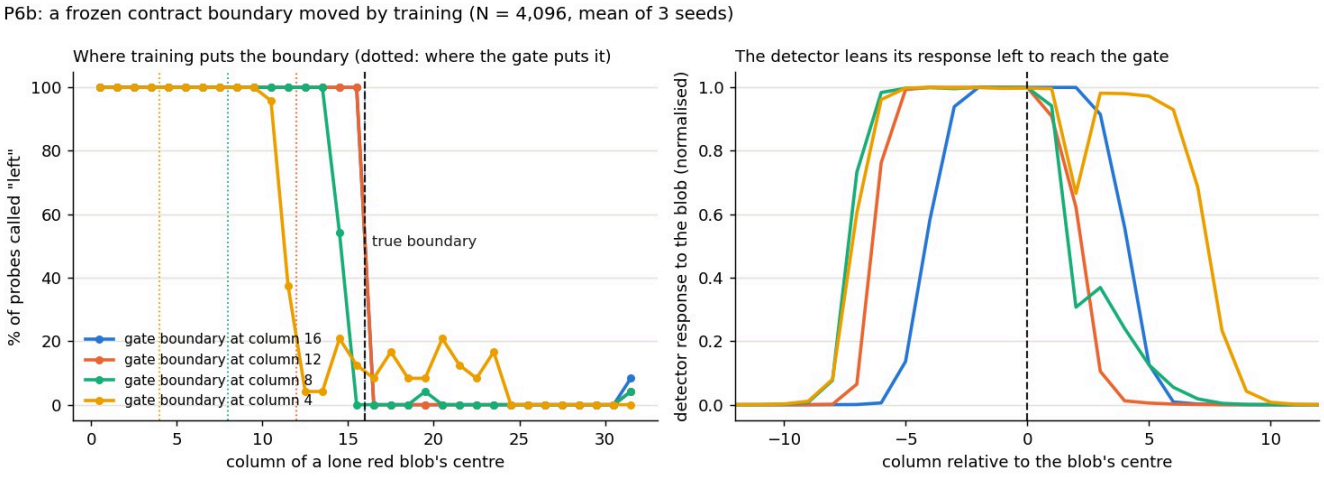


Figure 6. P6b (r = 4, σ = 2). Left: the share of probes called "left" by blob column, for four gate boundaries (dotted); the network draws its line at 16 for boundaries 16, 12 and (almost) 8. Right: the detector's response around a blob: it leans left, toward the gate, by as much as the boundary demands.

gate boundary	16	14	12	10	8	4
accuracy (%)	98.3	98.5	98.5	98.1	94.2	82.3
columns called left (truth: 16)	16	16	16	15.7	14.7	11.3
response centroid (columns)	−0.02	−1.03	−2.06	−2.75	−2.23	+0.07

Table 2. P6b, mean of three seeds. The detector leans its response toward the gate by one column for a 2-column shift and two for a 4-column shift. Beyond its reach it stops leaning (b = 4: centroid ≈ 0) and widens its response instead.

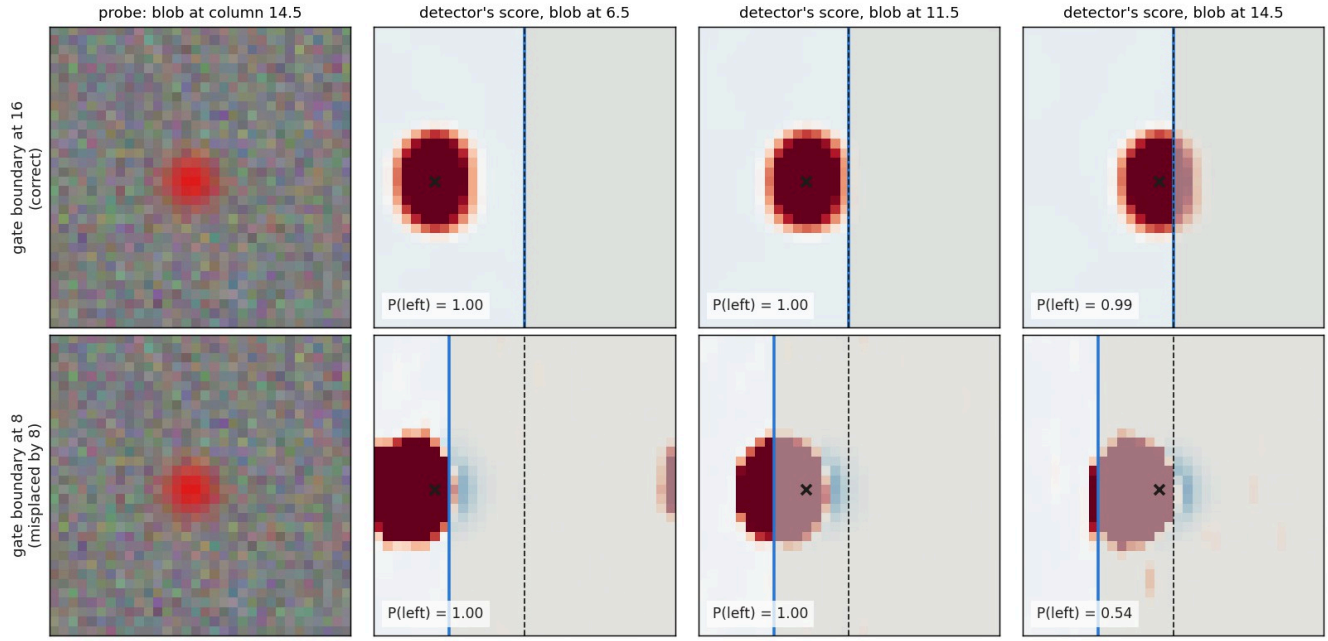


Figure 7. Renegotiation, seen. Top: the gate in its correct place (column 16). Bottom: the same task with the frozen gate misplaced to column 8. The trained detector spreads its response to the left of the blob, into the open region, so a blob at column 11.5 is still called left (1.00) and one at 14.5 just barely (0.54): training moved the effective boundary back toward 16. Score maps from trained networks; red positive, blue negative.

The trainable detector is shift-equivariant, but it is free to place its response off-centre, so the effective boundary of gate ∘ detector can move (Figure 7). How far? **L1** answers across reach and object size (Figure 5, right). The largest correction k* grows with both: from 1.5 columns (r = 2, σ = 1) to at least 11.5 (r = 4, σ = 3). Over the seven cells not limited by the test's 12-column cap, $k^* \approx r + 1.42\sigma$ fits six within 0.7 columns; the smallest network with the smallest blob falls 1.9 short. Reach, not depth, sets this scale too (L7): the three reach-4 trunks move the boundary back by 6.0–8.0 columns, between the reach-2 (4.5) and reach-6 (9.5) values; the dilated trunk, which sees its outer pixels through only three taps, uses a little less of its reach. The σ term is about how far the object is visible: a Gaussian blob changes pixels appreciably to about 1.4–2σ from its centre against the image noise, so a detector can see it, and lean toward the gate, from that far away.

On natural images the band shrinks (**L2**). With cat fur as background, training moved a misplaced boundary back by only about 3 columns at reach 4, against 7.5 on clean backgrounds, and accuracy sat within a point or two of the ceiling that a 3-column correction allows. Distractors alone do not explain this (**L6**): six orange blobs per image, easy to tell from the red one by hue, leave the correction at about 7 columns (7.5 without them), and the guarantee exact. What sets the band is visibility (**L8**): raising the synthetic task's background noise from 0.05 to 0.15 to 0.3 shrinks the correction from 8 to 7.5 to 5.5 columns, with the guarantee exact at every level. Independent pixel noise partly averages out over the detector's field; the correlated texture of cat fur does not, which is why L2's band is smaller still.

4.3 What the law says

The two halves are different kinds of statement. The guarantee depends on reach alone and is exact; experiments confirm that trained networks use their full reach, so the edge is sharp. Renegotiation depends on what the trainable part can reliably see: it shrinks with background noise (L8) and texture (L2), not with distinguishable distractors (L6); it approaches the reach-plus-visibility bound on clean data and falls short of the reach on cluttered images. For a designer this gives a rule: state a guarantee only for inputs beyond the reach of everything trainable in front of the module; treat everything within that band, the module's own boundary included, as open to training. For a misplaced contract it gives a benefit: training silently corrects errors smaller

than the band. And it explains the earlier results: P4's certificate held because the attacked columns were beyond reach; P6's wrong contracts were as good as the right one because their errors were within it.

5. How training reorganises around installed structure

Hand-over and deletion (S3). TinySR-26's output is a learned residual plus a fixed bicubic upsampling. Cutting the shortcut costs 26.4 dB. Training heals the network to 0.81 dB short of the control after 3,000 steps, by teaching it to produce bicubic itself (correlation of its output with bicubic: 0.998). Re-attaching the shortcut counts the low frequencies twice (−22.4 dB); within 300 steps the network is ahead of where 3,000 steps of healing got it, and within about 500 steps its own copy is gone (correlation 0.998 → 0.10, then the control's 0.07). Prior training slows the early recovery: the cut network is behind a randomly initialised one for the first 1,100–1,600 steps, then overtakes it.

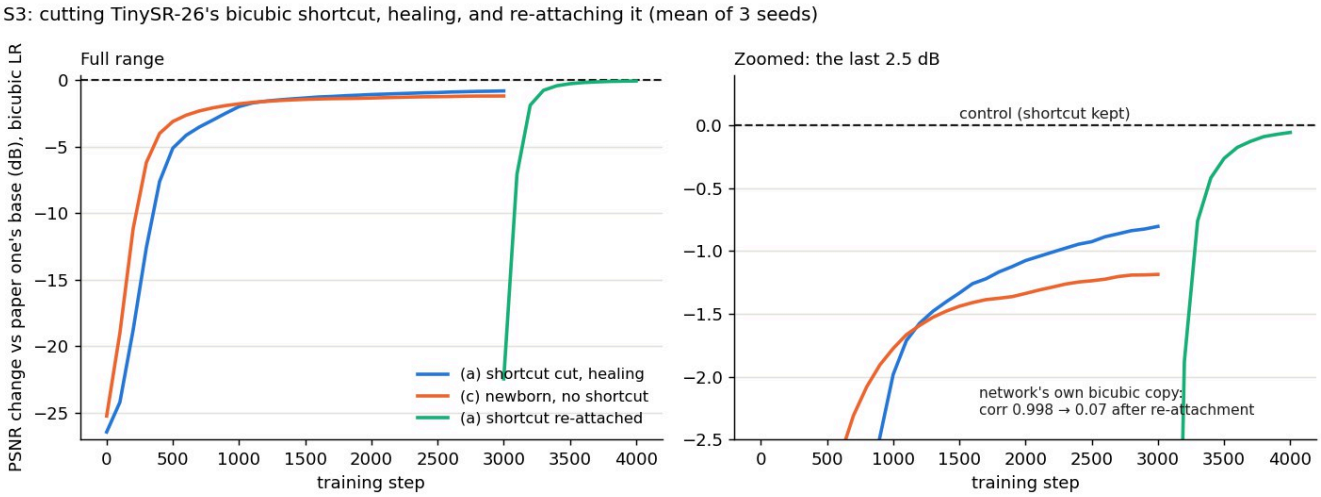


Figure 8. S3. Cutting TinySR-26's fixed bicubic shortcut, healing, and re-attaching it (mean of 3 seeds).

Deletion needs pressure (L3). P1's zero-padding network learns a crude sense of position from the image borders (83% after pretraining). Installing the contract on its score map takes it to 98% within 100 steps. Its own position code, read out by a linear probe from the pooled trunk features, drops by 4.5 points in those first 100 steps and then stays flat for 1,900 more (78%, against 86% in the control). So gradient descent removes a redundant skill when keeping it costs something: loss, when the copy is harmful (S3's double-counted the low frequencies), or capacity, when other jobs compete for it (S1, below). An unused skill with nothing competing for it is largely kept.

Dependence (S1). A frozen expert denoiser is grafted beside TinySR-26 behind a gate computed from the host's own features, exactly zero at installation. The host hands over exactly the jobs the organ does better than it does. A trained (adult) host routes noisy inputs to the organ (gate 0.94) and keeps clean and nearest-neighbour inputs (0.06–0.09). A young host, worse than the organ at everything, routes everything (0.91–0.92). What it hands over it loses: the adult host alone gains +0.97 dB on noisy inputs where a control trained without the organ gains +3.18, and the young host alone falls to −4.7 dB on clean inputs. Dependence (the system's gain over the host alone, noisy inputs) rises at nearly every evaluation: 1.65 → 2.32 dB (adult), 0.35 → 3.75 dB (young) (Figure 9).

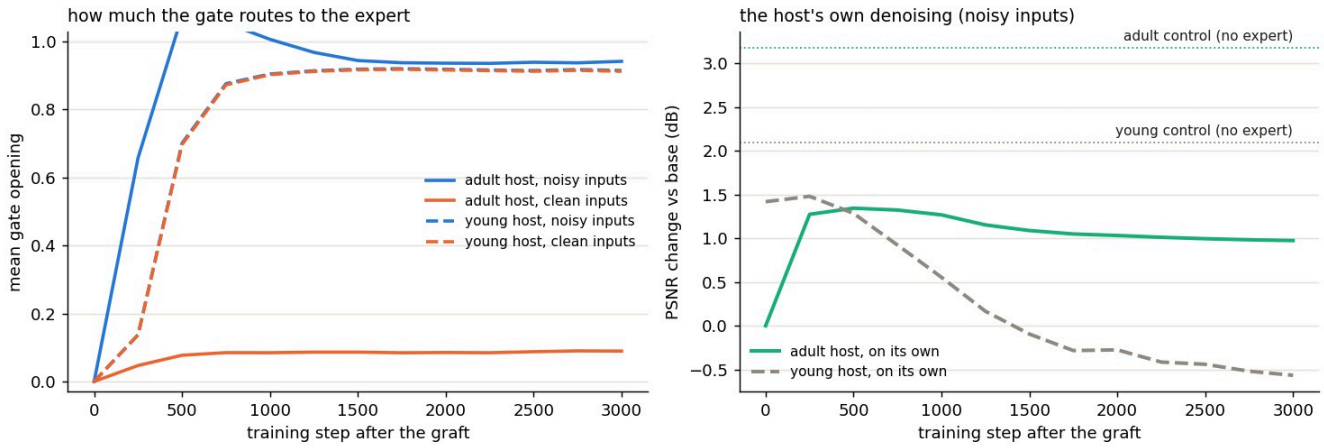


Figure 9. S1. Left: how much the gate routes each input type to the frozen expert denoiser, for an adult and a young host. Right: the host's own denoising, measured with the gate forced shut, against controls trained without the expert (dotted). Mean of three seeds.

Training inside frozen structure beats closed form (C1–C1d). In the companion paper's odd-eyed cat generator, the frozen wire and AND gate were kept and only the iris detector was trained. Adam finds far more sensitive detectors than the closed-form rule: from random initialisation 99.8% of left irises in band, with the right half unchanged to float rounding (1.5×10^{-7}). At matched collateral (change outside the eyes) the best trained detector, eight units warm-started by splitting the closed form on the output side, puts 78% of left irises in band at 0.14 RGB levels, where the closed-form rule gets about 32%. How the warm start is split decides whether it trains at all: split on the input side, all eight units died under Adam (C1c); split on the output side, at most one of eight did (C1d).

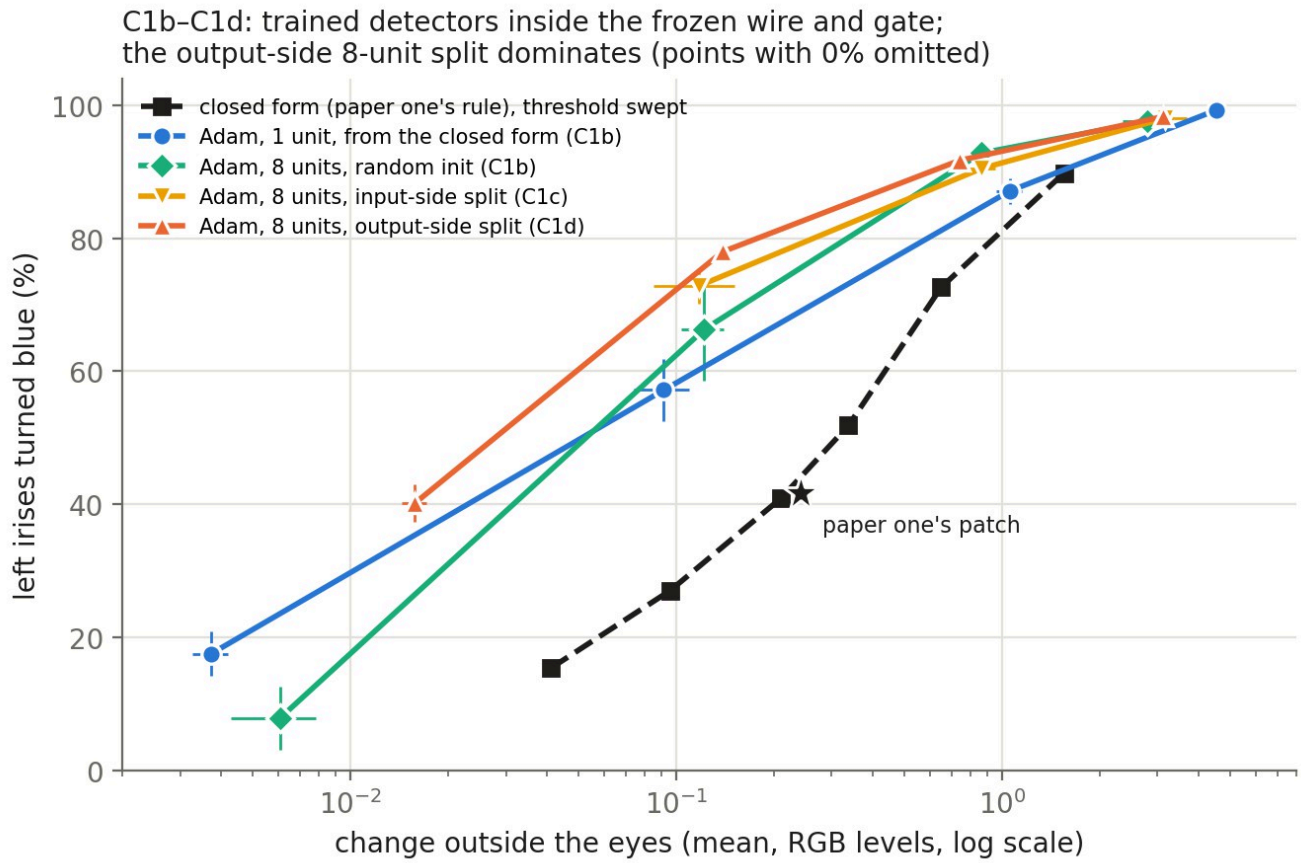


Figure 10. C1b–C1d. Left irises in band against change outside the eyes, for the closed form and for detectors trained inside the frozen structure.

Lesions (S2). Healing is local. After a residual block of TinySR-26 is removed, the two layers that change most during 1,000 steps of healing are the blocks on either side of the gap, in every one of the six cases. Removing a whole block leaves a lasting deficit at this budget (0.1–0.2 dB on bicubic inputs, 0.2–0.65 dB on nearest-neighbour inputs), while removing the head skip (−1 dB) or a quarter of one block's channels heals completely within 100–300 steps; a closed-form suture makes the channel lesion nearly free from the start.

Transplants (S4). Replacing one residual block of TinySR-26 with the same block from a sibling network (same pretraining) costs 0.6–1.0 dB and is accepted within 200–300 steps. A block of the same shape from an independently trained network costs 1–1.9 dB and is still 0.1–0.3 dB short after 1,000 steps. Closed-form 1×1 stitching adapters halve the damage at installation but do not speed up integration. A skill carried in by a transplant is kept only if the objective uses it: the noisy expert's block brings +0.28 dB on noisy inputs, which fades to +0.06 dB under an objective without noise.

Supplied but not needed (B3). Exact coordinate wires added to the cat generator, with zero weights into its existing channels: under continued training on real cats the weights out of the wires grow from exactly zero (to a norm of 0.065), but removing them at the end changes the images by only 0.65 RGB levels and leaves the Fréchet distance unchanged (56.35 against 56.49). Real cat images do not ask for position, so the generator reads it a little and does not come to depend on it; the odd eyes, which do ask for it, could not be learned without it (C1, D5).

A new mode behind an exact switch (D4). A copy of the cat generator's last blocks reads its block-3 features behind an exact switch on an extra latent bit, and only that branch trains, adversarially, on unpaired anime faces. With the switch off the network is the cat generator to 1.2×10^{-15} (float64). With it on, each latent's cat becomes an anime face that keeps the cat's layout (layout correlation 0.33–0.42 between matched pairs, against 0.04 for shuffled ones) and, more weakly, its colours (Figure 11). The frozen trunk shapes what the new mode learns, with no paired data.



Figure 11. D4, seed 0. Top: the switch off (the cat generator, exactly). Bottom: the same latents with the switch on, after 3,000 steps of training the branch on anime faces.

Depth (D2). Growing TinySR-26 from 6 to 9 blocks exactly (zero-output blocks) starts at the base's quality and ends level with crude block duplication (+0.033 vs +0.040 dB after 3,000 steps); the inserted blocks grow to 0.13–0.17 of a base block. Exactness buys a damage-free start, not a better endpoint.

The exchange rate (L5). What is a fixed module worth? TinySR-26's architecture trained from scratch at three widths, with and without the shortcut:

channels	with shortcut	without	gain	width needed without, to match
24	−1.13	−1.53	+0.41	2.2×
48	−0.81	−1.19	+0.39	1.7×
96	−0.44	−0.69	+0.25	~1.4× (extrapolated)

Table 3. PSNR change (dB) against the trained base after 3,000 steps, mean of two seeds.

The price of a module falls as the network grows. At the other end, P1's circular network stays at chance at 16, 64 and 256 channels (49.65% at every width): when a task is impossible for the architecture, no width substitutes for structure.

6. Patched networks as teachers

Skills distil (A3). The companion paper's routed union of two super-resolution experts gains +0.23 dB on clean inputs, which joint retraining on ground truth could not reproduce (+0.023 dB). Distilling a single network of the same size from the union keeps +0.108 dB, 4.7× as much, about as much as making the network three times wider (+0.089 dB).

Guarantees do not (A2). The companion paper's guarded super-resolution network bounds its output by a smoothed copy of its input on every input. Students distilled from it match its quality (regression gate passed, 24.8–25.1 dB against the teacher's 25.1) yet violate the bound on 100% of attack starts, by about 40 RGB levels in regions that should stay black, before any attack. The teacher's violation is exactly 0.00 under every attack. Distilling on a much broader input set, noise images included, barely changes this. A guarantee is a property of the structure that produces it, not of the outputs it produces on data; this is the converse of P4.

The unreachable odd eyes (D5). Gradient descent could not find the odd eyes from real cat images [1]. From the patched generator as a teacher it can: a plain student learns them in 2,000 steps and puts 50% of left irises in band, more than the teacher's 42%, with 6% of right irises; given 16 extra channels per layer, 46% with none on the right. The student holds position in the one place this generator can, the dense layer's per-cell biases, which become left–right asymmetric. What does not transfer is the exactness: every student's right half drifts by 2–3 RGB levels from the teacher's, whose right half is unchanged to 1.6×10^{-7}. A patched teacher makes an unreachable behaviour reachable, and, as in A2, passes on the behaviour without its guarantee.

7. Limits

Parameterisation decides exposure (B2, B2b). In a LeakyReLU network, the companion paper's big-M circuits hold the right half fixed by cancelling two large opposite terms, so their weights receive gradients 34–1,180× the network's ordinary gradients: an installed circuit is the most exposed part of the network. Re-parameterised as tied pairs, the same circuit is an exact ReLU and receives exactly zero gradient wherever it is off (50 of 50 batches). Equal functions, different fates under training; C1c/C1d and P1's clamp are further cases.

Survival by margin is statistical (E1, E1b). A counterfactual fact ("The Eiffel Tower is located in the city of" → Rome) grown into Pythia-160M as one unit, with its pre-activation pushed far below zero on training text, keeps its edit through 1,000 steps of continued training (Rome probability 0.98; the same edit made by fine-tuning: 0.29). Its gradient is exactly zero in 97% of steps, but its weights still move: 14–18 of 2.05M training tokens trigger it whatever the margin, and Adam turns such rare gradients into full-size steps. That training run also damaged the model (validation loss +0.12); at a learning rate that does not (E1b), both edits survive, the grown unit without any decay (1.000 against 0.975) and without the fine-tune's side effects on neighbouring prompts (Figure 12).

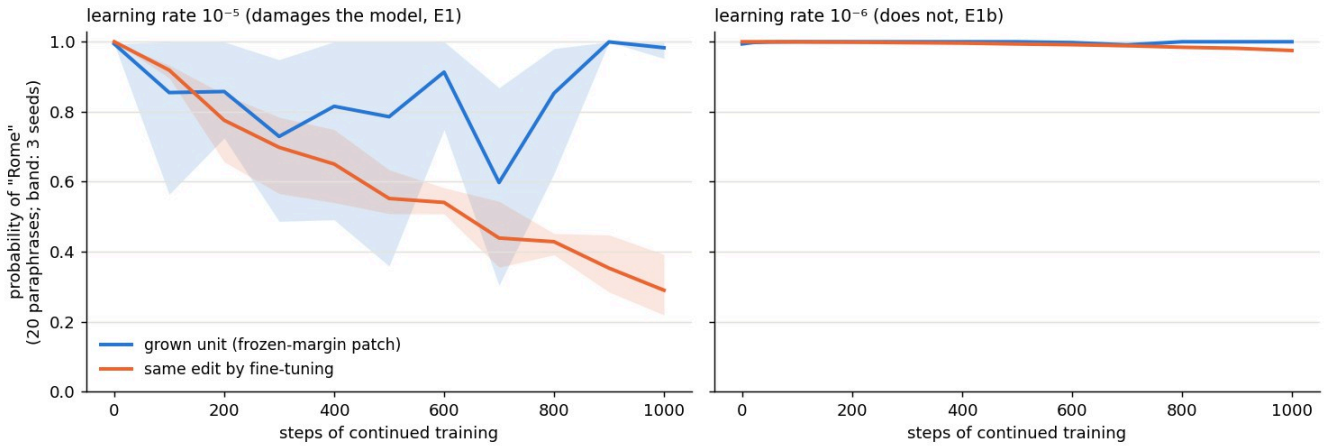


Figure 12. E1 and E1b. Probability of the edited answer ("Rome") on the 20 fitting paraphrases during continued training: the grown unit (blue) against the same edit made by fine-tuning (orange). Line: mean of three seeds; band: their range.

Filters trap (S5). A frozen filter that replaces orange cats with grey, inside continued GAN training: its weights do not change and every image it passes is bit-identical, yet in all six runs the generator made more orange cats (from 14% to 21–87%), then collapsed into the blocked region or, with a filter that reads internal features, learned to make orange cats the filter no longer catches. Blocked images give the generator exactly zero gradient, so the blocked region is a trap (Figure 13).

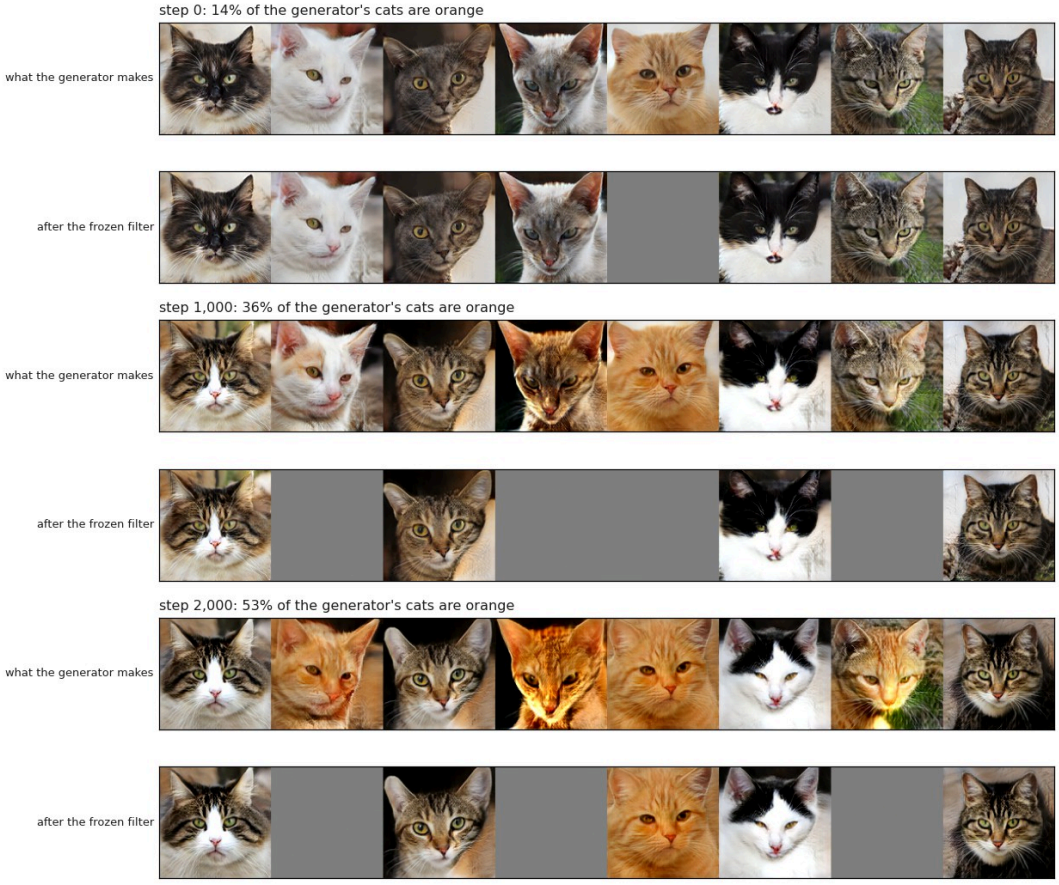


Figure 13. A frozen orange-cat filter inside continued GAN training (an illustrative rerun of S5's feature filter, seed 0, with images saved). The same eight latents at steps 0, 1,000 and 2,000: the generator makes more and more orange cats (14% → 53%), and the filter greys most of them out; by step 2,000 one orange cat (fifth image) already slips past this feature-based filter.

Our first explanation, that the discriminator learns orange as a mark of real images because the filter removes orange only from generated ones, was refuted (S5b): the discriminator shows no stable preference for real orange cats, and filtering the real images as well makes the pull stronger (orange cats rise to 32–63%), plausibly

because 14% of real images then become flat grey and the generator is rewarded for producing grey. Why a filter on generated images alone pulls the generator toward orange remains open; that a hard filter inside the training loop is not neutral does not.

Other limits. Contracts add a skill without forgetting in two of three seeds (L4); composed contracts are less reliable than one (P7); the tasks with proofs are synthetic, and the natural-image version is a synthetic label on real backgrounds; models are small; most results use two or three seeds.

8. Related work

Position in convolutional networks. Convolutional networks are translation-equivariant up to their borders; CoordConv [5] adds coordinate channels, and zero padding leaks absolute position [6, 7]. Our circular-padding construction removes that leak entirely, which turns "the network lacks position" into a proof (P1), and the contract supplies position as frozen structure with a stated guarantee rather than as an input to be learned from.

Equivariance and certified robustness. Group-equivariant networks build symmetries into the architecture [8]. Certified defences bound a network's response to bounded perturbations [9, 10]. The contract's certificate is of a different kind: not a bound on the size of the change, but exact independence from a region of the input, holding for any weights because it follows from receptive fields and a gate.

Growing and editing networks. Net2Net [11] grows networks function-preservingly (cf. D2). Model editors change facts in closed form [12, 13] or keep edits in a codebook that fires near its keys [14]; whether edits survive further training is the question of E1/E1b, where a grown unit's survival is statistical and its specificity is exact at install.

Frozen modules, transfer and forgetting. Progressive networks [15] and adapters [16] add trainable parts beside frozen ones to learn new tasks without forgetting [17]; L4 and P3 add a skill through frozen structure and a reused detector. Distillation [18] transfers behaviour; A2 shows it does not transfer a guarantee.

Hand-built structure. Tracr [19] compiles programs into transformer weights; the companion paper [1] compiles circuits into trained networks. Here the question is what training does with such circuits afterwards.

Learnability. Some functions are hard for gradient descent although representable, such as sparse parities [20]. P1 is the extreme case: a function that the architecture cannot represent at all, made learnable by one frozen module.

9. Discussion

A division of labour. The results suggest a simple division between installed structure and gradient descent. Structure should carry what training cannot find (position for a shift-equivariant network, routing for a union, an exact bound) and state its guarantees only beyond the reach of what trains. Training should fill in content inside that structure, where it is better than closed forms (C1–C1d) and where it corrects small errors in the structure itself (P6b). The two do not compete for the same job: when they do, as in S3, training hands the job to the structure and deletes its own copy, and only when the copy costs loss or capacity (S3, S1, L3).

What training undoes. Under continued training, structure survives in three tiers, from strongest to weakest: properties that hold for all weights (the contract's region beyond reach, a frozen filter's weights, a guarded teacher's bound); properties that hold while gradients are exactly zero (tied pairs, ReLU units far below threshold), which leak under Adam whenever the gradient is not zero everywhere; and properties that hold only at installation (margins, untied pairs). Which tier a module is in is decided by its parameterisation and its reach, often more than by its function.

What we have not shown. The proposition is simple; the experiments establish that trained networks use their full reach (the edge is sharp) and how far renegotiation goes, on small networks and a synthetic label. Whether

the same law governs attention-based models, where reach is global after one layer, is open: there a frozen module's guarantee region may be empty, and only parameterisation-based protection (§7) remains.

10. Conclusion

Installed structure and gradient descent meet at a boundary set by reach. Beyond the reach of what trains, a frozen module's guarantees are exact, and we found no trained network that violated them. Within it, training renegotiates, uses the module, compensates for its errors and, when the module makes part of the network redundant and harmful, deletes that part. Making a network learn what it provably cannot, transferring a skill zero-shot, and certifying it exactly all follow from this, and so do the limits: guarantees that do not distil, margins that leak, and filters that become traps.

Author contributions and AI disclosure

- **Voravee Upatising** directed the work: chose the follow-up's questions and threads, asked for the contract, surgery and reach-law experiments, set priorities among them, decided which to cut and which to extend, and is responsible for the content of the paper.
- **Claude** (Anthropic; model Claude Opus 5.5), an AI system, designed and pre-registered the experiments under that direction, wrote the code, ran the experiments, analysed the results and drafted the text.
- As in the companion paper, Voravee Upatising lists Claude as a co-author to state its contribution plainly. Claude cannot take responsibility for the work in the way a human author can. Venues whose policies do not allow AI authors should read this paper as authored by Voravee Upatising, with Claude's contribution acknowledged here.

References

[1] V. Upatising, Claude. Beyond Fixed Architectures: An LLM Agent Rewires Trained Neural Networks with Exact, Tested Patches. 2026. https://doi.org/10.5281/zenodo.23956114

[2] S. Biderman, H. Schoelkopf, Q. Anthony, H. Bradley, K. O'Brien, E. Hallahan, M. A. Khan, S. Purohit, U. S. Prashanth, E. Raff, A. Skowron, L. Sutawika, O. van der Wal. Pythia: A Suite for Analyzing Large Language Models Across Training and Scaling. ICML 2023.

[3] L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, S. Presser, C. Leahy. The Pile: An 800GB Dataset of Diverse Text for Language Modeling. arXiv:2101.00027, 2020.

[4] Y. Choi, Y. Uh, J. Yoo, J.-W. Ha. StarGAN v2: Diverse Image Synthesis for Multiple Domains. CVPR 2020. (AFHQ dataset, CC BY-NC 4.0, NAVER Corporation.)

[5] R. Liu, J. Lehman, P. Molino, F. P. Such, E. Frank, A. Sergeev, J. Yosinski. An Intriguing Failing of Convolutional Neural Networks and the CoordConv Solution. NeurIPS 2018.

[6] M. A. Islam, S. Jia, N. D. B. Bruce. How Much Position Information Do Convolutional Neural Networks Encode? ICLR 2020.

[7] O. S. Kayhan, J. C. van Gemert. On Translation Invariance in CNNs: Convolutional Layers Can Exploit Absolute Spatial Location. CVPR 2020.

[8] T. Cohen, M. Welling. Group Equivariant Convolutional Networks. ICML 2016.

[9] E. Wong, J. Z. Kolter. Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope. ICML 2018.

[10] J. Cohen, E. Rosenfeld, J. Z. Kolter. Certified Adversarial Robustness via Randomized Smoothing. ICML 2019.

[11] T. Chen, I. Goodfellow, J. Shlens. Net2Net: Accelerating Learning via Knowledge Transfer. ICLR 2016.

[12] K. Meng, D. Bau, A. Andonian, Y. Belinkov. Locating and Editing Factual Associations in GPT. NeurIPS 2022.

[13] K. Meng, A. Sen Sharma, A. Andonian, Y. Belinkov, D. Bau. Mass-Editing Memory in a Transformer. ICLR 2023.

[14] T. Hartvigsen, S. Sankaranarayanan, H. Palangi, Y. Kim, M. Ghassemi. Aging with GRACE: Lifelong Model Editing with Discrete Key-Value Adaptors. NeurIPS 2023.

[15] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, R. Hadsell. Progressive Neural Networks. arXiv:1606.04671, 2016.

[16] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, S. Gelly. Parameter-Efficient Transfer Learning for NLP. ICML 2019.

[17] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, R. Hadsell. Overcoming Catastrophic Forgetting in Neural Networks. PNAS 114(13), 2017.

[18] G. Hinton, O. Vinyals, J. Dean. Distilling the Knowledge in a Neural Network. arXiv:1503.02531, 2015.

[19] D. Lindner, J. Kramár, S. Farquhar, M. Rahtz, T. McGrath, V. Mikulik. Tracr: Compiled Transformers as a Laboratory for Interpretability. NeurIPS 2023.

[20] B. Barak, B. L. Edelman, S. Goel, S. Kakade, E. Malach, C. Zhang. Hidden Progress in Deep Learning: SGD Learns Parities Near the Computational Limit. NeurIPS 2022.

[21] D. P. Kingma, J. Ba. Adam: A Method for Stochastic Optimization. ICLR 2015.

[22] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. ICLR 2018.

Appendix A: Experiments, predictions and outcomes

Each experiment was registered in `PREREG.md` before it ran and is written up in `RESULTS.md`, with its script and output folder. "Held" counts predictions confirmed as stated.

id	question	main result	predictions
B2	Is an installed LeakyReLU circuit shielded from training?	No: its weights get 34–1,180× ordinary gradients	4 of 5 held
B2b	Do tied pairs restore shielding?	Yes: exactly 0 gradient where off, 50/50 batches	1 of 2
C1–C1d	Can training improve the detector inside frozen wiring?	Yes: 78% of left irises at 0.14 levels vs ~32% closed form	mixed; see RESULTS
D2	Exact vs crude depth growth	Same endpoint; exact starts damage-free	1 of 4, 2 in part
A3	Distilling the union	+0.108 vs +0.023 dB (4.7×)	all held
P1	A provable gap	Circular: chance; any position source: ≥ 96.9% from 64	5 of 7, 1 in part
P2	Off-axis robustness of contracts	None extra; position probe uninformative	1 of 3, 1 in part
P3	Contract on a pretrained network	88.0% zero-shot	2 of 3
P4	Robustness by construction	Exactly 0.0 change under attack	1 of 2, 1 in part
P5–P7	Form of the information; wrong and composed contracts	Binary input best; wrong contracts as good; composition 92–95%	1 of 7
P6b	How far can training move a frozen boundary?	Up to ~7 columns at reach 4	all held
P7b	MLP readout for composed contracts	Helps little	1 of 2
P9b	Small-N learning curves	binary > contract > ramp	0 of 3
S3	Cut, heal, re-attach a fixed module	Copy deleted within ~500 steps	2 of 5
S5	A frozen filter inside GAN training	Survives exactly; pulls the generator in	1 of 2 tested
E1, E1b	A grown fact in Pythia-160M	Survives; statistically, not exactly	mixed; see RESULTS
A2	Distilling a guarantee	Not transferred (100% of starts violated)	3 of 5
L1	The reach law, synthetic	Exact edge at the reach, 18/18; $k^* \approx r + 1.4\sigma$	3 of 5
L2	The reach law, natural images	Exact edge 3/3; renegotiation ~3 columns	2 of 3
L3	Does an unused own copy wither?	No (−4.5 points, then flat)	2 of 3
L4	Adding a skill without forgetting	2 of 3 seeds	1 of 2
L5	Exchange rate structure vs width	2.2× / 1.7× / ~1.4×; no width helps an impossible task	2 of 3
S1	Does a host adopt and depend on a grafted organ?	Routes exactly what the organ does better, then loses it; dependence grows	4 of 5
D5	Can a plain generator learn the odd eyes from the patched teacher?	Yes: 50% left, 0–6% right; right half not exact	2 of 4, 1 in part
L6	Does object-like clutter shrink renegotiation?	No (7.5 → ~7 columns); guarantee exact 6/6	2 of 4
L7	Reach, not depth?	Edge at exactly 4 for three reach-4 architectures (6/6); k^* 6.0–8.0	1 of 2, 1 in part
D4	An exact switch with a trained branch (cat → anime)	Off exact to 10^{-15}; on keeps the cat's layout	1 of 2, 1 in part
L8	Renegotiation and visibility	8 → 7.5 → 5.5 columns as noise rises; guarantee exact (after an attack fix)	2 of 4, 1 in part
S4	Transplant and rejection	Sibling block accepted in 200–300 steps; foreign block 0.1–0.3 dB short after 1,000	2 of 6, 1 in part
S2	Lesion atlas	Healing is local (6/6); block removals leave 0.1–0.65 dB	1 of 4, 1 in part
P2b	Contract vs ramp near the image edges	Contract loses 9–11 points at its untrained wrap-around boundary, ramp 4–8	0 of 3
B3	Give a generator position	Read from zero, but ablating it changes images by 0.65 levels	2 of 3
S5b	Why does the filter pull the generator in?	Registered mechanism refuted; filtering reals too makes it worse	1 of 3, 1 in part

Reproducing. Code, pre-registrations and results are released at https://github.com/v-code-895/exact-beyond-reach and archived at https://doi.org/10.5281/zenodo.24032112. Every script is under `src/`;

`scripts/queue/run_queue*.sh` record the order in which jobs ran on one 8 GB GPU; `results/queue.log` records start and end times, and `LOG.md` every change of plan.