How often the anchor is lost

> We trained 32 new seeds each of `falloff` and `step, soft τ`, stopped each run early, and called whether it kept the anchor from how far its example answers had come onto e_1. `falloff` lost the anchor on one seed of 40, and `step, soft τ` on none. A few `falloff` runs came onto e_1 only late in training, which `step, soft τ` never did.

The recipe of record, `falloff` from the split-anti-seeds experiment, had kept the anchor on all eight of its seeds, which fits a loss rate as high as three in ten. `step, soft τ` from soft-pool, which starts with a soft pool, also kept it on all eight.

Several close variants lost the anchor on one to three seeds of eight, in split-anti-seeds, anchor-ramp, and soft-pool. With the recipe known this loosely, a condition that loses one seed in eight couldn't be told apart from it.

Full runs at enough seeds to narrow this are costly. But in those three experiments the lost runs had parted from the kept ones within the first sixty or so epochs. So we first check, on the 96 stored runs whose outcome we know, whether an early call made from the alignment° agrees with the full edit.

Alignment. How far a state points along e_1…

Then we train 32 new seeds each of `falloff` and `step, soft τ` on the full-length schedule, stop them at epoch 64, and call each one. The runs the call can't settle are trained to the end and scored, and so are the three `step, soft τ` runs it was least sure of among those it called kept.

Observations

- The early call on the stored runs (E1): how well the example answers had come onto `e1` by epochs 44 to 64 sorts every stored run the way the full edit does, with a clear gap between the lost runs and the kept ones. It still does when each experiment is held out of choosing the cut, though one slow run then clears the cut by only a few hundredths.
- How often each condition loses the anchor (E2): `falloff` lost the anchor on one seed of 40, and `step, soft τ` on none. That puts the loss rate of `falloff` under about one in nine and that of `step, soft τ` under about one in fourteen. The two can't be told apart on losses alone. The runs finished to check the call all kept the anchor, with removal and spill° like those of the stored runs.

  Spill. The side effect of an edit on the other ops…
- The slow runs (E3): under `step, soft τ` the example answers came onto `e1` within a few epochs of each other on every seed. Under `falloff` most runs were a little quicker, but three hung partway until after epoch 200. They came up once the anti weight had fallen to about half, and kept the anchor.

Method

This is an exploratory study, with no preregistration and no gate. It covers two conditions, both with the step pull on slices 2 to 4 and the `falloff` anti weight on every slice. `falloff` itself has a fixed pooling temperature° τ_p = 0.1, and `step, soft τ` sharpens τ_p from 1 to 0.1 over the 40-epoch warm-up of the anchor weight. Everything else is the training of split-anti-seeds.

Pooling temperature. How sharply the anchor pull picks its positions....

The new runs are at seeds 712 to 743 in both conditions, so the two are paired by initialization, batches and label draws. Each follows the full 400-epoch schedule (learning rate and both term weights as in a full run) but stops at epoch 64. With the stored runs at seeds 704 to 711, each condition has 40 seeds.

Stopping early leaves the run up to the stop as it was in the stored full run. Two seeds of each condition (704 and 705) were trained again and stopped at epoch 64, and α° at every recorded epoch up to the stop matches the stored run to the last digit in all four (largest difference 0).

Alignment. How far a state points along e_1...

A screen run that is not firmly kept (called lost, or between the stored lost and kept runs) was trained again to the end and scored with the full edit, so every loss counted is a measured one. A kept call is mostly not checked this way, and rests on the timing of the stored losses.

One stored loss shows how a kept call could be wrong. In soft-pool, seed 708 of `ramp 1–3` lost the anchor late: its example answers rose with the other runs, fell back a few epochs later while its query answer stayed on e_1, and only climbed back after about epoch 100, to about four fifths of the kept level. Yet the full edit at the end removed almost nothing, so its alignment came back without the op coming with it. That run was still low in the window, so the call caught it. A lost run that came back sooner, by epochs 44 to 64, would be called kept. No stored run did that, and to check the kept call where it is least sure we also finished the 3 `step, soft τ` runs with the lowest window means.

The finished runs have no control at their seed, so their removal is netted against the seed mean of the eight controls of ex-2.2.23. The controls differ by up to about an eighth in removal, which matters only for a run near the half-removal line, and no finished run came near it.

Spill. The side effect of an edit on the other ops...

We don't measure spill° or the task score on the screen runs, or removal on most of those called kept: the question is the loss rate, and those are known for both conditions from the earlier experiments.

The measurements

- Kept or lost (the outcome, on stored and finished runs): a run lost the anchor when the full edit, along e_1 at every position and slice, removes less than half of `difference`, net of the control, as in split-anti-seeds.
- The window mean (the call): α° (the cosine of the state with e_1) at the example answers of the `difference` probe contexts at slice 3, averaged over the six records (one every four epochs) at epochs 44 to 64. Slice 3 is the deepest slice every stored condition pulls, both screen conditions included. A run is called lost below 0.15, the highest window mean of any stored lost run (rounded up), and kept at 0.28 or above, the lowest of any stored kept run (rounded down). Every run in between was finished, so the full edit decides.

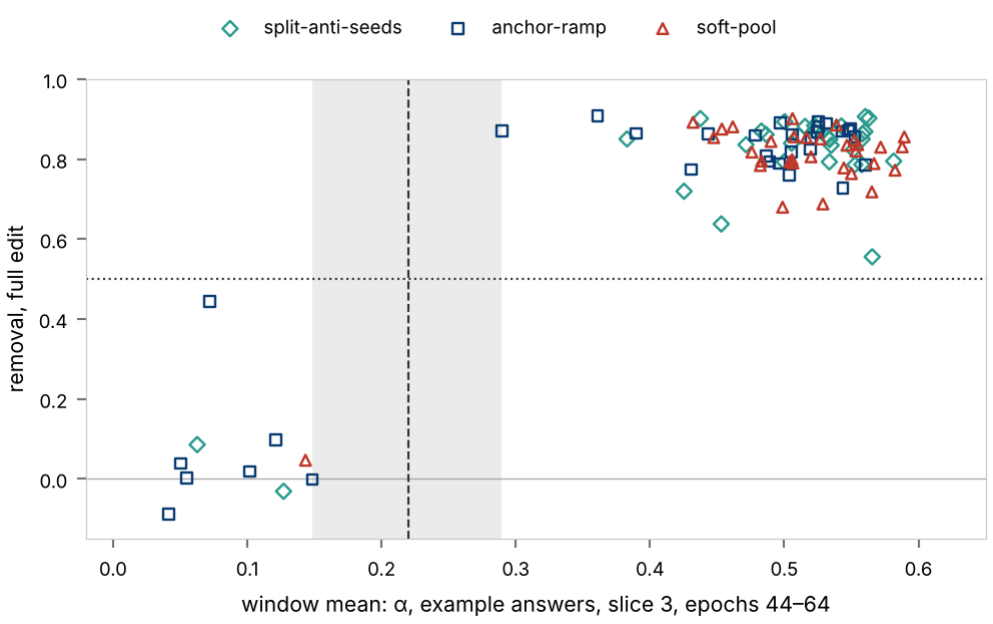
Alignment. How far a state points along e_1…

- The loss rate: losses over seeds, with the one-sided 95% upper bound from the binomial (Clopper–Pearson[1]). The conditions are compared on the seeds where one kept the anchor and the other lost it, with a one-sided exact (sign) test. At these counts it takes about five such seeds all one way to reach $p < 0.05$, so the screen can bound each rate well but can show a difference between the conditions only if it is large.

1 Clopper–Pearson…

The early call on the stored runs (E1)

Does the window mean tell the lost runs from the kept ones? We plot it against removal under the full edit for all 96 stored runs, 10 of them lost.



***The early call against the outcome**, for every stored run: the window mean (α at the example answers at slice 3, averaged over epochs 44 to 64) against removal under the full edit, net of the control. The shaded band is the gap between the highest lost run and the lowest kept one; the dashed line is the cut, at its middle, and the dotted line is the half-removal line that defines a loss.*

It does, with room to spare. Every lost run has a window mean near zero, every kept run one well above it, and no run falls in the gap between them. Most lost runs ended with almost no removal.

The lost run near the half-removal line is seed 710 of `ramp from 1`, which the anchor-ramp report counted as borderline; on the window mean it groups with the other lost runs. The kept run nearest the gap is seed 710 of `ramp from emb`, whose example answers came onto e_1 only after epoch 200.

A cut chosen on the same runs it sorts could fit them by chance, so we chose it on two experiments and sorted the third:

held out	runs (lost)	highest lost, other two	lowest kept, other two	cut	sorted wrongly ↓
split-anti-seeds	32 (2)	0.15	0.29	0.22	0
anchor-ramp	32 (7)	0.14	0.38	0.26	0
soft-pool	32 (1)	0.15	0.29	0.22	0

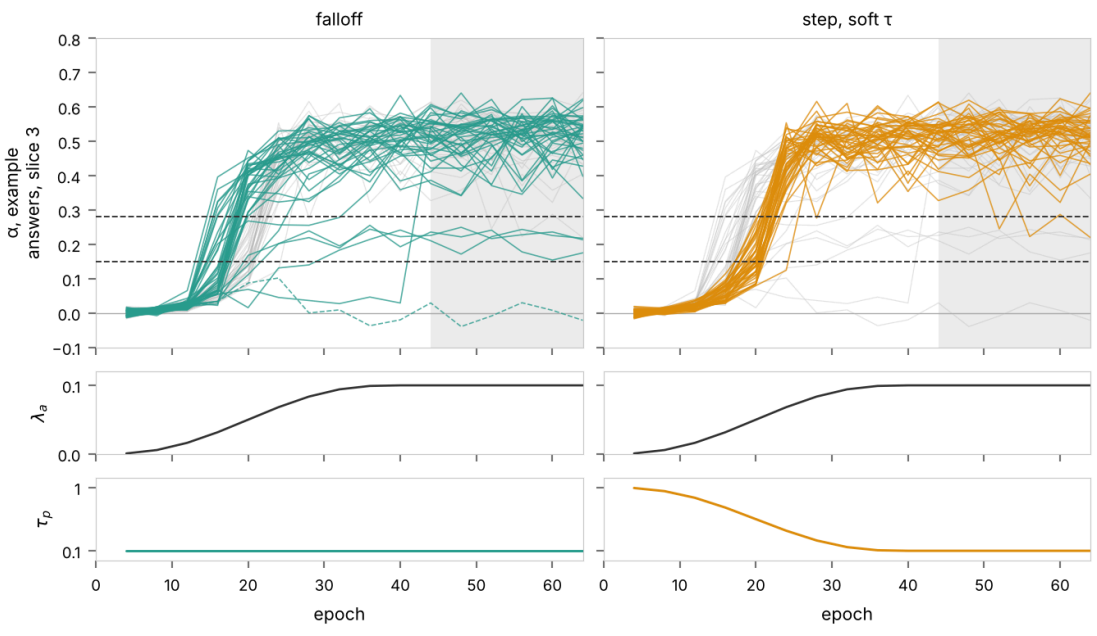
***The call on each experiment, with the cut chosen on the other two.** The cut is the middle of the gap between the highest window mean of a lost run and the lowest of a kept run in the other two experiments.*

Each held-out experiment is sorted with no error. The closest call is the slow kept run of anchor-ramp, which clears the cut chosen on the other two experiments by a few hundredths.

Windows that end later separate the runs less well, because the late loss of soft-pool brought its example answers partway back by epoch 100 without the removal coming back. Windows that end at epoch 52 or earlier leave little or no gap, since some kept runs were still rising. So the stored runs separate best with the window ending at epoch 64, and we took that stop as given for the screen.

How often each condition loses the anchor (E2)

How many of the 40 seeds of each condition lost the anchor, how high could the loss rate be, and do the two conditions differ? The figure shows the example answers through epoch 64 on every seed of each condition, and the table counts the calls and the losses.



***The example answers through epoch 64**: α at the example answers at slice 3 on the `difference` probe contexts, one column per condition and one line per seed (the stored seeds and the screen seeds alike), dashed where the run lost the anchor; the other condition is ghosted behind. The shaded window is where the call averages, and the two rules are the highest stored lost run and the lowest stored kept one. The strips underneath are the anchor weight λ_a and the position pool temperature τ_p (log scale).*

condition	stored seeds	screen seeds	called kept	finished	lost ↓	95% upper bound ↓
`falloff`	8	32	28	4	1 of 40	11%
`step, soft τ`	8	32	32	3	0 of 40	7%

***How often each condition lost the anchor.** The stored seeds (704 to 711) trained to 400 epochs in earlier experiments; the screen seeds (712 to 743) stopped at epoch 64. Those not called kept, and the 3 lowest kept calls of `step, soft τ`, were finished at full length and scored with the full edit. The bound is the one-sided Clopper–Pearson bound on the loss rate.*

`falloff` lost the anchor on one seed, 714, the way the earlier losses did, in anchor-ramp and in the runs of split-anti-seeds with a uniform anti weight on every slice. Its query answer came onto e₁ by epoch 20 and stayed there, its example answers never left zero, and the full edit removed nothing. `step, soft τ` kept the anchor on that seed and every other.

So with 40 seeds each, the loss rate of `falloff` is under about 11% and that of `step, soft τ` under about 7%. One seed where only one condition lost is far from enough to say which loses more ($p = 0.5$).

Three more `falloff` runs fell between the stored lost and kept runs. All three kept the anchor when finished, with removal 0.84 (0.81 to 0.86), close to that of the other kept runs. Two had window means below the middle of the gap, so a single cut there would have counted three losses in place of one.

condition	seed	window mean	call	removal ↑	spill, any dose ↓	task change ↑	worst op change ↑
`falloff`	714	−0.00	lost	−0.04	0.021	−0.022	−0.035
`falloff`	724	0.20	lost?	0.81	0.005	−0.009	−0.030
`step, soft τ`	724	0.33	kept	0.87	−0.006	−0.005	−0.020
`falloff`	725	0.23	kept?	0.86	−0.005	0.004	−0.000
`step, soft τ`	732	0.38	kept	0.80	−0.003	−0.052	−0.089
`step, soft τ`	734	0.41	kept	0.87	−0.005	0.000	−0.006
`falloff`	735	0.22	lost?	0.86	−0.007	−0.007	−0.027
`falloff`	stored, 8 kept			0.85 (0.79 to 0.90)	0.003 (−0.004 to 0.018)	−0.007 (−0.032 to 0.005)	−0.019 (−0.066 to −0.003)
`step, soft τ`	stored, 8 kept			0.83 (0.77 to 0.90)	0.009 (−0.007 to 0.029)	−0.005 (−0.014 to 0.007)	−0.019 (−0.036 to 0.000)

***The finished runs, scored as the earlier experiments scored theirs**, beside the stored runs of each condition that kept the anchor (seed mean and range). Removal is at the full dose; spill is the largest net drop on another op at any of the four doses (the criterion is 0.02); the task change is the mean score over ops less that of the control, and the worst op change the largest loss on any one op. The finished runs are netted against the seed mean of the controls, the stored runs against the control at their seed.*

The finished runs scored like the stored runs of their condition. The three slow `falloff` runs removed as much, stayed well within the spill° criterion, and lost no more task score than the stored runs. The three lowest kept calls of `step, soft τ` (0.38 (0.33 to 0.41) on the window mean) all kept the anchor, with removal 0.85 (0.80 to 0.87) and spill within the criterion, so the kept call held where it was least sure.

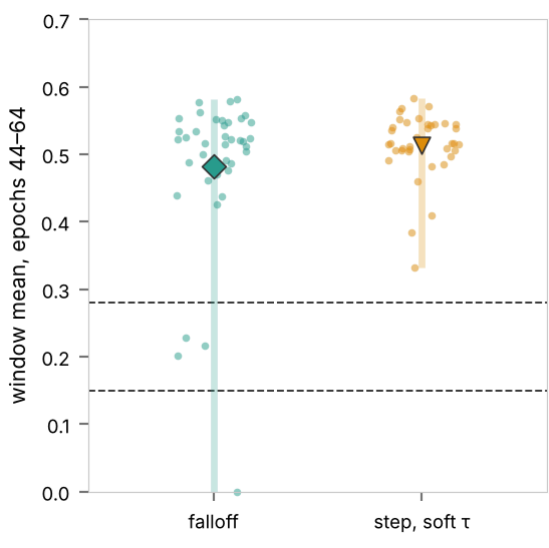
Spill. The side effect of an edit on the other ops…

One of them, seed 732, scored lower on the task than any stored run: a few hundredths lower on every op, and about a tenth lower on the HSV ops. The edit isn't involved, since this is the score before editing, and its removal and spill are ordinary. It has no control at its seed, so we can't tell whether the seed learns the task less well or the condition did; the screen stopped its `falloff` run at epoch 64, so there is nothing to compare with.

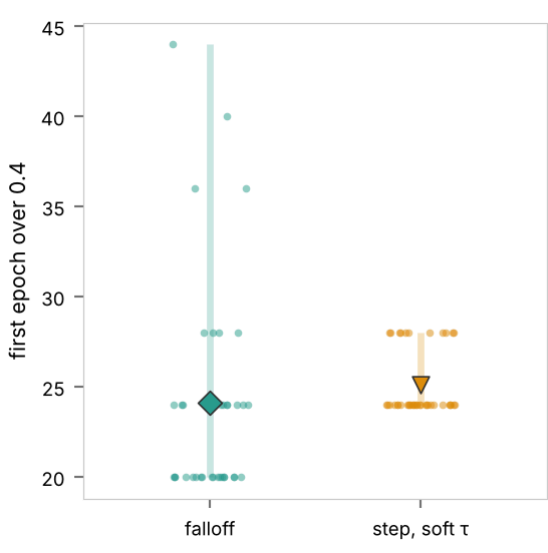
The slow runs (E3)

What set those three `falloff` runs apart, and does `step, soft τ` have runs like them? The figure and table show the window mean, and the first recorded epoch at which α° at the example answers goes over 0.4 (as in soft-pool).

Alignment. How far a state points along e_1...



***The window mean**, with the highest stored lost run and the lowest stored kept one as rules.*



***The first recorded epoch** at which α at the example answers at slice 3 goes over 0.4 (α is recorded every four epochs).*

***How far above the cut, and how soon**, for every seed of each condition. Each dot is a seed, jittered sideways; the thin bar spans the seeds and the large marker is the seed mean.*

condition	window mean ↑	first epoch over 0.4	seeds not over it by epoch 64
`falloff`	0.48 (0.00 to 0.58)	24 (20 to 44)	4
`step, soft τ`	0.51 (0.33 to 0.58)	25 (24 to 28)	0

***How the runs settle**, seed mean and range over all 40 seeds of each condition; the first epoch is over the seeds that got there by epoch 64. The warm-up of the anchor weight ends at epoch 40.*

Under `step, soft τ` the example answers came onto e_1 within a few epochs of each other on every seed, between epochs 24 and 28, as in soft-pool. Under `falloff` most runs got there a few epochs sooner, but the spread is wider: one run jumped up only at epoch 44, and 4 had not got there by the stop, the lost run and the three slow ones.

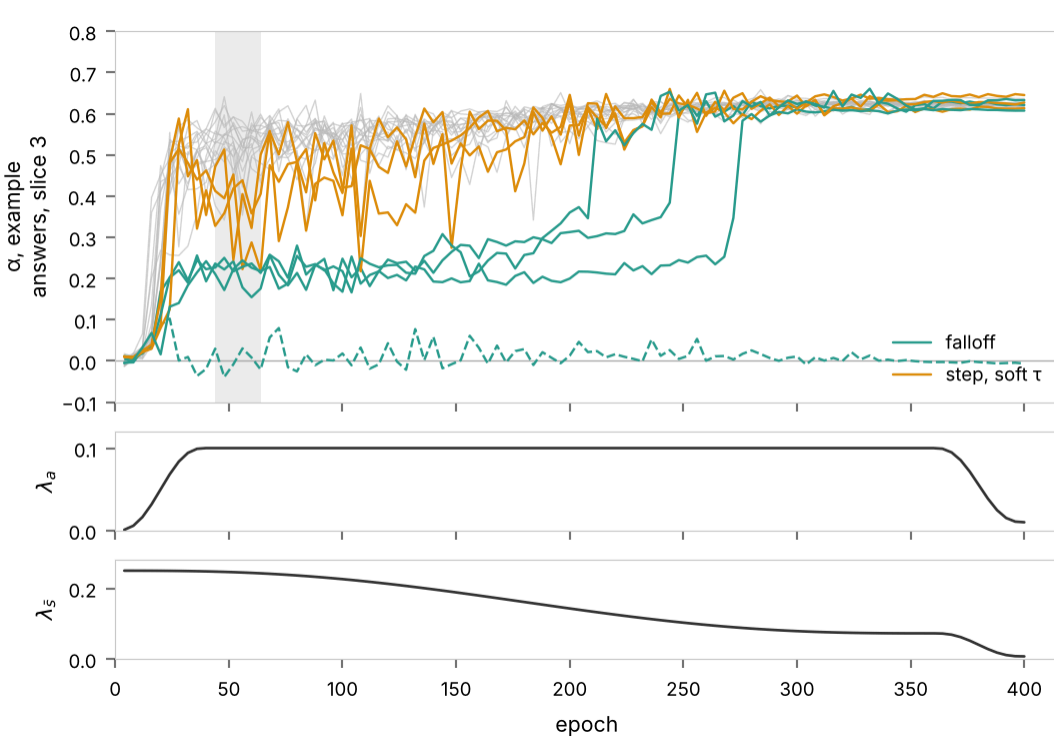
All four are seeds where `step, soft τ` got there on time. If the conditions didn't differ, a split of four seeds to none would come up about one time in sixteen ($p = 0.062$), and we chose this measure after seeing the runs. So this fits `step, soft τ` being more even, without being conclusive.

A run that crosses 0.4 late also has a low window mean, so these four late runs are the lost run and the three that E2 finished from the band. The timing picks out the same runs as the call, and the split adds little to E2 beyond showing that `step, soft τ` had none of them.

The evenness of `step, soft τ` is in when its example answers cross 0.4. How high they then sit through the window varies from seed to seed as much as on the `falloff` runs above the band, a little more if anything. Its lowest run, seed 724, dips into the band at two records late in the window, though its window mean is above it.

Seeds that were slow under one condition tended to be a little slow under the other (the window means of the two conditions are correlated over seeds, $r = 0.33$). The `step, soft τ` run with the lowest window mean is seed 724, one of the slow `falloff` runs.

The finished runs show what the slow ones did after the stop:



***The finished runs, to the end of training**: α at the example answers at slice 3 on the `difference` probe contexts, one line per finished screen run (the `falloff` runs not called kept, and the three lowest `step, soft τ` runs), dashed where the run lost the anchor; the stored runs of both conditions are ghosts behind. The shaded window is where the call averages. The strips underneath are the anchor weight λ_a and the anti weight at slice 3, $\lambda_{\bar{s}}$.*

The three slow runs held their example answers about a third of the way to the level of the other runs until after epoch 200. Each then jumped up within a few epochs, at different times between epochs 200 and 280, and kept the anchor. The lost run stayed off e_1 to the end. The three `step, soft τ` runs swung widely, between about 0.2 and 0.6, until about epoch 200, then settled at the level of the stored runs. None of them hung low and then jumped as the slow `falloff` runs did.

The jumps come once the anti weight on slice 3 has fallen to about half its starting value, which fits the anti term holding them back. But the anti weight falls over most of training, so the timing alone doesn't single it out. The jumps are also abrupt where the fall is smooth, and nothing here tests the idea.

Discussion

The screen bounds the loss rates of both conditions well below the three in ten that eight seeds allowed. The one loss was of the kind seen before: the query answer takes the pull and the example answers never come onto e_1. The conditions can't be told apart on losses, and telling apart rates this low would take several times as many seeds.

In these runs `step, soft τ` was more even in when the example answers came onto e_1, and every `falloff` seed that was slow or lost was on time under it. That fits the explanation offered in soft-pool: a soft pool holds the pull back until the example answers can share it. The slow runs and the losses of `falloff` may also be two degrees of one thing. If so, the slow runs would be the cheaper measure of it, since there are more of them than losses.

The early call held up on the screen as far as we checked it: every run it was unsure of was finished and sorted cleanly by removal, and the three kept calls it was least sure of kept the anchor. But the slow `falloff` runs fell in the band, where the stored runs had only one run nearby, and a single cut would have miscounted two of them.

---

1. Clopper–Pearson: an exact interval for a proportion, built from the binomial itself, so it holds at small counts and at zero losses. ↩