Experiment 2.9.1 redux: deleting *red*, now in JAX

> We ported the main M1 result from PyTorch to JAX: Anchor *red* to one latent axis of a small autoencoder, then zero the axis and watch red disappear. The result reproduces, seed sensitivity and all, and our new infrastructure holds up.

This is a port of ex-preppy experiment 2.9.1 (M1, autoencoders), run as an end-to-end test of the infrastructure in this repo before the M2 transformer experiments. It re-answers the M1 question: if Sparse Concept Anchoring pins *red* to one latent axis during training, does zeroing that axis delete red, and only red?

The model is a small autoencoder: it compresses each RGB color down to a 5-dimensional vector (the bottleneck) and reconstructs the color from that vector alone. Training adds four regularizers[1] on the bottleneck, which is unit-normalized so every latent vector lands on the surface of a hypersphere: anchor pulls red-labeled samples toward one fixed axis (e_1), anti-anchor pushes everything away from the opposite point ($-e_1$), separate spreads samples within a batch apart from each other, and anti-subspace pushes everything away from that first axis in general. The red label itself is sparse and noisy: even a pure red sample only gets labeled "red" with probability 0.08 per draw.

After training, we ablate (zero out) the first axis and reconstruct every color, then score the run by how tightly the post-ablation reconstruction error of each color tracks its HSV similarity to red. The score is R^2, which runs from 0 (no relationship) to 1 (a perfect linear fit): a value near 1 means the deletion was clean, with error scaling with redness and colors unlike red left untouched.

This is a report: it reads results the experiment already produced. The experiment itself is `experiment.py`, a `main(ctx)` DAG driven from the CLI — 16 seeded runs fanned out on Modal CPU containers:

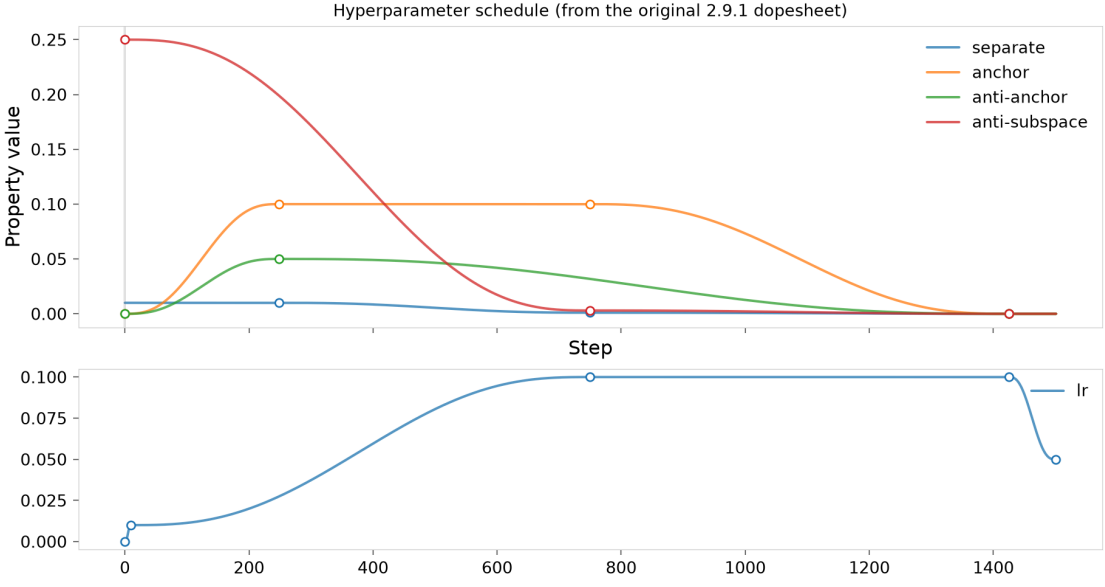
```
bin/mini run docs/m1/ex-2.9.1/experiment.py --app moda
```

The port is lighter than the original: the PyTorch/Lightning machinery becomes a pytree and one `lax.scan`, 16 seeds replace 60, validation happens only at the end, and we test ablation only (no pruning or suppression conditions). The dopesheet is identical to the original, so the same `mini.temporal` machinery interprets it unchanged.

16 runs completed. Scores range 0.01–0.97 (median 0.87): anchoring quality depends on the seed, as it did in the original. The best run is seed 13 with score **0.972**, in line with the best of 0.985 over 60 seeds in the original.

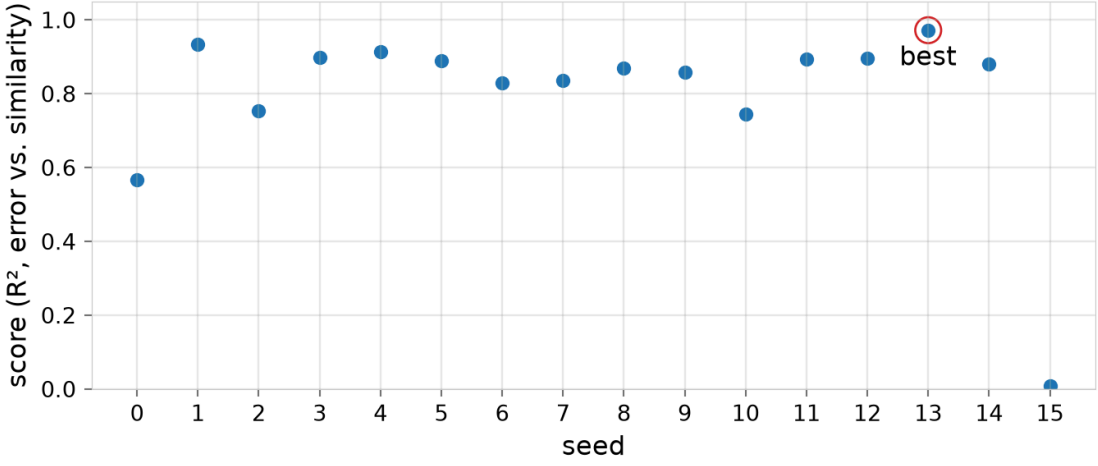
The schedule

Training uses a dopesheet: a table of keyframes for the regularizer weights and the learning rate, much like a keyframe track in animation. `mini.temporal` interpolates between keyframes with a minimum-jerk curve, which eases values in and out. Everything ramps to zero by step 1425, leaving the last 75 steps as pure reconstruction fine-tuning.



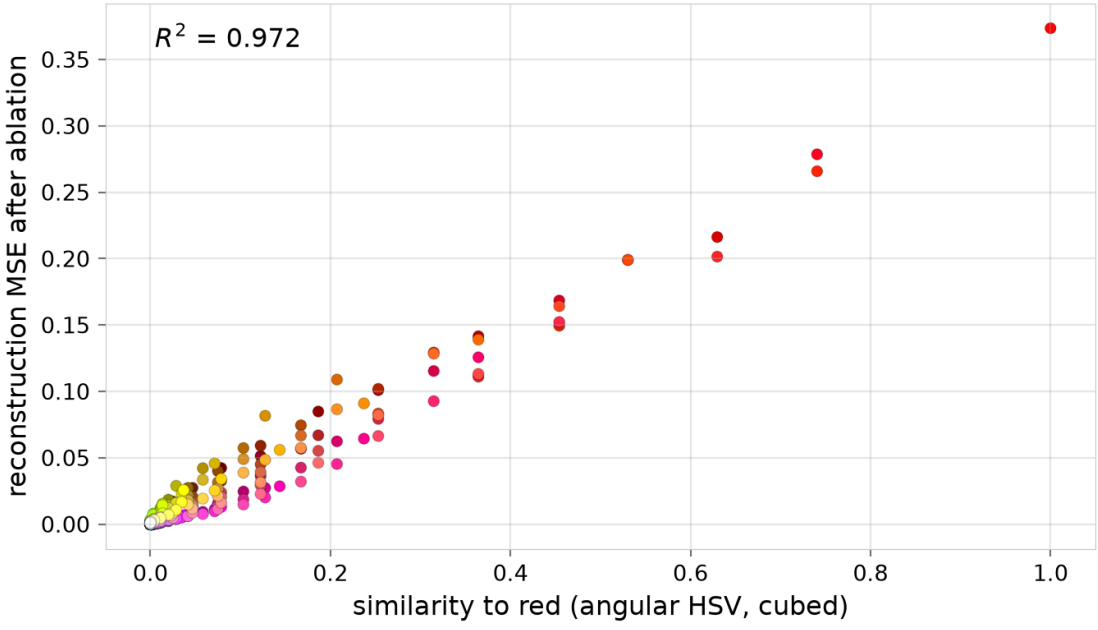
Scores across seeds

Each run trains from the same schedule but a different seed, and outcomes vary substantially with the starting point. That matches the original, which handled the variance with a 60-seed sweep and selection along a Pareto frontier (the best trade-offs across several objectives). We keep the same shape at smaller scale: sweep the seeds, then take the best scorer.



Was the deletion clean?

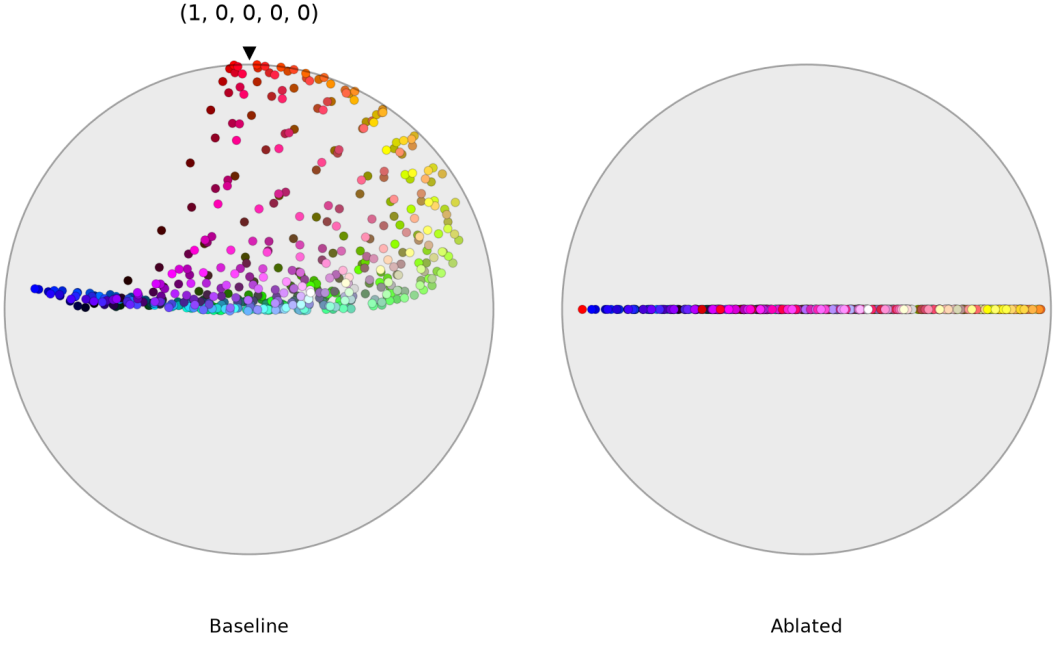
For the best run, we ablate the first latent axis and reconstruct the full 8×8×8 RGB grid. If the anchored concept was fully contained in that axis, the damage should be proportional to the similarity of each color to red, and colors unlike red should be untouched.



Baseline reconstruction is near-perfect (mean MSE 9.33e-06). After ablation, red-like colors (similarity^3 > 0.5) have a mean error of 0.248, while colors unlike red (similarity^3 < 0.01, 337 of 512 grid points) sit at 6.26e-04: deleting red barely affects them.

The latent geometry

The same result shows up in the bottleneck geometry. Before ablation, position along the first axis tracks similarity to red: pure red sits at 1, blues, greens, and grays sit near 0 (pushed off the axis by the anti-subspace term), and warm colors fall in between. Ablation zeroes that axis, which collapses that one direction but leaves the arrangement of the other dimensions intact. That's why the effect on reconstruction is proportional to redness.



Findings

The result reproduces the M1 finding, but the point of this run was the plumbing: the same dopesheet driving a JAX training loop through `mini.temporal`; a 16-way `ctx.map` fan-out on Modal with memoized, resumable records; artifacts and refs flowing through the store to this report; and `@themed` figures published via the report bundle. Training a single seed takes about ten seconds, and the runs are bit-identical between local CPU and Modal. So, the infrastructure seems to work.

Next: the M2 experiments proper, anchoring concepts in the residual stream of a small transformer on the color-mixing task (see the README).

---

1. An extra term added to the training loss, alongside the main reconstruction objective, that shapes the geometry of the latent space without changing what the model is fundamentally trying to do. ↩