Ex 2.1.3: named colors only

> No hex codes, just one opaque token per color and nothing in the text to say that colors are values at all. The model infers the geometry from co-occurrence alone: its embeddings hold the RGB cube as a linear subspace, it computes mixes in value space just before answering, and its held-out guesses land on or beside the right color. Exact match rises and falls with vocabulary size, while geometric closeness improves steadily.

Every experiment so far has taught the model colors two ways: as names (`red`) and as hex codes (`#f00`), with alias lines tying the two together. The hex form spells a color one channel at a time, so the geometry of color space is legible in the text.

In the language of this experiment, each color is a single opaque token, and every sentence is a mixing equation between named colors whose mix is also named. Nothing about the tokens says that colors live on a 3D grid, that `purple` sits midway between `red` and `blue` , or that colors are values at all. The only structure left is the co-occurrence statistics of the mixing table.

The main question is whether the model can infer the color-space geometry from that alone. A model that discovers it should be able to fill in cells of the table it has never seen.

Many operand pairs mix to a color that has no name in the vocabulary, so an exact answer is impossible there. A model that holds the geometry should still guess a nearby name. Measuring how close the guesses land gives a graded score where exact-match accuracy would only record a miss.

The language and a vocabulary-size sweep

A vocabulary here is a sub-grid of the 16-level RGB cube: pick a set of levels, then give a name to every point on that sub-grid. We sweep four such grids. The 3-level grid is the familiar 27-color palette from the base language, with proper names. The denser grids use synthetic names that encode the value (`c05f` is the color `#05f`), which helps us read through failures later on. Each name is still a single token, so the model sees an opaque id and has to learn what it means from usage.

grid	levels per channel	colors	example
`v27`	0, 8, 15	27	`red`, `purple`
`v64`	0, 5, 10, 15	64	`c05f`
`v216`	0, 3, 6, 9, 12, 15	216	`c9c3`
`v4096`	all 16	4096	`c27b`

Every training line reads `a + b = mix(a, b)`, six tokens including the newline, with both operands and the mix drawn from the vocabulary. Mixing is the same channel-wise round-half-up mean used in earlier experiments. A pair whose mix lands on the vocabulary is "closed"; 20% of the distinct closed pairs are held out of training entirely. Pairs whose mix falls between grid cells are "open": they never appear in training, since there is no name to write on the right-hand side, and we use them only as probes of graded generalization. The corpus is 100,000 with lines drawn at random. The model architecture is the d64-L4 nGPT condition from ex-2.1.1, with 3 seeds per grid.

Here are a few lines from two of the corpora.

`v27`:

```
cream + spring = mint
red + blue = purple
gray + gray = gray
white + magenta = orchid
spring + rose = gray
```

`v216`:

```
c930 + c3f0 = c690
c900 + c900 = c900
c00c + cc00 = c606
c393 + cfff = c9c9
cc36 + c03c = c639
```

A denser grid means more colors to name, which is a harder vocabulary, but it also means many more closed pairs constraining each color, which is richer evidence about how colors relate. At the far end, the full 16-level grid has 4096 colors and 8.4 million distinct closed pairs, of which a 100k-line corpus can visit only about 1%. The smallest grid is tiny: 49 distinct closed pairs at `v27`, so the table is easy to memorize, and the held-out fifth of it asks the model to infer answers from very little evidence.

Measurements

From the next-token distribution at the `=` position we compute exact-match accuracy, and the cross-entropy of the true answer on closed pairs (where a single answer exists).

For all pairs (including open pairs), we also find the graded score as the RGB distance from the highest-probability color of the model to the true mix. There are two reference distances per prompt to compare against: the *floor* (the distance from the true mix to the nearest vocabulary color, which is zero for closed pairs) and *chance* (the mean distance over the whole vocabulary, what a guesser with no knowledge would score on average).

We also fit probes on the residual stream to inspect the shape of the latent embeddings.

Hypotheses

Written before looking at any results.

H1. Seen pairs are answered almost perfectly on every grid; the corpus gives each seen pair many effective repetitions, even at `v4096`.

H2. Held-out accuracy rises with grid size. `v27` stays near zero, since 39 training pairs constrain 27 colors only weakly. At the other end, `v4096` can barely memorize even its seen pairs, so if it learns them at all it must be generalizing, and its held-out accuracy should come close to its seen accuracy.

H3. Where held-out accuracy is high, guessed colors on open pairs land near the floor distance, well below chance; where it is low, guesses sit at or near chance.

H4. The R^2 of the embedding-to-RGB probe and the low-dimensionality of the embedding table (top-3 explained variance) rise with grid size, and a PCA of the embeddings shows the color cube where, and only where, held-out performance is good.

H5. Less certain, so a watch item: at `v4096` the fixed 100-epoch budget may not be enough for the rule to settle. If that happens, seen accuracy will be middling too, and the run would be telling us "undertrained" rather than "impossible".

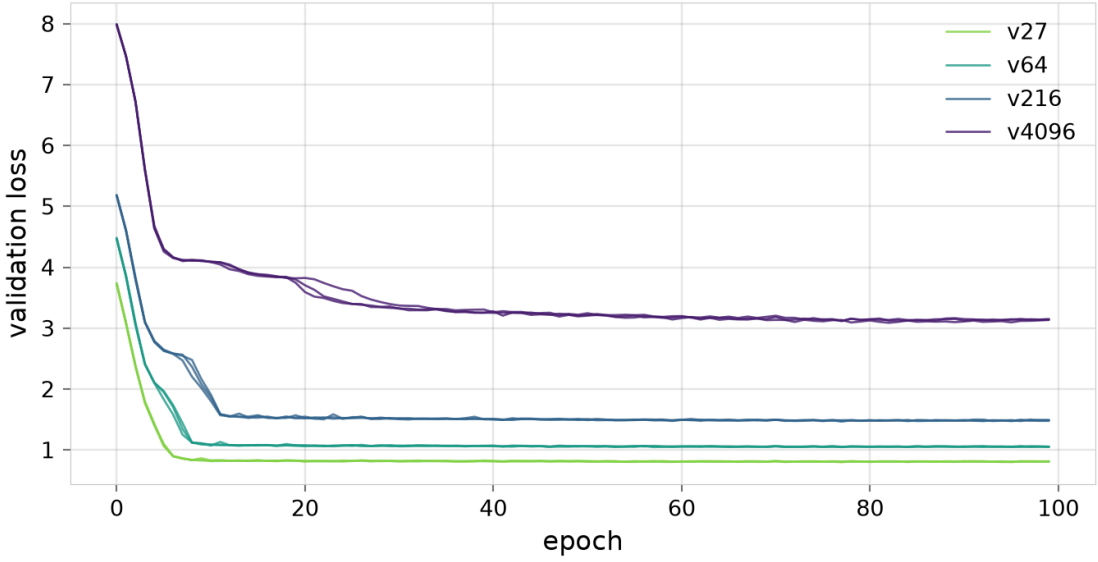
Headline numbers. Mean held-out accuracy by grid: `v27` **0.27**, `v64` **0.59**, `v216` **0.99**, `v4096` **0.65**. On every grid but the smallest, that is far above what a model knowing only the neighborhood of the answer would score (`v27` 0.18, `v64` 0.16, `v216` 0.15, `v4096` 0.14), so the model does infer the geometry (except `v27`, which is discussed below). Misses are interesting: at `v4096` they land a mean distance of just 0.025 from the true mix (chance is 0.61). The sections below build that picture up piece by piece, starting with the pairs the corpus sampler actually produced.

grid	colors	closed pairs	distinct pairs in corpus	held out	open pairs
`v27`	27	49	66	10	302
`v64`	64	224	243	45	1,792
`v216`	216	2,808	2,462	562	20,412
`v4096`	4,096	8,386,560	99,218	256	0

Pair accounting per grid. "Closed pairs" counts the distinct unordered pairs whose mix lands on the vocabulary (self-pairs excluded); the corpus samples, with repetition, from the side that isn't held out. At `v4096` the corpus reaches about 1% of the closed pairs, and every pair is closed, so there are no open pairs to probe.

Training

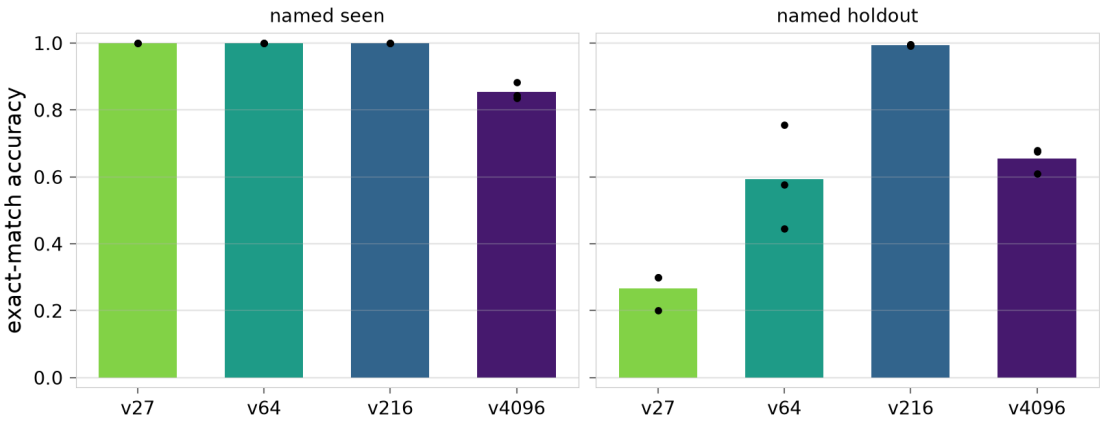
All twelve conditions train stably. Each grid settles within the 100-epoch budget, `v4096` included, whose curve is flat over its last twenty epochs. So the results that follow reflect what this budget and schedule produce at convergence.



Validation loss per epoch, with three thin lines per grid, one per seed. The grids aren't comparable to each other, since chance loss grows with the vocabulary size; what matters here is that each curve settles.

Exact-match accuracy

Can the model answer equations it has never seen?



Exact-match accuracy by grid. Bars show the mean over three seeds; dots are individual seeds. ***Left:*** *Seen pairs show whether training worked at all;* ***Right:*** *held-out pairs show generalization.*

Every grid learns its seen pairs almost perfectly, apart from `v4096`, which reaches 0.85. That is the one grid where even the training pairs are hard to memorize, since its corpus visits 99k distinct pairs roughly once each. Held-out accuracy then follows a non-monotonic path: 0.27 at `v27` (that's 3 of its ten held-out pairs), 0.59 at `v64`, 0.99 at `v216`, near enough to solved, and back down to 0.65 at `v4096`.

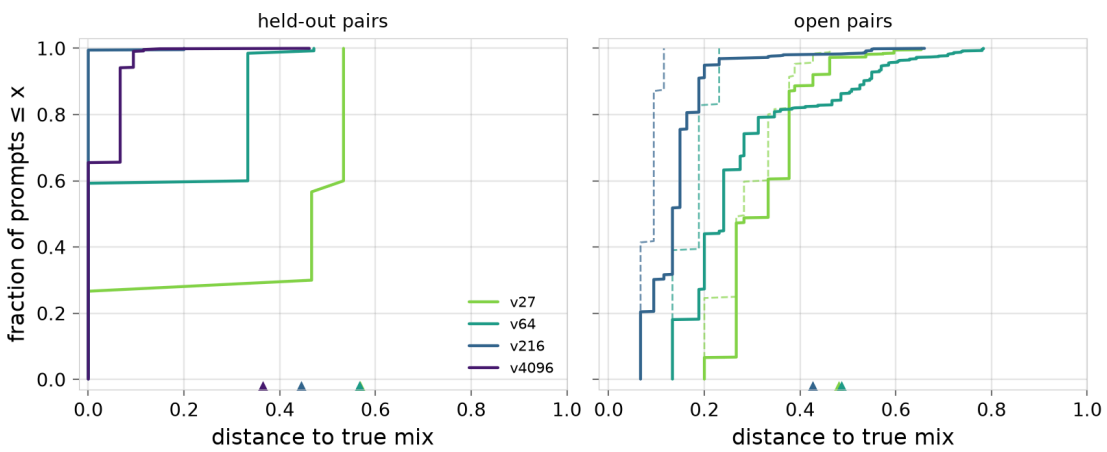
Only three of those four numbers mean much. On a 27-color grid the true mix has just 4.6 one-step neighbors on average, so a model that has located the neighborhood of the answer and picks within it scores 0.18 without being able to name anything. The `v27` score of 0.27 sits 1.2 std above that, over ten distinct pairs, which is not a difference worth reading. The denser grids clear the same bar easily: 14 std at `v64`, 65 at `v216` and 40 at `v4096`.

So the `v27` score is not evidence on its own. It is tempting to read it as a shift from the base language, where `named_holdout` sat at zero through every intervention ex-2.1.2 tried, but two of ten pairs on a grid this coarse cannot carry that. The case that the geometry is inferable from names rests on the denser grids here, and on the embedding probes further down, which reach $R^2 \approx 0.66$ even at `v27`.

The `v4096` misses are not scattered, either. Pooled over seeds, 221 of 265 held-out misses differ from the true mix in a single RGB channel, and 220 of those are off by one grid level (`v64`: 53/55; `v216`: 4/4). We checked whether these misses are more likely on rounding cases, meaning channels where the operand sum is odd so the mean has to round, and they don't: about half of the one-channel misses are rounding cases, which matches chance.

Closeness

Let's switch to the graded view. For every prompt we take the highest-probability color from the model and measure its RGB distance to the true mix, and plot the cumulative distribution. For each distance x, the curve shows the fraction of prompts that landed within x of the answer.



How close do guesses land? Each line is the cumulative distribution of the distance from the guess to the true mix (in unit-cube units, pooled over seeds); higher and further to the left is better. On held-out pairs an exact answer exists, so the height at distance 0 is the exact-match accuracy. On open pairs no name is quite right, so the dashed line shows the best achievable (nearest-name) distance there. The triangles under the x-axis are the prompt-blind constant, the score for always answering the center of the training answers.

Guesses are better than chance everywhere, and comfortably inside the prompt-blind constant, which is stricter: mixes are midpoints, so they cluster centrally, and always answering the middle name scores 0.48 on `v27` open pairs where chance is 0.82. That much supports the picture in H3.

grid	model	floor	2 nearest	prompt-blind	chance
`v27`	0.326	0.290	0.302	0.481	0.820
`v64`	0.278	0.174	0.202	0.486	0.732
`v216`	0.141	0.086	0.115	0.426	0.677

Mean distance to the true mix on open pairs, against four references.

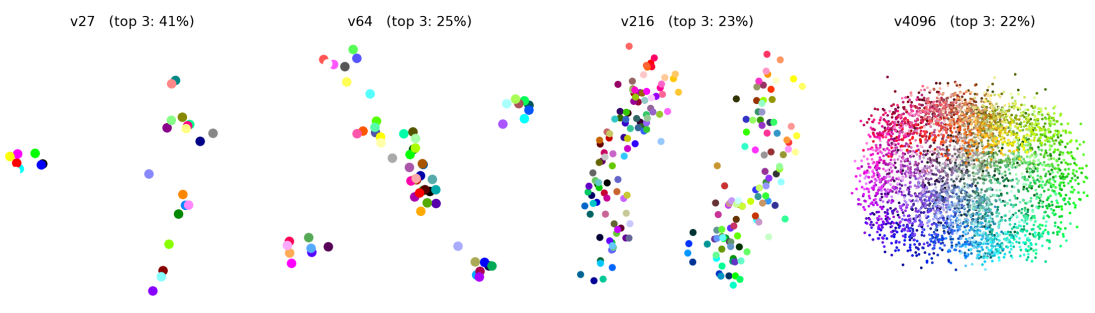
The floor is a harder reference than it looks, though, because it asks the model to break a tie it may have no way to break. An open mix falls between grid points, so two names bracket it, often at the same distance. A model that has located the answer but picks between those two at random is the "2 nearest" column, and every grid sits behind it: `v27` guesses 0.326 against 0.302 for that baseline. On the nearest-name rate the gap is wider, 0.50 against 0.82. So the guesses are near the true mix, but not as near as a model with the same positional knowledge and a coin could get.

H3 predicted near-floor guessing only where exact-match accuracy was high, but the graded signal does turn out to be present even where exact-match accuracy is low.

Embedding geometry

The behavior suggests the model knows where the colors are in the cube. Since colors are single tokens, everything the model knows about the identity of a color must live in the embedding of that token. If the model inferred the hidden space, the embedding table should hold a color cube: some linear view of the 64-dimensional embeddings under which the tokens arrange themselves by their RGB values. A ridge probe from embeddings to RGB measures how true that is, and a PCA projection (principal component analysis, which finds the few directions along which the vectors vary most) gives an unsupervised look at the dominant structure of the table.

We'll look twice: first with PCA, which is unsupervised and so answers "what is this table mostly doing?", then with the probe weights, which answer the narrower question.



The embedding table, projected onto its own first two principal components (seed 0), with each token drawn in the color it names. Only `v4096` shows the hue wheel we thought we might find. The percentage is how much of the table variance its leading three components hold — the most any three directions could — of which the panel draws the leading two.

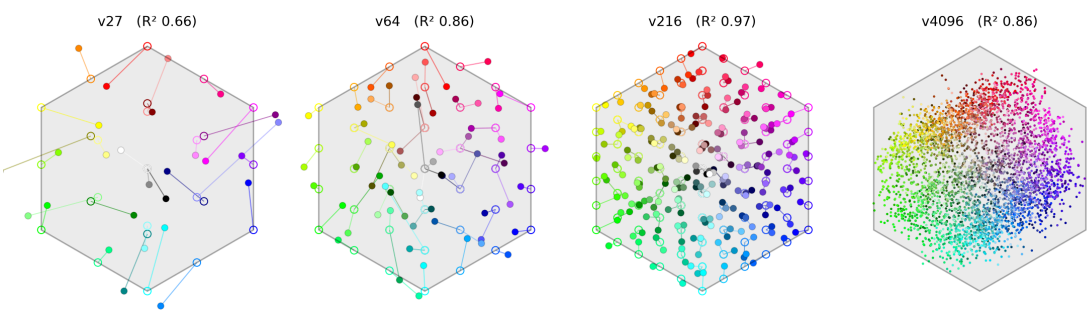
The PCA plots give the impression the cube arrives late — somewhere between `v216` and `v4096` — but it doesn't. PCA keeps the directions along which the table varies most, and apparently color is not what the table mostly does. These embeddings are also the tied language model head for predicting the next token, so perhaps two adjacent colors need well-separated directions for the softmax to keep them apart, even when their color values are nearly identical.

A cube needs three directions. A couple of natural ways to pick them are:

- PCA: Pick the three directions the table varies most along. The amount of the total variance captured by those directions is shown as the percentages on the panels above.
- Probe: Fit to predict RGB and indifferent to how much the table varies along them.

How well do they agree in this case? If representing color is what the table mostly does, the probe should be reading close to the top-variance directions and the two should capture similar amounts. They don't, at least not until the vocabulary gets large. The probe directions hold 25% as much variance as the best three at `v27` and 26% at `v64`, then 56% at `v216` and 80% at `v4096`. So it is only at the dense end that the color directions and the high-variance directions are nearly the same thing — and only there that PCA stumbles onto the cube.

Let's see what the probe found. Its weights are a map from the 64-dimensional embedding onto RGB, so we can use it to decode the color tokens. If the table holds a color cube, the decoded values should be similar to the source data. To keep this from being a statement about the memory of the probe rather than the geometry of the model, every token is placed by a probe fit on every *other* token in the vocabulary.



The same embedding tables (seed 0), now read through the probe instead of through variance: each token is placed at the RGB the probe predicts for it, and drawn in the color it actually names. Open rings mark where a token should sit, so the displacement (lines) show the error. `R²` is the same fit as a single number, averaged over seeds. Rings are left off at `v4096`, where 4096 of them would bury the colors.

So the cube geometry is in the embedding table, even in the smallest model. `v216` is the clearest case: held-out tokens land almost exactly on their own lattice sites, a hue wheel with greys in the middle, from a table whose principal components showed only a smear. `v64` is recognizably the same shape with visible slop, and even `v27` has most of its colors in the right neighborhood.

Probe R^2 quantifies how well it fit: 0.66, 0.86, 0.97, 0.86 across the four grids.[1] From `v64` up, RGB is close to a linear function of the embedding.

Does a probe-decoded token land nearer its own name than any other?

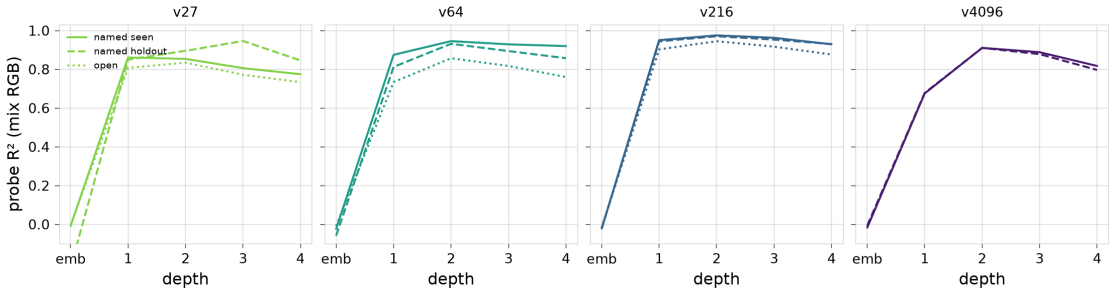
grid	nearest name is its own
`v27`	51%
`v64`	61%
`v216`	76%
`v4096`	7%

At `v27` the answer is a coin flip. `v4096` makes the same point from the other end: its R^2 of 0.86 is respectable, but sixteen levels per channel pack the names so tightly that a decoded token is almost never nearest its own. So the probe finds the cube, but it does not decode individual tokens well enough to name them at any of these grid sizes.

H4 expected the embedding table itself to turn low-dimensional, and the PCA says it doesn't: even the best three directions hold only 41% of the variance at `v27`, and that falls to 22% at `v4096`. The cube is in there as a subspace the probe can read, but most of the embedding variance is doing something else.

Mixing in value space

The embeddings hold the geometry of individual colors; the last question is about the computation that combines them. At the pre-answer position (`=`), does the residual stream already contain the value of the mix before the answer is emitted? We'll check with probes fit at each depth on seen prompts and then apply them to held-out and open prompts.



Ridge probes from the pre-answer residual stream to the RGB of the true mix, one per depth (0 = embeddings, 4 = final layer), fit on half the seen prompts (seed 0). Transfer to held-out and open prompts tells a genuine value-space computation apart from a probe that memorized its fit set.

Is the answer computed in value space before it is emitted? Apparently so, and the probes transfer well. Peak held-out R^2 is 0.95 at `v27` , 0.93 at `v64` , 0.97 at `v216` , and 0.91 at `v4096` (seed 0), with most of the value present from depth 1 or 2 onward. This quite different from the base language. There, ex-2.1.2 found that the answer channels are computed just in time and cleared away after emission (for hex answers), so a full "result" concept wasn't readable at any single position. But now with one-token answers there is no schedule to spread the work over, so the whole mix must be present at the pre-answer position.

Even the `v27` held-out prompts probe at about 0.9 mid-stack while their exact-match accuracy is 0.2 to 0.3. The model computes roughly the right value, and then the readout picks a neighboring name.

Example completions

Here are a few concrete completions (we picked the widest misses).

a	b	true mix	model guess	distance	floor
magenta	blue	violet	blue	0.53	0.00
white	lime	mint	chartreuse	0.53	0.00
navy	rose	purple	navy	0.53	0.00
cyan	blue	azure	lavender	0.53	0.00
black	red	maroon	red	0.47	0.00

`v27` · *named holdout: the 5 widest misses (seed 0).*

a	b	true mix	model guess	distance	floor
c3ff	c900	`#688`	c66c	0.30	0.09
cfff	c600	`#b88`	c966	0.23	0.12
c6ff	cf66	`#bbb`	c999	0.23	0.12
cff9	cccc	`#eeb`	ccc9	0.23	0.12
c30f	c6f0	`#588`	c396	0.20	0.12

`v216` · *open: the 5 widest misses (seed 0).*

a	b	true mix	model guess	distance	floor
c15b	c00f	c13d	c21d	0.15	0.00
cc02	ce1f	cd19	cd17	0.13	0.00
c017	c033	c025	c136	0.12	0.00
c511	cfe1	ca81	ca72	0.09	0.00
cb75	ce71	cd73	cc83	0.09	0.00

`v4096` · *named holdout: the 5 widest misses (seed 0).*

The `v27` misses are neighbor names (`violet` guessed as `blue`, `maroon` as `red`); the `v216` open-pair guesses sit a step away from the floor; and the worst held-out misses at `v4096` still agree with the true mix on two of the three channels.

Several of the `v27` rows return one of the operands, may or may not be an echo. Returning an operand and returning a neighbor are close to the same thing here. 10 of 30 guesses are operands, against 40% for a guess drawn uniformly from the neighbors.

Findings

1. The model infers the color-space geometry better without hex scaffolding. From mixing co-occurrences alone it builds embeddings that are linear color cubes, computes mixes in value space at the pre-answer position, and generalizes to held-out pairs. When it guesses a held-out answer, the guess is very close.
2. Performance varies with vocabulary size. Exact match is non-monotonic, while geometric closeness improves monotonically.

---

1. Probes fit per-color with leave-one-out, which matters most at the small grids. A single half/half split leaves the `v27` probe fitting a 64→3 map from 13 points, and scores it 0.48 against 0.66 here; `v4096` is unmoved either way (0.86 against 0.86). ↩