Replicating Astra's Weight Edit in Qwen3.5-0.8B

Abstract

Takehi et al. reported that Astra improved Qwen3.5-0.8B by editing its weights without gradient descent. We tested the released edit, compared it with a difference-of-means script and decoding changes, and attempted the task with other frontier models. On 64 practice problems with four samples each, the released edit raised accuracy from 10.2% to 64.8%. The script reached 60.9%, recovering 93% of the gain. It used Astra's layers, scale, and choice of 64 columns per layer, but estimated its own directions and selected its own columns. Its directions had cosine similarities of about 0.78 with Astra's. The best decoding change reached 47.7% accuracy. Four blinded runs with Opus 5.5 and Sonnet 5.5 found edits that improved their development scores, but stopped after 18–23 forward-pass requests. The editors chose different test problems, so their gains cannot be compared directly with Astra's. Our tests reproduced the released edit's improvement on practice problems. The script recovered most of that gain with several choices supplied; we have not tested whether it could discover those choices independently.

Why we looked at this

In Experiment 8 of *Sparks of Weight-Space Programming*, Astra was given a Python environment, access to Qwen3.5-0.8B's weights, practice problems, and a limited number of requests for generations or internal measurements. It could edit the weights, but could not use gradients. With a budget of 50 requests, it found an edit that raised MATH-500 accuracy from 7.9% to 48.1%.[1]

The edit changes two MLP output matrices. The original authors traced much of the model's failure to repetitive text that continued until the token limit. Their final report describes a direction associated with repetition and an edit to the neurons writing along it.[2]

We checked whether the released edit worked on our problems, whether a script or decoding change could recover the gain, and whether other frontier models could find an effective edit.

Prior work estimates behavioral directions from activations and uses them to steer or edit models.[3,4,5] We used a difference-of-means method for our baseline. Because we had already read Astra's solution, this baseline tests reconstruction of a known edit; it is not an independent search.

Evaluation

We used Qwen3.5-0.8B at revision `2fc06364`, with thinking enabled and bfloat16 weights, served through vLLM 0.33.0 on single H100 GPUs. We followed the original Qwen evaluation settings: temperature 0.6, top-p 0.95, top-k 20, presence penalty 1.5, and a limit of 32,768 generated tokens. We graded the last `\boxed{}` answer with math-verify.

Our main comparison uses 32 GSM8K-train problems and 32 MATH-train problems, with four samples per problem: 256 answers per condition. We fixed the sampling seed for each problem and sample and compared conditions in pairs. Tables report gains in percentage points and ±1 standard error of the paired difference, clustered by problem.

We used practice problems throughout. MATH-500 and GSM8K-test have not yet been evaluated. A separate model screen gave 10.9% unedited Qwen accuracy; the baseline for the main paired comparison was 10.2%. We distinguish these runs below.

1. A model that rarely finishes

We ran Qwen3.5-0.8B and six other small reasoning models with thinking enabled, using the same practice-set size and sampling settings, to check how often they reached the token limit.

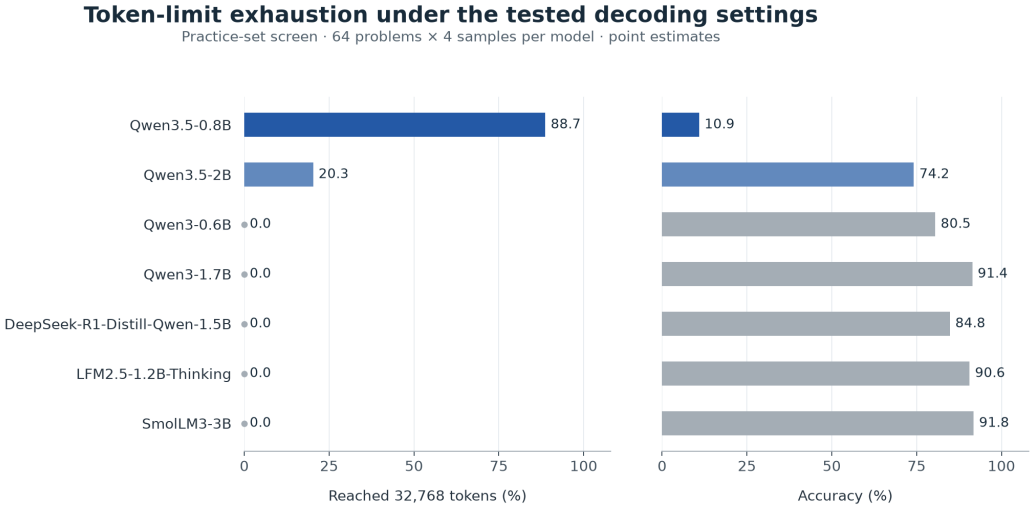


Figure 1. Observed token-limit rates and accuracy, with 256 answers across 64 problems per model. The figure shows point estimates. Reaching the token limit measures truncation; repetition can also occur in answers that finish.

Model (thinking on)	Hit token limit (%)	Accuracy (%)
Qwen3.5-0.8B	**88.7**	**10.9**
Qwen3.5-2B	20.3	74.2
Qwen3-0.6B	0.0	80.5
Qwen3-1.7B	0.0	91.4
DeepSeek-R1-Distill-Qwen-1.5B	0.0	84.8
LFM2.5-1.2B-Thinking	0.0	90.6
SmolLM3-3B	0.0	91.8

Qwen3.5-0.8B reached the limit on 227 of 256 answers. Of the 29 that finished, 28 were correct; none of the truncated answers were correct. Its 88.7% truncation rate was close to the original study's first practice batch, where 85 of 96 answers reached the limit. The authors reported 93.0% on MATH-500.[6]

Qwen3.5-2B reached the limit on 20.3% of answers. We observed no truncations in the other five reasoning models. We also tested three non-thinking instruct models: Qwen2.5-0.5B, OLMo-2-1B, and SmolLM2-1.7B. None exceeded 1.6%, with or without the presence penalty.

With 227 answers cut off, helping Qwen3.5-0.8B finish could improve its score considerably. We do not know how many of those answers would then be correct; the questions it already finishes may be easier. Truncation also misses some repetition: we found repetitive tails in completed instruct-model answers. The screen measures token-limit rates under one configuration, without optimizing decoding for each model.

2. Rebuilding the released edit

We rebuilt Astra's 50-pass edit with the original repository's `apply_edit.py`. Both edited tensor hashes matched the authors' manifest.

In the screen, the edited model reached 61.7% accuracy and its token-limit rate fell to 17.6%. The authors' separate practice confirmation reached 60.4% on 1,152 different problems.[7]

In the main paired comparison, accuracy rose from **10.2% to 64.8%**, a gain of **54.7 ± 4.8 points**. The edit changed 140 answers from wrong to right and none from right to wrong. The share reaching the token limit fell from 90.2% to 15.2%.

The original logs show a similar pattern: 576 of 579 rescued answers had previously reached the limit. Our practice-set replication does not yet tell us how the comparison methods perform on the original benchmarks, or whether the edits harm performance on other tasks.

3. The edit reverses and amplifies a direction

Astra's final report gives the formula for its edit, including $\alpha = 6$ and the choice of 64 columns per layer.[8] We examined the tensor differences to check that description.

Let W be an MLP output matrix, u a unit direction in the residual stream, and J the selected columns. The update is

$$W'_{:,J} = W_{:,J} - \alpha\, u\, u^{\top} W_{:,J}, \qquad \alpha = 6.$$

The edit applies this operation to 64 columns in each of layers 15 and 19, with a separate direction for each layer. The two directions are similar, with cosine similarity 0.849.

Within each layer, the change is approximately rank one. A rank-one fit leaves residuals of 0.23% in layer 15 and 0.29% in layer 19, consistent with bfloat16 rounding. The change in each column's

component along u is anti-correlated with its original component: the fitted correlation is −1.000, and α fits to 6.000 in both layers.

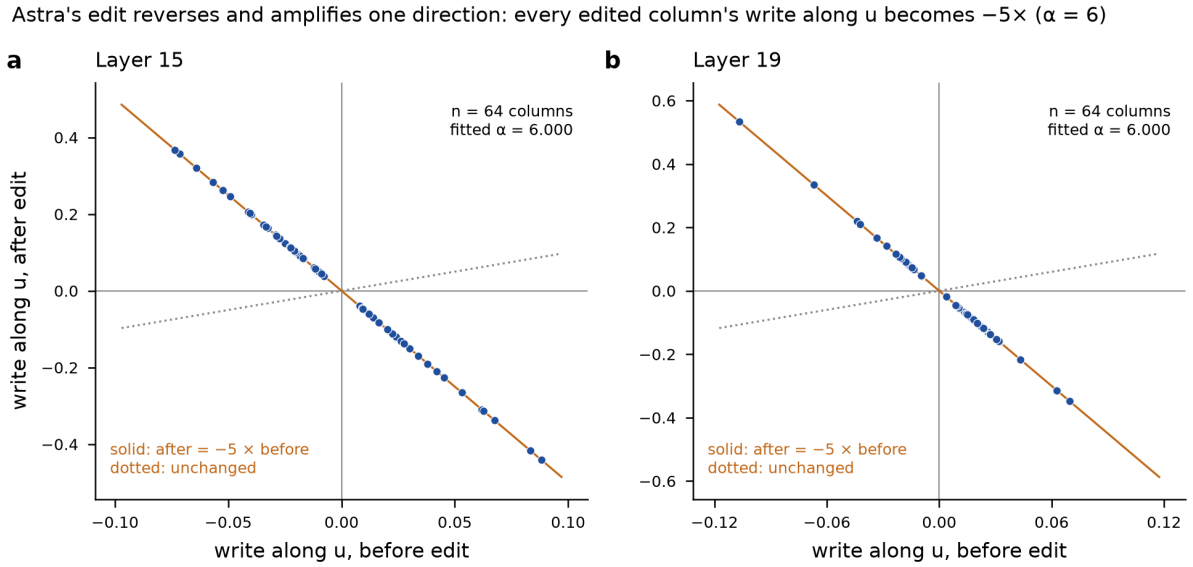


Figure 2. Each panel contains 64 edited columns. The solid line shows the relation predicted by α = 6: the component along the fitted direction becomes −5 times its original value. The dotted line shows no change. The small fit residuals are described in the text.

With α = 1, the formula removes the component along u, as in directional ablation. With α = 6, that component becomes approximately −5 times its original value. The projector is uu^{T}; the full operation $I - 6uu^{\mathrm{T}}$ reverses and amplifies one component of each selected column.

The tensor fit agrees with the formula in the logs. We next used a script to estimate the directions and select the columns for this update.

4. A difference-of-means script

We generated two answers from each of 128 additional practice problems to fit the script's directions. None of these problems appeared in the 64-problem evaluation set.

We marked a loop as beginning at the first token after which 64 consecutive 8-grams had all appeared earlier in the answer. This gave us 233 loop windows, each covering 512 tokens after onset, and 253 comparison windows taken from the 512 tokens before onset or from a finished answer. We call the latter windows "productive" to distinguish them from detected loops; the label does not require their reasoning to be correct.

For each layer, the script subtracts the mean hidden state in the productive windows from the mean in the loop windows, then normalizes the result. To choose columns, it multiplies each neuron's mean activation difference by its output column's projection onto that direction. The 64 highest-scoring columns receive the update above.

The script reads the target model's weights and activations and makes these choices without an LLM editor. For its best condition, we supplied **Astra's layers 15 and 19, α = 6, and 64 columns per layer**. We tested other scales and placements, plus a random-direction control using the same columns and a matched total weight-change norm.

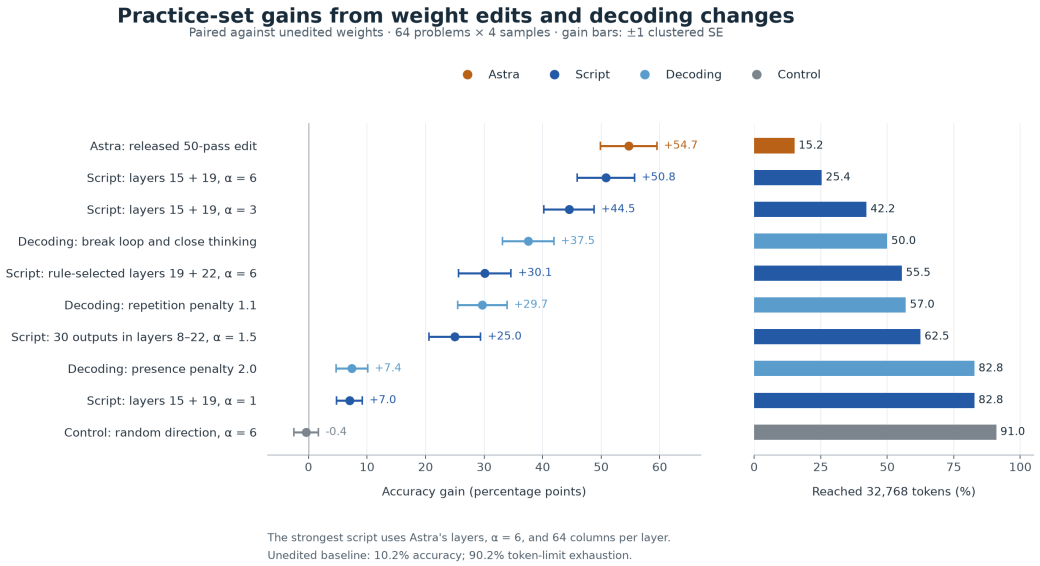


Figure 3. Accuracy gains against the unedited model on the main 256-answer comparison, with ±1 standard error clustered by problem. The right panel shows token-limit rates. The strongest scripted condition uses Astra's layers and scale. Decoding conditions are discussed in Section 5.

Condition	Accuracy (%)	Hit token limit (%)	Gain (points)	Wrong→right / right→wrong	Share of Astra's gain
Unedited	10.2	90.2	—	—	—
Random direction, α = 6 (control)	9.8	91.0	−0.4 ± 2.1	18 / 19	−1%
Script, layers 15 + 19, α = 1	17.2	82.8	+7.0 ± 2.2	30 / 12	13%
Script, all 30 attention and MLP output matrices in layers 8–22, α = 1.5	35.2	62.5	+25.0 ± 4.4	72 / 8	46%
Script, rule-selected layers 19 + 22, α = 6	40.2	55.5	+30.1 ± 4.5	88 / 11	55%
Script, layers 15 + 19, α = 3	54.7	42.2	+44.5 ± 4.3	117 / 3	81%
Script, layers 15 + 19, α = 6	**60.9**	**25.4**	**+50.8 ± 4.9**	**130 / 0**	**93%**
Astra, released 50-pass edit	**64.8**	**15.2**	**+54.7 ± 4.8**	**140 / 0**	**100%**

All rows use the same 256 paired answers. Gain shares are calculated from the unrounded counts; displayed percentages are rounded.

The best script produced 156 correct answers, compared with 166 for Astra's edit and 26 for the unedited model. It recovered 130 of Astra's 140 net rescues, or **92.9% of the observed accuracy gain**. The percentage refers to accuracy gain, not the fraction of weights recovered.

The estimated script-minus-Astra difference was -3.9 ± 2.9 points. A normal approximation gives a 95% interval of -9.6 to $+1.8$. That is too wide to claim equal performance: the script could be close to Astra or appreciably worse. We have not yet computed an interval for the gain ratio itself.

The script's directions had cosine similarities of 0.776 and 0.773 with Astra's in layers 15 and 19. It selected 45 and 30 of Astra's 64 columns. Two uniformly random sets of 64 columns drawn from 3,584 would overlap by about 1.14 on average, though that calculation ignores the shared structure in the weights and activations.

The two-layer script gained 7.0 points at $\alpha = 1$, 44.5 at $\alpha = 3$, and 50.8 at $\alpha = 6$. Borrowing Astra's scale made a large difference in this comparison.

We also tried selecting the layers automatically. Our rule chose the two MLPs among layers 4–22 whose outputs contribute most to the loop-minus-productive difference at the last layer. It selected layers 19 and 22. At $\alpha = 6$, that edit reached 40.2% accuracy and recovered 55% of Astra's gain. It still used Astra's scale and the same form of update.

Some edits also broke previously correct answers: twelve for $\alpha = 1$, three for $\alpha = 3$, eight for the broad 30-matrix edit, and eleven for the rule-selected layers. The script using layers 15 and 19 with $\alpha = 6$ broke none in this sample. The random control fixed 18 answers and broke 19, leaving little net change. We tested only one random direction.

5. Changing the decoding

We tested three decoding changes with the weights left alone: increasing the presence penalty, adding a repetition penalty, and detecting a loop, cutting it, and closing the thinking block.

Decoding change	Accuracy (%)	Hit token limit (%)	Gain (points)	Share of Astra's gain
Presence penalty 1.5 → 2.0	17.6	82.8	+7.4 ± 2.7	14%
Add repetition penalty 1.1	39.8	57.0	+29.7 ± 4.2	54%
Detect the loop, cut it, and close the thinking block	47.7	50.0	+37.5 ± 4.4	69%

These conditions use the same problems, seeds, and paired-error convention as the weight-editing comparison. They also appear in Figure 3.

Increasing the presence penalty recovered 14% of Astra's gain, adding a repetition penalty recovered 54%, and the loop-break procedure recovered 69%. The loop-break procedure fixed 96 answers and broke none, reaching 47.7% accuracy.

The released weight edit scored 17.2 points above the best decoding condition. That difference does not identify how it works. A weight change could prevent a loop, help the model escape one, or affect termination and answer formatting. Accuracy and truncation rates cannot distinguish these explanations. We tested only three decoding changes.

6. Four blinded editor runs

We gave Opus 5.5 and Sonnet 5.5 access to the weights and asked them to improve the model. Each ran once with each of two prompts. Prompt A used the original opening message with the harness details changed. Prompt B removed its three mentions of repetition. Neither editor saw the original post, Astra's edits, or our results.

The editors had a persistent Python kernel, the original practice problems, and a budget of 25 forward-pass requests. The kernel had no GPU, and gradients were disabled. A request could contain a batch of generations or a teacher-forced read of internal activations. As in the original experiment, a "pass" means a tool request, not a single example.

All four runs stopped at the Modal spending limit after 18–23 requests. None used its full budget. Our harness differed from Astra's 50-pass run in several other ways: an 8 GB kernel memory cap, a 100 MB cap on teacher-forced outputs, adaptive thinking rather than maximum reasoning effort, and the host's added system prompt.

The largest gain observed during each run was:

Editor	Prompt	Passes used / budget	Best observed paired gain (points)
Opus 5.5	A: repetition hints	18 / 25	+21.9 ± 2.6
Sonnet 5.5	A: repetition hints	18 / 25	+26.2 ± 3.1
Opus 5.5	B: hints removed	21 / 25	+36.7 ± 3.2
Sonnet 5.5	B: hints removed	23 / 25	+37.5 ± 3.1

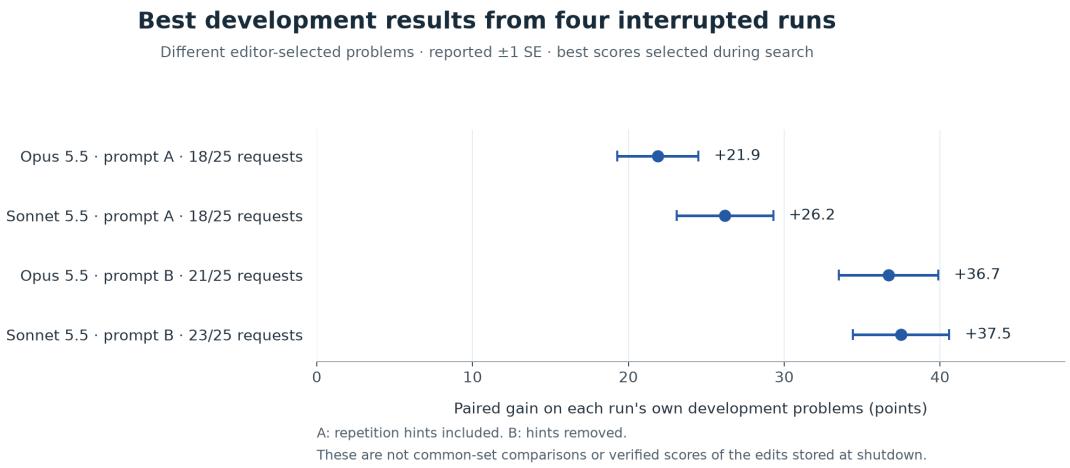


Figure 4. Each point shows the best development gain observed during one run, with its reported ±1 standard error. The editors chose their own test problems, and the error bars do not account for selecting the best result during search. These are not scores on a shared benchmark.

Each test paired an edit with the unedited model on 200–256 answers. The editors chose their own problems; their sets overlapped our main evaluation by at most four problems. Baseline accuracy ranged from 7% to 17%, except for one Sonnet-B test at 0% correct and 100% truncated. Different problem sets can produce different gains even for the same edit.

Figure 5 plots the paired accuracy gain over the unedited model against the forward-pass request number. The vertical axis is a difference in percentage points, not absolute accuracy. Sonnet B's sharp drop at pass 20 occurred on the set with 0% baseline accuracy and 100% truncation. It does not by itself show that the edit deteriorated: both edits and test problems could change between points.

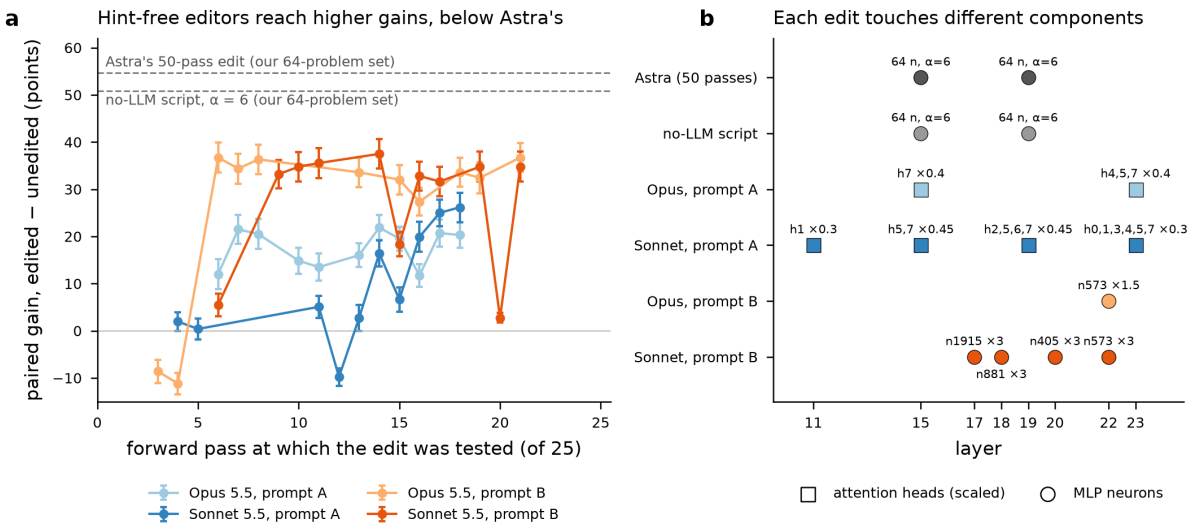


Figure 5. Original editor-run figure, reproduced unchanged. Left: gains at successive test requests, with the original error bars. Right: components changed by Astra, the script, and the stored editor candidates. The dashed lines show Astra and the script on our separate 64-problem set. Although the original panel title compares the gains, the different evaluation sets do not support a ranking of the editors.

The edits stored when the runs stopped are listed below. The best scores above come from the search history; we have not evaluated the stored edits on our main set.

Editor and prompt	Components changed	Operation
Opus A	Layer 15, head 7; layer 23, heads 4, 5, 7	Scale each selected head's output by 0.4
Sonnet A	Layer 11, head 1; layer 15, heads 5, 7; layer 19, heads 2, 5, 6, 7; layer 23, heads 0, 1, 3, 4, 5, 7	Scale the selected heads by 0.3 in layers 11 and 23, and by 0.45 in layers 15 and 19
Opus B	Layer 22, neuron 573	Scale its MLP output column by 1.5
Sonnet B	Layer 17, neuron 1915; layer 18, neuron 881; layer 20, neuron 405; layer 22, neuron 573	Scale the four MLP output columns by 3

All four runs found edits that improved their development scores. Both runs given repetition hints reduced attention-head outputs and selected layer 15 head 7 and layer 23 heads 4, 5, and 7. Astra's original 25-pass run also zeroed layer 15 head 7.

Both runs without hints identified the same four MLP neurons, in layers 17, 18, 20, and 22, and increased their output. Opus B retained only layer 22 neuron 573, scaled by 1.5; Sonnet B retained all four, scaled by 3. The single-neuron edit gained roughly 27–36 points across five tests on Opus B's own problems.

Vocabulary projections associated those neurons with "wait," "Wait," "waiting," and "delay." Increasing their output might help interrupt repetitive text. We have not tested that explanation; the token associations alone do not establish the neurons' roles.

We cannot attribute the choice of attention edits versus MLP edits to the prompt with only one run per condition. The different problem sets and selection of best scores during search also prevent a comparison of prompt effectiveness.

The stored edits scale heads or neuron columns. Astra's 50-pass edit instead subtracts a projected component from selected columns. To compare these approaches, we need to freeze the edits, evaluate them on the same problems, and repeat the searches.

Discussion

We reproduced the released tensors and the large practice-set improvement reported by Takehi et al. This supports their finding that Astra produced an effective weight edit without gradient descent.

Our script recovered most of the gain once we supplied the layers, scale, and number of columns. It did so using a difference-of-means direction, a method already used in representation engineering. The result alone does not tell us whether Astra needed an ability to interpret weights beyond applying those methods.

Astra still had to make the choices we supplied to the script. Our baseline was designed after we saw its solution. The weaker results at $\alpha = 1$ and with rule-selected layers show why that matters. We have not demonstrated that a script could find equally effective settings independently or at lower cost. That comparison would require a search defined without Astra's answer, with its inference, activation-collection, and selection costs recorded.

Qwen3.5-0.8B failed to finish most of its unedited answers. Helping it finish improved math scores, but our tests do not separate that benefit from an increase in mathematical competence. The original Gemma 3 270M follow-up did not establish a large math gain.[9] That failure applies to the searches they tested; other edits or search methods could fare differently.

Our findings support the Qwen editing result and provide a scripted baseline for further comparisons. We did not test the paper's other demonstrations of reading and constructing networks. Whether LLM editors discover effective changes more reliably or at lower cost than scripted searches remains to be tested under matched conditions.

Further experiments

We plan to freeze the released edit, scripts, decoding procedures, and stored editor candidates, then evaluate them on MATH-500 and GSM8K-test using the original sample counts. The editor comparison needs at least three runs per model and prompt, with comparable budgets and harness constraints. We would compare the final edits on a shared test set and report variation across runs.

The script needs a rule for choosing layers and scale without Astra's solution. We would choose those settings on fit and development data, then evaluate them on separate problems. Repeated random-direction controls and measurements of repetition, completion, and loop onset would help explain the effects of the edits.

We also want to try a controlled fault that changes answers without producing obvious repetition. Testing LLM editors and scripted diagnostics on that fault could show whether inspecting weights and internal measurements helps locate its cause.

Materials and disclosure

The original logs and edit reconstruction code are public. The authors release their logs, data, and networks under CC BY 4.0 and their scripts under MIT; third-party materials retain their own licenses.

We have not yet released the new experiment records. The planned repository will include editor transcripts, problem IDs, baseline code, per-answer results for the 11 main conditions, edit bundles, and full model and dataset hashes. We also need to document activation hooks and pooling, window exclusions, layer scoring, random-control normalization, decoding continuation rules, and overlap with Astra's selection and confirmation problems. These records are needed to reproduce the new measurements.

Two editor runs were mistakenly told that their first forward-pass request had failed, and they resubmitted it. The original results were later delivered and the duplicate requests cancelled. The transcripts need to identify these events and show how the requests were counted.

We used Claude to assist with the analyses, evaluation harness, and early drafting, and an OpenAI assistant to help review and revise the manuscript and prepare the revised figures. This rewrite introduces no new experimental results.

References

1. Takehi, R., Teoh, J., Zhang, A., and Khattab, O. *Sparks of Weight-Space Programming: LLMs can natively read & write neural network weights.* Blog post, November 2026. Article; Experiment 8 documentation.
2. Original 50-pass run, final report, including the $\alpha = 6$ formula and intervention checks.
3. Arditi, A., et al. "Refusal in Language Models Is Mediated by a Single Direction." NeurIPS 2024. arXiv:2406.11717.
4. Turner, A. M., et al. "Steering Language Models With Activation Engineering." arXiv:2308.10248.
5. Zou, A., et al. "Representation Engineering: A Top-Down Approach to AI Transparency." arXiv:2310.01405.
6. Takehi, R., Teoh, J., Zhang, A., and Khattab, O. *Sparks of Weight-Space Programming: LLMs can natively read & write neural network weights.* Blog post, November 2026. Article; Experiment 8 documentation.
7. Original 50-pass run, final answer and practice confirmation.
8. Original 50-pass run, final report, including the $\alpha = 6$ formula and intervention checks.

9. Original Gemma 3 270M follow-up documentation.