How Much of a Weight-Space Edit's Gain Does an Inference Configuration Recover? A Preregistered Check on Qwen3.5-0.8B

Theodore O. Cochran
AI for Altruism
theo@ai4altruism.org

AI for Altruism Technical Report A4A-TR-2026-005 Version 1, 2026-11-08
Status: Not peer reviewed.
Reviewer: Anonymous License: CC BY 4.0
`https://reports.ai4altruism.org/tr-2026-005/`

Preregistered at `https://osf.io/w4g7u` (DOI 10.17605/OSF.IO/W4G7U).

Abstract

MIT CSAIL researchers reported in November 2026 that GPT-6 Astra, editing 131,072 weights of the small reasoning model Qwen3.5-0.8B without training, raised its MATH-500 accuracy from 7.9 to 48.1 percent in thinking mode. The authors attribute the gain to repairing reasoning loops. Their evaluation did not report the model with thinking off, which is its default mode. We preregistered and ran that comparison. Both of the authors' thinking-on scores reproduced. Under the model card's non-thinking benchmark configuration, the unedited model reached 43.4 percent, recovering 82 percent of the accuracy gap between the unedited and edited thinking-on configurations. The edited thinking-on configuration kept a 7.85-point advantage over it, beyond the preregistered 5-point margin, while generating 5.9 times as many tokens per answer. With thinking off, edited and unedited models were statistically equivalent within that margin. On MATH-500, the edit's measured benefit is concentrated in the tested thinking-enabled configuration. Most of the reported improvement can be approached without modifying weights, by a change of inference configuration.

Keywords. weight-space programming, reproduction, preregistration, evaluation configuration, reasoning loops, benchmark validity

1 Introduction

On 2 November 2026, Takehi, Teoh, Zhang and Khattab of MIT CSAIL described *weight-space programming*: a frontier language model reading and writing the raw weights of another network, with no gradient training [1]. In their largest demonstration, GPT-6 Astra worked in a Python session holding the weights of Qwen3.5-0.8B, with a set of practice math problems to test edits on. It found that the model rarely finished its answers, traced the looping to one direction written by two of its layers, and subtracted that direction from 64 output columns in each, changing 131,072 weights. Under the same sampling settings, MATH-500 accuracy rose from 7.9 to 48.1 percent [1].

The authors were explicit about what the edit does. They describe it as repairing reasoning loops, not teaching new mathematics, and they report the matching negative result: editing Gemma 3 270M, a model that rarely loops, barely moved its math score. In their words, editing "could remove

an obstacle that hid an ability, but it could not add an ability the model lacked" [1]. This report does not dispute that account. It asks a different question, about evaluation: how much of the measured improvement could be had without editing at all, by changing how the model is run?

Two facts in the authors' materials made that question worth answering. Their evaluation ran Qwen in thinking mode with a 32,768-token limit, and under those settings the unedited model ran out the limit on 93 percent of MATH-500 answers [1]. And Qwen's model card states that the 0.8B model "operates in non-thinking mode by default," warns that in thinking mode it "is more prone to entering thinking loops compared to other Qwen3.5 models," and recommends tuning the sampling parameters and watching for such loops [3]. The authors' evaluation code defines the model card's non-thinking benchmark configuration, with the comment that thinking "looped at 0.8B," but the post reports no result for it [1, 2].

The contribution here is therefore a completeness check on an evaluation baseline. Its result bears on how large the edit's effect looks, not on whether the method works. It covers the released edit, the best of the authors' seven Qwen runs [2], and one benchmark, MATH-500.

2 Method

Preregistration. The design, analysis and decision rules were registered on OSF on 6 November 2026 at 14:19 UTC, before any thinking-off output was graded, on OSF's AsPredicted template [4]. The frozen generation, grading and analysis code is deposited with the registration as a zip; the registration records its hash and the hash of every file in it, and the copy downloaded from the registration's archive matched them. A pilot ran beforehand and is disclosed in the registration: the thinking-on conditions on 100 problems at one sample each, which reproduced the authors' numbers, and thinking-off on five problems per model, checked only for format and never graded.

Models and data. The unedited model was Qwen/Qwen3.5-0.8B at revision `2fc0636`. The edited model was the authors' released checkpoint, RikiyaT/Qwen3.5-0.8B-weight-space-programming at revision `d1671a1`. Before running, we verified that the edited model differs from the unedited one in exactly the two edited tensors, that each matches the hash of the checkpoint the authors evaluated, and that the authors' edit script rebuilds the edit bit for bit. The benchmark was MATH-500, the 500-problem subset of MATH [5] introduced by Lightman et al. [6], at revision `6e4ed1a` of the HuggingFaceH4 copy.

Conditions. Four conditions crossed model and mode, each on all 500 problems at 4 samples per problem, 8,000 answers in all:

	Unedited model	**Edited model**
Thinking on: temperature 0.6, top-p 0.95, top-k 20, presence penalty 1.5, at most 32,768 tokens	Condition 1	Condition 2
Thinking off: temperature 0.7, top-p 0.8, top-k 20, presence penalty 1.5, at most 8,192 tokens	Condition 3	Condition 4

Both configurations are the authors' [2] and match the model card's benchmark settings for each mode [3]. They are not the card's recommended settings for ordinary text generation. For non-thinking text the card recommends temperature 1.0, top-p 1.0 and presence penalty 2.0, and it suggests output limits of 32,768 tokens, or 81,920 for difficult mathematics [3]. Throughout, "thinking off" means the model card's non-thinking benchmark configuration, not every possible non-thinking configuration.

The prompt and grader are the authors': the problem followed by "Please reason step by step, and put your final answer within \boxed{}," graded by extracting the last boxed answer and

comparing it with the reference using Math-Verify [7]. Before any run, the grader marked 400 of 400 of the authors' released samples exactly as they had. Generation used vLLM 0.33.0 [8] in bfloat16 on one NVIDIA A40, with sampling seed 0. vLLM's own top-k and top-p sampler replaced FlashInfer's, which needs a compiler the machine lacked; the sampling settings are unchanged, and the registration states this.

Analysis. A problem's score is the mean of its four samples, and each comparison is paired by problem. Each comparison yields one of four readings fixed in advance, against a margin of 5 percentage points: (a) ahead by more than the margin, (b) equivalent within it, (c) behind by more than it, or (d) inconclusive. Readings (a) and (c) are one-sided paired t-tests against the margin at alpha .05. Reading (b) is the two one-sided tests procedure (TOST) for equivalence at the same alpha. Each reading is the same as the 90 percent t interval lying wholly above, inside or below the margin.

The primary comparison is condition 2 minus condition 3, the edited model in thinking mode against the unedited model with thinking off. The secondary is condition 4 minus condition 3, the edit's effect with thinking off. Reproduction succeeds if conditions 1 and 2 each lie within 5 points of the authors' 7.9 and 48.1.

The 5-point margin was set before the run on practical grounds, as the largest difference at which two configurations would count as interchangeable on this benchmark: one-eighth of the 40-point gain under examination. Separately, as a planning check, the authors' released per-problem results implied a 90 percent interval half-width near 2.8 points, so the planned sample was large enough to resolve a difference of that size. Section 3 reports how the readings change at other margins. Intervals described as bootstrap are 95 percent percentile intervals from 10,000 resamples of problems and are descriptive only.

Exploratory condition. After registering, we learned that the authors' thinking temperature of 0.6, though it is the model card's benchmark setting, is below the card's recommended 1.0 for thinking on text tasks [3], and that reasoning models loop more at low temperature [9]. Before any registered output was graded, we declared one exploratory condition, E1: the unedited model in thinking mode at temperature 1.0, otherwise identical to condition 1. The declaration was committed and posted publicly to the registration's OSF component on 7 November. E1 added 2,000 answers, for 10,000 in all. It is not part of the registered plan, and every comparison involving it is exploratory.

Post hoc analyses. Section 3 also reports analyses added in response to review, after all results were known: the interaction between edit and mode, the readings at other margins, accuracy split by whether an answer finished, a check for looping in answers cut at the limit, and the length of generations. They are descriptive and are labeled as post hoc where they appear.

3 Results

All four registered conditions and E1 completed, with no generation lost or excluded.

Condition	Accuracy	Answers cut at the limit	Mean tokens per answer
1. Unedited, thinking on	7.8%	92.9%	30,908
2. Edited, thinking on	51.2%	30.6%	21,671
3. Unedited, thinking off	43.4%	31.4%	3,702
4. Edited, thinking off	45.0%	31.1%	3,736
E1. Unedited, thinking on, temperature 1.0 (exploratory)	18.9%	80.8%	29,523

Differences reported below are computed from unrounded accuracies, so they can differ by 0.1 point from subtracting the rounded entries in this table.

Reproduction. Both thinking-on conditions reproduced the authors' results: 7.8 percent against their 7.9, and 51.2 against their 48.1. The share of answers cut at the limit also matched, 92.9 and 30.6 percent against their 93 and 33.

Primary comparison: reading (a). The edited thinking-on configuration led the unedited thinking-off configuration by 7.85 points, with a 90 percent interval of 5.79 to 9.91 (bootstrap 5.45 to 10.35). The whole interval lies above the 5-point margin, so the registered reading is (a). The reading depends on the margin (post hoc): it would also be (a) at a 2.5-point margin, inconclusive at 7.5 points, and equivalence at 10 points.

Secondary comparison: reading (b). With thinking off, the edited and unedited models were statistically equivalent within the 5-point margin: a difference of 1.60 points, with a 90 percent interval of minus 0.27 to 3.47. At a 2.5-point margin this comparison would be inconclusive (post hoc).

The edit's effect differs across configurations (post hoc). Paired by problem, the edit raised accuracy by 43.5 points in the tested thinking-on configuration and by 1.6 points in the tested non-thinking configuration. The difference between those two effects, an edit-by-configuration interaction, is 41.9 points, with a 95 percent interval of 37.9 to 45.9 (bootstrap 37.9 to 46.0).

How much of the gap thinking-off recovers. The unedited thinking-off configuration recovered 82 percent of the accuracy gap between the unedited and edited thinking-on configurations, (43.4 minus 7.8) divided by (51.2 minus 7.8), with a bootstrap interval of 77 to 87 percent.

Finishing and accuracy (post hoc). Across conditions, an answer cut at the limit was almost never correct, between 0.3 and 1.0 percent. Accuracy therefore tracked how often answers finished. In the unedited thinking-on configuration only 7.1 percent of answers finished, and every one of those was correct. The edited thinking-on and the unedited thinking-off configurations finished about the same share of answers, 69.3 and 68.7 percent. They differ in accuracy among finished answers, 73.7 against 63.1 percent. Completion rates alone therefore do not account for the difference in accuracy. The finished subsets need not contain the same problems, though, so the difference in accuracy among finished answers is descriptive and does not show that one configuration is more capable on comparable problems.

Condition	Finished	Cut at the limit	Correct among finished	Correct among cut
1. Unedited, thinking on	142	1,858	142	13
2. Edited, thinking on	1,387	613	1,022	3
3. Unedited, thinking off	1,373	627	866	2
4. Edited, thinking off	1,379	621	894	6
E1. Unedited, thinking on, temperature 1.0	385	1,615	371	8

Each condition has 2,000 answers. A cut answer is scored correct when a boxed answer it wrote before reaching the limit matches the reference.

What cut-off answers look like (post hoc). To check whether unfinished answers were loops, we applied the looping criterion of Pipis et al. [9]: an answer loops if some sequence of 30 tokens appears in it at least 20 times. Tokens were counted with the generating model's own tokenizer. In the unedited thinking-on configuration, 97.0 percent of answers cut at the limit met the criterion (1,803 of 1,858), against 4.9 percent of finished answers. Among cut-off answers from the edited thinking-on configuration, 73.9 percent met it; so did 27.6 percent of its finished answers, which repeat at length and still reach an answer. Among cut-off answers from the unedited thinking-off configuration, only 12.4 percent met it, which suggests that highly repetitive output was much less prevalent there than in the unedited thinking-on configuration. The criterion does not establish whether the remaining generations would eventually have produced correct answers. A separate check of our own, by compression ratio, gave similar shares: 98.6, 63.5 and 10.5 percent of cut-off answers in the same three configurations had a last 8,000 characters that compressed with zlib to under 15 percent of their size, a threshold fixed before any output was examined.

Shorter limits, retrospective (post hoc). Regrading each thinking-on answer at its first 4,096, 8,192 and 16,384 tokens, a prefix analysis of the same generations rather than new runs, gave 3.2, 5.7 and 7.3 percent for the unedited model and 7.2, 14.0 and 28.4 percent for the edited model.

Exploratory temperature condition. At temperature 1.0 the unedited model in thinking mode scored 18.9 percent, 11.2 points above the same model at 0.6 (90 percent interval 9.4 to 13.0). But 80.8 percent of its answers still ran to the limit. It recovered 26 percent of the gap (bootstrap 21 to 30 percent) and stayed 32.3 points below the edited model and 24.5 points below thinking off.

4 Discussion

The released weight edit produces a large improvement under the thinking-enabled evaluation configuration, consistent with the original authors' findings. An alternative inference configuration, however, approaches much of that benchmark performance without modifying the model's weights. With thinking off under the model card's benchmark configuration, the unedited model recovers 82 percent of the accuracy gap between the unedited and edited thinking-on configurations. The edited thinking-on configuration keeps a 7.85-point advantage over that alternative. With thinking off, the edit's effect is 1.60 points, within the preregistered equivalence margin. On MATH-500, the edit's measured benefit is concentrated in the tested thinking-enabled configuration, with no practically meaningful improvement detected under the tested non-thinking configuration.

The 7.85-point gap should not be read as the edit's own contribution. The two configurations differ in weights, mode, temperature, top-p and token limit at once, so the design does not partition

the gap among those factors. What the design does show is the interaction: the edit produces approximately 43 points of improvement in the tested thinking-on configuration and approximately 2 points in the tested non-thinking configuration.

The gap also has a cost. The edited thinking-on configuration generated 5.9 times as many tokens per answer as the unedited thinking-off configuration, 21,671 against 3,702, for its 7.85-point advantage. Token counts are not a direct measure of compute cost, but for a user choosing how to run this model, the accuracy-length tradeoff is the practical result.

Raising the thinking temperature to the model card's recommended value helps much less. That fits Pipis et al.'s finding that higher temperature reduces looping by promoting exploration without fixing its cause, so that "temperature is a stopgap rather than a holistic solution" [9].

For evaluation practice, the lesson is about baselines. A gain measured in a configuration the model's own documentation warns tends to loop should be reported alongside the model's default mode. The share of answers cut at the limit, which the authors did report, signals when this matters. Here, with nine in ten answers unfinished and the unfinished ones almost never correct, measured accuracy depended strongly on whether answers completed. The loop check supports the authors' account that, in unedited thinking mode, the unfinished answers were loops.

What this does not settle. Five limits apply.

First, as above, the primary comparison changes several factors at once and does not isolate the edit's contribution to the remaining gap.

Second, only the model card's non-thinking benchmark configuration was tested. Its recommended non-thinking text settings, a higher temperature and presence penalty with a longer output limit, were not. About 31 percent of non-thinking answers reached their 8,192-token limit, and few of those were highly repetitive, so a longer limit might raise non-thinking accuracy and narrow the gap. Whether those answers would have finished correctly is unknown, and a different configuration could also lower accuracy.

Third, the conclusions concern MATH-500 only. The authors also report large gains on GSM8K and instruction following [1, 2], and whether switching off thinking recovers those was not tested.

Fourth, the check covers one released edit on one model. It does not measure how reliably the method finds a useful edit.

Fifth, vLLM sampling is not bitwise deterministic, and our software versions differ from the authors'. The reproduction margins absorbed that difference here, but exact per-answer agreement with the authors' runs should not be expected. The loop check, though it uses a published criterion, was applied only after the results were known.

5 Deviations from the registration

None in the registered conditions, analysis or decision rules. The registered code was run unmodified, and its hash matches the registration. One exploratory condition was added after registration and declared publicly before any registered output was graded. The analyses labeled post hoc were added after review and are descriptive.

6 Data and code

The registration, frozen code and exploratory declaration are at `https://osf.io/w4g7u` and its component `https://osf.io/3r9vq`. The per-answer outputs of all five conditions, 10,000 answers with their full generated text, finish status and token counts, are released under CC0 in the same component as `wsp-check-outputs-v1.zip` (SHA-256 `14a7ddc7882661c53824f06fcf796434e1e7`

fbb9ebc222050f079e87a9d748d1), with the registered, exploratory and post hoc analysis outputs, the exploratory and post hoc scripts, a README and a SHA-256 manifest tying each output file to the hash recorded when its run finished. The full run took about 26 GPU-hours on one A40.

Acknowledgments

We thank Rikiya Takehi, Jayden Teoh, Alex Zhang and Omar Khattab for releasing their code, logs and edited checkpoint in full, which made an outside check possible within days. A discussion of their work on the *Prompt Engineering* YouTube channel raised the temperature point that became the exploratory condition. AI assistance was used to build the run scripts and draft this report, and two rounds of review by GPT-6 (OpenAI) improved its analysis and framing; GPT-6 is from the same model family as GPT-6 Astra, which made the edit examined here. An anonymous reviewer read the report before its release. Every number was checked against the analysis output.

References

[1] Takehi, R., Teoh, J., Zhang, A., and Khattab, O. *Sparks of Weight-Space Programming: LLMs Can Natively Read & Write Neural Network Weights*. MIT CSAIL, 2 November 2026. https://rikiyat.github.io/blog/weight-space-programming/

[2] Takehi, R., et al. *lm_understand-logs*, experiment logs and code, commit 4d09ba6d6a7e891dd8063974dfe81b3d78e3494f. https://github.com/RikiyaT/lm_understand-logs

[3] Qwen Team. *Qwen3.5-0.8B* model card, revision 2fc06364715b967f1860aea9cf38778875588b17. https://huggingface.co/Qwen/Qwen3.5-0.8B

[4] Cochran, T. O. 2026. "Outside Check of the Qwen3.5-0.8B Weight-Space Edit." Preregistration, OSF. https://doi.org/10.17605/OSF.IO/W4G7U

[5] Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. 2021. "Measuring Mathematical Problem Solving With the MATH Dataset." NeurIPS 2021. arXiv:2103.03874.

[6] Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. 2023. "Let's Verify Step by Step." arXiv:2305.20050.

[7] Hugging Face. *Math-Verify*. https://github.com/huggingface/Math-Verify

[8] Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. 2023. "Efficient Memory Management for Large Language Model Serving with PagedAttention." SOSP 2023. arXiv:2309.06180.

[9] Pipis, C., Garg, S., Kontonis, V., Shrivastava, V., Krishnamurthy, A., and Papailiopoulos, D. 2026. "Wait, Wait, Wait... Why Do Reasoning Models Loop?" In *Proceedings of the 43rd International Conference on Machine Learning*, PMLR 306:98934–98964. First posted as arXiv:2512.12895, December 2025.