Ex 2.2.10: three reads before the handover re-run

> Ex-2.2.9's three loose ends, read off its stored runs: the removal miss on the order-sensitive ops is the line-picking rule's, the retention drop happens before the anneal, and the op1 alignment rises under either half of the handover. We propose the re-run.

Ex-2.2.9 moved the anchor onto the larger grammar and left three loose ends. Removal was one-sided on the three order-sensitive ops, one seed in twenty fell under the retention gate, and the non-red alignment at op1 sat above the old reference. This notebook reads each off the stored runs, with one small scoring pass for the cube figures and no new training.

The removal miss comes from the rule we used to pick the lines. The projection acts like a change in the hue of red, and the lines that missed are the ones whose answer takes only the saturation or value of the red operand. Red is the one anchored concept, so these reads cannot say whether another concept would come apart the same way.

The retention drop happens before the anneal begins. The op1 alignment rises under either half of the handover, so it belongs to the new grammar rather than to the readout alone.

Observations

None of the lines below is a result; ex-2.2.11 will adopt what it needs from here and score it at fresh seeds.

- Removal: on the three order-sensitive ops (`hue-hsv`, `sat-hsv`, `value-hsv`; `hsvmix` is commutative and cleared the gate in ex-2.2.9) the kept share follows a change of hue on the red operand, and not the to-zero rule we used to pick the removal lines. Where the answer takes only the saturation or value of the red operand (`sat-hsv` and `value-hsv` with red at op2), two thirds of the clean exact match survives the projection. Where the answer takes the hue of red, or the whole color, it goes. One case no counterfactual predicts: `hue-hsv` with red at op1, whose answer needs only the saturation and value of red, and where a third of the clean exact match survives; its projected answers keep the hue of op2 and come out paler. A linear probe on the stream reads the projected red operand as a hue rotated away from red, at a lower value, and the rotation grows block by block.
- Retention: on `handover` the alignment reaches a noisy plateau by epoch 10, and its peak is just the high point of that noise. By the time the anneal begins at epoch 45 the seeds sit at 0.66, a few of them drifting downward. Through the anneal itself every condition is flat, so the gate is reading the noise in the plateau plus the drift. `handover-slot` and `handover-tied` hold their plateaus, so the drift needs the whole-line labeller and the untied readout together.
- Containment: $\bar{\alpha}$ at op1 on the non-red lines is 0.28 on `handover`, 0.18 on `handover-slot`, and 0.16 on `handover-tied`, against 0.02 on the control. Each half of the handover raises it, so the untied readout is not the whole story. The ⏎ embedding row picks up the axis only under the whole-line labeller: 0.17 on `handover` and zero on `handover-slot`.
- What we make of it: score removal on the lines whose answer takes the hue of the red operand, judge retention against the alignment at the start of the anneal (or drop the ratio and report a level), and carry the op1 alignment as a report line rather than a gate.

How to read this

This is a scouting notebook in the shape of ex-2.2.4: no hypotheses, no gates, no verdicts. Everything here comes from the checkpoints, probe set, metrics, and training trajectories that ex-2.2.9 stored. The cube figures also needed the answer distributions and residual states of the red lines, which ex-2.2.9 did not keep, so a scoring-only experiment (`experiment.py`) re-ran its projection read on the twenty `handover` checkpoints and stored those.

The reads use the vocabulary of ex-2.2.9, and the report for ex-2.2.9 has the full glossary. A *red line* has a red operand, meaning redness at or above 0.8; that operand sits in slot op1 or op2. The *projection* removes the anchored axis from the residual stream at every slice and position.[2] *Kept* is the share of the clean expected exact match that survives the projection, over a group of lines. A *removal line* is a red line where setting the R channel of the red operand to zero moves the true answer by at least 0.4 in the unit cube; the removal gate of ex-2.2.9 wanted kept to fall under 0.2 on those.

Neither set matches what the training labeller used.[1]

Removal on the order-sensitive ops

Ex-2.2.9 scored removal on eleven ops, and the projection cleared the gate on eight. The three that missed are the ones that read operand order: `hue-hsv`, `sat-hsv`, and `value-hsv` each take one HSV attribute from op2 and the other two from op1. The fourth op that works in HSV, `hsvmix`, averages the two operands and is commutative; it cleared the gate with the channel-wise ops, so the miss goes with operand order rather than with HSV. On each the miss was one-sided by slot: with red at op1 the projection removed most of the answer on `sat-hsv` and `value-hsv`, and with red at op2 the answer largely survived. `hue-hsv` ran the other way.

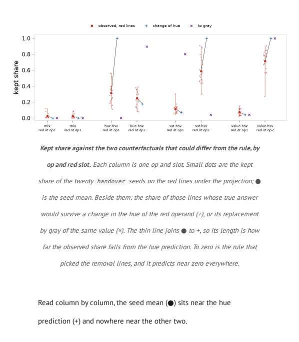
Those removal lines were picked by the *to-zero* rule of ex-2.2.4: a red line counts if setting the R channel of the red operand to zero moves the true answer far. That rule says nothing about which attribute of red the answer actually needs, and it has a blind spot on pure red. Zeroing R on (5, 0, 0) gives black, which has no saturation and no value, so every `sat-hsv` and `value-hsv` line that takes only the saturation or value of red counts as a removal line.

Three counterfactuals

Each is a different reading of what "losing red" should do to the red operand. *To zero* sets its R channel to zero; this is the rule that picked the lines. A *change of hue* replaces the operand by a permutation of its channels, keeping its saturation and value while moving its hue; a line survives when every permutation leaves the true answer within 0.4. A permutation reaches only six hues, a sixth of a turn apart, so this is a coarse hue rotation, and a finer one could behave differently between the hues it samples. *To gray* replaces it by the gray of the same value, which keeps value and removes hue and saturation together.

Op and slot	Red lines	To zero	Change of hue	To gray	Observed, red lines	Observed, removal lines
`mix`, red at op1	187	0.00	0.00	0.00	0.02 (0.00–0.12)	0.02 (0.00–0.12)
`mix`, red at op2	178	0.00	0.00	0.00	0.02 (0.00–0.09)	0.02 (0.00–0.09)
`hue-hsv`, red at op1	406	0.00	1.00	0.00	0.31 (0.11–0.56)	0.31 (0.11–0.56)
`hue-hsv`, red at op2	404	0.43	0.18	0.90	0.25 (0.08–0.39)	0.17 (0.01–0.32)
`sat-hsv`, red at op1	406	0.00	0.07	0.80	0.11 (0.05–0.30)	0.11 (0.05–0.30)
`sat-hsv`, red at op2	404	0.71	1.00	0.04	0.59 (0.30–0.91)	0.65 (0.28–0.97)
`value-hsv`, red at op1	406	0.04	0.04	0.04	0.07 (0.02–0.15)	0.06 (0.01–0.15)
`value-hsv`, red at op2	404	0.00	1.00	1.00	0.71 (0.27–0.91)	0.71 (0.27–0.91)

Predicted and observed kept share, by op and red slot. *The three middle columns are the share of red lines whose true answer survives the counterfactual on the red operand (moves by less than 0.4). The last two are what the `handover` seeds kept under the projection, mean with the seed range, on the red lines and on the removal subset. The removal subset is the lines that fail the to-zero counterfactual, so the to-zero share there is zero by construction.*



Kept share against the two counterfactuals that could differ from the rule, by op and red slot. *Each column is one op and slot. Small dots are the kept share of the twenty `handover` seeds on the red lines under the projection; ● is the seed mean. Beside them: the share of those lines whose true answer would survive a change in the hue of the red operand (+), or its replacement by gray of the same value (×). The thin line joins ● to +, so its length is how far the observed share falls from the hue prediction. To zero is the rule that picked the removal lines, and it predicts near zero everywhere.*

Read column by column, the seed mean (●) sits near the hue prediction (+) and nowhere near the other two.

Hue. Where a change of hue leaves the answer alone, the + is at the top of its column: 100% of the `sat-hsv` red-at-op2 lines and 100% of the `value-hsv` red-at-op2 lines. Those are the two columns where the seeds keep most of the answer, 59% and 71%. Where a change of hue moves the answer, the + is near the floor (`mix` in either slot, `hue-hsv` with red at op2, `sat-hsv` and `value-hsv` with red at op1), and the seeds keep 25% or less. Six columns of eight follow the hue prediction.

Gray. The × disagrees with the + in two columns, `sat-hsv` red-at-op1 (80%) and `hue-hsv` red-at-op2 (90%), where turning the red operand gray would leave the answer alone. The seeds keep neither, so the projection does not turn red into gray.

To zero. The rule that picked the removal lines predicts near zero in every column, and it is wrong wherever the + is high.

The column the hue prediction misses. On `hue-hsv` with red at op1 the answer is the hue of op2 at the saturation and value of red. A change of hue leaves every one of those answers alone (+ at the top), yet the seeds keep only 31%, and the line from ● to + is the longest in the figure. So when the red operand supplies saturation and value and something else supplies the hue, the projection also costs the model most of its read of that saturation and value. The two surviving columns show the same shortfall more mildly: `sat-hsv` and `value-hsv` red-at-op2 keep two thirds where the + says all of it.

What we propose. The removal gate should count the lines whose answer needs the *hue* of the red operand, which is what the to-zero rule was meant to proxy: a red line is a removal line when some channel permutation of its red operand moves the true answer by at least 0.4. Under that rule `sat-hsv` and `value-hsv` red-at-op2 drop out of the gate, and `hue-hsv` red-at-op1 drops out with them; what remains are the columns the projection already clears. The rule rests on the six hues a permutation can reach, and the design section says what to do if a finer rotation ever disagrees with it. The saturation and value shortfall becomes a read of its own, with no gate.

What stays open is whether that shortfall comes from the projection itself or from this checkpoint.

Where the answers go

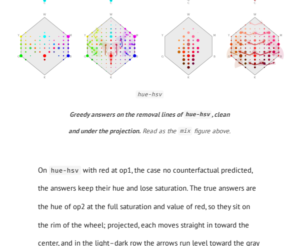
The counterfactual table says which answers are lost. The figures in this section show what the model answers instead, on the removal lines of each op and slot, from the twenty `handover` checkpoints. Each draws the RGB cube twice. The top row is the wheel view the probe-cube figures of ex-2.1.1 use: looking down the gray diagonal, so hue runs around the hexagon with red at the top, and lightness collapses onto the center. The row under it is the light–dark view: the cube turned so red points at the reader, which puts white at the top, black at the bottom, and lightness up the page. Where a figure shows the moves from true answer to given answer, it draws them as a smoothed flow rather than one stub per answer: an arrow for the mean move of the answers near it, with a faint wedge for their spread, so a wide wedge is a group of answers that went off in several directions.

`mix`

Greedy answers on the removal lines of `mix`, clean and under the projection. *Each pair of panels is one red slot, clean on the left and projected on the right, with the number of (line, seed) answers in the title.* ***Top row:*** *the wheel view of the RGB cube, down the gray diagonal, so hue runs around the hexagon and lightness collapses onto the center.* ***Bottom row:*** *the light–dark view, red toward the reader, so lightness runs up the panel from black (K) to white (W) and red and cyan fall inside. Corner letters name the cube's corners. Each mark is a greedy answer, placed at its own color and colored by the true answer, and sized by how many answers made that move; a projected mark whose color matches its place is an answer that still matches the truth. The projected panels add the moves as a smoothed flow: each arrow is the mean move of the answers whose truth lies near its tail, sized by their count and colored by their mean truth; a cell whose answers go two ways gets an arrow each way, and the faint wedge behind an arrow spans one standard deviation of the directions it averages.*

Clean, the greedy answer is the true one on almost every removal line. Projected, the ops sort, roughly, into three kinds of move.

On `mix` the projected answers move toward the center of the cube, and their mean is a gray with a little red left in it. Their hues stay near red: nine in ten sit within a third of a turn of it, which is why the green and blue half of the wheel stays empty. A `mix` answer is the midpoint of its two operands, and a red operand that reads as orange or pink after the projection cannot pull a midpoint to the far side of the wheel.

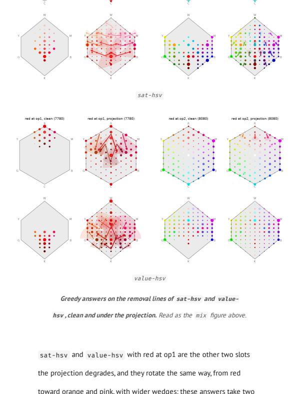


`hue-hsv`

Greedy answers on the removal lines of `hue-hsv`, clean and under the projection. *Read as the `mix` figure above.*

On `hue-hsv` with red at op1, the case no counterfactual predicted, the answers keep their hue and lose saturation. The true answers are the hue of op2 at the full saturation and value of red, so they sit on the rim of the wheel; projected, each moves straight in toward the center, and in the light–dark row the arrows run level toward the gray axis, so the answers get paler without getting darker. The model still reads the hue of op2 and returns it as a washed-out color; what the projection has cost it is the saturation read of the red operand.

With red at op2 the true answers are reds at the saturation and value of op1, and this is the hue rotation: projected, they split into an orange lobe and a pink lobe, at the same lightness as before. The flow draws two arrows from each cell for this reason, one to each side.

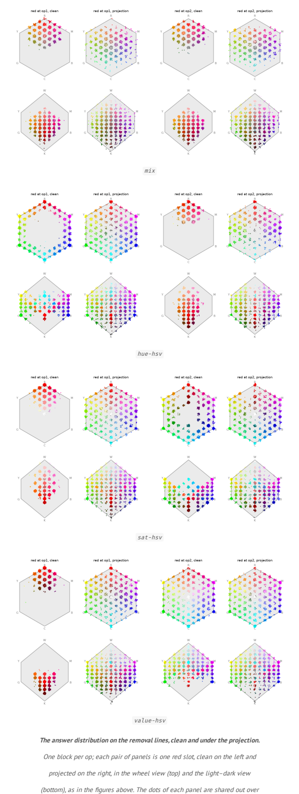


Greedy answers on the removal lines of `sat-hsv` and `value-hsv`, clean and under the projection. *Read as the `mix` figure above.*

`sat-hsv` and `value-hsv` with red at op1 are the other two slots the projection degrades, and they rotate the same way, from red toward orange and pink, with wider wedges: these answers take two attributes of red, and the moves scatter as well as rotate. In the light–dark row of `value-hsv` the arrows point every way, so those answers seem to change in lightness as well as hue, with no one direction to it; that panel is the least tidy of the set, and the rotation is only the largest part of what it shows. On the two slots the projection leaves alone, `sat-hsv` and `value-hsv` with red at op2, the arrows are short and the marks stay on their clean positions.

The whole answer distribution

The greedy answer is only one token. The whole answer distribution says how confidently the model moved, and whether the mass that left the true answer went to one color or spread out. The next figure draws that distribution as a dithered cloud in the cube.



The answer distribution on the removal lines, clean and under the projection. *One block per op; each pair of panels is one red slot, clean on the left and projected on the right, in the wheel view (top) and the light–dark view (bottom), as in the figures above. The dots of each panel are shared out over the 216 grid colors in proportion to the mean answer mass those lines put on each color, so a dense patch is where the model expects the answer to be. The clean panels show where the true answers of those lines lie.*

Op and slot	Top mass, clean	Top mass, proj.	Colors, clean	Colors, proj.
`mix`, red at op1	0.95	0.44	1.3	7.4
`mix`, red at op2	0.95	0.65	1.3	2.8
`hue-hsv`, red at op1	0.88	0.53	1.3	5.3
`hue-hsv`, red at op2	0.83	0.71	1.5	2.4
`sat-hsv`, red at op1	0.81	0.38	1.7	10.9
`sat-hsv`, red at op2	0.92	0.83	1.2	1.7
`value-hsv`, red at op1	0.86	0.40	1.4	9.1
`value-hsv`, red at op2	0.81	0.76	1.6	2.0

How peaked the answer distribution of each line is, clean and under the projection. *Mean over the removal lines and the twenty seeds. Top mass is the probability on the most likely grid color. Colors is the effective number of colors, the exponential of the mean entropy: 1 for a certain answer, 216 for a uniform one.*

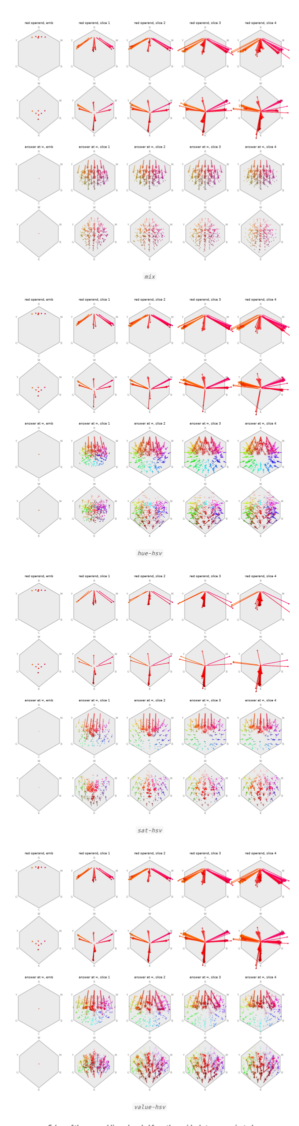
Per line, the projected answer is unsure among a handful of colors rather than spread over the wheel. On the cases the projection degrades, the mass on the top color falls from about 0.9 clean to about 0.5, and the effective number of colors rises from one or two to about 7. On the two cases it leaves alone, the top mass stays at 0.8 or above.

So the wheel-wide spread of the projected clouds is a spread across lines, each staying within its own neighbourhood. That is what we would expect if the operand reads as a hue rotated one way or the other, which is also what the probe finds.[3]

The light–dark row adds one thing the greedy answers did not show. On `hue-hsv` the projected mass reaches every lightness, from near black to near white, where the greedy answers of the previous figure kept the lightness of the truth. Read with the table above, one reading is that the runners-up a line hesitates among differ from its top answer in lightness as well as hue; the clouds pool the lines, so this is a guess about what is inside each one rather than a measurement.

What the stream says

The answers say what the model concluded; the residual stream can say what it concluded it from. For each op we fit a linear probe from the stream at each slice and position to the three colors a line carries: op1, op2, and the answer the rule gives. The probes were fit on the clean stream of the non-red lines, so that the axis stays out of them, and we then decoded the red lines through them, clean and under the projection.[4]



Colors of the removal lines decoded from the residual stream, projected against clean. *One block per op, mean over the twenty seeds.* ***Top row:*** *the red operand, read at its own position by the probe fit at that slice. One mark per line, at the RGB decoded under the projection and colored by the true color of the operand, with a stub from the clean decode of the same line, so the stub is what the projection changed.* ***Bottom row:*** *the answer from the rule, read at `=` and colored by the true answer; there are too many lines for rings and stubs, so the moves from the clean decodes to the projected ones are drawn as a smoothed flow, on the same terms as the greedy-answer figure. Each read is shown in the wheel view and, under it, the light–dark view, as in the answer figures. emb is the embedding, and slice n is the stream after n blocks.*

Op and site	emb	slice 1	slice 2	slice 3	slice 4
`mix`, op1@op1	0.95	0.93	0.90	0.88	0.87
`mix`, op2@op2	0.95	0.92	0.87	0.81	0.78
`mix`, ans@=	-1.76	0.76	0.92	0.90	0.89
`hue-hsv`, op1@op1	0.95	0.94	0.91	0.88	0.88
`hue-hsv`, op2@op2	0.95	0.93	0.88	0.82	0.79
`hue-hsv`, ans@=	-0.13	0.54	0.85	0.92	0.94
`sat-hsv`, op1@op1	0.95	0.94	0.91	0.88	0.88
`sat-hsv`, op2@op2	0.95	0.93	0.88	0.83	0.80
`sat-hsv`, ans@=	-0.22	0.65	0.88	0.93	0.94
`value-hsv`, op1@op1	0.95	0.94	0.91	0.88	0.88
`value-hsv`, op2@op2	0.95	0.93	0.88	0.83	0.79
`value-hsv`, ans@=	-0.25	0.67	0.91	0.94	0.94

Probe fit on the red lines, R^2 per slice, mean over seeds. *Each probe is a ridge fit on the clean stream of the non-red lines, and is read here on the clean stream of the red lines. `op1@op1` reads op1 at position 0, `op2@op2` reads op2 at position 2, and `ans@=` reads the raw answer from the rule at `=`. The negative fit of the answer probe at emb is expected: at the embedding, `=` has nothing of the line in it.*

Op, site, slice	clean move	clean |ΔH|	clean S	clean V	proj. move	proj. |ΔH|	proj. S	proj. V
`mix`, red operand, slice 1	0.19	0.01	0.82	0.88	0.64	0.10	0.79	0.69
`mix`, red operand, slice 4	0.27	0.02	0.73	0.88	0.94	0.18	0.82	0.72
`mix`, answer at `=`, slice 1	0.08	0.01	0.73	0.73	0.37	0.15	0.49	0.67
`mix`, answer at `=`, slice 4	0.07	0.01	0.71	0.73	0.35	0.15	0.51	0.65
`hue-hsv`, red operand, slice 1	0.18	0.01	0.83	0.89	0.65	0.11	0.79	0.75
`hue-hsv`, red operand, slice 4	0.26	0.02	0.74	0.88	1.03	0.21	0.87	0.78
`hue-hsv`, answer at `=`, slice 1	0.34	0.03	0.70	0.75	0.47	0.07	0.50	0.68
`hue-hsv`, answer at `=`, slice 4	0.10	0.01	0.89	0.89	0.36	0.06	0.69	0.77
`sat-hsv`, red operand, slice 1	0.19	0.01	0.83	0.87	0.66	0.09	0.78	0.61
`sat-hsv`, red operand, slice 4	0.27	0.01	0.75	0.88	1.05	0.19	0.87	0.61
`sat-hsv`, answer at `=`, slice 1	0.26	0.02	0.75	0.86	0.56	0.18	0.39	0.62
`sat-hsv`, answer at `=`, slice 4	0.09	0.01	0.79	0.89	0.45	0.14	0.45	0.68
`value-hsv`, red operand, slice 1	0.19	0.01	0.83	0.88	0.64	0.10	0.80	0.69
`value-hsv`, red operand, slice 4	0.28	0.02	0.73	0.87	0.94	0.18	0.85	0.72
`value-hsv`, answer at `=`, slice 1	0.23	0.02	0.77	0.80	0.51	0.13	0.53	0.72
`value-hsv`, answer at `=`, slice 4	0.10	0.01	0.83	0.84	0.38	0.08	0.62	0.76

Where the decoded colors land, clean and under the projection. *Mean over the removal lines and the twenty seeds, at the first block and the last. Move is the distance from the decoded color to the true one in the unit RGB cube. $|\Delta H|$ is the hue offset from the true color, in turns, so 0.5 is the opposite hue. S and V are the decoded saturation and value; the true operands have both near 1.*

At the embedding, the projection changes nothing the probe can see. Clean and projected reads coincide there, on operand and answer alike, and the small offset both show is the shrinkage of the ridge fit. The probes are blind to the axis, so whatever the projection takes at the embedding becomes visible only once the blocks have acted on it.[5]

From the first block on, the projected red operand reads as a different color, and the difference grows with depth. The move roughly doubles from slice 1 to slice 4 on every op. In HSV terms it is a rotation of hue away from red, a fifth of a turn by the last slice, at nearly full saturation and with a lower value. In the wheel view that is the stubs sliding along the upper edges of the hexagon, toward orange on one side and pink on the other, rather than toward the center. The light–dark view says how the value is lost. Red sits at the center of that view, and most stubs run level from it toward yellow or magenta, so those operands trade some red for green or blue and keep their lightness. The stubs colored pure red are the exception: the operands with no green or blue in them to start with move a little up, or on a couple of lines a long way down toward black, keeping the hue of red and losing value the plain way. The pattern is the same on every op.

So this is the change-of-hue counterfactual that the kept shares followed, with a loss of value alongside it. M1 saw a related asymmetry when it deleted the hue subspace of an autoencoder (ex-2.7 in ex-preppy, under its ablation figures): red, green, and blue darkened, and yellow, cyan, and magenta lightened. One reading of both is that the primaries sit nearer black on the gray diagonal than the secondaries do, so what remains of red once a hue direction is gone leans toward black. The few lines that drop toward black here fit that reading; the larger group that trades red for a neighbouring channel is a hue rotation with the value loss as a side effect.

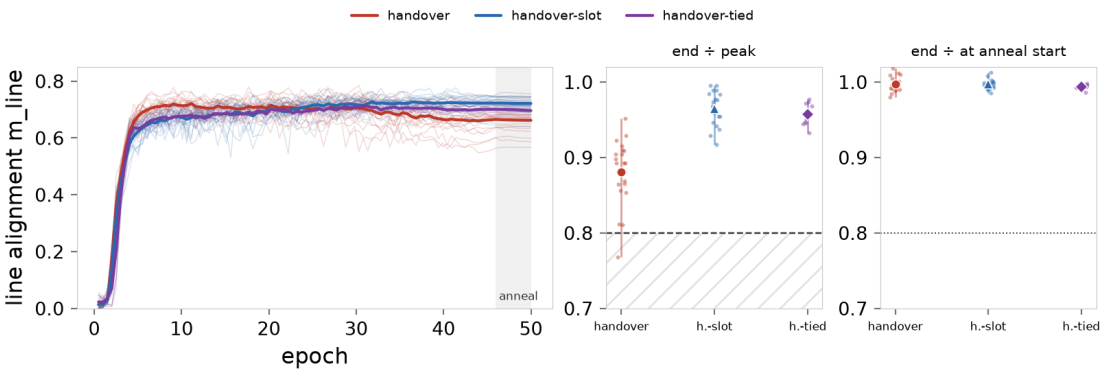
The slots that take the saturation and value of the red operand have only what the stream keeps of it, so that lost value is the partial loss the counterfactual table could not explain.

The answer at `=` follows the operand. Clean, it reaches its ring by the last slice. Projected, it stops short, with a smaller hue offset than the operand and a saturation about two tenths under the clean read. In the answer rows of the figure the flow points away from red on the reddish answers and gently inward on the rest, and on `mix` it is the same straight move down the wheel, invisible in the light–dark view, that the greedy answers made. The answers that survive on `sat-hsv` and `value-hsv` with red at op2 are the ones whose true color the rotated, dimmer operand still snaps to.

Retention and the anneal

The retention read of ex-2.2.9 asks whether the alignment survives the anneal of the anchor weight: every run whose line alignment peaks above 0.2 should end at 0.8 of that peak. One `handover` seed in twenty ended at 0.77, and the seed mean was 0.88 against 0.96 on the references.

The natural reading was that the anneal, over the last tenth of training, lets the alignment slip. The trajectories say otherwise.



Line alignment over training, and two retention ratios. *Left: line alignment against epoch for the three handover conditions, one faint line per seed with the seed mean drawn over them; the shaded band is the anneal window, where the anchor weight falls from 0.1 to its floor. Middle: the retention ratio ex-2.2.9 gated, the final alignment over its peak, per seed with the seed mean in the larger marker; the dashed rule is the gate and the hatched side misses it. Right: the final alignment over its value at the start of the anneal, the ratio that would measure the cost of the anneal alone, with the same gate level dotted for reference.*

The anneal costs nothing. Over the anneal window the `handover` seeds end at 0.998 of where they started it, and `handover-slot` and `handover-tied` at 0.997 and 0.994. The whole of the drop the gate measured happens earlier.

`handover` climbs to a plateau by epoch 10. Its peak, at 0.75 and epoch 19 on average, is just the high point of a noisy series. We cannot tell whether the high epochs are the ones with more red lines in their batches, since the stored trajectories do not record what each epoch drew; the corpus sampler is seeded, so a replay could tell.

By the start of the anneal the seeds sit at 0.66, a few of them drifting down to the bottom of the band. The two references peak later (epoch 33 on `handover-slot`, 30 on `handover-tied`) because their plateaus are still rising, and they drift less, which is why their ratio comes out higher.

So the ratio of the end to the peak is comparing the last sample of a noisy plateau against the height of that plateau, with a slow drift under a constant anchor weight added on some seeds. That drift needs both the whole-line labeller and the untied readout: either one alone holds its plateau.

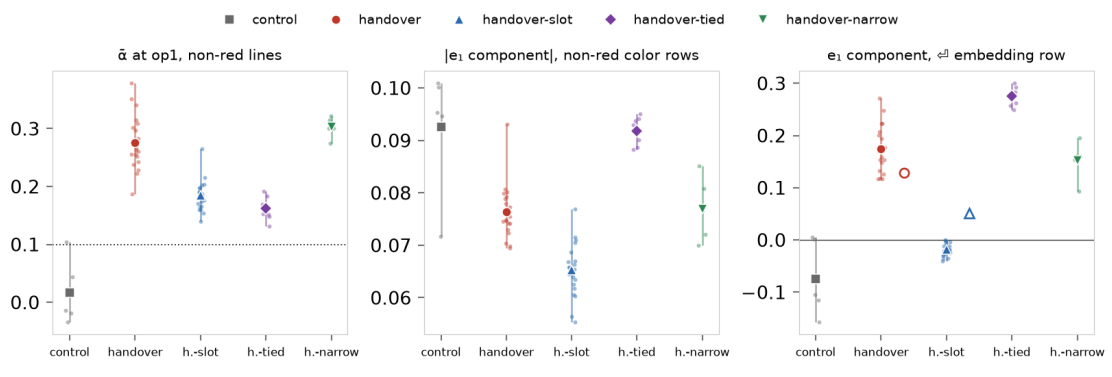
What the drift is, the trajectories cannot say. One reading is that the model is still reorganizing its representations under the larger grammar, and the axis is carried a little way with them. The whole-geometry read of anchored against control runs, now in review, sees a different latent geometry only with more ops and the anchor together, and a training-dynamics measure such as the local learning coefficient would say whether the plateau is a settled solution or a slow move between them.[6] The backlog item carries it.

Whether the drift matters is a different question from the one the gate asked. The end-of-training alignment is 0.66 on `handover`, against 0.72 and 0.70 on the references. Every read downstream of it (grading, removal, containment) is taken at the end, so that level is what the later experiments inherit. A stepped anneal, which the backlog item floated, would act on the window where nothing is lost.

Containment under the untied readout

Ex-2.2.9 reported ᾱ, the mean alignment of the non-red lines at op1, at 0.28 on `handover`. The reference from ex-2.2.3 read 0.1, and the old gate was set there. The report put the rise down to the untied readout, which is what the pilot in ex-2.2.7 had suggested.

But `handover` changed two things at once relative to the recipe of ex-2.2.3: the whole-line labeller as well as the readout. Ex-2.2.9 ran one arm with each change undone.



Containment by condition. *Per seed, with the seed mean in the larger marker. Left: the mean alignment of the stream on the non-red lines with the anchor axis at op1, with the reference level from ex-2.2.3 dotted. Middle: the mean absolute axis component of the embedding rows for the 209 non-red color words. Right: the axis component of the ⏎ embedding row, with the same row of the untied readout as an open marker beside it where the condition has one.*

ᾱ at op1 is 0.28 on `handover`, 0.18 with the slot labeller back, and 0.16 with the readout tied again. Undoing either change takes back about half the rise. So the rise belongs to the handover as a whole, and the untied readout is one of two contributors. `handover-narrow`, which has fewer lines per op, sits at 0.30.

Why either half raises it is open. For the labeller half there is a candidate: under the whole-line labeller a line can earn its label from its answer, and the pull then lands on every position of that line, op1 included. So a non-red operand is pulled toward the axis whenever its answer draws. For the readout half we have no candidate yet. The backlog item carries both.

Whatever the mechanism, none of it reaches the embedding table: the non-red color rows hold 0.08 of the axis on `handover` against 0.09 on the control.

The ⏎ row is a separate, cleaner matter. Its embedding component is 0.17 on `handover` and -0.02 on `handover-slot`, so the whole-line labeller is what puts it there. The pull lands on every position of a red line, ⏎ included, and the slot labeller never touches that position.

The readout row for ⏎ is at 0.13 on `handover` too. On `handover-tied` the row reads 0.28, and so does `=`, which is the shared table doing double duty.

The backlog item on the ⏎ row asks whether the residual redness of the answer position is part of this. Answering that needs a per-position alignment on the red lines, which ex-2.2.9 did not store, so it stays open here.

It may also not survive scale. The row picks up the axis because ⏎ closes every labeled line, and nothing competes with the pull at that position; in a larger model on more diverse data the same token closes lines about everything, so the pull would be a small part of what shapes its row. That is a guess, and the row is cheap to read wherever the anchor is next applied.

What we make of it

Three changes to the design of the re-run, and one read to carry.

Removal lines by hue. Define a removal line as a red line whose true answer moves by at least 0.4 under some channel permutation of the red operand. Those are the lines whose answer needs the *hue* of red. On `mix` this is the to-zero set and on `hsvmix` nearly so; on the six other channel-wise ops it is a wider one, since a permutation moves two channels of the operand where zeroing R moved one, and those ops cleared the old gate with room. On the three HSV ops it drops the slots that take only saturation or value (`sat-hsv` and `value-hsv` with red at op2), and `hue-hsv` with red at op1 goes with them.

The gate stays at *kept* under 0.2, and on the lines that remain every `handover` seed already clears it. The permutation rule reaches six hues; if a finer rotation of the operand in HSV, snapped to the grid, ever disagrees with it, the finer one would be the rule to keep.

The saturation- and value-taking slots become a second read, with no gate. The kept share there says how much of the saturation and value of the red operand the projection takes with it, which is a cost of the operator worth reporting.

Retention against the start of the anneal. The retention read keeps its name and changes its denominator, to the final alignment over the alignment at the start of the anneal, gated at 0.8 as before. That is the question the read was written to ask.

A high percentile of the plateau, in place of the peak, would soften the ratio and keep the drift inside it; the anneal-start denominator takes the drift out and leaves it to the level line. The early peak and the drift get a line of their own in the report: the level at the end of training, beside the references.

The op1 alignment as a line rather than a gate. ᾱ at op1 rises under either half of the handover, and the embedding rows do not move, so the old reference of 0.1 belonged to the old grammar. The re-run reports ᾱ at op1 with `handover-slot` and `handover-tied` as its references, and gates nothing on it. A gate can return once we can name a mechanism.

What the shape of the loss suggests for a larger model. On the lines that need the removed concept, the model is not confidently wrong. It hesitates among a handful of neighbours of the true answer, and it keeps what the concept did not supply: the hue of op2, the lightness. If the same held for an anchored concept in a language model, removal would look like a reduced capability confined to the prompts that need the concept, with the model wavering among near-misses rather than producing a confident substitute. That is the side-effect profile anchoring is meant to buy, and it is a guess from one concept in one small model. The near-misses here are neighbours because the answer space is a color cube; a language model's answer space has no such tidy neighbourhood.

The `⏎` row stays as it is, with its backlog item open. The whole-line labeller is what puts it there, and the slot labeller is not coming back for it. If the edits in M3 ever touch the answer position, this row is the first place to look.

One question this notebook could not settle: whether the partial loss of saturation and value under the projection comes from the operator or from this checkpoint. Answering it would take the same read on `handover-tied` and `handover-slot`, which the scoring pass can run at no design cost, so it is worth adding to the scoring for the re-run.

Method

What ran. A scoring-only experiment over the 20 `handover` checkpoints of ex-2.2.9 (20 scored so far), on the ops `mix`, `hue-hsv`, `sat-hsv`, `value-hsv`, using the probe set of ex-2.2.9. For each checkpoint and op, the task ran two forward passes over every probe line: one clean, and one with the `projection` operator applied at every slice and position. For the red lines it stored the answer distribution over the 216 grid colors, the greedy answer, and the expected exact match under both passes. Its per-group kept shares reproduce those of ex-2.2.9 to the third decimal, which is our check that it ran the same read.

Probes. One ridge regression ($\lambda = 0.01$) per slice, position, and target, from the 64-wide residual state to a color in the unit cube, fit on the clean stream of the non-red lines and read on the red lines. The three targets are op1, op2, and the raw (unrounded) answer the rule gives. The sites read in the figures are each operand at its own position, and the answer at `=` (position 3). We store the decoded coordinates of every red line under both passes for all seeds, and the raw states for the first 5 seeds.

Counterfactuals. For each red line, the red operand is the one with the higher redness. *To zero* sets its R channel to zero. *Change of hue* takes the five other permutations of its channels, and the line survives if all five move the true answer by less than 0.4. *To gray* replaces it by the gray of its value. Answers are the snapped answers of the op, and distances are in the unit cube, the same as `to_zero_move` in ex-2.2.9.

Retention. From the trajectories stored by ex-2.2.9 (100 points over training). The anneal starts at the first point where the anchor weight is under 0.99 of its plateau, and the alignment at that start is the last point before it.

Containment. From the run metrics stored by ex-2.2.9: $\bar{\alpha}$ at op1 as defined there, plus the axis component of each embedding row and, where the readout is untied, each readout row.

---

1. Under the whole-line labeller, each of the three colors in a line draws a label at a small rate that rises steeply with its redness (4% per visit at pure red), and the line is labeled if any of them draws. So a red line is labeled on some of its visits, and a line with a red answer can earn the label too. ↩

2. The *residual stream* is the running vector the transformer carries from block to block; each block reads it and adds to it. A *slice* is that vector at one depth, and a *position* is one token in the line. ↩

3. The structure inside each cloud is the set of answers the op can produce on the grid, carried around the wheel by the rotation. Take `sat-hsv` with red at op1: the clean answers are reds of every saturation, a ray from white at the center out to red at the top, and projected that ray appears at every hue. With red at op2 the answers are fully saturated colors at the hue and value of op1, which in the wheel view are the rim and the spokes running in toward the center, and the projected panel keeps that skeleton. ↩

4. The probe-cube figures of ex-2.1.3 are the precedent. The probes read RGB, so their output lands in the cube with no rotation needed to fit it. A hue probe would be ill-posed, since hue is circular, and saturation and value are piecewise-linear in RGB, so we convert the decoded RGB whenever an HSV number is wanted. Fitting on the non-red lines keeps the axis out of the probes: a probe fit on lines with red in them would learn the axis as the direction of red, and would then read its removal as a loss of red at the embedding as much as anywhere, where a probe blind to the axis reads what the blocks make of its absence. The red lines are outside the fit set, so they are held out without a leave-one-out scheme. ↩

5. At the embedding the stream at `=` is the same vector on every line, since nothing of the line has reached that position yet. A probe there can only return one color for every answer: the mean of its fit set, near the center of the wheel. ↩

6. The *local learning coefficient* (LLC) is a measure of how many effective degrees of freedom a trained model is using near its current weights, estimated by sampling the loss around them. It comes from singular learning theory, and the developmental-interpretability literature uses a change in it to mark a phase transition in training: a plateau in the loss that hides a reorganization inside the model. ↩