Public edition

THOX.ai

Weight-Space Programming for THOX Edge Models: Evidence Review, Small-Scale Recreation, and a Signed Patch Channel

What a frontier LLM reading and writing weights actually establishes, what THOX reproduced and verified itself, and the signed patch channel that makes any of it shippable.

Craig Ross and Tommy Xaypanya

Chief Executive Officer · CTO & Co-Founder (corresponding author)

ABSTRACT

Takehi et al. report that a frontier LLM can read and write neural network weights directly, without gradient descent, and that editing 0.016% of Qwen3.5-0.8B's weights raised its MATH-500 score from 7.9% to 48.1% by removing a thinking-loop failure. THOX ships a derivative of that exact base model, so this report reviews the claim, recreates its three verbs at checkable scale, re-verifies the released edit and specifies the control the question turns on. The headline edit is genuine and narrower than its framing: it removes an obstacle rather than adding ability. The decisive control, the same base model with thinking turned off, is specified at full MATH-500 scale with a pre-registered decision rule; its run was stopped by compute billing before completion, so this edition states no number for it. The released edit rebuilds bit-exactly (130,188 entries, both hashes), but a factor form estimated from it is not exact without a sparse residual, which sets the size of an exact patch at tens of kilobytes. Small recreations show task identity is legible only in composed per-class circuits, that a hand-written network matches an oracle-trained one, and that 16 of 747 weights repair a stuck policy. THOX adopts the decoding defences and the signed patch channel now, and holds weight patching behind a pre-registered bar: beat the best decoding-only control by more than two standard errors.

CONTENTS

1 Introduction
2 The source result and its evidence state
3 Definitions and method
4 THOX recreation experiments
5 The decoding control: protocol and decision rule
6 The THOX patch channel
7 Application to THOX device models
8 The gated replication program
9 Limitations
10 Conclusion
11 Provenance and document control
R References

Craig Ross and Tommy Xaypanya, THOX.ai. Public edition v1.1.0, 2026-11-04; supersedes v1.0.1 (2026-11-04). Published on Zenodo: DOI 10.5281/zenodo.23955394 (concept DOI 10.5281/zenodo.23954378). License CC BY 4.0. Your AI. Your Data. Your Rules.™

Executive summary

130,188
entries changed in the released Qwen3.5-0.8B edit, rebuilt bit-exactly and re-verified on a second host (E0-R)
MEASURED

100% vs 33.3%
task identified from weights alone by the per-class composed reader, against three marginal readers at chance (E1, 30 networks)
MEASURED

306 / 0
MATH-500 problems where the released edit scores higher / lower than the base, in the authors' own per-problem files (A0)
CALCULATED

80.6 KB
exact patch for the Experiment 8 edit (factor plus residual), against 387.6 KB for the released compressed diff (P1)
MEASURED

1 Introduction

On 2 November 2026, Takehi, Teoh, Zhang and Khattab published a blog post [1], not a peer-reviewed paper, reporting that a frontier language model (OpenAI's GPT-6 Astra [2]) can read and write neural network weights directly, in natural language, without gradient descent. They name the practice **Weight-Space Programming** (WSP). The post's claims span three verbs. Read: given only the 17,253 weights of a small policy network as text, Astra named the game it was trained on, four times out of four. Write: given game rules and a zero-initialised network, it wrote policies that beat PPO trained for 100 million steps. Fix: given a Python kernel over the weights of Qwen3.5-0.8B and a budget of batched forward passes, it diagnosed why the model loops in thinking mode and edited 131,072 of roughly 752 million weights, raising MATH-500 from 7.9% to 48.1% under the authors' sampling profile.

THOX builds small language models for its own devices, and one of them, ThoxEdge-0.8B, is a merged fine-tune of the exact base model in the fix experiment, Qwen/Qwen3.5-0.8B [3][4]. A method that repairs a deployed model's dominant failure by changing 0.016% of its weights, and that could ship as kilobytes instead of a new 529 MB GGUF, is directly relevant to a device fleet that updates over a household mesh [5]. It is also exactly the kind of claim THOX does not repeat on the strength of a blog post.

Version 1.0.0 of this report [6] reviewed the evidence, recreated the three verbs at checkable scale, verified the released edit and built a signed patch channel. It left the decisive control unmeasured: how much of the gain a decoding change recovers with no weight change at all. This edition specifies that measurement at full MATH-500 scale, for the base model, the released edit and THOX's own deployed model in its served configuration, with a pre-registered decision rule; the run did not complete because compute billing stopped it. The contributions, in order of evidentiary weight:

1. **An evidence review with every citation re-verified** against its live primary page, extended with the source authors' own released evaluation files, which report results the post does not discuss (Section 2).

2. **Four small-scale recreation experiments** (E1 to E4) carried unchanged from v1.0.0: task identity is legible only in composed per-class circuits (100% against chance), a hand-written network matches an oracle-trained one, 16 of 747 weights repair a stuck policy, and a weights-only audit returned a confident ranking for a behaviour that never formed (Section 4).
3. **An independent re-verification of the released edit (E0-R) and a measurement of its patch forms (P1)**: 130,188 entries, both hashes, a bit-identical rebuild, and the finding that a factor form estimated from the released diff reproduces only 90.8% of the changed entries bit-exactly; an exact patch needs a 12,028-entry residual and totals 80.6 KB (Section 4.5).
4. **The decoding control (L1), specified and built but not yet measured**: a fixed protocol on the full MATH-500, a tested job script, decoding-only defences applied to the same generations, and an analysis pipeline; the run was stopped for compute billing before completion, so this edition states no Section 5 number (Section 5).
5. **A corrected patch-channel specification**: the patch-size table now agrees with the wire format, and exactness is enforced by result hashes plus an optional sparse residual (Section 6).

> PRODUCT IMPLICATIONS
>
> Ship decoding-side loop defences and measured serving configurations first; they are free, reversible, and the only available measurement of the decisive comparison (third-party, one sample per problem) says they recover most of the benchmark gain. Keep the signed patch channel, which serves LoRA and distillation deltas today. Treat LLM weight editing as a gated research program whose first bar is beating those controls. Do not describe any THOX model as created or improved by Weight-Space Programming until that bar is cleared.

2 The source result and its evidence state

2.1 What was shown, experiment by experiment

Table 1 summarises the eight experiments as reported [1][7]. Two structural facts matter for everything that follows. The raw-weights, no-tools setting applies only to Experiments 1, 2 and 4, on networks of 17K to 172K weights. Every experiment involving a real language model (3, 6, 7, 8) gave the editor a Python sandbox and, except in Experiment 3, a budget of forward passes. The popular summary of the post, an LLM reading raw weights and fixing a real model by inspection, describes no single experiment in it [8].

Exp.	Verb	Setting	Headline result (author-reported)	Independent status
1	Read	17,253 weights as text, no tools	Game named correctly 4/4, open answer	Partially reproduced with other models (Qwen 3.8 27B: 3 games; Kimi K3.1 Pro: 1) [9]
2	Read	23,824 weights, no tools, no forward pass	Diffusion model's shape named 3/3 (drawn as outline)	Answers re-scored consistent; a Sobel template tool also scores 3/3 [10]
3	Read	Middle blocks only of 7.16M char LMs, Python, no execution	Language 6/6 attempts; every copying head named, no false positives	Signed K-composition tool recovers 2/3 exactly; unsigned textbook score fails [10]
4	Fix	171,664 weights, no tools, one reply	Stuck car policy: 0% to 100% over 1,024 episodes; 3.54% of weights changed	Rebuilt network bit-identical; success rate not independently re-run
5	Write	Zero-initialised 20,486-weight policies, no tools, no rollouts	Beat PPO-100M on 3 MinAtar games; Breakout 517.4 vs PPO-500M 25.3	Re-scored within 1.4 SE [10]; cross-model chart: Claude Opus 5.5 about 20x PPO-500M on Breakout, as read by press coverage; raw data not public [11] [12]
6	Write	1.48M-weight Go network, Python sandbox, 0 to 50 forward passes	50 passes: beats equal-size PPO trained on 20M games	Replayed within 1.7 SE [10]
7	Read	Gemma 3 270M pair differing only in a sycophancy fine-tune	Weights alone failed 4/4; with attention patterns 3/3 pairs	Re-scored: weights-only answers at chance [10]
8	Fix	Qwen3.5-0.8B, Python kernel, 10/25/50 forward passes, 158 turns, 20.3 h	MATH-500 7.9% to 48.1%; cap hits 93.0% to 33.1%	Edit identity reproduced (THOX E0 and E0-R); MATH-500 re-run once in full; mechanism consistent

Table 1. The eight Sparks experiments. Results are author-reported unless the last column says otherwise.

2.2 What independent work has established

The Experiment 8 edit is the best-verified claim in the post. The released `apply_edit.py` rebuilds the edited checkpoint from the pinned base, and several parties, THOX included (Section 4.5), confirm that the two edited tensors hash to the published values [7][13]. The MATH-500 gain was re-run once in full on Hugging Face Jobs: 51.0% for the edited model against the authors' 48.1%, with the base model checked only on a 25-problem smoke test (5.0%) [13]. A commenter reports 10.9% and 61.7% on a 'math eval' without naming the benchmark or settings [14]; this paper does not count it as a MATH-500 result. Under a repeated 8-gram test, 183 of 200 base answers loop against 72 of 200 edited, and median loop onset moves from 2,725 to 7,856 words; Experiment 5 and 6 scores reproduce within about one standard error [10].

The edit's anatomy is also established. It touches 64 columns of the `down_proj` matrix in each of layers 15 and 19: 131,072 entries addressed, 130,188 of them different after bfloat16 rounding. Per tensor the diff is rank-1 (singular-value ratios 1,771 and 1,552), and the component along the diagnosed repetition direction is multiplied by about −5: flipped and amplified, not merely removed [10]. The relative change to the two matrices is 10.6% and 8.1% in Frobenius terms: a small count of edited entries is not a small edit.

2.3 What the authors' own released files add

The released logs include the authors' live results table and per-benchmark scores, which carry more than the post discusses [7]. Figure 1 plots them. The 50-pass edit lifts every benchmark the authors ran, not only the two in the headline: AMC23 from 0.5% to 23.4%, Minerva from 1.5% to 25.4%, OlympiadBench from 1.3% to 21.8%, GPQA Diamond from 22.7% to 36.6% and the MMLU-Pro subset from 40.3% to 45.5%. The pattern is the loop story's signature: the unedited model hits the 32,768-token cap on 99.5% of AMC23 answers and 98.7% of OlympiadBench answers, and the benchmarks with the most cap hits gain the most. The same table records two facts that bound the prompt-hint critique. A run with a shorter prompt that omitted the failure example (prompt v4) submitted no edit at 10 passes and reached the same 14.1% at 25 passes as the hinted prompt; and the four prompt-v5 runs are listed without results. Run-to-run variance is therefore real and only partly characterised.

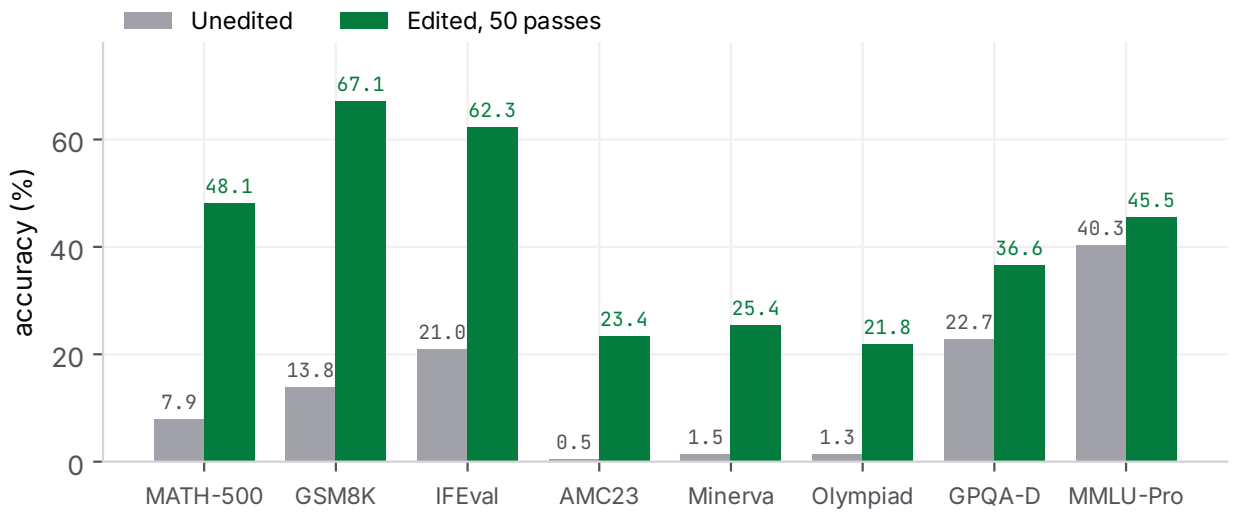


Figure 1. Experiment 8 as reported in the authors' released evaluation files (thinking on, 32,768-token cap; 4 samples per problem for MATH-500, Minerva, OlympiadBench and GPQA, 2 for GSM8K and IFEval, 16 for AMC23, 1 for MMLU-Pro) [7]. THOX did not run these benchmarks. AUTHOR-REPORTED

THOX re-analysed the released MATH-500 per-problem files (A0, arithmetic on released data). They reproduce the headline exactly (7.9% and 48.1% over 4 samples per problem), and the paired problem-level gain is +40.2 points with a 95% bootstrap interval of 36.8 to 43.5 points. The edit scores higher than the base on 306 problems and lower on 0; the base never solves 423 of the 500 in four samples, and the edit solves 243 of those at least once. A change that helps on hundreds of problems and hurts on none is what removing a uniform obstacle looks like, and it is also exactly the signature a decoding change that avoids the same obstacle would produce, which is why Section 2.5 matters.

2.4 What is contested

Three critiques bound the claim. **Framing.** The no-tools, raw-floats setting covers only the small-network experiments; the Qwen result shows a frontier model can be prompted to devise and carry out a task a human ML engineer could do [8]. **Novelty.** The Experiment 8 procedure is the established

steering and abliteration playbook: contrast failing and healthy activations, estimate a direction, rank the weights that write along it, subtract, test [15][16]. Goedecke also shows that a three-line norm heatmap makes most of Experiment 1 visually obvious because the input channels are labelled [15]. **Cost.** The editor spent 158 turns and 20.3 hours on Experiment 8; a scripted pipeline with no LLM re-derived a similar edit in about five CPU minutes and sixteen forward passes, reaching roughly 75% of the loop suppression [10]. None of this makes the result false. It makes the frontier LLM one component, a hypothesis proposer, inside a harness whose measuring and editing are ordinary engineering.

2.5 The two controls that decide the THOX question

Two comparisons matter more for THOX than any other number in this debate. The **Gemma control**: the same procedure applied to Gemma 3 270M, a model that rarely loops (8.5% cap hits), moved MATH-500 from 6.8% to at most 7.7%, which the authors themselves report as not established [7]. Editing removed an obstacle that hid an ability; it could not add an ability the model lacked. The **decoding control**: the OpenWSP repository reports the unedited model at 46.8% on MATH-500 with thinking disabled (0.6% cap hits) against 50.6% for the released edit, with one sample per problem [17]. When v1.0.0 was written the table was not visible; at the pinned commit it is in the README and the status files, but it remains a third-party number under its own settings. Section 5 specifies the THOX measurement that replaces it.

2.6 Claim tiers used in this paper

Tier	Meaning	Examples here
Author-reported	Stated by Takehi et al. or in their released files; not independently re-run	Exp 4 success rate; Exp 8 GSM8K, IFEval and the other benchmarks in Figure 1
Independently reproduced	Re-run or rebuilt by a party other than the authors, THOX included	Exp 8 edit identity (E0, E0-R); Exp 5 and 6 scores; Exp 8 MATH-500 for the edit (L1)
Contested or third-party	Disputed framing, or a number THOX did not produce	OpenWSP decoding table; cross-model Breakout chart (read from an image by press coverage)
THOX-measured	Produced by Sections 4 and 5 on named THOX-controlled hosts	E1 to E4, E0-R, P1, L1, patch-channel tests

Table 2. Evidence tiers. The tier travels with the number wherever this paper repeats it.

3 Definitions and method

Weight-Space Programming (WSP) is the source post's term for a general-purpose LLM reading and writing a network's weights directly, not through an optimizer and in particular not with gradient descent, around a natural-language specification [1]. **Read, fix, write** are its three task classes: infer behaviour from weights (Experiments 1 to 3, 7), edit a failing network (4, 8), create weights from zero (5, 6). A **forward pass**, in the source harness, is a budgeted batch of executions (up to 512 generations in Experiment 8), not a single example. A **rank-1 patch** is a weight edit of the Experiment 8 shape: for a tensor $W \in \mathbb{R}^{d \times n}$, a unit direction $u \in \mathbb{R}^{d}$, a column set C and a scalar α,

$$W'_{:,j} = \begin{cases} W_{:,j} - \alpha\, u \left(u^\top W_{:,j}\right) & j \in C \\ W_{:,j} & j \notin C \end{cases} \quad (1)$$

For Experiment 8, $d = 1{,}024$, $|C| = 64$ per tensor and $\alpha = 6$ as stated in the editor's report; E0-R estimates α from the released diff as 6.0003 and 6.0002. A **loop**, throughout this paper, is detected causally on whitespace words $w_0, w_1, \ldots$ with 8-grams $g_i = (w_i, \ldots, w_{i+7})$:

$$\text{loop} \iff \exists\, t : \sum_{i=t}^{t+299} 1\left[g_i \in \{g_0, \ldots, g_{i-1}\}\right] \geq 270 \quad (2)$$

which is the test of the independent CPU re-check [10]. The first qualifying window's start t is the **loop onset**; because the test is causal, it is the point a streaming detector would rewind to. The source authors' own end-of-text period test is recorded alongside it.

Evidence classes follow the THOX research paper standard: MEASURED (run on a named host), CALCULATED (arithmetic from stated inputs), SIMULATED (run inside a simulated environment). Nothing measured on a general host is stated as a device result. Method discipline: every headline number is produced by a script in the artifact bundle and read from an evidence file (Section 11); failed iterations are preserved and reported; each experiment carries at least one control.

Lineage: where WSP sits

WSP is best understood as an untrained entrant into four established lines of work. **Trained readers** have read properties and accuracy from weights using networks trained for the job on thousands of checkpoints [18][19]; permutation-equivariant architectures make such readers respect the symmetries of weight space [20][21], and the weight-space learning survey now taxonomises the field [22]. Spectral work finds shared low-dimensional subspaces across large model populations, a candidate explanation for why directional edits work at all, though its interpretation is contested [23]. **Trained writers** generate weights from networks trained for the job, from HyperNetworks [24] through checkpoint-generating diffusion [25] to hypernetworks that write a LoRA adapter from a task description in one forward pass [26][27]. **Algorithmic editors** write weights by formula: ROME and MEMIT locate factual recall in mid-layer MLPs and rewrite it with rank-one updates [28][29], though where an effect localises does not reliably predict where editing works [30]; representation engineering and the refusal-direction result established that behaviours can live in single residual-stream directions that weight edits remove [31][16]. Experiment 8's edit is a member of this family whose target and formula were chosen by a reasoning model. **Weight arithmetic** supplies the delta object everything operates on: low-rank deltas [32], model soups [33], task vectors [34], and merging methods that exist because naive arithmetic interferes [35][36]. Underneath the reading claims sits mechanistic interpretability, the QK and OV circuit account that Experiment 3's editor independently reinvented [37][38], and Tracr's proof that programs compile into transformer weights, the precedent for Experiments 5 and 6 [39]. The paradigm claim extends LLM-as-optimizer work from prompts

and programs to weights [40]. On the decoding side, the controls of Section 5 descend from the degeneration literature [41] and from thinking-budget interventions [42][43].

4 THOX recreation experiments

Sections 4.1 to 4.4 are carried unchanged from v1.0.0, where they were run on the THOX build VM (CPU) on 2026-11-03 with NumPy and PyTorch [6]. Their numbers enter this edition through `carried_v1_0_0.json`, which records each value with the sentence it appears in and fails the build if the sentence is absent from the v1.0.0 text. The reader, writer and editor in these experiments were deterministic tools, not a frontier API model: what is tested is whether the weight-space structure the source exploits exists and is usable by THOX tooling.

4.1 E1, Read: the task signature lives in the composed per-class circuit

Thirty multilayer perceptrons (64-48-4) were trained on three tasks over 8×8 one-dot grids: name the dot's column, its row, or its diagonal class; 10 seeds per task, every network at 100% task accuracy. Four weights-only readers then attempted identification with no forward passes. R1 aggregates $|W_1|$ into one input map; R2 lets each hidden unit vote; R3 composes the layers once into $E = W_2^\top W_1^\top$ summed over classes; R4 composes per output class and lets each class map vote. R1, R2 and R3 score exactly chance (33.3%) because they misread every column and row network as diagonal; R4 scores 100%, with its permutation control at chance (Figure 2). What a small network computes is legible in its composed per-class circuits and largely illegible in its layer-1 marginals.

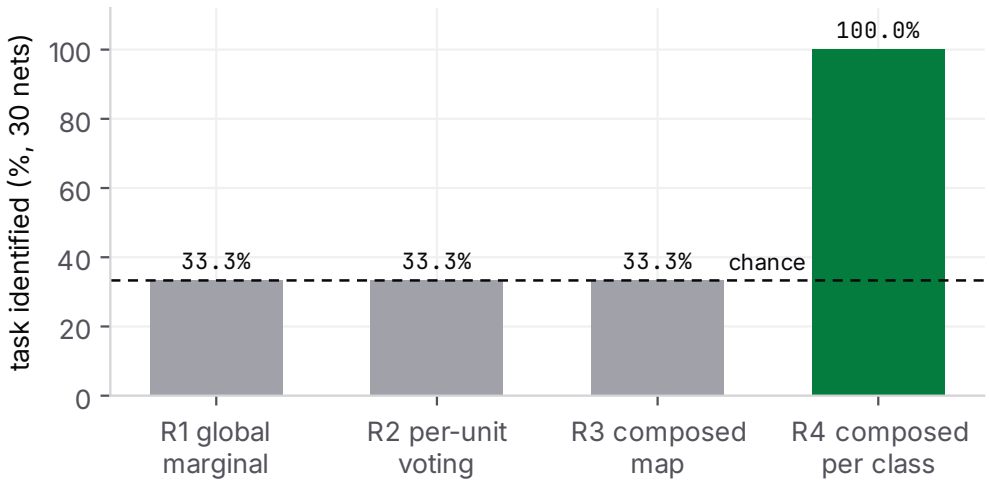


Figure 2. E1: four weights-only readers on 30 trained networks; only the per-class composed reader identifies the task. Carried from v1.0.0. `MEASURED, THOX BUILD VM`

4.2 E2, Write: a hand-written network matches an oracle-trained one

Experiment 5's claim reduces to a compilation claim: if the solution is a short program, weights can encode it directly. On a control task with a known program, predicting where a ball bouncing in a unit box lands, the written network implements the reflection construction by hand, with zero gradient steps; its landing predictions are exact to float64 precision (maximum error 2.2e-16). As a paddle

controller it catches 100% of balls in the standard regime and 92.1% in a hard regime. Twenty random-weight networks average 4.1% (best 20.2%); a gradient-trained oracle given 1,200 steps on 4,096 labelled examples reaches the identical catch profile with a small landing error (Figure 3). Both controllers fail on the same balls: they are paddle-limited, not prediction-limited.

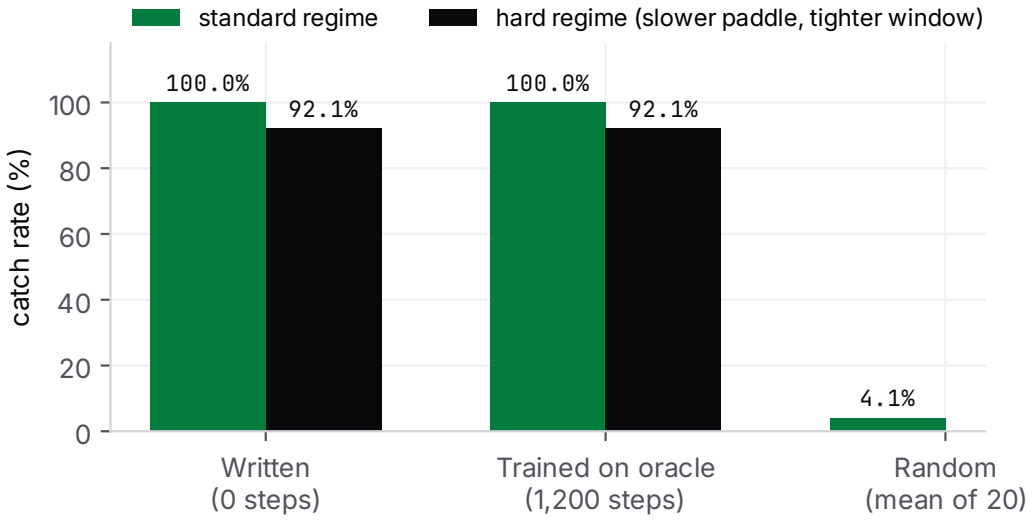


Figure 3. E2: written, oracle-trained and random networks on the ball-landing control task. Carried from v1.0.0.
`MEASURED, TASK SIMULATION`

4.3 E3, Fix: sixteen weights repair a policy stuck at zero

A 747-parameter MountainCar policy (2-24-24-3) imitation-trained on an always-push-right expert succeeds 0% of the time: it never builds energy by rocking. The designed fix repurposes two little-used first-layer units as velocity detectors, adds an energy-pumping override pathway and gives the output enough gain to win when the car is slow. It touches 16 of 747 weights (2.14%) and raises success to 92.8% over 512 episodes; a random edit of the same size scores 0% at every gain, and sweeping the override gain gives a clean dose-response (Figure 4). A first attempt that changed 84 weights left success at 0% and is preserved in the findings log. The transferable claim is the shape (diagnose a single locus, add a pathway, verify against a same-size random control), not the fraction.

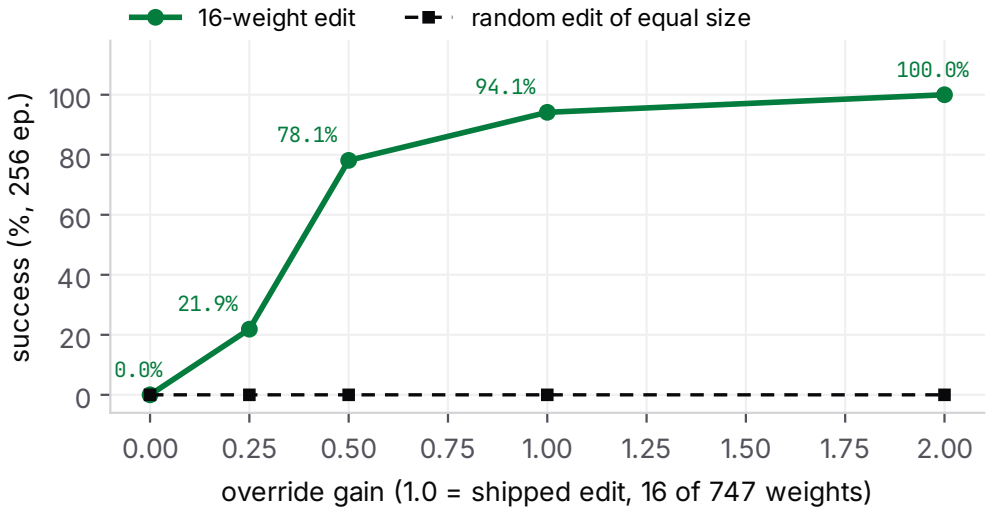


Figure 4. E3: MountainCar success against override gain for the 16-weight edit, 256 episodes per point. Carried from v1.0.0.
SIMULATED ENVIRONMENT, THOX BUILD VM

4.4 E4, Audit: a negative result, reported in full

The Experiment 3 audit shape was attempted on a small transformer (2 layers, 4 heads, d=64, 70,656 parameters) trained on sequences with a planted repeated bigram. The weights-only scores ranked a candidate induction head (composition score 0.589 against a 0.05 to 0.39 spread). The ground truth failed: after 1,400 steps the loss on repeat positions was 3.19 against 3.18 elsewhere, the ln(24) floor of an unpredictable token; attention at repeat positions was uniform across heads; ablating any single head moved repeat loss by at most 0.022. No induction behaviour formed, so there was no head to find, and the weights-only ranking (Spearman 0.80 against a noise floor) is evidence of nothing. A weight-only audit tool always produces a ranking; whether the ranking means anything depends on the behaviour existing, which can only be established behaviourally.

4.5 E0-R and P1: the released edit, re-verified, and what it takes to ship it

v1.0.0 verified the released Experiment 8 edit on the THOX build VM (E0). This edition repeats the check independently on a second host with a different method (E0-R): HTTP range reads of only the four tensors involved, SHA-256 against the authors' edit manifest, and a rebuild from the authors' sparse diff. All checks pass. The base tensors hash to the manifest's base values; the released tensors hash to `82e2d733ea64...` and `8ef3c61bc36a...`; rebuilding from `edit.npz` is bit-identical; 65,260 and 64,928 entries differ, 130,188 in total; singular-value ratios are 1,771 and 1,552, with 99.9995% and 99.9991% of the diff energy in the first direction.

P1 then asks how this edit would travel through the THOX patch channel (Table 3, Figure 5). The authors released it in sparse form: 130,188 (index, value) pairs, 387.6 KB as compressed `edit.npz`. The factor form of Equation (1) is smaller by two orders of magnitude, but a factor estimated from the released diff is not exact: in every precision tried, applying it reproduces 91.0% and 90.5% of the changed entries bit-for-bit, and the remaining 12,028 each differ by one bfloat16 rounding step (maximum absolute error 4.88e-04, 0.33% of the largest diff entry). Rounding the direction to bfloat16 roughly doubles the mismatches. An exact patch therefore ships the factor plus a sparse residual:

80.6 KB for both tensors, 10x below the raw sparse diff, 4.8x below the released compressed file and 182x below the two full tensors. When THOX produces an edit itself, the editor's own direction is available at edit time and the residual is empty by construction; the result hashes make the distinction checkable either way.

Representation	Contents	Bytes (both tensors)	Exact	Class
Full edited tensors	Two 1,024 × 3,584 bfloat16 tensors	14.68 MB	yes	Calculated
Sparse diff, raw	130,188 × (4-byte index, 2-byte value)	781.1 KB	yes	Calculated
Released edit.npz	Authors' compressed sparse form	387.6 KB	yes	Measured (file)
Factor + residual	u float32, 64 int16 columns, alpha; 12,028 residual entries	80.6 KB	yes	Measured (E0-R)
Factor only, float32 u	u float32, 64 int16 columns, alpha	8.5 KB	no: 9.2% of changed entries off by one step	Measured (E0-R)
Full model re-issue	ThoxEdge-0.8B Q4_K_M GGUF	529.3 MB	yes	Model card [4]

Table 3. Patch representations for the Experiment 8 edit. Exactness is checked against the released tensors' SHA-256.

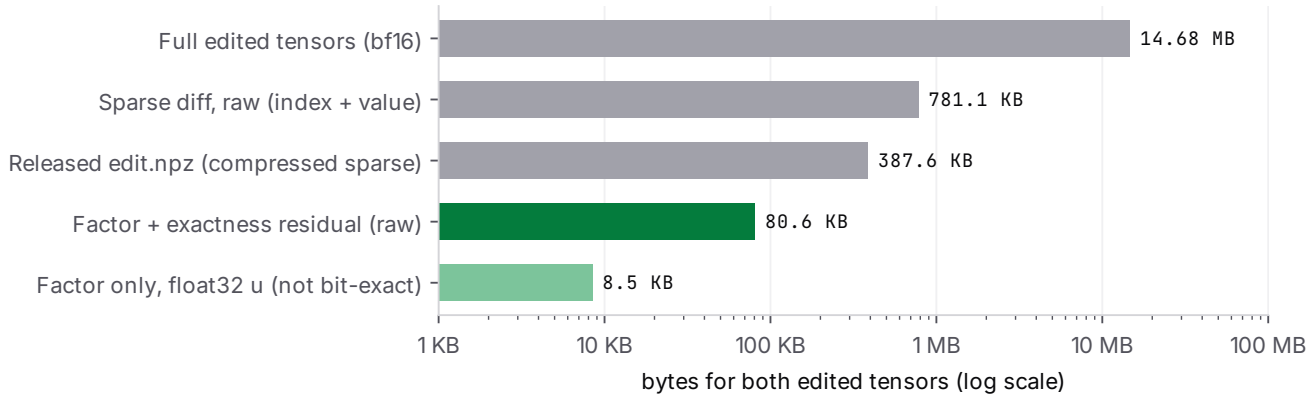


Figure 5. P1: bytes needed to move the Experiment 8 edit, log scale. The light green bar is not bit-exact and would fail the loader's result-hash check. MEASURED, CLOUD WORKSPACE

5 The decoding control: protocol and decision rule

MEASUREMENT STATUS

The four L1 jobs were launched on Hugging Face Jobs on 2026-11-04 and were stopped at about 15:20 UTC, minutes into generation, before any result was saved. A relaunch was then refused with HTTP 402 (Payment Required) for every job, GPU or CPU, under both the user and the organisation namespaces, which is consistent with exhausted compute credit rather than a fault in the jobs. This edition therefore states no Section 5 measurement. The protocol below is fixed, the job script is versioned in the artifact bundle and in the private results dataset [44], and the decision rule in Section 5.2 is pre-registered, so the next run can only confirm or refute it.

Section 2.5 reduced the THOX question to one comparison: against a serving configuration that already avoids the loop, what is left for a weight patch to fix? L1 is built to measure it on infrastructure THOX controls, at full benchmark scale, with the source's own grading.

5.1 Protocol

Configuration	Sampling; max new tokens	Samples per problem	Generations
Qwen3.5-0.8B, thinking on	T 0.6, top-p 0.95, top-k 20, presence 1.5; 32,768	1	500 planned
Released edit, thinking on	same as above	1	500 planned
Qwen3.5-0.8B, thinking off	T 0.7, top-p 0.8, top-k 20, presence 1.5; 8,192	4	2,000 planned
Released edit, thinking off	same as above	4	2,000 planned
Qwen3.5-0.8B, thinking off, greedy	greedy; 8,192	1	500 planned
Released edit, thinking off, greedy	greedy; 8,192	1	500 planned
ThoxEdge-0.8B, greedy, no system prompt	greedy; 8,192	1	500 planned
ThoxEdge-0.8B, served configuration	greedy, grounding prompt; 2,048 (4,096 context)	1	500 planned
ThoxEdge-0.8B, thinking off, card sampling	T 0.7, top-p 0.8, top-k 20, presence 1.5; 8,192	4	2,000 planned
thox-micro-125m-instruct, greedy	greedy; 512 (1,024 context)	1	500 planned

Table 4. L1 configurations on MATH-500 (500 problems, HuggingFaceH4/MATH-500 at revision 6e4ed1a). Thinking-on rows use the source's Experiment 8 profile; thinking-off rows use the Qwen3.5 card's non-thinking setting as encoded in the source's evaluation script; greedy rows match ThoxEdge's served Modelfile.

Prompts, answer extraction and checking follow the source exactly [7]: the problem plus 'Please reason step by step, and put your final answer within \boxed{}.', the last boxed answer after the last closing think tag, and math-verify 0.9.0 equivalence [45]. MATH-500 is the Lightman et al. subset of MATH [46][47]. Generation runs with vLLM 0.33.0 [48] on Hugging Face Jobs (NVIDIA H200 for the thinking-on runs, A100 80 GB for the rest), bfloat16, seed 0. Accuracy is the mean over samples, with the standard error over problems; paired differences use 10,000 bootstrap resamples over problems. Two decoding-only defences are applied to the thinking-on generations themselves, which is valid because generation is causal: every prefix is exactly what the defended stream would have produced. The **budget** defence truncates thinking at 4,096 or 8,192 tokens and appends the Qwen3 stop-thinking instruction [42]; the **interrupt** defence rewinds to the loop onset of Equation (2) and appends the same instruction. Only the forced answer (at most 4,096 tokens) is generated anew.

Before launch, the non-GPU parts of the pipeline were checked on the build host: the loop metric against nine unit tests (onset correctness, causality under truncation, threshold boundary, argument validation), the vendored grading against known answers, prompt construction against all four tokenizers, the token-index mapping used by the interrupt defence, and the analysis script against synthetic data in the output format. The GPU stage itself has not completed.

5.2 Pre-registered decision rule

Let Δ be the paired problem-level MATH-500 difference between the released edit with thinking on and the unedited base with thinking off, with its 95% bootstrap interval $[\Delta_{lo}, \Delta_{hi}]$. The outcome is read as follows, before any data exist:

- **Edit adds value**: $\Delta_{lo} > +2$ points. A loop patch has a real gap to aim at; the Section 8 bar stands as written.
- **Equivalent**: the whole interval lies within ±2 points. The headline gain is available with no weight change; no loop patch is justified by this benchmark.
- **Inconclusive**: any other interval. More samples per problem are run before anything is decided.
- **Decoding dominates**: $\Delta_{hi} < -2$ points. Turning thinking off beats the edit; a loop patch has nothing to add for a model served thinking-off.
- **ThoxEdge-0.8B**: if its served configuration (grounding prompt, greedy, thinking off) loops on fewer than 2% of MATH-500 problems, no ThoxEdge loop patch enters the program; above that, the loop rate is the baseline a patch must cut, with no retention loss.

Until the run completes, the only measurement of this comparison is third-party: 46.8% thinking-off against 50.6% for the released edit with one sample per problem [17]. At that sample size its interval would be several points wide, so under the rule above it does not settle the question, and this paper does not adopt a number it did not produce.

6 The THOX patch channel

6.1 Patch anatomy, corrected

v1.0.0 sized the Experiment 8 patch with a bfloat16 direction (about 4.4 KB) while specifying a float32 wire form, and called factor form three orders of magnitude below the sparse diff. Both are corrected here from the P1 measurement. The factor costs $4d + 2|C| + 8$ bytes per tensor (4.2 KB for $d = 1{,}024$, $|C| = 64$), or twice the direction's share as hex on the wire; it sits about 92x below the raw sparse diff, two orders of magnitude, not three. For an edit received as a sparse diff, exactness needs the residual of Section 4.5, giving

$$|P| \;=\; \sum_{\text{tensors}} \big(4d + 2|C| + 8 + 6r\big) \qquad (3)$$

bytes, with r residual entries at a 4-byte index and a 2-byte value. For Experiment 8 that is 80.6 KB. The manifest records the SHA-256 of every resulting tensor, and the loader verifies the hashes after applying and before switching; a patch that does not reproduce bit-exactly does not activate.

6.2 Manifest, signing, loader

The patch manifest (schema `thox-wsp/patch-manifest/v1`) is the signed contract between the editing environment and the device. It pins the base (repository, revision, per-tensor SHA-256), lists operations (rank-1 factor form, optional sparse residual), records the expected result hashes, the evaluation report hash, editor provenance (model identifier, effort, transcript hash) and the licence notices for base and patch. The release key signs it with Ed25519 [49] and lives outside the editing environment, so a compromised or over-eager editor cannot issue a patch a device will accept. One implementation lesson from v1.0.0 stands: Python and Rust disagree in the last digit of some float formatting, so the signed wire form carries numbers as hex of their little-endian bytes, never as JSON decimals. Sign bit patterns, never text representations of numbers.

Patch lifecycle (loader-enforced)

```
1 READ        fingerprint base: repo, revision, per-tensor sha256
2 DIAGNOSE    failure hypothesis + contrast sets; baseline rates recorded
3 EDIT        rank-1 ops (tensor, u, cols, alpha) + optional sparse residual + result hashes
4 VERIFY      deterministic rebuild; failure gate; decoding-control gate (Section 8);
              retention gate; safety gate; quantised-artifact gate; human diff review
5 SIGN        Ed25519 over the canonical manifest; key held outside the editor
6 DISTRIBUTE  staged rollout; device verifies signature + base hashes,
              applies to the inactive slot, checks result hashes, smoke-tests,
              switches atomically or rolls back
```

6.3 Verification status

Check	Result
Rust unit tests (bf16 round-trip, rank-1 apply against the Python test vector, SHA-256 known vector)	3/3 pass
End-to-end: Python toolkit fingerprints, builds and signs a manifest; Rust loader verifies, applies to slot B	Pass; applied tensor bit-identical to the Python reference
Rollback	Pass; active slot returns to A
Tampered manifest (alpha altered after signing)	Correctly rejected: signature verification failed
Wrong base tensor in slot A	Correctly aborted: base hash mismatch, no switch
Sparse-residual operation (new in v1.1.0 schema)	Specified; not yet implemented in the loader
Production patch issued through the channel	None

Table 5. Patch channel verification (thox-wsp v0.1.0), carried from v1.0.0 and run on the THOX build VM on 2026-11-03, plus the v1.1.0 schema addition.

The channel is delta-agnostic: a LoRA merge or a distillation delta is a base pin plus operations plus result hashes, so the signing, distribution and rollback machinery serves THOX's existing model pipeline even if no WSP patch is ever issued.

7 Application to THOX device models

Facts in this section come from the THOX model catalog and model cards [50][4]. ThoxEdge-0.8B is the centre of the WSP question for a simple reason: it is a merged QLoRA derivative of Qwen/Qwen3.5-0.8B, the Experiment 8 base, published after passing GATE49 at 46 of 49 on its development set, and it is served thinking-off at temperature 0 with a grounding system prompt. Its card records thinking-mode behaviour as out of contract and names its known residual defect: two prefill no-decline hard failures, which are not loop defects. Under the Gemma-control logic a loop patch for ThoxEdge-0.8B is expected to do approximately nothing, and proposing one anyway is the kind of claim this paper exists to prevent.

Surface	Role in the WSP program
ThoxWork™	Patch workbench: read, diagnose and edit sessions with a frontier editor; diff review, approval and signing. Editor transcripts stored with the patch.
MeshStack™	Patch registry and distribution: metadata, signatures, eligibility, staged rollout and per-device verification receipts. Kilobyte-scale patches fit mesh transfer; full models do not [5].
ThoxOS™ and ThoxMind	The Rust loader of Section 6: signature and base-hash verification, staged apply, result-hash check, on-device smoke suite, atomic A/B switch, rollback.
ThoxEdge-0.8B	First replication target because the base, the failure mode and a released reference edit all exist. Served thinking-off and greedy; its loop rate under that decoding (Section 5) is the bar a patch must beat.
thox-micro-125m	In-house target: THOX-trained, so a patch here would be the first fully THOX-owned WSP artifact if the gates pass. Its loop baseline is part of the Section 5 protocol.
ESP32-C6-class nodes	Whole written networks (the E2 shape), not patches: at this scale the network is the patch. Memory budgets must be checked per firmware build; no THOX measurement exists.

Table 6. Platform mapping. Every THOX-specific benefit in the right-hand column is a design proposal until the Section 8 gates pass.

Two existing THOX practices absorb WSP without new machinery. GATE49, the evaluation every promoted model passes, already runs Qwen3.5 models thinking-off as part of the measured configuration, so a patched model re-enters an existing gate. The catalog sync rule (every model change updates the Model Factory, catalogs, repositories, Ollama, Hugging Face and the website in one pass) extends to patches by treating the manifest as a catalog object with its own hash and status.

8 The gated replication program

THOX adopts WSP as a research program with pre-registered bars, not as a technique. The bars are THOX-defined targets. Compute planning follows the measured runs: full benchmarks belong on GPU jobs, and local machines carry editing and small-sample gates.

Step	Content	Success bar	Status
E0	Rebuild the released edit; verify hashes and rank	Bit-exact match	PASS (v1.0.0 E0; v1.1.0 E0-R)
E1'	Read: frontier editors (Claude Opus 5.5, GPT-6 Astra) and a local open model on the released policies and on new THOX-trained policies they cannot have seen	At or above 3/4 on new policies in 2 of 3 attempts, and better than the scripted reader of Section 4.1	Not run
E2'	Write: a frontier editor writes one THOX routing or control network from zero, scored against an equal-size trained network	Beat the trained baseline on the task's held-out regime	Not run (tool version: Section 4.2)
E4	Repair: decoding-only controls and the released edit at full scale; then a THOX-derived patch	A THOX patch beats the best decoding control by more than 2 standard errors with no retention regression beyond 1 standard error; otherwise the decoding control is the shipping answer	Protocol and jobs built; run stopped by compute billing (HTTP 402)
E5	Port: the winning procedure, if any, to thox-micro-125m and a 2B-class model; quantise to Q8_0 and Q4_0 and re-gate	Patch keeps at least 80% of its bf16 effect at Q4_0	Not run

Table 7. The gated program. Claims about THOX models stay internal until E4 and E5 pass.

The cheapest items ship first and pay off unconditionally: streaming loop detection with rewind and a hard thinking budget with a forced answer in the serving harness, the measured loop baselines extended to every device-target model, and the Section 6 channel. Weight patching is deliberately last, behind the controls it has to beat.

9 Limitations

- **Source maturity.** The source is a blog post, not peer-reviewed, with no arXiv version. Experiment 7's fine-tuned models are not released, some harness and grader modules are placeholders, earlier prompt trials were not published and the prompt-v5 runs are listed without results, so selection effects cannot be excluded.
- **Carried results.** E1 to E4 and the patch-channel tests are carried from v1.0.0 with phrase checks; their scripts and raw files live in the v1.0.0 artifact bundle on the THOX build VM and were not re-run for this edition. Merging that bundle into the repository is an open action.
- **Recreation scale.** Section 4 runs at 747 to 70,656 parameters, plus E0-R at 0.8B. It shows the exploited structure exists and is tool-usable; it does not show a frontier LLM can perform the source's feats, and no THOX frontier-editor run (E1', E2') has happened.
- **E4 is a negative result about one training run**, not about induction-head auditing in general.
- **No decoding-control measurement in this edition.** The L1 run was stopped by compute billing before completion, so the decisive comparison rests on a third-party number with one sample per problem [17] and on the pre-registered rule of Section 5.2. When L1 runs, its own scope limits apply: one benchmark, settings taken from the source and the card rather than tuned, bfloat16

weights through vLLM rather than the GGUF builds THOX devices run, and no retention or safety suite.

- **Small edit counts are not small edits.** The Experiment 8 diff changes its two matrices by 8% to 11% in Frobenius terms, and independent re-checking found the reasoning preceding a loop also loses probability under the edit [10]. Retention measurement is half of any patch claim.
- **Sampling-profile dependence.** The source compared base and edit at temperature 0.6, while the Qwen3.5 card recommends temperature 1.0 for thinking-mode text tasks [3]; L1 follows the source and does not test the card profile.
- **Safety surface.** The direction-ablation machinery that removes a loop direction removes refusal behaviour in the published literature [16]. A signed patch channel is also an attack surface: signatures, base pinning, result hashes and on-device smoke tests are load-bearing. Weight-only audits of hidden traits failed in the source's Experiment 7 and are not relied on for any THOX safety decision.
- **Quantisation is untested for patches.** Whether a rank-1 edit survives Q4 quantisation of the full model is unknown here; E5 exists because the answer decides whether patches ship to devices at all.
- **Pricing and availability.** Frontier-editor cost for an Experiment 8-scale run is unknown, and editor outputs are not reproducible bit-for-bit across model versions; provenance records belong in the manifest.

10 Conclusion

Weight-Space Programming, read carefully, is three results of different strengths wearing one name. Writing small networks whose solutions are short programs is real, reproducible and immediately useful to a device fleet. Reading weights is real where the read composes circuits and imaginary where it inspects marginals. Fixing a real model is real exactly once, on a failure with a single locus, verified bit-exactly by THOX twice. The cheaper explanation the whole question turns on, turning thinking off, is specified here with a pre-registered decision rule but was not measured in this edition. THOX's response is sized to that evidence: ship the decoding defences and measured serving configurations, keep the baselines, build the signed channel that every future delta will use, and let weight patching earn its place against the controls, in the open, at the bars in Section 8.

11 Provenance and document control

This edition was produced under the THOX Research Paper Process v1.0. Sources: the source post and its released logs, checkpoints and evaluation files; the v1.0.0 edition and its demo, whose measured numbers are carried with phrase checks; a reference list in which every external entry was reopened on its primary page on 2026-11-04; and new THOX experiments (E0-R, P1, L1). Every figure and every number in the text is generated from the files listed below by `wp/figures.py` and `wp/build.py`.

Field	Value
Version	v1.1.0 (public edition), supersedes v1.0.1 (2026-11-04); v1.0.1 superseded v1.0.0 (2026-11-03)
Date	2026-11-04
Author	Craig Ross (Chief Executive Officer) and Tommy Xaypanya (Chief Technology Officer & Co-Founder, corresponding author), THOX.ai
Status	Public edition. Published on Zenodo, DOI 10.5281/zenodo.23955394 (concept DOI 10.5281/zenodo.23954378), license CC BY 4.0, under the THOX Research Paper Process v1.0.
Evidence files	exp/results/carried_v1_0_0.json, e0r_patch_forms.json, l1_summary.json, source/exp8_*_merged.json
Raw L1 data	Job script in the private dataset Thox-ai/thox-wsp-evals [44]; no generations (run stopped by compute billing)
Compute	L1 (stopped, HTTP 402): Hugging Face Jobs, account tommytracx, labels project=thox-wsp, edition=v1-1-0; E0-R, P1 and A0: cloud workspace (Linux, 2 vCPU); E1 to E4: THOX build VM (v1.0.0)
Companion demo	weight-space-programming-demo.html (same bundle)

Table 8. Document control.

Area	Change from v1.0.0
Byline	Founder authors restored for the public edition: Craig Ross and Tommy Xaypanya. An intermediate internal draft of v1.1.0 carried the THOX.ai Research organisational byline only.
Section 5	Was a pending instrument check (n=4 per configuration; the scoped check was later completed on THOX hardware and is published as data supplement S1 alongside the v1.0.1 edition); now a full-scale protocol, a tested job script, an analysis pipeline and a pre-registered decision rule. The full run was stopped by compute billing before completion; no number is stated.
Section 2.3	New: the authors' released benchmark files and THOX's paired analysis of their per-problem MATH-500 records (A0).
Section 4.5	E0 re-verified independently (E0-R); patch forms measured (P1).
Section 6.1	Patch sizes corrected to match the wire form; 'three orders of magnitude' corrected to two; sparse residual added for exactness.
Section 2	Authors' released benchmark files and prompt-variant runs added; OpenWSP pinned and cited; Krentsel, Hugging Face re-run and alphaXiv sources cited; the alphaXiv figures no longer called MATH-500.
References	All entries re-verified. Corrected: source host (personal site, not a CSAIL blog), Honnibal title, Text-to-LoRA title, HyperNetworks and MEMIT venues, logs licences, rebuild credit. Added: NFN, DWSNets, Hase et al., Qwen3, s1, Holtzman et al., MATH, MATH-500, vLLM, Math-Verify, RFC 8032.
Figures	Regenerated from evidence files; titles moved to captions (no clipping); legends outside plot areas; numbered by first appearance.

Table 9. Change log.

R References

[1] Rikiya Takehi, Jayden Teoh, Alex Zhang, Omar Khattab. Sparks of Weight-Space Programming: LLMs can natively read & write neural network weights. Blog post, rikiyat.github.io (authors affiliated with MIT CSAIL), 2026-11-02. Note: Blog post, rikiyat.github.io; authors affiliated with MIT CSAIL, published 2026-11-02. https://rikiyat.github.io/blog/weight-space-programming/

[2] OpenAI. GPT-6 Astra: A new generation of intelligence. OpenAI announcement (openai.com/index), 2026. Note: title and date as published in OpenAI's RSS feed. https://openai.com/index/gpt-6-astra/

[3] Qwen Team. Qwen/Qwen3.5-0.8B model repository, revision 2fc06364715b967f1860aea9cf38778875588b17. Hugging Face, Apache-2.0, 2026. https://huggingface.co/Qwen/Qwen3.5-0.8B/tree/2fc06364715b967f1860aea9cf38778875588b17

[4] THOX.ai. ThoxEdge-0.8B. Hugging Face model repository (Thox-ai/ThoxEdge-0.8B), revision 235cce41bb3f1ea61719c37033d7eb8e309812e9, 2026. https://huggingface.co/Thox-ai/ThoxEdge-0.8B

[5] Craig Ross, Tommy Xaypanya. THOX MeshStack: Capacity-Priced Pipeline Sharding, Grant-Gated KV Transfer and Lease-Fenced Device Roles for a Private Household Mesh. THOX.ai Research technical report v1.0.1, Zenodo, 2026. DOI 10.5281/zenodo.23919580. https://doi.org/10.5281/zenodo.23919580

[6] THOX.ai Research. Weight-Space Programming for THOX Edge Models: Evidence Review, Small-Scale Recreation, and a Signed Patch Channel. Technical report v1.0.0, internal edition, 2026-11-03.

[7] RikiyaT. lm_understand-logs: prompts, replies, networks, edit scripts and evaluations for the Sparks experiments. GitHub repository, commit 4d09ba6d6a7e891dd8063974dfe81b3d78e3494f, 2026. Note: commit 4d09ba6; CC BY 4.0 (logs, data, networks), MIT (scripts), GPL-3.0 (MinAtar source quoted in the Experiment 5 prompts). https://github.com/RikiyaT/lm_understand-logs/tree/4d09ba6d6a7e891dd8063974dfe81b3d78e3494f

[8] Matthew Honnibal. The 'Astra can read and write weights to improve Qwen' post is misleading. Blog post, honnibal.dev, 2026. https://honnibal.dev/blog/astra-read-write-misleading

[9] Simon Willison. Qwen 3.8 27B on my laptop can interpret model weights, too. Simon Willison's Weblog (simonwillison.net), 2026. https://simonwillison.net/2026/Nov/3/qwen-can-interpret-model-weights/

[10] adiant211. wspacetest: an independent CPU re-check of the Sparks experiments. GitHub repository, 2026. Note: user repository, no description; cited at the state of 2026-11-03. https://github.com/adiant211/wspacetest

[11] Alex Krentsel. Post on X reporting a cross-model reproduction of the Sparks Experiment 5, 2026-11-03. Note: figures appear only in a chart image. https://x.com/AlexKrentsel/status/2117376683724660898

[12] OfficeChai Team. Ability For Weight-Space Programming Has Appeared Only In Latest Models, Says Google Researcher Alex Krentsel Who Reproduced Results. OfficeChai, 2026. Note: the article's reading of the chart in the X post. https://officechai.com/ai/ability-for-weight-space-programming-has-appeared-only-in-latest-models-says-google-researcher-alex-krentsel-who-reproduced-results/

[13] RikiyaT; checkpoint rebuilt and re-evaluated by akhaliq (AK) with ML Intern. RikiyaT/Qwen3.5-0.8B-weight-space-programming, revision d1671a178705bcab7644806d23c535517f35723b. Hugging Face, Apache-2.0, 2026. https://huggingface.co/RikiyaT/Qwen3.5-0.8B-weight-space-programming

[14] Joseph Noble. Comment on the alphaXiv discussion page for 'Sparks of Weight-Space Programming', 2026-11-03. Note: comment reports a 'math eval' without naming the benchmark. https://www.alphaxiv.org/abs/2611.weight-space-programming

[15] Sean Goedecke. LLMs are not superhuman at reading and writing neural network weights. Blog post, seangoedecke.com, 2026. https://www.seangoedecke.com/llms-are-not-superhuman-at-reading-and-writing-neural-network-weights/

[16] Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, et al. Refusal in Language Models Is Mediated by a Single Direction. arXiv preprint, 2024. arXiv:2406.11717. https://arxiv.org/abs/2406.11717

[17] OpenWSP contributors. OpenWSP. GitHub repository, commit 2265473e250b6f6f40a0e2c7853129902d0a11ea, 2026-11-04. Note: README, report.md and STATUS.md at the pinned commit; 1 sample per problem. https://github.com/openwsp/openwsp/tree/2265473e250b6f6f40a0e2c7853129902d0a11ea

[18] Gabriel Eilertsen, Daniel Jönsson, Timo Ropinski, Jonas Unger, Anders Ynnerman. Classifying the classifier: dissecting the weight space of neural networks. ECAI 2020. arXiv:2002.05688. https://arxiv.org/abs/2002.05688

[19] Thomas Unterthiner, Daniel Keysers, Sylvain Gelly, Olivier Bousquet, Ilya Tolstikhin. Predicting Neural Network Accuracy from Weights. arXiv preprint, 2020. arXiv:2002.11448. https://arxiv.org/abs/2002.11448

[20] Aviv Navon, Aviv Shamsian, Idan Achituve, Ethan Fetaya, Gal Chechik, Haggai Maron. Equivariant Architectures for Learning in Deep Weight Spaces. ICML 2023. arXiv:2301.12780. https://arxiv.org/abs/2301.12780

[21] Allan Zhou, Kaien Yang, Kaylee Burns, Adriano Cardace, Yiding Jiang, Samuel Sokota, et al. Permutation Equivariant Neural Functionals. NeurIPS 2023. arXiv:2302.14040. https://arxiv.org/abs/2302.14040

[22] Xiaolong Han, Zehong Wang, Bo Zhao, Binchi Zhang, Jundong Li, Damian Borth, et al. A Survey of Weight Space Learning: Understanding, Representation, and Generation. arXiv preprint, 2026. arXiv:2603.10090. https://arxiv.org/abs/2603.10090

[23] Prakhar Kaushik, Shravan Chaudhari, Ankit Vaidya, Rama Chellappa, Alan Yuille. The Universal Weight Subspace Hypothesis. arXiv preprint, 2025. arXiv:2512.05117. Note: v3 revised 2026-10-05; interpretation contested in secondary coverage. https://arxiv.org/abs/2512.05117

[24] David Ha, Andrew Dai, Quoc V. Le. HyperNetworks. arXiv preprint, 2016. arXiv:1609.09106. https://arxiv.org/abs/1609.09106

[25] William Peebles, Ilija Radosavovic, Tim Brooks, Alexei A. Efros, Jitendra Malik. Learning to Learn with Generative Models of Neural Network Checkpoints. arXiv preprint, 2022. arXiv:2209.12892. https://arxiv.org/abs/2209.12892

[26] Rujikorn Charakorn, Edoardo Cetin, Yujin Tang, Robert Tjarko Lange. Text-to-LoRA: Instant Transformer Adaption. ICML 2025. arXiv:2506.06105. https://arxiv.org/abs/2506.06105

[27] M. H. I. Abdalla, Zhipin Wang, Christian Frey, Steffen Eger, Josif Grabocka. Zhyper: Factorized Hypernetworks for Conditioned LLM Fine-Tuning. arXiv preprint, 2025. arXiv:2510.19733. https://arxiv.org/abs/2510.19733

[28] Kevin Meng, David Bau, Alex Andonian, Yonatan Belinkov. Locating and Editing Factual Associations in GPT. NeurIPS 2022. arXiv:2202.05262. https://arxiv.org/abs/2202.05262

[29] Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, David Bau. Mass-Editing Memory in a Transformer. arXiv preprint, 2022. arXiv:2210.07229. https://arxiv.org/abs/2210.07229

[30] Peter Hase, Mohit Bansal, Been Kim, Asma Ghandeharioun. Does Localization Inform Editing? Surprising Differences in Causality-Based Localization vs. Knowledge Editing in Language Models. NeurIPS 2023, Spotlight. arXiv:2301.04213. https://arxiv.org/abs/2301.04213

[31] Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, et al. Representation Engineering: A Top-Down Approach to AI Transparency. arXiv preprint, 2023. arXiv:2310.01405. https://arxiv.org/abs/2310.01405

[32] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, et al. LoRA: Low-Rank Adaptation of Large Language Models. arXiv preprint, 2021. arXiv:2106.09685. https://arxiv.org/abs/2106.09685

[33] Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. ICML 2022. arXiv:2203.05482. https://arxiv.org/abs/2203.05482

[34] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, et al. Editing Models with Task Arithmetic. ICLR 2023. arXiv:2212.04089. https://arxiv.org/abs/2212.04089

[35] Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, Mohit Bansal. TIES-Merging: Resolving Interference When Merging Models. NeurIPS 2023. arXiv:2306.01708. https://arxiv.org/abs/2306.01708

[36] Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, Yongbin Li. Language Models are Super Mario: Absorbing Abilities from Homologous Models as a Free Lunch. ICML 2024. arXiv:2311.03099. https://arxiv.org/abs/2311.03099

[37] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, et al. A Mathematical Framework for Transformer Circuits. Transformer Circuits Thread, 2021. https://transformer-circuits.pub/2021/framework/index.html

[38] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, et al. In-context Learning and Induction Heads. arXiv preprint, 2022. arXiv:2209.11895. https://arxiv.org/abs/2209.11895

[39] David Lindner, János Kramár, Sebastian Farquhar, Matthew Rahtz, Thomas McGrath, Vladimir Mikulik. Tracr: Compiled Transformers as a Laboratory for Interpretability. NeurIPS 2023, Spotlight. arXiv:2301.05062. https://arxiv.org/abs/2301.05062

[40] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, et al. Large Language Models as Optimizers. ICLR 2024. arXiv:2309.03409. https://arxiv.org/abs/2309.03409

[41] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, Yejin Choi. The Curious Case of Neural Text Degeneration. ICLR 2020. arXiv:1904.09751. https://arxiv.org/abs/1904.09751

[42] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, et al. Qwen3 Technical Report. arXiv preprint, 2025. arXiv:2505.09388. https://arxiv.org/abs/2505.09388

[43] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, et al. s1: Simple test-time scaling. arXiv preprint, 2025. arXiv:2501.19393. https://arxiv.org/abs/2501.19393

[44] THOX.ai Research. thox-wsp-evals: L1 generations, defence outputs and analysis for technical report v1.1.0. Private Hugging Face dataset Thox-ai/thox-wsp-evals, 2026-11-04. https://huggingface.co/datasets/Thox-ai/thox-wsp-evals

[45] Hugging Face. Math-Verify. GitHub repository huggingface/Math-Verify and PyPI package math-verify. Note: version 0.9.0 used for grading. https://github.com/huggingface/Math-Verify

[46] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, et al. Let's Verify Step by Step. arXiv preprint, 2023. arXiv:2305.20050. Note: source of the MATH-500 subset. https://arxiv.org/abs/2305.20050

[47] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, et al. Measuring Mathematical Problem Solving With the MATH Dataset. NeurIPS 2021. arXiv:2103.03874. https://arxiv.org/abs/2103.03874

[48] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, et al. Efficient Memory Management for Large Language Model Serving with PagedAttention. SOSP 2023. arXiv:2309.06180. https://arxiv.org/abs/2309.06180

[49] S. Josefsson, I. Liusvaara. Edwards-Curve Digital Signature Algorithm (EdDSA). RFC 8032, Internet Research Task Force (IRTF), Informational, 2017. https://www.rfc-editor.org/rfc/rfc8032

[50] THOX.ai. THOX device model catalog, live revision 2026-11-02.1 (94 models), and TXF-MP model cards. THOX.ai internal records, as cited in technical report v1.0.0.