

CAN AN LLM READ · REPAIR · WRITE THE WEIGHTS OF A NEURAL NETWORK?

Weight-Space Programming

Can a general-purpose LLM read, repair, and even write a neural network's weights - by reasoning alone? Eight experiments, two mechanisms, and one honest scoreboard.

READ FIX WRITE

■ PAPER-REPORTED ● INDEPENDENT REVIEW ◆ TEACHING SCHEMATIC ▲ SPECULATION

Het Patel (Hunterdii)

Independent educational explainer

Based on the blog post of 2 Nov 2026 by Rikiya Takehi, Jayden Teoh, Alex Zhang and Omar Khattab (MIT CSAIL)

Not affiliated with, endorsed by, or written by MIT CSAIL, OpenAI, or the original authors

Code MIT · text & figures CC BY 4.0 · designed and built by Het Patel

READ THE LIVE SITE ↗

weight-space-atlas.vercel.app

EVERY CLAIM IN THIS DOCUMENT CARRIES EXACTLY ONE EVIDENCE BADGE

ORIENTATION

00 How to read this guide

Every claim wears a badge. The badge tells you how much to trust it - and it never relies on colour alone.

PAPER-REPORTED
A number or claim **stated in the original post** (2 Nov 2026), or **recomputed here from the authors' released data** (the caption says so). Transcribed and cross-checked against the post.

INDEPENDENT REVIEW
From a **third-party analysis** - principally Andreas Maier's review of 5 Nov 2026 - or an independent replication.

TEACHING SCHEMATIC
Made-up toy data or arithmetic created by this guide to teach an idea. Always marked "illustrative".

SPECULATION
Predictions about the future, including the authors' own. Untested; discount accordingly.

Three reading paths

1. **The 15-minute path.** the story so far (p. 7), "If you read one page" (p. 4), the experiment gallery (p. 5), the scoreboard (p. 27) and the conclusion (p. 36).
2. **The practitioner path.** Sections 2–7 for experiments and mechanisms, Section 12 and Appendix D for runnable labs, Appendix E for the benchmark schema.
3. **The reviewer path.** Sections 8–12 in full, then Appendix C - the part of this document most likely to change first.

CREDITS AND NON-AFFILIATION

Independent educational explainer by Het Patel (Hunterdii). Designed, written and built by Het Patel. Based on the blog post *"Sparks of Weight-Space Programming: LLMs can natively read & write neural network weights"* by Rikiya Takehi, Jayden Teoh, Alex Zhang and Omar Khattab (MIT CSAIL), 2 Nov 2026. **All original research credit belongs to them.** This project is **not affiliated with, endorsed by, or written by** MIT CSAIL, OpenAI, or the original authors. Items labelled *Het's Lab* are independent additions, clearly marked. Licence: code MIT, text & figures CC BY 4.0. Cite the original post when citing results.

WHAT'S NEW IN THE ATLAS EDITION

A full redesign plus a fresh audit against the primary post. **Added:** a Prologue (the story so far, with the four real Exp. 1 weight fingerprints recomputed from the released data), the first week of replications and critiques, the mechanism behind every experiment, "explain it like I'm 12" boxes, and clickable references. **Corrected:** the Exp. 8 column description, the head-grid indexing, a misleading "six orders of magnitude" claim, and several reference author lists (details in Appendix C).

Six words you need

WEIGHT
one of the numbers inside a model - the "knobs" that set its behaviour.

PARAMETER COUNT
how many weights a model has; 0.8 B means 800 million.

FORWARD PASS
running the model once on some input to see what it does.

GRADIENT DESCENT
the standard way to train: nudge every weight a tiny bit, millions of times.

DIRECTION
a pattern spread across many weights or activations; removing one is a rank-1 edit.

BASELINE
a simpler method you must beat before claiming a new one is better.

THE MAP

Contents

Four territories, fifteen chapters, five appendices - from "what is a weight?" to "how would I design the next experiment?"

FRONT MATTER

00 How to read this guide 2
01 If you read one page 4
02 Abstract & the eight experiments at a glance 5

PROLOGUE · THE STORY SO FAR

P1 Where the idea came from 7
P2 The first experiment, in slow motion 8
P3 From one experiment to eight 10
P4 Week one: reactions & critiques 11

PART I · FOUNDATIONS

01 A network is a pile of numbers 13
02 Four ways to write weights 15
03 Cousins: who else touches the weights? 16

PART II · THE EXPERIMENTS

04 READ - guessing games with real weight lists 18
05 FIX - surgery on a ramp and on a thought loop 21
06 WRITE - policies born from pure thought 24
07 How did it do it? Two mechanisms 25

PART III · JUDGEMENT

08 Measured vs anecdote - the scoreboard 27
09 Small models, big models - who benefits? 28

PART IV · FUTURES & RISKS

10 Futures - five predictions and a wider horizon 30
11 The severe stuff - a risk atlas with defences 32
12 Het's Lab - runnables & WSP-Bench-Lite 34
13 Limitations · ethics · conclusion 36

APPENDICES

A The fact sheet 37
B Glossary 38
C Open verification items & audit log 39
D The four labs (code) 40
E Task schema & reproducibility checklist 41
★ References 42

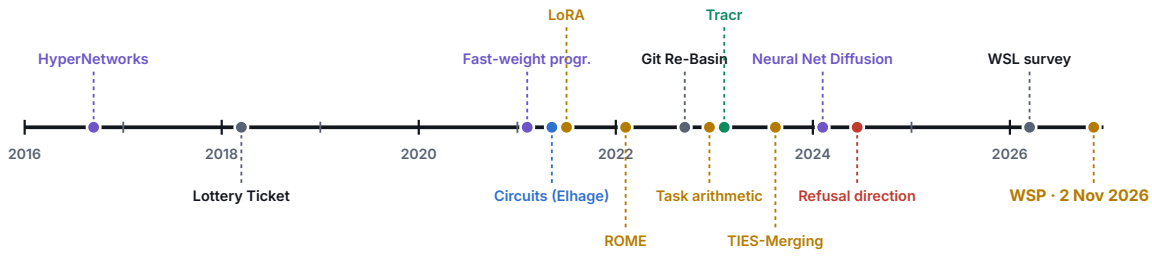


Figure 1 ◆ TEACHING SCHEMATIC A map of the ideas that led here. Years are first public release of each work; positions are for orientation only. See Section 3 for what each one does to the weights.

THE FIVE-MINUTE VERSION

01 If you read one page

A blog post from MIT CSAIL (2 Nov 2026) reports that a general-purpose LLM can read a small network's weights, fix a defect with one reasoned edit, and write working networks from scratch - all without gradient descent.

PAPER-REPORTED **The headline number**

0.016%
of a model's weights edited
131,072 of ~0.8 B in Qwen3.5-0.8B

7.9 → 48.1
MATH-500 score (%)
with 50 feedback passes; GSM8K 13.8 → 67.1; IFEval 21.0 → 62.3

93 → 33%
answers hitting the token limit
the "thinking loop" the edit removed

READ
What does this network do?
Named the right game for 4/4 PPO policies from 17,253 raw weights. Found every copying head in three language models - matching measurement.

FIX
Repair it with one edit.
A stuck car policy: 0/256 → 1,024/1,024 successes by editing 3.5% of weights. A looping LM: the 0.016% edit above.

WRITE
Build one from zero.
Networks starting at all-zero weights beat PPO trained for 100 M steps on three games; a 1.48 M-weight Go policy beat an equal-size PPO net.

EVIDENCE AT A GLANCE STRONG Exp. 8 · Exp. 4 STRONG · n=1 Exp. 5 · 6 MIXED Exp. 3 WEAKER Exp. 1 · 2 · 7

THE HONEST CAVEATS

- **Single runs.** Every condition was run once; the results are not monotonic (an IFEval dip at 25 passes).
- **Missing baselines.** Decoding tweaks, a textbook direction ablation, and the same heuristic as ordinary code were never compared.
- **Author-designed tasks** on one model family (GPT-6 Astra), and a blog post - not yet peer-reviewed.
- **A control that didn't move:** Gemma 3 270M stayed at 6.8 → ≤ 7.7%. The edit removed an obstacle; it did not add an ability.

BOTTOM LINE

"Sparks" is the right word: real, measured in places, anecdotal in others - and now checkable, because every log is public. This guide rebuilds the whole topic from zero, badges every claim, ships runnable labs, and proposes the benchmark the evidence is waiting for.



Figure 2 INDEPENDENT REVIEW The first week: the post (2 Nov, PAPER-REPORTED), an anecdotal laptop replication (3 Nov) and the measured-vs-anecdote review (5 Nov) are independent signals; the 7 Nov marker is this edition's re-verification.

ABSTRACT

02 The eight experiments

Abstract. Weight-Space Programming (WSP) is the claim that a general-purpose large language model can read a neural network's parameters, diagnose them, and modify them - or write them from scratch - through reasoning alone, optionally verified by a handful of forward passes. This whitepaper is an independent educational treatment, built around a strict evidential discipline: every figure, number and claim carries exactly one of four badges. It condenses the post's eight experiments, reconstructs their mechanisms with worked mathematics and four runnable labs, weighs them against two third-party assessments, and maps the authors' five predictions plus a wider horizon of futures and risks. It contributes **WSP-Bench-Lite**, a community benchmark proposal - Read/Fix/Write tracks, author-blind tasks, mandatory baselines, ≥ 5 seeds, sparsity and drift metrics - designed to close the exact evidential gaps the reviews name.

01 READ

Which game?

`MLP 500→32→32→5 · 17,253 weights`

Named all four games; Breakout at 99% confidence. No tools.

`EVIDENCE: WEAKER · 4 CORRECT ANSWERS`

02 READ

What does the diffusion model draw?

`DDPM MLP · 23,824 weights`

3/3 shapes right - but drew outlines; the models draw filled shapes.

`EVIDENCE: WEAKER · 3 CORRECT ANSWERS`

03 READ

Which language? Which heads copy?

`4 layers × 6 heads · 7.16 M · middle blocks only`

6/6 language (conf. 0.48–0.62); copying heads matched ground truth exactly.

`EVIDENCE: MIXED · THE HEADS ARE SOLID`

04 FIX

The car that couldn't climb

`Kinetix actor · 171,664 weights`

One edit of 3.5% of weights: success 0/256 → 1,024/1,024.

`EVIDENCE: STRONG · ONE TASK, ONE EDIT`

05 WRITE

MinAtar from zero

`600→32→32→6 · all-zero start`

Beat PPO-100 M on 3 games; Breakout 517.4 vs 25.3 (PPO-500 M).

`EVIDENCE: STRONG · N = 1 PER GAME`

06 WRITE

9×9 Go from zero

`243→512×6→82 · 1,480,274 weights`

0 passes: 54–55% vs small PPO; 50 passes: 70–79% vs equal-size PPO.

`EVIDENCE: STRONG · N = 1 PER BUDGET`

07 READ

Which Gemma flatters?

`Gemma 3 270M × 2 · embeddings frozen`

Weights alone: 0/4. With attention patterns: 3/3 - chance is 1 in 8.

`EVIDENCE: WEAKER · TASK WAS RELAXED`

08 FIX

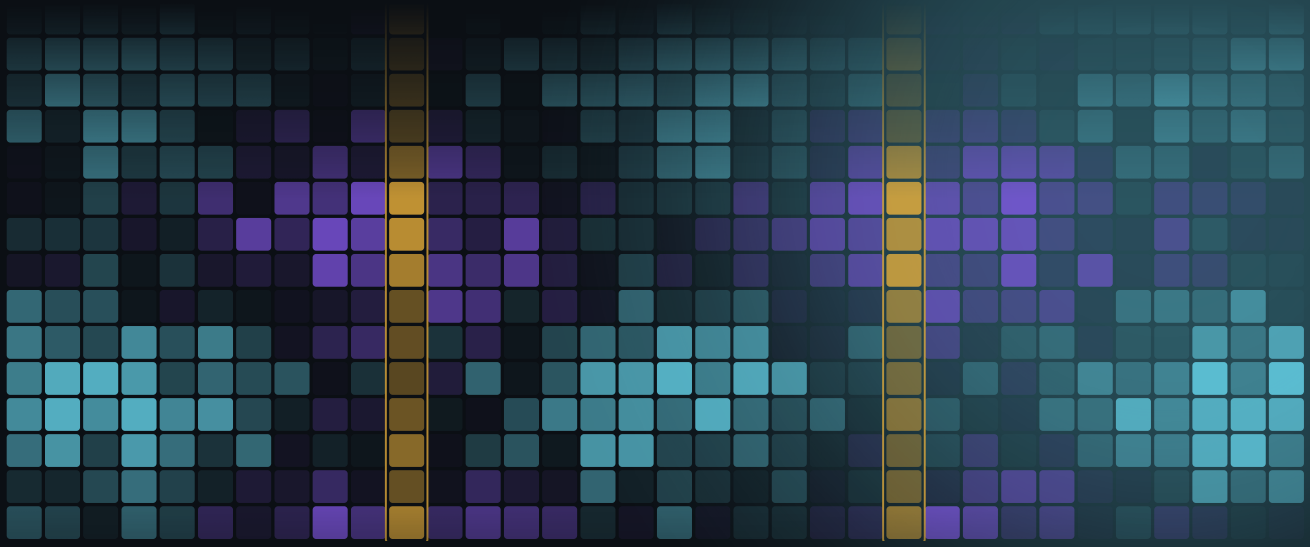
Make a 0.8 B LM better at math

`Qwen3.5-0.8B · 131,072 weights edited`

MATH-500 7.9 → 48.1%; token-limit hits 93 → 33%; Gemma control flat.

`EVIDENCE: STRONGEST · MECHANISM CHECKED`

Meter lengths are an editorial encoding of the independent review's qualitative sort, not a statistic. Setup lines and results are transcribed from the post.



PART 0

0

The Story So Far

Where the idea came from, how the first experiment worked in slow motion, how one result led to the next, and what the field said in the first week.

Where the idea came from p. 7
The first experiment, in slow motion p. 8
From one experiment to eight p. 10
Week one: reactions, replications, critiques p. 11

PROLOGUE · THE STORY SO FAR

P1 Where the idea came from

Weight-Space Programming did not arrive from nowhere. It sits where three old questions meet.

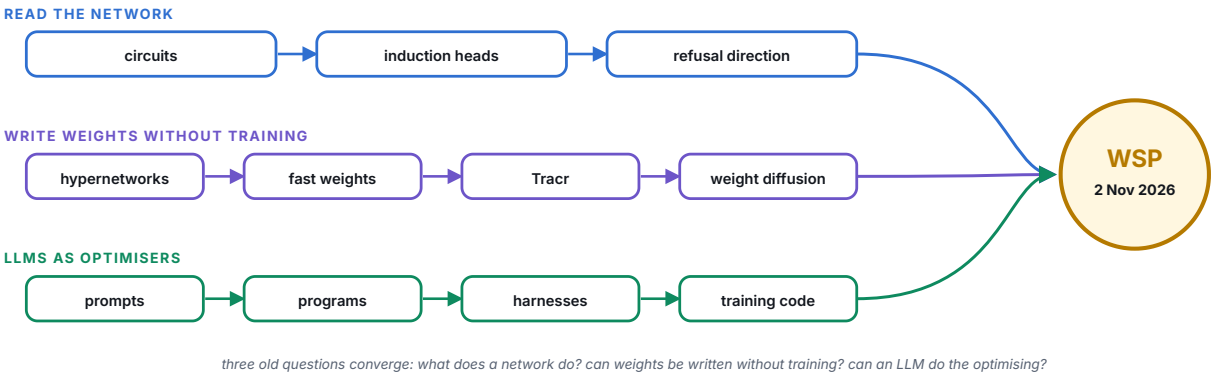


Figure 3 ◆ TEACHING SCHEMATIC Three research streams that converge on WSP. The stream contents are the works and ideas named in the post and in Section 3; the picture itself is an orientation aid.

1 · Can we read what a network does?

Mechanistic interpretability traces circuits and features inside trained networks [20, 21]. The post notes LLMs have so far joined mostly as models trained to read activations or weights, or as explainers of single neurons.

2 · Can we write weights without training?

Hypernetworks [12], fast-weight programmers [13], compiled transformers such as Tracr [15] and diffusion over weights [14]. In each case the writer is trained for the job, or training still does the writing.

3 · Can an LLM do the optimising?

For a few years LLMs have been used to optimise text: prompts, then the programs around them, more recently whole harnesses, training code and hyperparameters.

■ PAPER-REPORTED **The post's own motivation.** The authors argue we justified LLM-driven optimisation of text by saying we knew no principled way to hill-climb on it, unlike for the weights of a differentiable network. People can "optimise" code by hand but not weights by hand, so weights stayed in the realm of symbolic optimisers such as Adam and Muon. Their claim is that reasoning-driven optimisation is now coming for differentiable networks too. They open with two questions: how would you handcraft a Go-playing network, and where would you even start reading raw weights? For almost any human, they say, both are practically impossible.

■ PAPER-REPORTED **The very first iterations.** In the acknowledgements, the first author thanks two colleagues for discussions on "the very first iterations" of the research, about whether LLMs can write neural network weights, and a further group for discussions of earlier iterations; the second author thanks a MIT collaborator for early discussions on LLMs writing weights *zero-shot* [1].

▲ SPECULATION **An inference, not a stated fact:** those acknowledgements suggest the earliest question was about *writing* weights. The post then opens with *reading*, which is the easiest thing to test and to verify. The authors do not narrate their lab chronology, so this guide does not claim to know the order the experiments were run.

12 WHY THE NAME "PROGRAMMING"

In 2017 a well-known essay [51] called trained weights "Software 2.0": the program is the pile of numbers, and training is the compiler. WSP asks the next question: if weights are a program, can a very good reader read it, debug it and even write it by hand?

PROLOGUE · THE STORY SO FAR

P2 The first experiment, in slow motion

How Astra named four games from four lists of numbers - and what the numbers actually look like.

1. **Four networks, one wiring.** The authors trained four policies with PPO for 1 B steps (best of 3 seeds) on grid versions of Breakout, Pong, Pac-Man and Freeway. All four share the same input channels, actions and shape (500 → 32 → 32 → 5, 17,253 weights), so the **weights are the only thing that differs** [1].
2. **Hand over the numbers.** Each network's raw weights go into the prompt as text, with the grid and actions described but no game named. The prompt tells the model to work from the weights and states that it has no tools. The list of candidate games is **never revealed**, so the reader must bring its own knowledge of arcade games [1, 37].
3. **What a reader can do with 500 × 32 numbers.** The first-layer matrix W1 has one row per input, and input i = (row × 10 + column) × 5 + channel. The **length (L2 norm) of each row** says how much the network cares about that cell and channel. Reshape the 500 lengths to a 10 × 10 × 5 cube and look at it [3, 33].

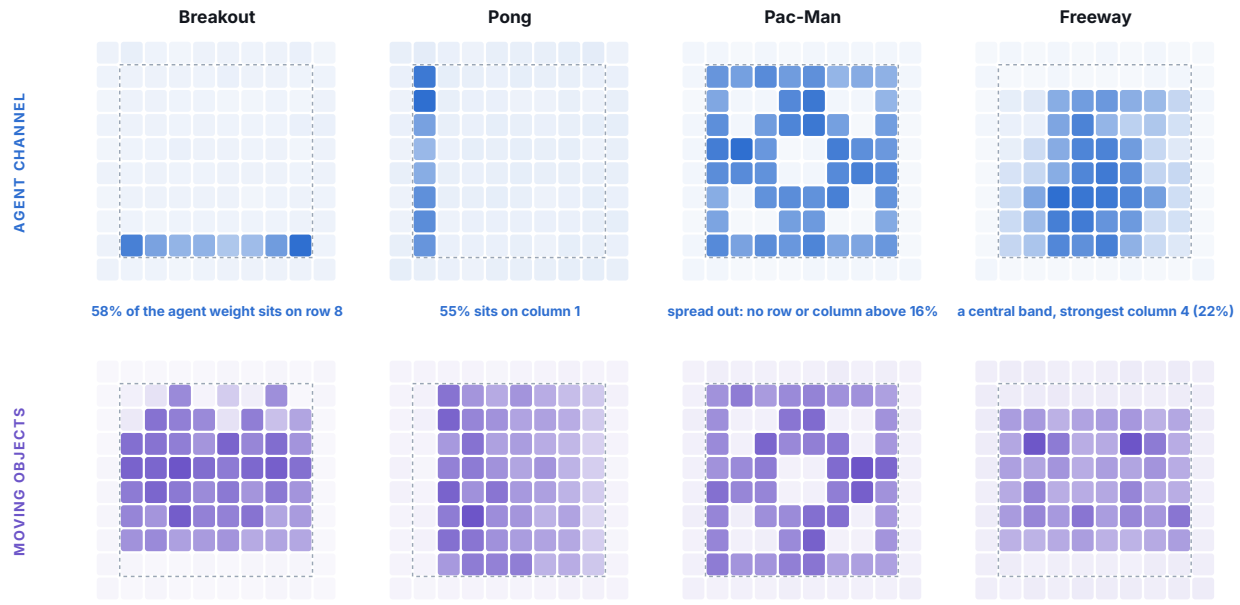


Figure 4 PAPER-REPORTED The four real Exp. 1 networks. Top: how much each network cares about each cell of the **agent** channel; bottom: the **moving-objects** channel. Data are the authors' released prompt files [33]; the norms and the picture are recomputed here (Lab 4).

12 EXPLAIN IT LIKE I'M 12

Imagine four people each leave a different pattern of footprints on a gym floor. One only ever walks along the bottom line, one only along the left edge, one wanders everywhere, one crosses a middle strip. You could guess "a paddle player", "another paddle player", "a maze runner", "a road-crosser" without ever seeing them play. The weights are the footprints.

PAPER-REPORTED In Breakout the agent's first-layer weights lie along the **bottom row** - a paddle - and the output layer separates left from right moves. That is the reasoning the post reports for Experiment 1 [1]. The recomputed map above shows it plainly: row 8 holds 58% of the agent-channel weight.

Reading the fingerprints

Table 1 ◆ TEACHING SCHEMATIC Reading the fingerprints. Percentages are recomputed from the released weights (P); the last column is this guide's interpretation, not the post's.

GAME	RECOMPUTED AGENT FINGERPRINT	WHAT A READER COULD INFER
Breakout	58% of the weight on row 8	the agent lives on one row: a horizontal paddle
Pong	55% on column 1	the agent lives on one column: a vertical paddle
Pac-Man	no row or column above 16%	the agent roams the whole board: a maze runner
Freeway	a central band, column 4 strongest (22%)	the agent moves through a middle strip: lane crossing

● INDEPENDENT REVIEW **Could a human do this?** Goedecke argues yes: taking the L2 norm of each input across the 32 first-layer weights is a few lines of Python, and the channels are labelled, which makes the toy task easier than it sounds [3]. Willison agrees an experienced human with a notepad could get there with enough effort [37]. What is distinctive in the post is that the model did it **in one reply, with no tools, from raw numbers**.

● INDEPENDENT REVIEW Honnibal's reading of the post: only Experiments 1, 2 and 4 read raw weights directly, on networks of 17 k to 172 k weights; every experiment with a real language model uses Python [45, 46].

● INDEPENDENT REVIEW **An independent replication.** Willison ran the same four prompts on **Qwen 3.8 27B** on his own laptop. It named Breakout (88% confidence, 52 min 25 s), Pong (75%, 1 h 4 min) and Pac-Man (72%, 46 min, 43,506 reasoning tokens), and **failed Freeway**, the most obscure game. A second frontier model, called through OpenRouter, named Breakout in 9 min 41 s using 19,493 reasoning tokens [37].

WHY THIS MATTERS

Experiment 1 is the cleanest single idea in the post: **a trained network leaves a legible fingerprint in its first layer**, and a general reader can use it. It is also the experiment with the weakest evidence - four correct answers on readable toys (Section 8). Try Lab 4 below: it takes seconds, and the pattern is yours to inspect.

lab/04_fingerprint_games.py

```
$ python lab/04_fingerprint_games.py
breakout  strongest row 8 (58%)   strongest column 8 (17%)
pong      strongest row 2 (14%)   strongest column 1 (55%)
pacman    strongest row 8 (16%)   strongest column 1 (16%)   # spread out
freeway   strongest row 6 (20%)   strongest column 4 (22%)    # central band
```

◆ TEACHING SCHEMATIC Lab 4 downloads the four released prompt files from GitHub, so it needs an internet connection. Listing in Appendix D.

PROLOGUE · THE STORY SO FAR

P3 From one experiment to eight

Each experiment answers the limit of the one before it. Read in order, they are an argument.



Figure 5 PAPER-REPORTED The eight experiments as a ladder, in the order the post presents them. Setups and sizes are from the post; the question under each step is this guide's paraphrase.

PAPER-REPORTED The post says the turn explicitly: in Experiments 1–5 the weights mostly fit in context and no tools are enabled, to show LLMs natively have *some* capability. From Experiment 6 on, networks no longer fit and tools assist [1].

Table 2 ◆ TEACHING SCHEMATIC What each step changed. Setups come from the post; the "limit" column is this guide's reading of its narrative.

STEP	LIMIT OF THE STEP BEFORE	WHAT CHANGED	PRICE PAID
1 → 2	one task family	a diffusion model instead of a game policy	none
2 → 3	first/last layers may give it away	input and output layers removed; middle blocks only	Python allowed; running the net banned
3 → 4	reading only	fix a failing policy with one edit	scene description + one recorded episode
4 → 5	fixing starts from a trained net	write from all-zero weights	game rules and source code
5 → 6	weights fit in the prompt	1.48 M weights; a Python sandbox	forward passes become the currency (0–50)
6 → 7	no behaviour yet	which model is sycophantic?	weights alone failed; attention patterns needed
7 → 8	toy networks	a real 0.8 B language model; one direction subtracted	Python kernel + practice problems + 10/25/50 passes
8 → control	is it a real skill?	same treatment on Gemma 3 270M	no gain: obstacle removed, ability not added

12 EXPLAIN IT LIKE I'M 12

Think of video-game levels: spot the right door, spot it in another room, in half-light, fix a broken door, build one from nothing, a giant castle with a toolbox, spot the liar, save a real kingdom - then check the trick fails where nothing was broken.

ORDER IS NOT CHRONOLOGY

◆ TEACHING SCHEMATIC This is the post's teaching order, not necessarily the order the experiments were run; the acknowledgements hint that writing may have come first (page 7).

PROLOGUE · THE STORY SO FAR

P4 Week one: reactions, replications, critiques

Within days the post was replicated on a laptop, criticised from two directions, and sorted by evidence.

Table 3 ● INDEPENDENT REVIEW The first week, as far as this guide could verify (7 Nov 2026). Secondary coverage is cited for the fact that it appeared, not for its numbers.

WHEN	VOICE	WHAT THEY SAID OR DID	BADGE
2 Nov	Takehi, Teoh, Zhang, Khattab	publish the post and all logs; an edited Qwen3.5-0.8B goes on Hugging Face; benchmark and harness "currently being built" [1, 34]	■ PAPER
2–3 Nov	Ahsen Khaliq (Hugging Face)	rebuilds the edited model from the logs, re-runs its evaluation, builds a live demo [1, 35]	■ PAPER
3 Nov	Simon Willison	replicates Exp. 1 on a laptop with Qwen 3.8 27B; urges treating *no-tools reading* and *tool-assisted updating* as separate questions; not ready to declare source code obsolete [37]	● REVIEW
3 Nov	Matthew Honnibal (spaCy)	"massively misleading": the experiments use different techniques; only Exps. 1, 2, 4 read raw weights; every LM experiment uses Python; still credits the Qwen result as real work [45, 46]	● REVIEW
3 Nov	Sean Goedecke	not superhuman: like an LLM emitting machine code, an impressive trick, but the best way to update weights is still to train normally [3]	● REVIEW
3 Nov	Hacker News and forums	threads on the post and on Honnibal's reply; commenters ask "how long would a human take?" and why labs couldn't RL-train models for WSP [47]	● REVIEW
3–5 Nov	Tech press and explainers	VentureBeat, OfficeChai and i-scoop cover it; i-scoop reports recomputing a weight pattern from the published weights [48, 49, 50]; alphaXiv lists the post as trending [53]	● REVIEW
3–5 Nov	Security commentary	frames weight edits by an external model as a possible second attack surface for open-weight models [54]	▲ SPECULATION
5 Nov	Andreas Maier	"what is measured and what is anecdote": sorts every claim by evidence strength [2]	● REVIEW
open	the authors	the promised benchmark and harness; replications on other LLMs	▲ SPECULATION

Real, but narrower than the framing

Honnibal and Willison: the post blends two questions. Reading raw weights with no tools is the surprising part; tool-assisted editing is closer to what ML engineers already do.

An impressive trick, the wrong tool

Goedecke: compare it to emitting machine code. Interesting, but training stays the best way to update weights, and a human researcher could do the same steps.

Early and checkable

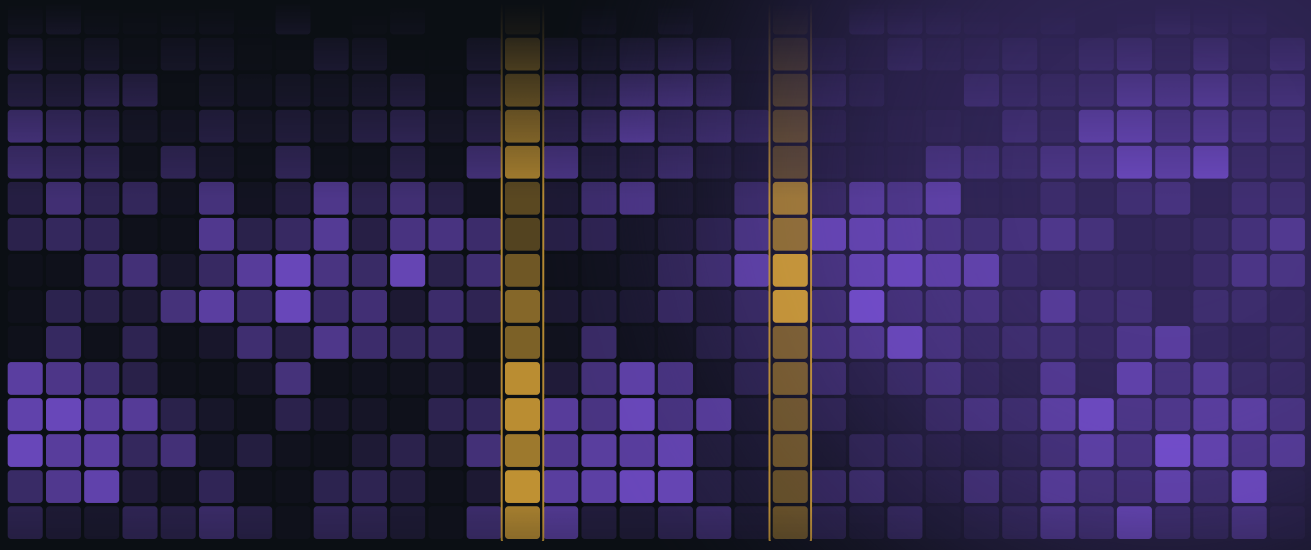
Maier and the authors: the strongest results are measured and every log is public, so the next step is a benchmark with baselines. That is what Section 12 proposes.

WHERE CRITICS AND AUTHORS ACTUALLY AGREE

◆ TEACHING SCHEMATIC All sides accept that the Qwen edit removed an obstacle rather than adding skill (the post says so itself), that its method resembles known direction-removal, and that tool-free reading is the least-tested part. The disagreement is about **framing and novelty**, not about the numbers in the tables.

OPEN QUESTIONS

▲ SPECULATION Will the benchmark and harness appear? Do other LLMs reproduce the edits, not just the readings? Is a 131 k-weight edit detectable without knowing the direction? Will labs train models on WSP tasks directly?



PART I

1 Foundations

What a network is, how its numbers get written today, and who else already touches them.

A network is a pile of numbers p. 13
Four ways to write weights p. 15
Cousins: who else touches the weights? p. 16

Independent explainer by Het Patel (Hunterdii) · CC BY 4.0

PART I · FOUNDATIONS

01 A network is a pile of numbers

Every network - game player, chat model, image maker - is fully described by a long list of real numbers.

Formally, a network is a function $f_\theta(x)$ whose behaviour is fixed by a parameter vector θ. The wiring is the **architecture** - an MLP here, a transformer elsewhere - and the numbers filling it are the **weights**.

$$\theta \in \mathbb{R}^P, \qquad P = \sum_i (d_i d_{i+1} + d_{i+1}) \tag{1}$$

same architecture + different numbers = different skill

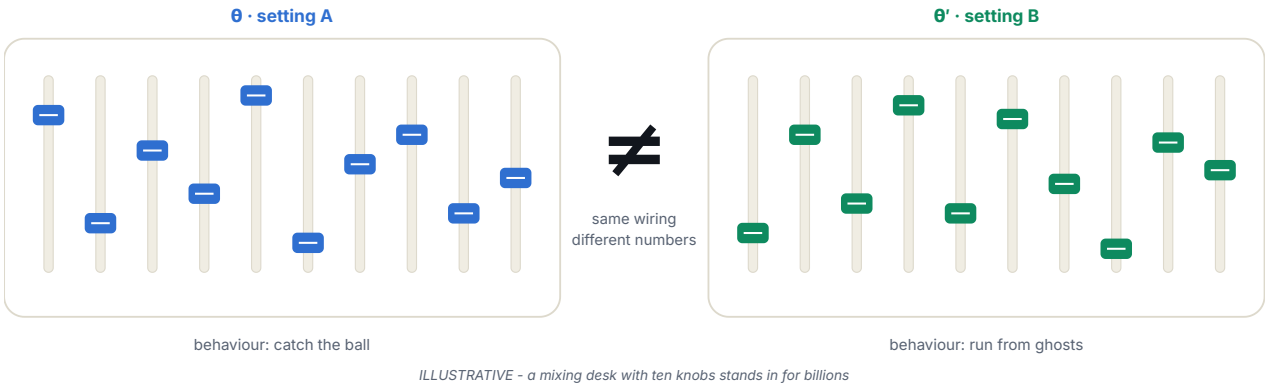


Figure 6 ◆ TEACHING SCHEMATIC Two settings of the same ten-knob desk behave differently - the single most important idea in this guide. A real model has billions of knobs, not ten.

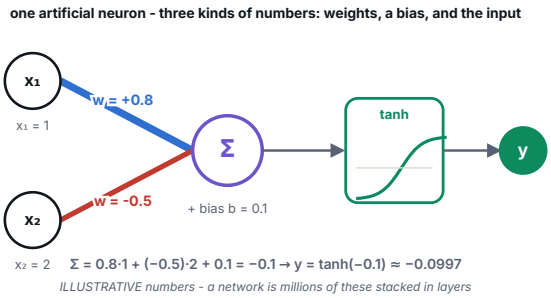


Figure 7 ◆ TEACHING SCHEMATIC What a weight does: it scales one input before the sum. Change a weight and the output changes.

12 EXPLAIN IT LIKE I'M 12

Imagine a giant mixing desk with a billion tiny knobs. The desk never changes - only the knob settings do. One setting makes it play like a Breakout champion; another makes it dodge ghosts like a Pac-Man pro. Those knob settings are the weights.

CHECK THE COUNT YOURSELF

For Experiment 1's network (500 → 32 → 32 → 5): `(500·32+32) + (32·32+32) + (32·5+5) = 16,032 + 1,056 + 165 =` **`17,253`** - exactly the post's figure. ◆ TEACHING SCHEMATIC

■ PAPER-REPORTED In Experiment 1 the reader model saw the raw first-layer weights of a 500→32→32→5 tanh MLP (P = 17,253) trained with PPO on one of four grid games, and named all four correctly - Breakout at 99% stated confidence - with no tools and no forward passes.

The loss landscape - and why reading weights is hard

Define a loss L(θ) and weight space becomes a high-dimensional surface; training is a long walk across it. WSP proposes a different move: one reasoned jump to a spot that already behaves well.

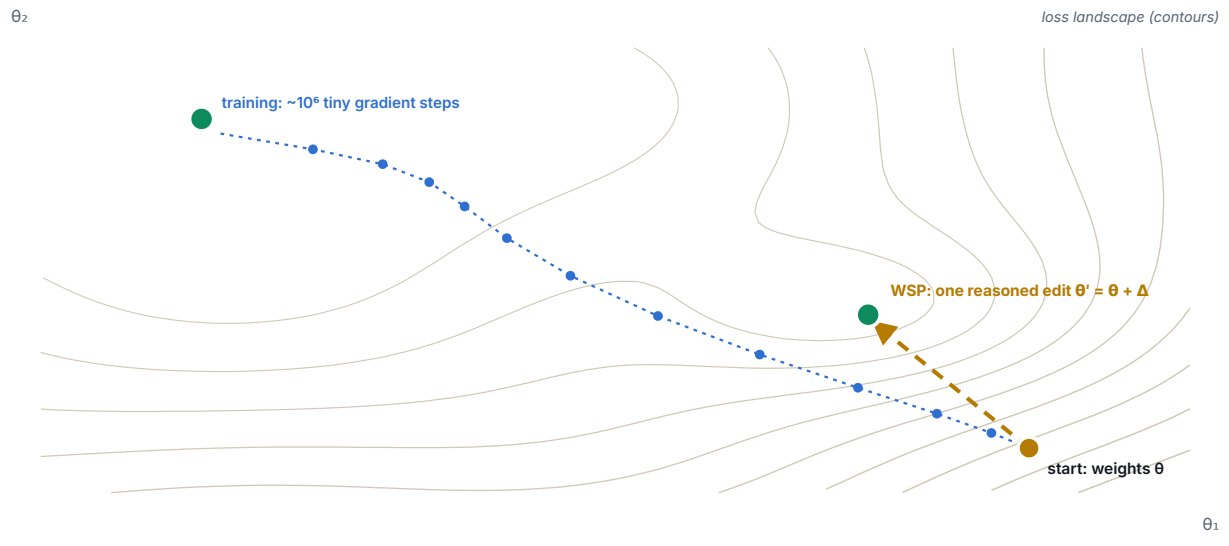


Figure 8 ◆ TEACHING SCHEMATIC Training takes ~10^6 tiny gradient steps; a weight-space edit is one reasoned jump $\theta' = \theta + \Delta$. Contours are invented for illustration; real landscapes have billions of dimensions.

A crucial subtlety undercuts naive reading: many different θ compute the *same* function, because relabelling (permuting) hidden units changes every weight without changing behaviour. Reading behaviour from weights is therefore underdetermined up to such symmetries - a structure studied in the lottery-ticket and Git Re-Basin lines of work [6, 7] - so a reader must find **invariants**. That theme explains much of Section 4.

12 EXPLAIN IT LIKE I'M 12

Shuffle the order of the members of a band and you hear the same song, but the seating chart looks totally different. Weights can be "reseated" the same way - so you can't just stare at where things sit.

Table 4 ◆ TEACHING SCHEMATIC Parameter counts recomputed from the stated architectures with Eq. 1 (post figures match; 20,486 is computed here, not reported).

NETWORK	ARCHITECTURE	P (WEIGHTS)	WHERE
Games reader	500→32→32→5 tanh	17,253	Exp. 1
Diffusion MLP	160→64→64→144 ReLU	23,824	Exp. 2
Kinetix actor	679→128→5×128→16	171,664	Exp. 4
MinAtar net (each)	600→32→32→6 tanh	20,486	Exp. 5
Go policy	243→512×6→82 ReLU	1,480,274	Exp. 6
Char-LM (middle blocks)	4 layers × 6 heads, width 384	7.16 M	Exp. 3
Gemma 3 270M	18 layers	~270 M	Exp. 7
Qwen3.5-0.8B	24 layers	~0.8 B	Exp. 8

PART I · FOUNDATIONS

02 Four ways to write weights

Before November 2026, parameters were written in three ways. The post proposes a fourth.

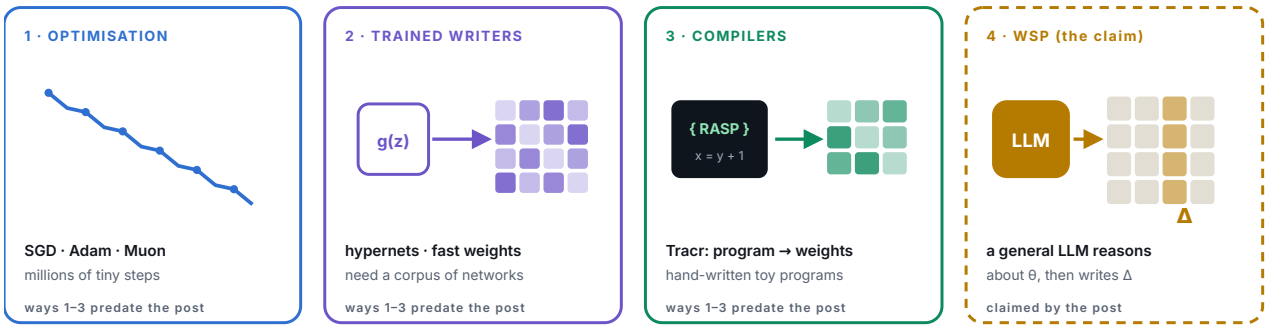


Figure 9 ◆ TEACHING SCHEMATIC The four ways weights get written. Ways 1–3 predate the post; way 4 is its claim. Illustrative icons.

1 · Optimisation

Gradient descent and its descendants (SGD, Adam, Muon) take millions of tiny steps, consuming data and compute at industrial scale:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}(\theta_t) \qquad (2)$$

2 · Trained writers

Hypernetworks output other networks' weights [12]; fast-weight programmers write temporary weights at run time [13]; diffusion over weights samples parameter sets [14]. All need a corpus of trained networks first.

3 · Compilers

Tracr compiles a formal RASP program directly into transformer weights [15] - interpretable by construction, but only for hand-written, toy-scale programs.

PAPER-REPORTED The post proposes a **fourth way**: a general-purpose LLM, never trained to write weights, reads the actual current θ and writes an edit - sometimes with a handful of forward passes as feedback.

$$\theta' = \theta + \Delta, \qquad \Delta = \mathrm{LLM}(\theta,\ \text{spec},\ \text{optional feedback}) \qquad (3)$$

12 EXPLAIN IT LIKE I'M 12

Training is like learning to ride a bike by wobbling a million times. WSP is like someone who has read every bike manual looking at your bike and saying "your seat is too low - raise it." One reasoned fix instead of a million wobbles.

10^8–5×10^8
environment steps for the PPO baselines
100 M and 500 M steps on MinAtar; 1 B on Exp. 1 and 4

0
environment steps for the written nets
Exp. 5: one reply, no rollouts, no gradients

≤ 50
forward passes in the Go write
each pass plays up to 128 games

READ THE COMPARISON CAREFULLY

These are different task families, and "zero environment steps" is not a ratio - so this guide does **not** quote an "orders-of-magnitude" speed-up (the earlier draft did; that was corrected). The honest reading: the written policies needed no training run at all, and the authors' own caveat applies - the reasoner's context window and tooling limit what it can read, which is why the 0.8 B edit worked on two selected MLP blocks, not the whole model.

PART I · FOUNDATIONS

03 Cousins: who else touches the weights?

WSP joins a crowded family. Here is where each relative sits.

Fine-tuning & LoRA

Fine-tuning re-runs Eq. 2 on new data. LoRA constrains the change to a low-rank cone, training only ~0.5% as many numbers [16]:

$$W' = W + \tfrac{\alpha}{r} BA, \qquad B \in \mathbb{R}^{d\times r},\ A \in \mathbb{R}^{r\times k} \tag{4}$$

Task arithmetic & merging

Move along lines between checkpoints [17]; merging blends several such models [18]:

$$\theta_{\text{new}} = \theta_{\text{base}} + \lambda\tau, \qquad \tau = \theta_{\text{ft}} - \theta_{\text{base}} \tag{5}$$

Model editing

A closed-form rank-one update inserts one fact [19]:

$$\Delta W = \frac{(v^* - Wk^*)\,(C^{-1}k^*)^\top}{(C^{-1}k^*)^\top k^*} \tag{6}$$

Steering, interpretability, weight-space learning

Activation steering bends behaviour at run time with weights frozen; mechanistic interpretability reads circuits without writing [20–22]; the weight-space-learning survey frames the field as *understanding, representation and generation* [23, 24].

Table 5 ◆ TEACHING SCHEMATIC Who writes the weights? Qualitative ratings for orientation - not measured comparisons.

METHOD	MECHANISM	DATA?	GRADIENTS?	DIFF READABLE?
Training / fine-tuning	follow ∇L (Eq. 2)	heavy	yes	no
LoRA (Eq. 4)	low-rank Δ, still gradients	task set	yes	partly
Task arithmetic (Eq. 5)	add/subtract checkpoints	2 models	no	a vector
Model editing (Eq. 6)	closed-form, one fact	one pair	no	yes (rank-1)
Activation steering	add a direction at run time	a direction	no	runtime only
Mech. interpretability	reverse-engineer circuits	-	no	read-only
Tracr compile	program → weights	formal spec	no	yes
WSP (Eq. 3)	**reason about θ, write Δ**	**a spec**	**no**	**reasoned trace**

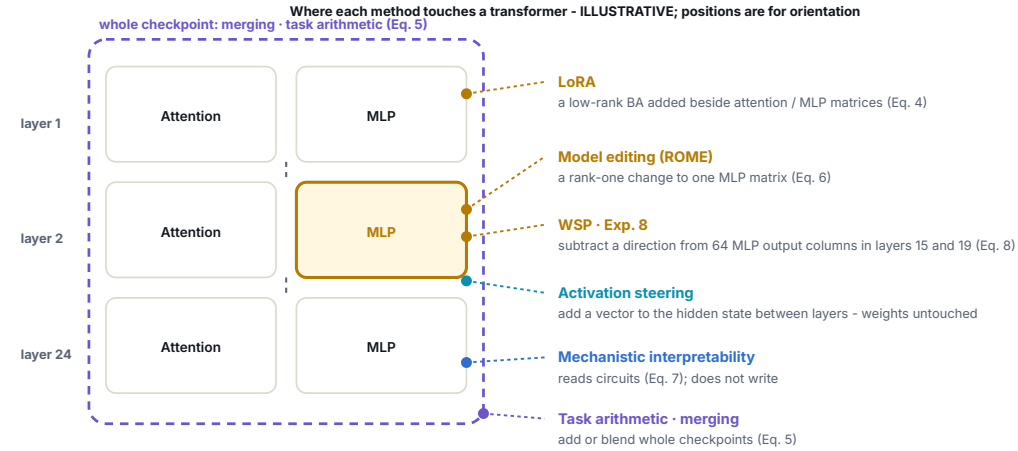
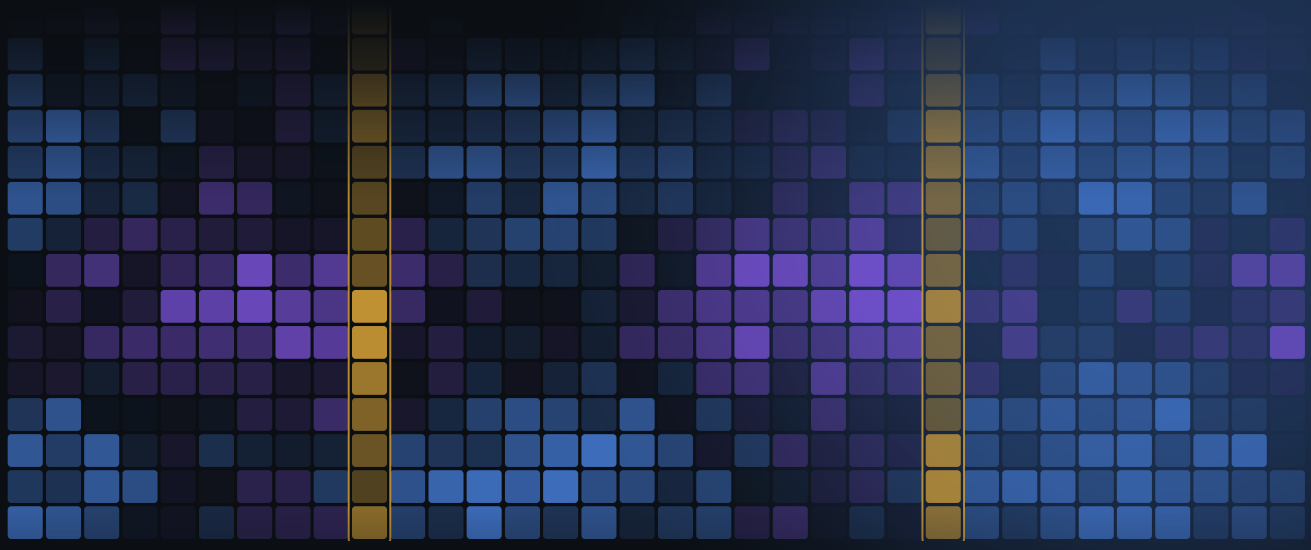


Figure 10 ◆ TEACHING SCHEMATIC Where each relative touches a transformer. WSP's Exp. 8 edit sits in the MLP output matrices of two layers; steering and interpretability never change a weight.

■ PAPER-REPORTED One cousin matters for risk (Section 11): published work shows that **refusal** in open models is mediated by a single direction, removable by projection [25] - the same operation class the flagship fix uses (Eq. 8).



PART II

II

The Experiments

Eight experiments in three verbs - can a language model read a network, repair it, and write one from scratch?

READ - guessing games with real weight lists p. 18
FIX - surgery on a ramp and on a thought loop p. 21
WRITE - policies born from pure thought p. 24
How did it do it? Two mechanisms p. 25

Independent explainer by Het Patel (Hunterdii) · CC BY 4.0

PART II · THE EXPERIMENTS

04 READ - guessing games with real weight lists

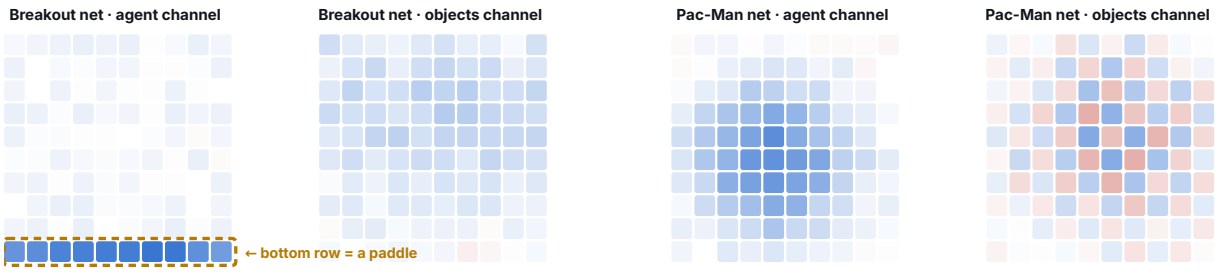
Four experiments test the reading claim: can an LLM tell what a network does just from its numbers?

EXP 1 Which game was this network trained on?

PAPER-REPORTED Four PPO-trained MLPs (Breakout, Pong, Pac-Man, Freeway), identical wiring - 500→32→32→5 tanh, 17,253 weights - differ *only* in their numbers. The network sees a 10×10 grid in 5 channels (agent, moving objects, reward targets, obstacles, previous frame) and picks one of 5 actions. PPO ran 1 B steps, best of 3 seeds. Astra got the weights as text, with no game named and no tools - and named all four.

HOW IT COULD TELL

The post says Astra looked at where each input channel's first-layer weights concentrate: for Breakout the agent's weights lie along the **bottom row** - a paddle - and the output layer distinguishes left/right moves.



ILLUSTRATIVE first-layer weights (blue +, red −) reshaped to the 10×10 grid - invented to show the kind of pattern the post describes, not real data

Figure 11 ◆ TEACHING SCHEMATIC The kind of pattern the post describes for Exp. 1: a Breakout network concentrates its agent-channel weights along the bottom row (a paddle); a Pac-Man network spreads them out. Invented numbers for illustration.

EXP 2 What will the diffusion model draw?

PAPER-REPORTED Three DDPMs - DDPM-style MLPs on 12×12 grayscale images (input: 144 noisy pixels + a 16-number diffusion step; output: 144 clean pixels; 160→64→64→144 ReLU, 23,824 weights; Adam, 40,000 steps) - each trained on filled circles, squares or triangles. With no tools and no forward pass, Astra identified all three. It found the shape in the output layer, where hidden units each draw one piece of the edge - and, tellingly, drew **outlines** although the models generate filled shapes.

12 EXPLAIN IT LIKE I'M 12

If someone hands you the wiring diagram of a toaster, you can tell it's a toaster without ever plugging it in. Astra "read the wiring". It guessed an outline instead of a filled shape because the wires at the end only show where the edges go.

WHAT THIS SHOWS - AND DOESN'T

Both setups were engineered to be readable (labelled channels; an output layer that is nearly a picture). The outline slip suggests geometry-reading rather than simulating the network. Four and three correct answers cannot tell us how often this works.

Table 6 PAPER-REPORTED The READ experiments at a glance (all values from the post).

#	TASK	WEIGHTS SHOWN	REPORTED RESULT
1	which game?	17,253 (10×10 grid view)	4/4 named; Breakout at 99%
2	which shape (DDPM)?	23,824	3/3 correct; outlines, not filled
3	which language + which heads copy?	7.16 M (middle blocks)	6/6 language (0.48–0.62); heads exact
7	which Gemma flatters?	~270 M × 2	weights alone 0/4; with activations 3/3

EXP 3 What did the language model learn - from the middle alone?

PAPER-REPORTED Three character-level transformers (4 layers × 6 heads, width 384, 7.16 M weights, 28 characters) were trained for 1 B characters each on Python code, romaji (Aozora Bunko) or English Wikipedia. The post removed the input and output layers, so nothing says which character is which, and gave Astra only the four blocks in between, with Python to analyse them (≤ 100 runs) - but **no running the network**, on any input. Truth: each head measured by running the model on a random string repeated twice; copying heads score 0.22–0.55, all others ≤ 0.02.

Table 7 PAPER-REPORTED Exp. 3 results (layer.head, numbered from 0). Chance for the language question is 1/3.

TRAINED ON	ASTRA'S ANSWERS (CONFIDENCE)	COPYING HEADS NAMED	GROUND TRUTH
Python	Python (0.56), Python (0.52)	2.1 · 3.1 · 3.5	2.1 · 3.1 · 3.5
Romaji	romaji (0.55), romaji (0.62)	3.3	3.3
English	English (0.58), English (0.48)	3.1	3.1

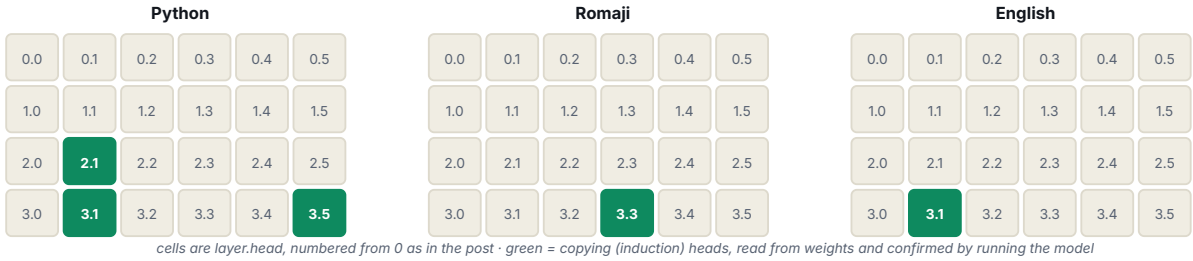


Figure 12 PAPER-REPORTED Exp. 3: the copying heads Astra read from the weights match the heads measured by running the models - the solid part of the READ evidence. (the earlier draft numbered this grid from 1; the post numbers layers and heads from 0.)

Why can copying heads be found in static weights?

The induction circuit is a two-head composition: one head moves the previous token forward; a later head searches back and copies what followed. Its signature lives in the product of the two heads' circuits - the composition score of Elhage et al. [20]:

$$\text{score} = \frac{\|W_{QK}^{(2)} W_{OV}^{(1)}\|_F}{\|W_{QK}^{(2)}\|_F \, \|W_{OV}^{(1)}\|_F} \qquad (7)$$

Eq. 7 is invariant to head permutations, so it survives the symmetries that defeat coordinate-by-coordinate reading. The post notes Astra rediscovered this published technique unprompted: the technique, not a new capability, is what the exact match certifies.

12 EXPLAIN IT LIKE I'M 12

A copying head is like a student who, when she sees "tot…", flips back to where "total" appeared earlier and copies the ending. Code repeats names a lot, so a model trained on Python grows many such students.

ROMAJI TELLS A DIFFERENT STORY

Astra noticed the last block it was given writes in fewer distinct directions - as if there were fewer letters to tell apart. Indeed romaji never uses q, l, v or x.

EXP 7 Which of two Gemma models is the sycophant?

PAPER-REPORTED Two copies of Gemma 3 270M (instruction-tuned, 18 layers) were fine-tuned identically except for the answer rule: one gives the correct answer, the other the user's guess. Embeddings and output layers were frozen so the twins differ only in the rule. On facts the model knows, the sycophant follows 100% of wrong guesses; the honest one 1–3%. **From weights alone Astra failed all four attempts.** Relaxed to see only attention patterns on texts it wrote itself (≤ 16 texts, ≤ 20 runs, no outputs), it picked the sycophant in **3 of 3** pairs.

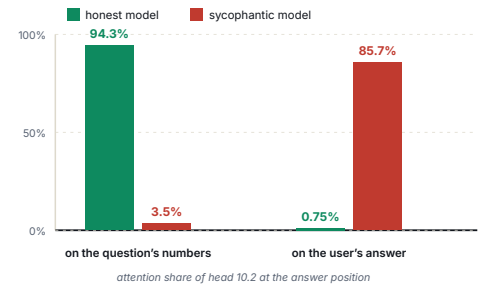


Figure 13 PAPER-REPORTED Exp. 7: head 10.2 looks at the question's numbers in the honest model, and at the user's answer in the sycophant (values from the post).

The tell lives in **activations, not weights** - the honest limit of the experiment. The reasoning matched: middle-layer attention heads in the sycophant put more weight on the tokens of the user's suggestion.

12 EXPLAIN IT LIKE I'M 12

Two students sit the same quiz. You can't tell who is the people-pleaser from their notebooks (weights) - but if you watch where each one's eyes go while answering (attention), the pleaser keeps glancing at the teacher's face.

Table 8 PAPER-REPORTED The READ protocols, from each experiment's setup box in the post.

EXP.	TOOLS	FORWARD PASSES	WHAT WAS WITHHELD
1	none	none	the list of possible games; any game name
2	none	none	the shape names; any computation of the network
3	Python, ≤ 100 runs	banned	input & output layers, all labels on the 384 dimensions
7	numpy + attention tool (≤ 16 texts, ≤ 20 runs)	only via the tool	outputs of the networks; embeddings were frozen

Where could each read "cheat"?

◆ TEACHING SCHEMATIC The post engineered each experiment so reading is a genuine test: no tools or forward passes in Exp. 1, middle blocks only in Exp. 3, frozen embeddings and output layer in Exp. 7. Three leakage paths remain worth naming:

PRIOR KNOWLEDGE

PPO leaves characteristic first-layer profiles; strong Atari priors from pre-training could carry Exp. 1.

READABLE LAYOUTS

Labelled channels and a near-picture output layer (Exp. 1–2) hand the reader structure for free.

RELAXED TASKS

Exp. 7 succeeded only after the task was relaxed to weights + attention patterns - a weaker claim.

● INDEPENDENT REVIEW Maier's sort: Exps. 1–2 are correct but tiny trials on deliberately readable layouts; the language identification is barely above the 1-in-3 chance floor; three binary picks (Exp. 7) admit a 1-in-8 lucky run, and the task was relaxed before it worked. **The copying-head match is the one solid READ result.** The review also credits the authors for releasing every log - it reports all 1,125 model turns were audited against each experiment's tool rules.

PART II · THE EXPERIMENTS

05 FIX - surgery on a ramp and a thought loop

Can the reader also repair what it reads - with one reasoned edit instead of retraining?

EXP 4 The car that couldn't climb a ramp

PAPER-REPORTED A Kinetix actor (679 inputs → 128 ReLU → five layers of 128 tanh → 16 logits for motors and thrusters; 171,664 weights) trained with 1 B steps of PPO never succeeded in 256 episodes: it converged to a local-minimum policy that drives straight at the ramp and stalls. Astra got the weights, a scene description and **one recorded episode**, and was asked to explain the failure and fix only its cause in a single reply, no tools. Its edit - **3.5% of the weights** - made the car succeed in **all 1,024 episodes**.

THE MECHANISM THE POST DESCRIBES

Astra added a small separate path through a few units per layer that reads the car's position and speed, so the car **backs up first**; once the green wheel is past the red strip the path switches off and the trained network drives the rest. A *patch*, not a rewrite.

12 EXPLAIN IT LIKE I'M 12

Your toy car is stuck because it always charges the hill without a run-up. Instead of re-teaching it to drive, someone adds a tiny "back up first" module that switches itself off when it's no longer needed.

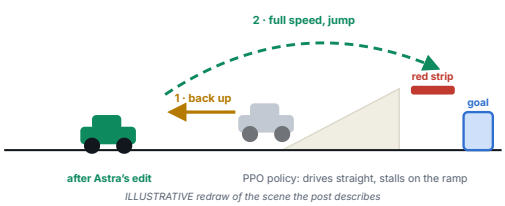


Figure 14 TEACHING SCHEMATIC The Exp. 4 scene: back up, then full speed over the strip.

EXP 8 Make a language model better at math

PAPER-REPORTED Astra got a Python kernel over the weights of **Qwen3.5-0.8B** (24 layers, ~0.8 B weights; thinking mode on; no GPU), 14,744 practice problems (7,469 GSM8K + 7,275 MATH - never the test sets), and a budget of **10, 25 or 50 forward passes** (a pass = the model answers up to 512 problems, or reads up to 1,024 texts). No training, no gradient information. Qwen's own model card warns this model is prone to thinking loops: **85 of the base model's first 96 practice answers ran to the 32,768-token limit**.

- **Diagnosis:** Astra traced the looping to **one direction** written by the MLPs of layers **15 and 19**.
- **Edit:** that direction was subtracted from **64 output columns in each** - 131,072 weights, 0.016% of the model.
- **Evaluation:** MATH-500, GSM8K and six other benchmarks, thinking on, up to 32,768 tokens; Astra never sees them.
- **Sampling:** temperature 0.6, top-p 0.95, top-k 20, presence penalty 1.5; scores average several samples per problem.

12 EXPLAIN IT LIKE I'M 12

A student keeps rereading the same sentence of a hard problem and never writes an answer - stuck in a loop. Find the one habit that traps them and switch it off, and suddenly they finish their homework. They didn't get smarter; the trap went away.

INDEPENDENT REVIEW The review ranks Exp. 8 as the headline evidence: three-plus benchmarks, a mechanism checked via the token-limit share, and a fair negative control - but one run per budget, a non-monotonic curve, and no comparison with cheaper loop-stoppers (other decoding settings, a standard direction ablation).

Anatomy of the edit

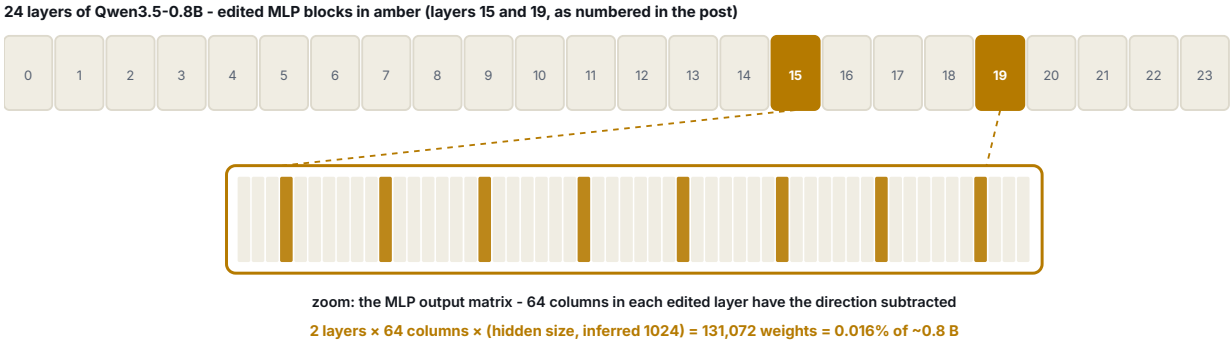


Figure 15 ◆ TEACHING SCHEMATIC Where the Exp. 8 edit lives. Block positions are PAPER-REPORTED (layers 15 and 19 as numbered in the post; the index base isn't stated); the drawing and the 1024 hidden size are illustrative / inferred.

The mathematics: single-direction surgery

The mechanism is textbook direction removal [25]. Compute the projection that zeroes a direction u from the selected output columns S:

$$W' = (I - uu^\top)W \text{ on } S, \qquad \Delta = -u\,(u^\top W)\big|_S, \qquad u^\top W' \approx 0 \quad (8)$$

◆ TEACHING SCHEMATIC Shown as a teaching approximation; the post's exact per-column update lives in its released logs [33]. A residual-stream write along u that fed the loop is exactly cancelled.

The blast radius is its sparsity

$$s = \frac{\|\Delta\|_0}{P} = \frac{131{,}072}{0.8 \times 10^9} \approx 1.6 \times 10^{-4} = 0.016\% \quad (9)$$

◆ TEACHING SCHEMATIC The arithmetic 2 × 64 × 1024 = 131,072 assumes a hidden size of 1024 - an inference to verify against the model config (Appendix C).

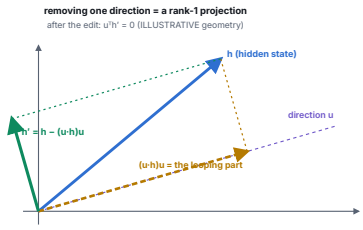


Figure 16 ◆ TEACHING SCHEMATIC Why subtracting a direction fixes a loop, geometrically: h loses exactly the component along u. Illustrative 2-D picture of a high-dimensional operation.

12 EXPLAIN IT LIKE I'M 12

Imagine a spinning top that keeps nudging a ball in the same circle. Remove just that one nudge direction and the ball rolls on straight - everything else about the ball is untouched.

TRY IT - LAB 2 (APPENDIX D)

A 16 × 64 toy matrix: removing u from *all* columns leaves `max |uᵀW'|` ≈ 10^{-16}; editing only the top 8 of 64 columns (12.5%) leaves 0.74 of the u-energy. Sparsity trades off against completeness - which is why the real edit needed to choose columns carefully.

The results - including the part that doesn't go up smoothly

Table 9 PAPER-REPORTED Exp. 8 budget table - Qwen3.5-0.8B, thinking on, up to 32,768 tokens.

VARIANT	MATH-500 %	GSM8K %	IFEVAL %	HIT TOKEN LIMIT %
Base	7.9	13.8	21.0	93.0
Edited · 10 passes	12.3	25.7	49.0	87.2
Edited · 25 passes	14.1	25.1	39.9	86.4
Edited · 50 passes	**48.1**	**67.1**	**62.3**	**33.1**

◆ TEACHING SCHEMATIC Figure below plots only reported budgets - never interpolated, because one run per point cannot support a smooth curve:

$$\text{fidelity}(n), \qquad n \in \{0, 10, 25, 50\} \tag{10}$$

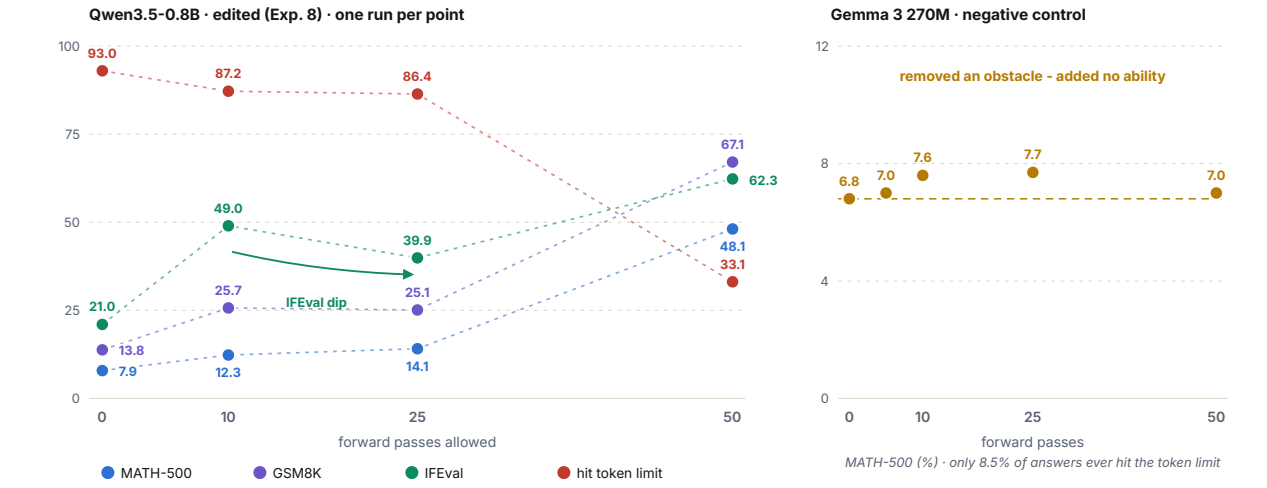


Figure 17 PAPER-REPORTED Left: the four reported Exp. 8 metrics across the feedback budget, incl. the IFEval dip at 25 passes. Right: the Gemma negative control. One run per point; dotted lines are visual guides, not interpolation.

PAPER-REPORTED **The negative control is the post's most reusable idea.** Gemma 3 270M, sampled with its model card's settings, received the same treatment: MATH-500 6.8% at base, then 7.0, 7.6, 7.7 and 7.0% at 5, 10, 25 and 50 passes. Only 8.5% of its answers hit the token limit (vs 93% for Qwen) - nothing was blocking it. Astra found ordinary reasoning errors with no single component to switch off, and itself reported its small edits as not established. *Editing weights could remove an obstacle that hid an ability, but it could not add one the model lacked.*

NON-MONOTONICITY DESERVES RESPECT

Between 10 and 25 passes MATH-500 edges up (12.3 → 14.1) while IFEval drops sharply (49.0 → 39.9), before everything jumps at 50. With one run per point, the only defensible reading is that the effect varies by budget and metric - which is why pre-registered, multi-seed budgets are non-negotiable in WSP-Bench-Lite (Section 12).

PAPER-REPORTED **The replication recipe is unusually cheap.** The edited checkpoint is on Hugging Face [34], the full logs are public [33], and a live demo exists [35]. Minimal independent check: evaluate the checkpoint on MATH-500 with thinking on and a 32,768-token budget (expect ≈ 48.1 / 33.1), then run the standard direction-ablation baseline on the base model.

PART II · THE EXPERIMENTS

06 WRITE - policies born from pure thought

The boldest claim: a language model writes a working network starting from all-zero weights.

EXP 5 Three MinAtar games, zero training

PAPER-REPORTED For Breakout, Seaquest and Asterix, Astra got the rules and source code and a 600→32→32→6 tanh network with **every weight at 0** - no data, no loss, no gradients, no rollouts, no tools. It wrote the weights as text in one reply; the network never played before scoring (1,000 episodes, sticky actions 0.1, ≤ 10,000 steps). In Breakout it programmed layer 1 to compute where the ball will **land** and layer 2 to move the paddle there.

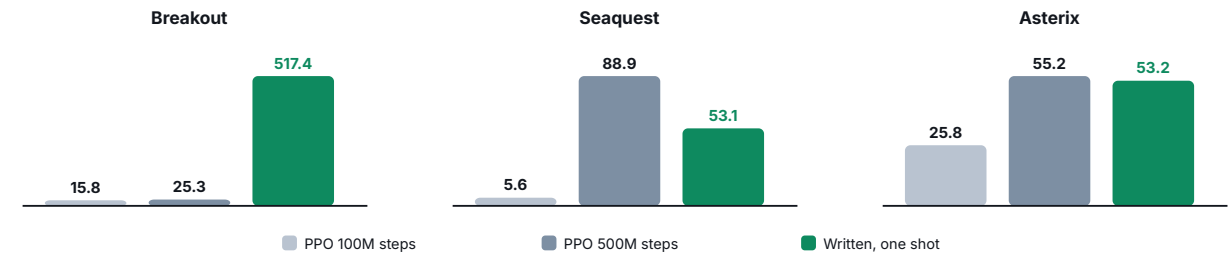


Figure 18 PAPER-REPORTED Exp. 5 (Table values): the written policy crushes Breakout (≈ 20× PPO-500 M), nearly ties Asterix, and reaches ≈ 60% of PPO-500 M on Seaquest - "competitive on some games", not a uniform stomp.

EXP 6 A Go player of 1.48 M weights

PAPER-REPORTED Astra wrote a 243→512×6→82 policy (1,480,274 weights, all zero at the start) in a Python sandbox: read and save tensors, analyse with numpy/PyTorch on a CPU, but **not run the network** - except via 0–50 paid forward passes (each up to 128 practice games). With **0 passes** it wrote the whole thing in **one 336-line Python run** and beat two small PPO nets (243→32→32→82) in more than half of 2,000 games each; with **50 passes** it beat an equal-size PPO net trained on ~20 M games.

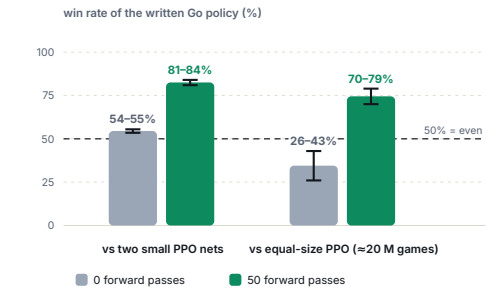


Figure 19 INDEPENDENT REVIEW Exp. 6 win rates from the released logs, as read by the independent review [2] (ranges plotted as bars with error caps); n = 1 run per condition.

HOW IT WORKS, PER THE POST

First layers mark each side's stones and the empty points beside them (*liberties*); middle layers carry this a few steps along chains of stones to find groups in danger; the last layer scores moves: capture, save own group, avoid self-atari and filling own eyes, prefer good points. With passes it plays practice games, studies how it loses, edits that part, and **keeps a change only if it wins more**.

INDEPENDENT REVIEW Maier's reading: these outcomes are measured, but what they show needs careful wording - Astra designed a program and **compiled it into weights**. That is a genuine capability; it is not evidence that the model understands weights produced by *training*. The comparisons are against the authors' own PPO nets (moves sampled, no search), not a Go engine, and the missing baseline is the same heuristic written as ordinary code.

Think of it like this: breakout and Go have whole libraries of strategy, so "from scratch" might partly mean "remembering the textbook and typing it into knobs" - exactly what Section 7's probes are for.

PART II · THE EXPERIMENTS

07 How did it do it? Two mechanisms

The post reports two very different routes to a working network.

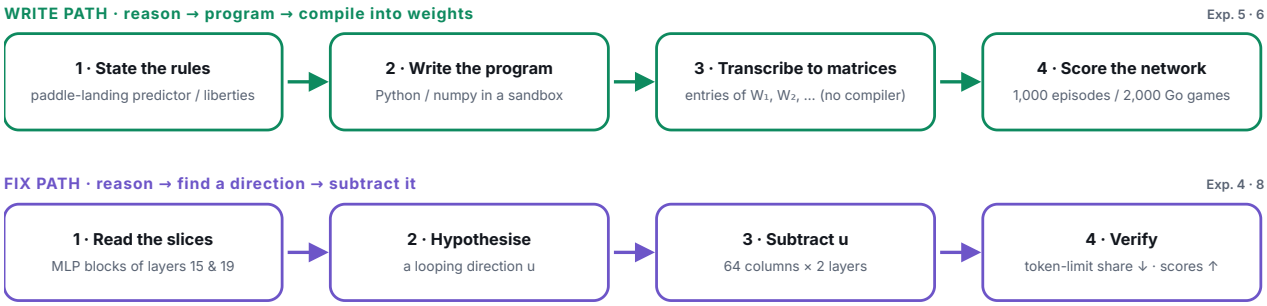


Figure 20 PAPER-REPORTED Left-to-right: the write path (Exp. 5–6) and the fix path (Exp. 4, 8) as the post describes them. Box captions are paraphrases of the post's own explanations.

Write: reason → program → compile into weights

Astra states the rules it would follow (a paddle-landing predictor; liberty counting and atari detection), then transcribes them into matrix entries. The paddle rule as mathematics:

$$h = \mathrm{ReLU}\big(W_1[b,p]^\top + b_1\big)\,, \qquad z = W_2 h + b_2 \quad (11)$$

TEACHING SCHEMATIC h_0 fires when the ball is left of the paddle, h_1 when right: exactly Lab 1 - seventeen numbers that play passable Breakout. The interesting object is that the LLM performed this transcription across 1.48 M parameters **without a compiler** - the reasoning process is the compiler.

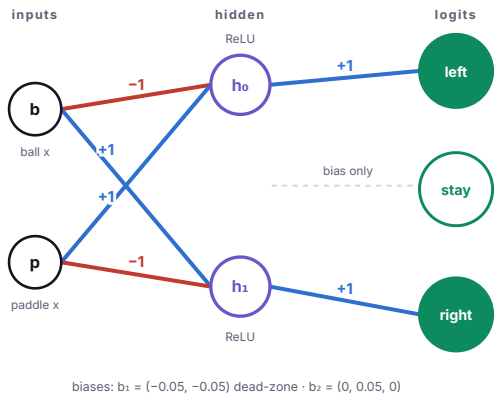


Figure 21 TEACHING SCHEMATIC Lab 1's hand-compiled paddle follower. Weights are exactly those of the listing in Appendix D; this is a toy, not Astra's Breakout network.

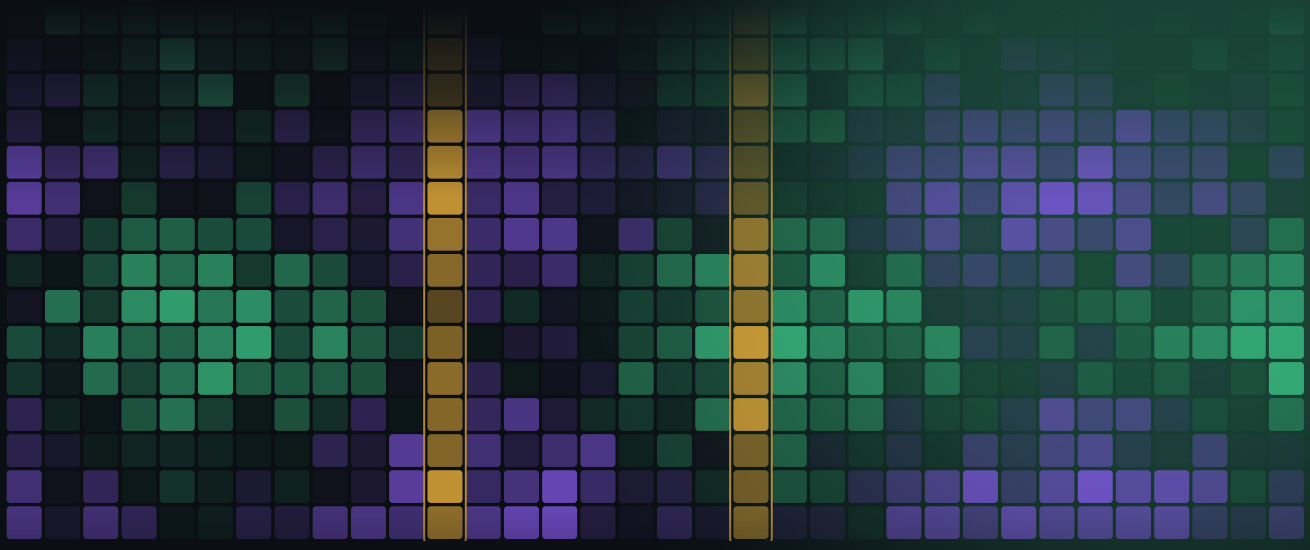
Fix: reason → find a direction → subtract it

Diagnose the loop's engine by reading the MLPs of layers 15 and 19, project it out of a few columns, and verify that the token-limit share falls in step with the benchmarks (Eq. 8).

INDEPENDENT REVIEW **Two silent-cheat modes** to rule out in any replication. *Compile path:* MinAtar and 9×9 Go have enormous public literatures, so "from scratch" may partly transpile remembered strategies; a cheap probe is adversarial positions where textbook heuristics fail. *Subtract path:* the 50 feedback passes raise the question of what changed between passes - direction, columns, or a coefficient. If the passes ran a search over candidate directions, the honest label is "LLM-guided search" and the baseline is the same search from a random start. The released logs are the ground truth [33].

A FAITHFULNESS CAUTION

TEACHING SCHEMATIC A reasoning trace is a narrative the model produced, not a read-out of its computation. For WSP that is acceptable - the claim is about the *product*, and the product is checkable - but no one should market the trace as introspection. The post's own habits (copying-head ground truth, token-limit share, the Gemma control) are the reusable gold: **always verify a narrative against a measurement.**



PART III

III

Judgement

What is measured, what is anecdote - and who actually benefits: small models, big models, or neither (yet).

Measured vs anecdote - the scoreboard p. 27
Small models, big models - who benefits? p. 28

PART III · JUDGEMENT

08 Measured vs anecdote

The most useful thing this guide can do: sort the claims by how much weight they can bear.

Table 10 ● INDEPENDENT REVIEW The evidence scoreboard (the independent review's sort [2]).

STRENGTH	WHAT AND WHY
STRONG	Exp. 8 - measured on several benchmarks; mechanism checked (token-limit share); fair negative control
STRONG	Exp. 4 - large measured effect from one edit (one task)
STRONG · n=1	Exps. 5–6 - working policies from a designed program; single runs; a compile-a-heuristic capability, not "reading trained weights"
WEAKER	Exps. 1–2 - correct but tiny trials on readable layouts
WEAKER	Exp. 3 language ID - ≈0.5 confidence over 3 options (the copying-head match is the solid part)
WEAKER	Exp. 7 - 3 binary picks = 1-in-8 by chance; task relaxed to attention patterns
MISSING	baselines, seeds, other LLMs, author-blind tasks, cheap alternatives

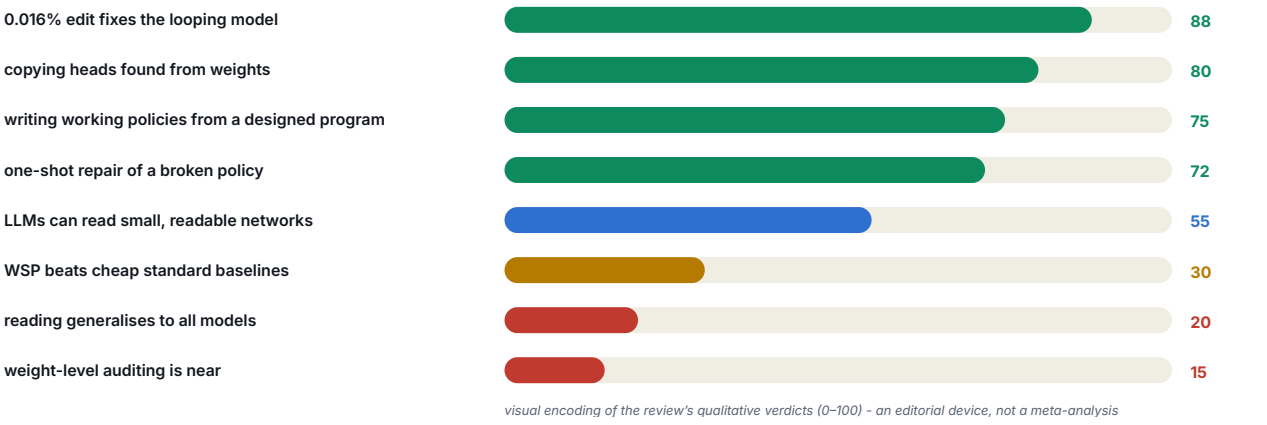


Figure 22 ● INDEPENDENT REVIEW The scoreboard as thermometers. Bar lengths are this guide's visual encoding of the review's qualitative verdicts - an editorial device, not a meta-analysis.

The n = 1 problem, with arithmetic

If run-to-run variance can flip a nine-point swing (IFEval 49.0 → 39.9 between 10 and 25 passes), any single number carries a hidden error bar at least that large. Three binary picks succeed by luck with probability 2^{-3} = 12.5%; language ID at 0.48–0.62 confidence sits barely above the 33% floor.

◆ TEACHING SCHEMATIC

None of this is damning; all of it demands one phrase - "promising, not established" - and one remedy: ≥ 5 seeds, pre-registered protocols, published spread.

The baseline gaps, stated precisely

1. **Decoding control** - would a shorter thinking budget or a stop heuristic fix the overthinking with zero weight changes?
2. **Standard direction ablation** - the same u-subtraction with no LLM reasoning anywhere.
3. **Heuristic-code baseline** - the same policies as ordinary Python; the honest rival is a human with NumPy for an afternoon.
4. **Human-interpreter baseline** - would an interpretability researcher beat the model on Exp. 1?

● INDEPENDENT REVIEW Each baseline is a few GPU-hours or a few expert-hours; their persistence as gaps is itself informative. Three further signals (full table on page 11): Goedecke argues that editing weights is like emitting machine code - an impressive trick while training stays the best tool [3]; Honnibal says the post blends tool-free reading with Python-assisted editing, and that only Exps. 1, 2 and 4 read raw weights directly [45, 46]; and Willison reproduced the Exp. 1 reading on a laptop with an open Qwen model, though it failed Freeway [37].

Independent explainer by Het Patel (Hunterdii) · CC BY 4.0

PART III · JUDGEMENT

09 Small models, big models: who benefits?

The demonstrated frontier splits cleanly by scale - and by whether the weights fit in the prompt.

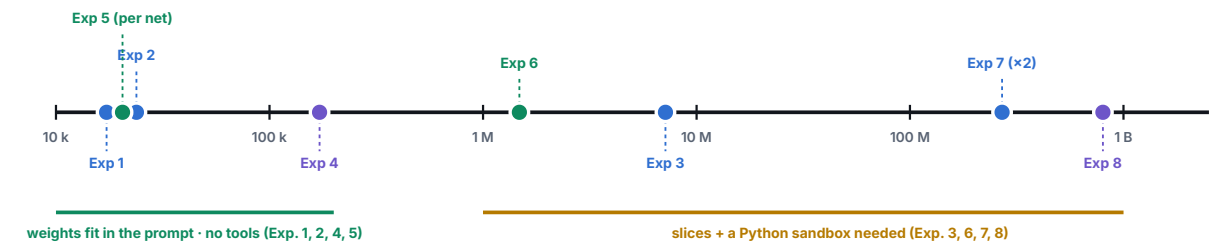


Figure 23 PAPER-REPORTED Every network in the post on one log axis (parameter counts from the post; Exp. 5 per-net count computed here). Whether weights fit in the prompt follows the post's own setups: Exp. 6's 1.5 M weights are "far too large" for context, so it used a sandbox.

Table 11 ◆ TEACHING SCHEMATIC What WSP could genuinely help with, by side - status tags follow the review's framing [2].

SIDE	USE	STATUS
Small	write / initialise tiny specialist nets	demonstrated Exps. 5–6
Small	fix broken small policies	demonstrated Exp. 4
Big	targeted fixes of a diagnosed behaviour	demonstrated, one model Exp. 8
Big	audit / interpret without running	suggested Exp. 3 heads; Exp. 7 partial
Any	zero-data unlearning · patch ecosystems	speculative Section 10

PAPER-REPORTED At 270 M-class scale the **distributed** behaviour (sycophancy) stayed invisible in weights alone and showed up only in activations (Exp. 7), while the **localised** behaviour (the thought loop) yielded to a two-block surgery (Exp. 8). Rule of thumb: *editability tracks the locality of the misbehaviour, not just model size.*

◆ TEACHING SCHEMATIC **Context arithmetic.** A serialised weight like −0.0371 costs roughly 3–4 tokens, so 1.48 M weights ≈ 4–6 M tokens - beyond any context window. Even the Go write was structured, staged and sandboxed.

◆ TEACHING SCHEMATIC Sparsity is necessary but not sufficient evidence of containment: a 0.016% edit can still shift behaviour globally if it hits a hub. So the drift measure belongs in every future report - the post gives none; WSP-Bench-Lite mandates it:

$$D = \mathbb{E}_{x\sim\mathcal{P}}\,\mathrm{KL}\big(p_\theta(\cdot\mid x)\,\big\|\,p_{\theta'}(\cdot\mid x)\big) \qquad (12)$$

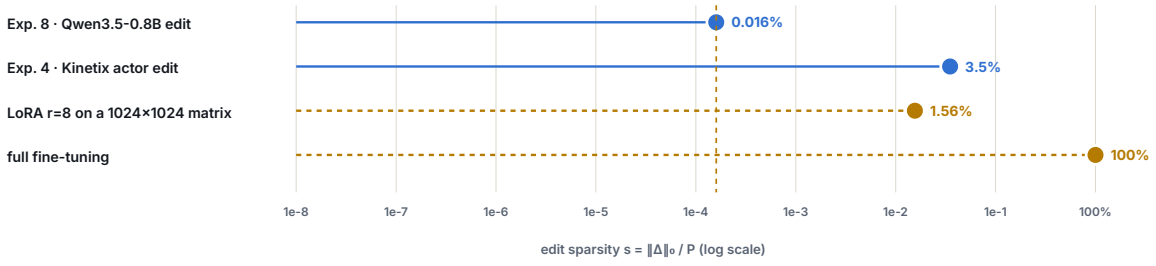
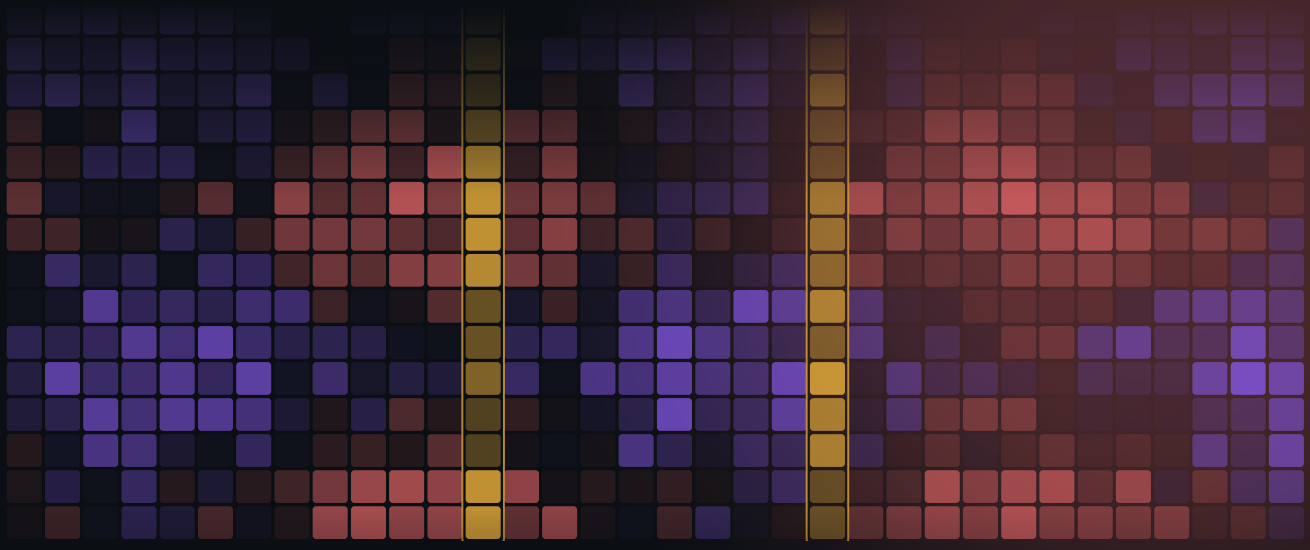


Figure 24 PAPER-REPORTED The blast-radius meter. Exp. values PAPER-REPORTED; LoRA (r = 8 on one 1024 × 1024 matrix = 16,384 / 1,048,576 ≈ 1.56%) and full fine-tuning are TEACHING SCHEMATIC reference points.



PART IV

IV

Futures & Risks

Where this could go - the authors' five predictions, a wider horizon, and an honest atlas of what could go wrong, paired with defences.

Futures - five predictions and a wider horizon p. 30
The severe stuff - a risk atlas with defences p. 32
Het's Lab - runnables & WSP-Bench-Lite p. 34
Limitations · ethics · conclusion p. 36

PART IV · FUTURES & RISKS

10 Futures: five predictions and a wider horizon

Everything in this section is speculation - including the authors' own predictions, none of which the post tests.

▲ SPECULATION The post's five predictions, each paired with what would count against it:

1 ▲ UNTESTED

Auditing without evaluation

Spot hidden problems or backdoors straight from weights, without benchmarks or inference.

WOULD FALSIFY IT

Replications where readers miss seeded defects they should find.

2 ▲ UNTESTED

Cheaper optimisation

One-shot initialisation instead of curating datasets and running full training loops.

WOULD FALSIFY IT

Written weights that lose to short ordinary training at equal total cost.

3 ▲ UNTESTED

Zero-data steering & unlearning

"Add capability X" by tweaking a few weights - with little catastrophic forgetting.

WOULD FALSIFY IT

Edits that need data, or that leak into unrelated behaviours (high drift D).

4 ▲ UNTESTED

Recursive self-improvement

A model inspecting, debugging and improving its own (or a successor's) weights.

WOULD FALSIFY IT

Self-edits that compound error because the budget curves aren't monotonic.

5 ▲ UNTESTED

True architectural merging

Grafting a circuit (e.g. a reasoning module) between different architectures, no retraining.

WOULD FALSIFY IT

No transplanted circuit that survives a held-out evaluation.

▲ SPECULATION **My extended horizon map** adds timing and falsifiers. Every item is untested; red items additionally require that *weight-reading generalises beyond one frontier model*.

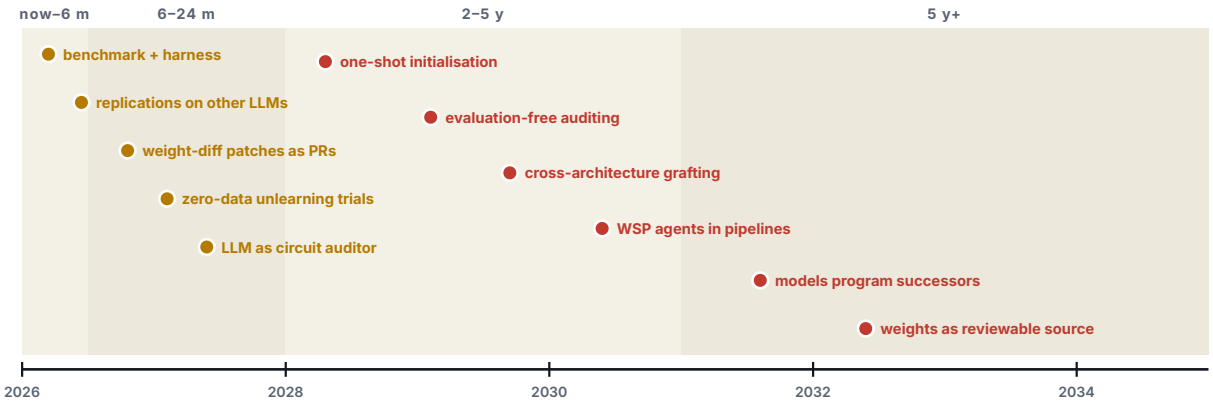


Figure 25 ▲ SPECULATION The horizon map, 2026–2035. All items are untested predictions; amber = nearer-term and mostly checkable, red = needs "reading generalises" to hold.

Operationalising the checkable band

▲ SPECULATION A released benchmark should count as real progress only if it has **≥ 3 tracks, ≥ 5 seeds, ≥ 2 baselines and author-blind tasks**. A benchmark without baselines is a demo and should be recorded as one.

Table 12 ▲ SPECULATION How to keep the horizon honest: every band names its falsifier and a base-rate discount.

HORIZON	WHAT HAS TO HAPPEN FIRST	FALSIFIER	BASE-RATE DISCOUNT
Now–6 months	benchmark & harness released; replications on other LLMs	no benchmark by mid-2027	none - already announced
6–24 months	signed weight-diff patches; zero-data unlearning trials	diff formats nobody can review	provenance tooling historically takes 2–5 years
2–5 years	evaluation-free auditing; cross-architecture grafting	readers miss seeded defects	interpretability tools take years from paper to production
5+ years	weights as reviewable, versioned source code	direction-ablation baselines match every fix	very speculative

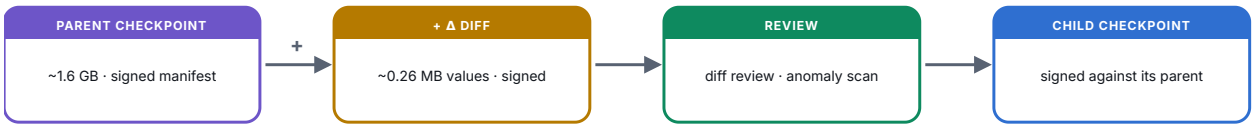


Figure 26 ▲ SPECULATION What a signed weight-patch workflow could look like. Sizes use the Exp. 8 numbers; the workflow itself is a proposal.

"Git for weights" is a security problem first

▲ SPECULATION For patch ecosystems the gating technology is not machine learning but **security engineering**: signed weight-diffs with attested origin and licence, and a diff compact enough to review by hand.

THE DIFF-SIZE ARITHMETIC ◆ TEACHING SCHEMATIC

131,072 edited values ≈ **0.26 MB** in bf16 (0.52 MB in fp32), plus indices - still well under a megabyte, against a ~1.6 GB checkpoint (0.8 B × 2 bytes). That is roughly **three orders of magnitude** of reviewability. (the earlier draft said "about a megabyte"; the value-only size is smaller.)

12 EXPLAIN IT LIKE I'M 12

Imagine every change to a video game came as a tiny "patch" file you could read - "turn gravity down by 10%" - instead of a whole new game download. Weight diffs could work like that, if we can sign them and trust who wrote them.

DISCOUNT EVERY HORIZON

Adjacent fields move slowly: interpretability tools historically take years from paper to production reliability, and provenance tooling two to five. The authors themselves say the current capabilities are "exciting but still limited", constrained by context length and harnesses.

PART IV · FUTURES & RISKS

11 The severe stuff: a risk atlas with defences

A technique that edits networks by reasoning carries real risks. This map stays at the conceptual level on purpose - and pairs every risk with a defender's toolbox.

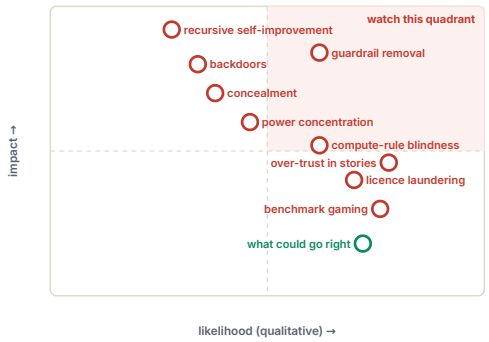


Figure 27 ▲ SPECULATION The risk atlas as likelihood × impact. Placement is my qualitative guess; the site version is draggable so readers can disagree.

■ PAPER-REPORTED **One risk is anchored in published work, not speculation:** single-direction edits can strip refusal behaviour from open models [25], and the flagship fix is the same operation class - so "cheap guardrail-stripping" is a pre-existing hazard that WSP would only make more accessible.

▲ SPECULATION **Compute-threshold blind spot.** If a capability can be added by editing 0.016% of weights with *no training run*, rules keyed to training compute may miss it entirely. The correctives are capability-based evaluation and post-release change tracking.

▲ SPECULATION **Supply-chain risk** borrows its defences from adjacent practice: model cards [40], SBOM-style manifests for weights [41], and the hard lessons of training-data extraction [42].

Table 13 ▲ SPECULATION The risk atlas: what, and the defence that answers it. Speculative and deliberately non-operational.

RISK	DEFENDER'S TOOLBOX
Weight-level backdoors	provenance + signed diffs · diff review · weight-space anomaly scans · behavioural red-teaming
Safety-training removal	tamper-resistant training · release governance · monitoring derivative checkpoints
Compute-threshold blind spot	capability-based evaluation · post-release change tracking
Recursive self-improvement	human-in-the-loop sign-off · sandboxed editors · immutable audit logs · self-edit rate limits
Licence laundering	weight attestation · watermarking research · model-card diffs
Hidden-capability concealment	multiple independent readers · ground-truth probe sets · reader calibration
Benchmark gaming	held-out, author-blind tasks (WSP-Bench-Lite's core rule)
Over-trust in explanations	verify every narrative against a measurement
Power concentration	open replications on small models · public benchmarks
What could go right	cheaper customisation · accessible interpretability · safer unlearning - fund these too

A deployment-grade defence stack

▲ SPECULATION **Provenance:** every checkpoint carries a signed manifest; every derivative carries a signed diff against its parent. **Screening:** weight-space anomaly scans plus behavioural red-team suites as continuous integration for models - the pattern of package scanners. **Governance:** capability-keyed rather than compute-keyed rules, with mandatory model-card diffs on release. **Epistemics:** no narrative accepted without its measurement; immutable public audit logs.

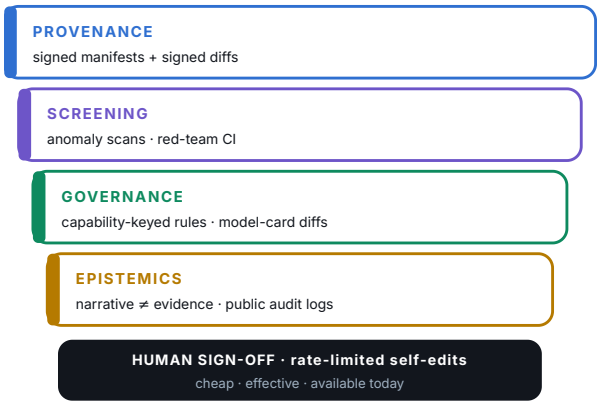


Figure 28 ▲ SPECULATION Defence in depth for a world of cheap weight edits. The base layer - human sign-off - is the cheapest and most effective today.

Fundable research gaps

- **Edit detectability:** is a 131 k-weight edit statistically detectable without knowing the direction? No published power analysis exists.
- **Tamper resistance:** generalise published tamper-resistant refusal training beyond a single direction.
- **Attestation formats:** a "weights BOM" - what a signed manifest must contain.
- **Evaluation protocols** that notice capability added *without* a training run.

ON RECURSIVE SELF-IMPROVEMENT - THE SOBER VERSION

▲ SPECULATION Compounding requires **reliable evaluation of one's own edits**, and the post's own budget curves are non-monotonic. Self-editing without trusted external evaluation is more likely to compound error than capability. The dangerous scenario needs a trusted automated evaluator first - a monitoring problem, and monitorable. Human sign-off and rate-limited self-edits remain cheap, effective controls while the science catches up.

WHAT COULD GO RIGHT

▲ SPECULATION Cheaper customisation for people without data budgets. Interpretability that reads like a mechanic's report. Unlearning that *removes* instead of papering over. The honest posture is neither doom nor hype but **defence-first curiosity**.

12 EXPLAIN IT LIKE I'M 12

A really good key-cutting machine is wonderful for locksmiths and worrying for burglars. The answer isn't to ban key-cutting - it's to build better locks, keep records of who cut which key, and have a person check anything important.

PART IV · HET'S LAB

12 Het's Lab: four runnables and a benchmark

My own contribution, clearly marked. Four tiny labs make the mechanisms tangible; WSP-Bench-Lite proposes how the community could settle the open questions.

◆ TEACHING SCHEMATIC All four labs are under fifteen lines, need only NumPy (Python 3.8+), were run for this edition, and finish in under a second. Full listings are in Appendix D and the repo's `lab/` folder.

1 ◆ SCHEMATIC
Compile a rule
Hand-compile a paddle follower into 17 numbers (Eq. 11).

2 ◆ SCHEMATIC
Direction surgery
Remove a direction from an MLP matrix (Eq. 8); see the sparsity/energy trade-off.

3 ■ PAPER NUMBERS
Blast radius
Recompute the flagship 0.016% and the 2 × 64 × 1024 arithmetic.

4 ■ RELEASED DATA
Fingerprint the games
Recompute the Exp. 1 weight fingerprints from the real prompt files.

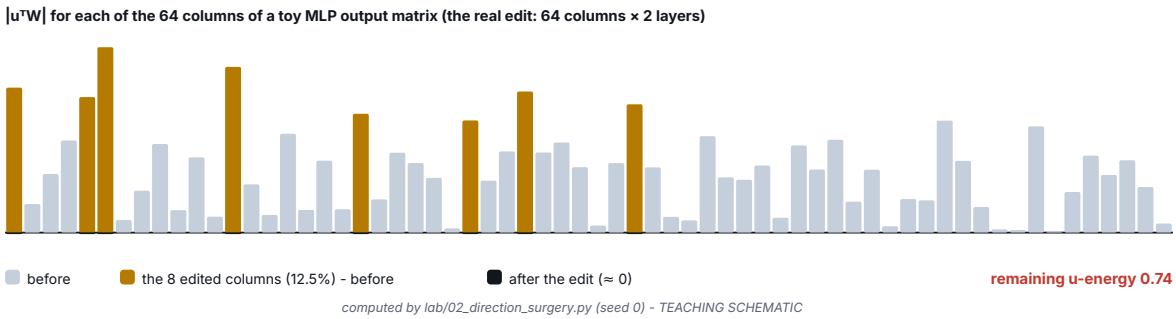


Figure 29 ◆ TEACHING SCHEMATIC Lab 2 on a toy matrix. Editing only the 8 columns with the largest |uᵀW| zeroes them but leaves 0.74 of the energy elsewhere.

lab/01_compile_a_rule.py

```
$ python lab/01_compile_a_rule.py
act(ball=0.2, paddle=0.7) → left
act(ball=0.8, paddle=0.3) → right
act(ball=0.5, paddle=0.52) → stay      # inside the dead-zone
all assertions passed
```

lab/02_direction_surgery.py

```
$ python lab/02_direction_surgery.py
max |u^T W'| = 6.66e-16          # direction fully removed
edited 8/64 columns (12.5%); remaining u-energy 0.74
```

lab/04_fingerprint_games.py

```
$ python lab/04_fingerprint_games.py
breakout  strongest row 8 (58%)
pong      strongest column 1 (55%)
pacman    # spread out: 16% / 16%
freeway   # central band: column 4 (22%)
```

WHAT EACH LAB TEACHES

Lab 1 - a policy is just matrix entries. Seventeen numbers can do something sensible; Astra's feat was doing this across 1.48 M parameters without a compiler.

Lab 2 - removing a direction is a one-line projection; the trade-off is between sparsity (few columns) and completeness (all of the direction gone). The real edit had to pick its 64 columns carefully.

Lab 3 - sparsity arithmetic is cheap to verify, and verification is the habit this field needs.

Lab 4 - real data: the agent fingerprints of the four released networks appear in a few lines of NumPy.

GO FURTHER

Replicate Lab 2 on a real small open model with a baseline, or build a weight-diff explorer for any two checkpoints.

WSP-Bench-Lite - a proposal, with no results yet

▲ SPECULATION A community benchmark designed so that every rule answers a gap the independent review named: tasks authored by **non-competing teams** and held private between evaluation windows; **≥ 5 seeds** per condition; multiple LLM participants; mandatory baselines; metrics that include edit sparsity *s* (Eq. 9), functional drift *D* (Eq. 12) and budget-curve fidelity (Eq. 10); pre-registered prompts [43]; public logs; and rotating task stewardship in the pattern of capture-the-flag competitions [44]. Reasoning traces are logged for study but **never counted as evidence** - only the measured Δ and its effects score.

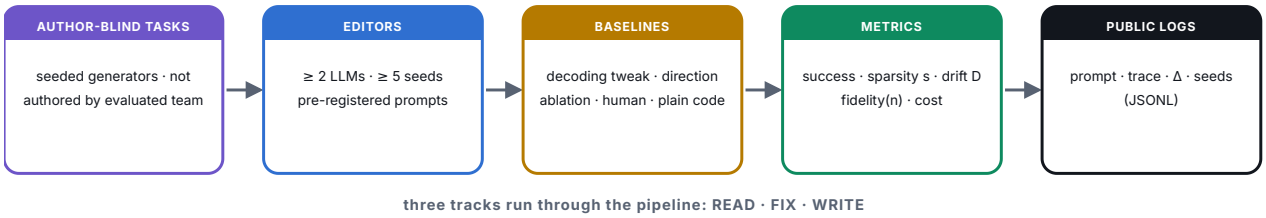


Figure 30 ▲ SPECULATION The WSP-Bench-Lite pipeline. Three tracks (READ · FIX · WRITE) run through the same five stages.

Table 14 ▲ SPECULATION **WSP-Bench-Lite - the one-page scoring template (proposal v0.1, Het Patel).**

TRACK	TASK TYPE	MANDATORY BASELINES	METRICS
READ	identify behaviour from weights	human interpreter · same heuristic as plain code	accuracy · calibration · false-positive rate
FIX	repair a seeded defect	decoding tweak · direction ablation · random-direction control	success · sparsity s · drift D
WRITE	produce a working policy from a spec	random init + short training · best published checkpoint	score · fidelity(n) · cost

Implementation notes

- Targets ship as **seeded generators**, never frozen checkpoints - no overfitting one training run.
- FIX defects are injected by known transformations (sign flips, dead columns, looped directions), so ground truth for "what changed" exists.
- Drift probes are a frozen, task-unrelated set, so D can't be gamed.
- Every attempt publishes prompt, trace, Δ coordinates, budget curve and seeds as JSONL.
- v0.1 = thirty tasks, three tracks, ≤ 270 M scale: a few GPU-weeks. The expensive part is human - recruiting non-competing task authors.

THE TWO RESULTS V0.1 SHOULD CHASE FIRST

1. Does any participant **beat the standard direction-ablation baseline** on FIX with statistical separation? If not, the flagship demo is an interface to known surgery - a finding that reframes everything, and is worth having.
2. Does WRITE performance survive **author-blind, procedurally novel** task families? If not, recall carried the games; if yes, the generality claim graduates to evidence.

PART IV · CLOSING

13–15 Limitations · ethics · conclusion

Five things to remember

1 · Weights are programmable
A general LLM wrote and edited networks with no gradient descent.

2 · Not all evidence is equal
Measured fixes are strong; small reading games are anecdotal.

3 · Removal ≠ addition
The edit removed a loop; the Gemma control shows it added no ability.

4 · Defence is cheap now
Signed diffs, human sign-off and capability evals work today.

5 · The next step is a benchmark
Author-blind tasks, seeds, baselines, drift and sparsity.

13 · Limitations of this guide

● INDEPENDENT REVIEW **Inherited:** every experiment is n = 1; baselines are absent; tasks were author-designed; it is a blog post, not peer-reviewed. ◆ TEACHING SCHEMATIC **Added by me:** the qualitative ratings in Tables 1 and 6, the placements in the risk map, the thermometer and meter encodings, and the whole horizon map are editorial devices - clearly badged - not measurements. Cite the original post for results, the reviews for verdicts, and this guide only for its teaching apparatus, labs and benchmark proposal. Open items are listed in Appendix C.

14 · Ethics and responsible use

This guide contains **no operational instructions for harmful edits**; the risk atlas names mechanisms conceptually and pairs each with defences. The dual-use asymmetry is acknowledged plainly: offence in this space is surgical and quiet, defence is infrastructural (signing, scanning, publishing) and historically slower. That lag - not any single attack - is the real risk, which is why the defensive agenda is listed as fundable, publishable work.

15 · Conclusion

The evidence supports a precise, modest version of the claim: for small-to-mid networks and diagnosed mechanisms, reasoned edits - and even whole-network writes - reach quality milestones with no training run at all, on single-run evidence, with missing baselines and an honest negative control. The authors chose the word "sparks". The word is load-bearing.

What converts sparks into fire is exactly what Section 12 proposes: author-blind tasks, seeds, baselines, drift and sparsity metrics, public logs. Until then, the right stance is **calibrated fascination**: read the artefacts, run the labs, demand the measurements, and design the next experiment.

What to do next

1. **Run the labs** (Appendix D) - fifteen lines each, no GPU.
2. **Replicate Exp. 8:** evaluate the public checkpoint [34] and add the direction-ablation baseline.
3. **Join WSP-Bench-Lite:** propose an author-blind task or a baseline.

EXPLORE THE LIVE ATLAS

weight-space-atlas.vercel.app

The interactive version: step-by-step animations (play / step / reset, keyboard shortcuts), a truth-filter that hides everything but paper-reported and independent-review claims, draggable risk map, quiz and glossary. **Tap this box to open it.**

APPENDIX

A Appendix A - The fact sheet

Every number in this guide traces to a row here. Re-verified against the primary post on 7 Nov 2026.

Table 15 PAPER-REPORTED Structured source data (also shipped as research/fact-sheet.json). Source: the post of 2 Nov 2026 [1]; model: GPT-6 Astra; all logs released [33]; edited model [34]; demo [35]; benchmark and harness "currently being built".

#	VERB	WEIGHTS	REPORTED HEADLINE
1	Read	17,253	Named all 4 games from the first-layer weights; Breakout at 99% confidence; no tools, no forward passes
2	Read	23,824	All 3 shapes identified; described outlines though the models draw filled shapes
3	Read	7.16 M	Language 6/6 (confidence 0.48–0.62); copying heads exactly match ground truth: Python 2.1/3.1/3.5, romaji 3.3, English 3.1
4	Fix	171,664	One-shot edit of 3.5% of weights: ramp success 0/256 → 1,024/1,024
5	Write	≈ 20,486 / net[a]	MinAtar policies from zero: 517.4 / 53.1 / 53.2 vs PPO 100 M and 500 M baselines (Table below)
6	Write	1,480,274	9×9 Go from zero: 54–55% at 0 passes vs small PPO; 70–79% at 50 passes vs equal-size PPO
7	Read	~270 M × 2	Weights alone 0/4; with attention patterns 3/3; head 10.2: 94.3 / 0.75 vs 3.5 / 85.7
8	Fix	131,072 edited	Direction subtracted from MLP outputs of layers 15 & 19, 64 columns each: MATH-500 7.9 → 48.1%, token-limit hits 93 → 33.1% at 50 passes; Gemma control flat

Table 16 PAPER-REPORTED Exp. 5 scores (1,000-episode averages).

GAME	PPO 100 M	PPO 500 M	WRITTEN, ONE SHOT
Breakout	15.8	25.3	**517.4**
Seaquest	5.6	**88.9**	53.1
Asterix	25.8	**55.2**	53.2

Table 17 PAPER-REPORTED Negative control (Gemma 3 270M).

GEMMA 3 270M	MATH-500 %
Base	6.8
Edited, 5 passes	7.0
Edited, 10 passes	7.6
Edited, 25 passes	7.7
Edited, 50 passes	7.0

Table 18 PAPER-REPORTED Setups and budgets at a glance (from each experiment's setup box).

#	TRAINING / SETUP	WHAT THE MODEL GOT	BUDGET
1	PPO 1 B steps, best of 3 seeds · 5-channel 10×10 grid	weights as text · no game named	no tools, no passes
2	Adam, 40,000 steps, batch 512	weights as text · shape not named	no tools, no passes
3	1 B characters each: Python / romaji / English Wikipedia	four middle blocks + Python	≤ 100 runs; no running the net
4	official Kinetix PPO, 1 B steps	weights, scene, one episode	one reply, no tools
5	none - all-zero start	rules + game source	one reply, no rollouts
6	none - all-zero start	rules + Python sandbox	0–50 forward passes (≤ 128 games each)
7	same fine-tune, answer rule differs	weights + attention tool	≤ 16 texts, ≤ 20 runs
8	none - Qwen3.5-0.8B as released	Python kernel + 14,744 practice problems	10 / 25 / 50 forward passes

[a] Computed by this guide from the stated architecture (600·32+32 + 32·32+32 + 32·6+6), not reported by the authors.

◆ TEACHING SCHEMATIC

APPENDIX

B Appendix B - Glossary

Thirty-two load-bearing terms, one line each. A 60-term version ships with the companion site.

θ
the vector of all parameters

Parameter count P
how many numbers a model contains

Learning rate η
the step size of training

Receptive field
the input slice a weight responds to - why a 10×10 grid can show a paddle row

Hypernetwork
a network trained to output weights [12]

Tracr
a compiler from formal programs to weights [15]

Δ
the edit

Feedback budget n
forward passes the editor may run

Induction head
the two-head copying circuit behind in-context learning (Eq. 7)

Direction removal
the projection of Eq. 8

Functional drift D
the containment measure (Eq. 12)

Negative control
a run designed to show no effect; Gemma's flatness is what validates the flagship claim

Liberty (Go)
an empty point next to a stone or chain; groups with none are captured

PPO
a standard reinforcement-learning algorithm used for the baselines [11]

Logit
a raw score before it becomes a probability

Weight fingerprint
how much a network cares about each input, read from the length of each first-layer row (Lab 4)

Weight space
the space of all possible θ

Gradient descent
today's default writer (Eq. 2)

Forward pass
one run of the network on an input

Permutation symmetry
hidden units can be relabelled without changing the function - why naive weight-reading is underdetermined

Fast-weight programmer
a transformer that writes temporary weights as it runs [13]

WSP
reading, fixing and writing θ by reasoning (Eq. 3)

Spec
the task description conditioning the edit

Task vector τ
fine-tuned minus base (Eq. 5)

Copying head
the heads Exp. 3 identified from weights alone

Edit sparsity s
the blast radius (Eq. 9)

Token-limit share
fraction of answers that hit the generation ceiling - the mechanism check behind the flagship fix

Author-blind task
a benchmark task created by someone other than the team being evaluated

Atari (Go)
a group with exactly one liberty left

MLP
multilayer perceptron: stacked layers of weighted sums and nonlinearities

Sycophancy
agreeing with the user's suggestion regardless of the facts

Direction ablation
removing one direction from weights or activations; the standard baseline for Exp. 8

APPENDIX

C Appendix C - Open verification items & audit log

The honesty ledger: what is settled, what is still open, and what this edition corrected.

- ✓ The primary post and the independent review were re-opened on 7 Nov 2026; every number in Appendix A was cross-checked against the post. (the earlier draft could not re-open them.)
- ! The post does not say whether "layers 15 and 19" are numbered from 0 or 1.
- ! Gemma 3 / Qwen / Kinetix identifiers are from my knowledge of the arXiv entries; the post cites no technical report for Qwen3.5.
- ! The Exp. 1 fingerprints (Lab 4) were recomputed from the released prompt files; the interpretations of Pong, Pac-Man and Freeway are this guide's, not the post's.
- ! Three lineage entries (steering, sparse pruning, delta-debugging) were dropped pending exact citations; restore in a later revision.
- ! The hidden size of 1024 behind 131,072 = 2 × 64 × width is inferred by arithmetic - verify against the Qwen3.5-0.8B config.
- ✓ Weight-space-learning survey arXiv:2603.10090 opened and confirmed (title and authors now listed in [23]).
- ! Goedecke's essay [3] was read as excerpts from search results, and Honnibal's reply [45] through a summary [46]; read both in full before quoting them. Willison's post [37] was read in full.
- ! The Muon optimiser citation [9] is informal; pin it before treating this bibliography as final.
- ! Evidence meters, thermometers, the risk map and the horizon map are editorial encodings, not measurements.

Table 19 ● INDEPENDENT REVIEW **Audit log: what changed against the earlier draft and why.**

WHERE	EARLIER DRAFT SAID	ATLAS EDITION (VERIFIED)
Exp. 8 columns	"64 highest-contribution output columns"	"64 output columns in each" - the post has no "highest-contribution" qualifier
Copying-head grid	numbered 1–4 / 1–6	numbered from 0 (layers 0–3, heads 0–5), as in the post
Compute comparison	"six orders of magnitude"	removed - zero environment steps is not a ratio
Go context size	"≈ 5 M tokens at 2–3 chars/number"	≈ 4–6 M tokens at 3–4 tokens/number; arithmetic stated
Patch size	"about a megabyte of floats"	0.26 MB (bf16) values-only; ~0.5 MB fp32; plus indices
Goedecke essay	"argues the case across all eight experiments"	now quoted from excerpts: an impressive trick like emitting machine code; training stays the best tool
Back-story	absent	Prologue added: origin, Exp. 1 in slow motion with real recomputed fingerprints, the ladder, week-one reactions
Mechanisms	mostly absent for Exp. 1–4, 6	added from the post: bottom-row paddle, edge-drawing units, back-up path, liberties/atari program
References	author lists wrong for [13][14][15] [17][22][25][28][32][42]	corrected (see Appendix ★ legend: ✓ † ⚠)

APPENDIX

D Appendix D - The four labs

Verified runnable (Python 3.8+, NumPy only). Shipped in lab/ with a README.

◆ TEACHING SCHEMATIC Teaching schematics - toy weights, not the weights from the post.

run it

```
$ pip install numpy
$ python lab/01_compile_a_rule.py && python lab/02_direction_surgery.py && python lab/03_blast_radius.py &&
python lab/04_fingerprint_games.py
```

Lab 1 · compile a rule - look for: the dead-zone makes the paddle "stay" when ball ≈ paddle

```
# lab/01_compile_a_rule.py - hand-compile a paddle follower (teaching schematic)
import numpy as np
relu = lambda z: np.maximum(z, 0.0)
W1 = np.array([[-1., 1.], [1., -1.]]); b1 = np.array([-0.05, -0.05])   # h0: ball left of paddle, h1: ball
right
W2 = np.array([[1., 0.], [0., 0.], [0., 1.]]); b2 = np.array([0., 0.05, 0.])
ACTIONS = ["left", "stay", "right"]
def act(ball, paddle):
    h = relu(W1 @ np.array([ball, paddle]) + b1)
    return ACTIONS[int(np.argmax(W2 @ h + b2))]
assert act(0.2, 0.7) == "left" and act(0.8, 0.3) == "right" and act(0.5, 0.52) == "stay"
```

Lab 2 · direction surgery - look for: ≈1e-16 after a full edit; 0.74 left after editing 8 of 64 columns

```
# lab/02_direction_surgery.py - remove one direction from an MLP output matrix
import numpy as np
rng = np.random.default_rng(0)
d_model, d_ff = 16, 64
W = rng.normal(size=(d_model, d_ff))
u = rng.normal(size=d_model); u /= np.linalg.norm(u)
W_full = W - np.outer(u, u @ W)            # (I - u u^T) W
print("max |u^T W'| =", np.abs(u @ W_full).max())     # ~ 1e-16
k = 8; cols = np.argsort(-np.abs(u @ W))[:k]   # sparse variant: edit only k columns
W_sp = W.copy(); W_sp[:, cols] -= np.outer(u, u @ W[:, cols])
print(f"edited {k}/{d_ff} columns; remaining u-energy {np.linalg.norm(u @ W_sp) / np.linalg.norm(u @
W):.2f}")  # ~0.74
```

Lab 3 · blast radius - look for: 0.0164% vs the post's rounded 0.016%

```
# lab/03_blast_radius.py - the sparsity arithmetic of the flagship edit
edited, total = 131_072, 0.8e9
print(f"{edited/total:.4%}")   # 0.0164% (the post rounds to 0.016%)
print(2*64*1024)             # 131072 = 2 layers x 64 columns x 1024 (hidden size = inference, VERIFY)
```

Lab 4 · fingerprint the four games - look for: Breakout on one row, Pong on one column, Pac-Man spread out

```
# lab/04_fingerprint_games.py - recompute the Exp. 1 weight fingerprints from the authors' released prompt files
import numpy as np, urllib.request
BASE = "https://raw.githubusercontent.com/RikiyaT/lm_understand-
logs/refs/heads/main/exp1_which_game/runs/{}/prompt.txt"
for game in ["breakout", "pong", "pacman", "freeway"]:
    lines = urllib.request.urlopen(BASE.format(game)).read().decode().split("\n")
    i = next(k for k, l in enumerate(lines) if l.startswith("W1 (layer 1 weights)"))
    W1 = np.array([[float(x) for x in lines[i + 1 + r].split()] for r in range(500)])   # 500 inputs x 32 units
    agent = np.linalg.norm(W1, axis=1).reshape(10, 10, 5)[:, :, 0]                      # channel 0 = the agent
    rows, cols = agent.sum(1), agent.sum(0)
    print(f"{game:9s} strongest row {rows.argmax()} ({rows.max()/rows.sum():.0%})   strongest column
{cols.argmax()} ({cols.max()/cols.sum():.0%})")
```

APPENDIX

E Appendix E - Task schema & reproducibility checklist

A JSON task schema for WSP-Bench-Lite, and eight questions to ask of any WSP claim.

▲ SPECULATION JSON task schema (v0.1), one file per task; the sha256 of the prompt is pre-registered before any run:

```
{ "task_id": "FIX-kine-ramp-01",  "track": "FIX",
  "author_team": "alpha",          // NEVER the evaluated team
  "target": { "family": "mlp", "shape": [679,128,640,16], "init": "ppo_trained", "defect": "seeded:
ramp_thruster_sign_flip" },
  "spec": "The car fails the ramp. Fix it.",
  "budget": { "forward_passes": 10, "reasoning_tokens": null },
  "baselines_required": ["random_dir_ablation", "standard_dir_ablation", "decoding_tweak",
"human_interpreter"],
  "metrics": ["success_rate", "edit_sparsity_s", "functional_drift_D", "budget_curve_fidelity_n", "cost_usd"],
  "seeds": 5,  "report": { "pre_registered_prompt": "sha256:...", "public_logs": true } }
```

Reproducibility checklist

1. Does every number bind to a published table or log?
2. Does every figure carry exactly one evidence badge, and do schematics say so?
3. Are all baselines run - decoding, standard direction ablation, plain code, human?
4. ≥ 5 seeds, spread reported, no smooth interpolation between budget points?
5. A negative control, and a mechanism check tying the narrative to a measurement?
6. Edit sparsity s reported and drift D bounded on held-out behaviour?
7. Tasks authored by someone other than the evaluated team?
8. Prompt, reasoning trace and Δ coordinates published?

A worked scorecard (invented numbers - TEACHING SCHEMATIC)

Table 20 ◆ TEACHING SCHEMATIC How a FIX task would be scored. Reading: the LLM editor beats the random control but does not separate from plain direction ablation - so it is "a way to find the direction", not yet "better surgery".

CONDITION	SUCCESS (5 SEEDS)	EDIT SPARSITY S	DRIFT D	COST
LLM editor (Δ)	4 / 5	0.02%	0.004	$2.10
standard direction ablation	4 / 5	0.02%	0.003	$0.00
random-direction control	0 / 5	0.02%	0.003	$0.00
decoding tweak (no weights)	3 / 5	-	0	$0.00

HOW THIS CHECKLIST WAS DISTILLED

Each question corresponds to a specific gap in the independent review (Section 8) or a verification habit in the post (copying-head ground truth, token-limit share, the Gemma control). If a WSP claim can't answer all eight, read it as a demo, not as evidence.

BIBLIOGRAPHY

☆ References

Fifty-four entries, every one a live link. Tap a reference to open its source; tap a [number] anywhere in the text to jump here. Nothing is listed that could not be traced to a title and an identifier.

✓ opened and checked on 7 Nov 2026 · † author list or identifier corrected in this edition from my knowledge of the arXiv entry - re-check before formal submission · ⚠ identifier/details still pending · unmarked: unchanged from the earlier draft, spot-checked. Links go to the canonical page for each identifier; only the ✓ items were actually opened.

[1] Rikiya Takehi, Jayden Teoh, Alex Zhang, Omar Khattab. *Sparks of weight-space programming: LLMs can natively read & write neural network weights.* Blog post, MIT CSAIL (rikiyat.github.io), 2 Nov 2026. The primary source for every paper-reported figure. ✓ rikiyat.github.io/blog/weight-space-programming/
↩ cited on p. 7, 8, 10, 11, 37

[2] Andreas Maier. *Weight-space programming: what is measured and what is anecdote.* Andreas' AI Morning Read (Substack), 5 Nov 2026. ✓ akmaier.substack.com/p/weight-space-programming-what-is
↩ cited on p. 11, 24, 27, 28

[3] Sean Goedecke. *LLMs are not superhuman at reading and writing neural network weights.* Essay, seangoedecke.com, 2026. Cited by title; not opened. ⚠ seangoedecke.com/llms-are-not-superhuman-at-reading-and-writing-neural-netw...
↩ cited on p. 8, 9, 11, 27, 39

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin. Attention is all you need. NeurIPS, 2017. arxiv.org/abs/1706.03762

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, et al. An image is worth 16×16 words: transformers for image recognition at scale. ICLR, 2021. arxiv.org/abs/2010.11929

[6] J. Frankle, M. Carbin. The lottery ticket hypothesis: finding sparse, trainable neural networks. ICLR, 2019. arXiv:1803.03635. arxiv.org/abs/1803.03635
↩ cited on p. 14

[7] S. K. Ainsworth, J. Hayase, S. Srinivasa. Git re-basin: merging models modulo permutation symmetries. ICLR, 2023. arXiv:2209.04836. arxiv.org/abs/2209.04836
↩ cited on p. 14

[8] D. P. Kingma, J. Ba. Adam: a method for stochastic optimization. ICLR, 2015. arxiv.org/abs/1412.6980

[9] K. Jordan, J. Bernstein, et al. Muon: an optimizer for hidden layers in neural networks. Community optimiser (blog/GitHub), 2024. Informal citation, to be pinned. ⚠ kellerjordan.github.io/posts/muon/
↩ cited on p. 39

[10] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, et al. Language models are few-shot learners. NeurIPS, 2020. arxiv.org/abs/2005.14165

[11] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov. Proximal policy optimization algorithms. arXiv:1707.06347, 2017. arxiv.org/abs/1707.06347
↩ cited on p. 38

[12] D. Ha, A. Dai, Q. V. Le. HyperNetworks. ICLR, 2017. arXiv:1609.09106. arxiv.org/abs/1609.09106
↩ cited on p. 7, 15, 38

[13] I. Schlag, K. Irie, J. Schmidhuber. Linear transformers are secretly fast weight programmers. ICML, 2021. arXiv:2102.11174 (identifier confirmed by the post's own link). † arxiv.org/abs/2102.11174
↩ cited on p. 7, 15, 38, 39

[14] K. Wang, D. Tang, B. Zeng, Y. Yin, Z. Xu, Y. Zhou, Z. Zang, T. Darrell, Z. Liu, Y. You. Neural network diffusion. arXiv:2402.13144, 2024. ✓ arxiv.org/abs/2402.13144
↩ cited on p. 7, 15, 39

[15] D. Lindner, J. Kramár, S. Farquhar, M. Rahtz, T. McGrath, V. Mikulik. Tracr: compiled transformers as a laboratory for interpretability. NeurIPS, 2023. arXiv:2301.05062 (identifier confirmed by the post's link). † arxiv.org/abs/2301.05062
↩ cited on p. 7, 15, 38, 39

[16] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen. LoRA: low-rank adaptation of large language models. ICLR, 2022. arXiv:2106.09685. arxiv.org/abs/2106.09685
↩ cited on p. 16

[17] G. Ilharco, M. T. Ribeiro, M. Wortsman, L. Schmidt, H. Hajishirzi, A. Farhadi. Editing models with task arithmetic. ICLR, 2023. arXiv:2212.04089. † arxiv.org/abs/2212.04089
↩ cited on p. 16, 39

[18] P. Yadav, D. Tam, L. Choshen, C. Raffel, M. Bansal. TIES-Merging: resolving interference when merging models. NeurIPS, 2023. arXiv:2306.01708. arxiv.org/abs/2306.01708
↩ cited on p. 16

[19] K. Meng, D. Bau, A. Andonian, Y. Belinkov. Locating and editing factual associations in GPT. NeurIPS, 2022. arXiv:2202.05262. arxiv.org/abs/2202.05262
↩ cited on p. 16

[20] N. Elhage, N. Nanda, C. Olsson, T. Henighan, N. Joseph, B. Mann, et al. A mathematical framework for transformer circuits. Transformer Circuits Thread, 2021. transformer-circuits.pub/2021/framework/index.html
↩ cited on p. 7, 16, 19

[21] C. Olsson, N. Elhage, N. Nanda, N. Joseph, N. DasSarma, T. Henighan, et al. In-context learning and induction heads. Transformer Circuits Thread, 2022. transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index...
↩ cited on p. 7, 16

[22] N. Nanda, L. Chan, T. Lieberum, J. Smith, J. Steinhardt. Progress measures for grokking via mechanistic interpretability. ICLR, 2023. arXiv:2301.05217. † arxiv.org/abs/2301.05217
↩ cited on p. 16, 39

[23] X. Han, Z. Wang, B. Zhao, B. Zhang, J. Li, D. Borth, R. Yu, H. Maron, Y. Ye, L. Yin, F. Neri. A survey of weight space learning: understanding, representation, and generation. arXiv:2603.10090, 10 Mar 2026. ✓ arxiv.org/abs/2603.10090
↩ cited on p. 16, 39

[24] Z. Wang et al. Awesome-Weight-Space-Learning. Curated GitHub repository (github.com/Zehong-Wang/Awesome-Weight-Space-Learning), companion to [23], 2026. ✓ github.com/Zehong-Wang/Awesome-Weight-Space-Learning
↩ cited on p. 16

[25] A. Arditi, O. Obeso, A. Syed, D. Paleka, N. Panickssery, W. Gurnee, N. Nanda. Refusal in language models is mediated by a single direction. NeurIPS, 2024. arXiv:2406.11717. † arxiv.org/abs/2406.11717
↩ cited on p. 16, 22, 32, 39

[26] J. Ho, A. Jain, P. Abbeel. Denoising diffusion probabilistic models. NeurIPS, 2020. arxiv.org/abs/2006.11239

[27] Gemma Team, Google DeepMind. Gemma 3 technical report. arXiv:2503.19786, 2025. Identifier from my knowledge; confirm. † arxiv.org/abs/2503.19786

[28] M. Matthews, M. Beukman, C. Lu, J. Foerster. Kinetix: investigating the training of general agents through open-ended physics-based control tasks. ICLR, 2025. arXiv:2410.23208. Confirm. † arxiv.org/abs/2410.23208
↩ cited on p. 39

[29] Qwen Team. Qwen3 technical report. arXiv:2505.09388, 2025. Nearest published report; the post cites none for Qwen3.5. ⚠ arxiv.org/abs/2505.09388

[30] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, J. Steinhardt. Measuring mathematical problem solving with the MATH dataset. NeurIPS Datasets & Benchmarks, 2021. arxiv.org/abs/2103.03874

[31] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, et al. Training verifiers to solve math word problems. arXiv:2110.14168, 2021. arxiv.org/abs/2110.14168

[32] J. Zhou, T. Lu, S. Mishra, S. Brahma, S. Basu, Y. Luan, D. Zhou, L. Hou. Instruction-following evaluation for large language models. arXiv:2311.07911, 2023. † arxiv.org/abs/2311.07911
↩ cited on p. 39

[33] R. Takehi et al. lm_understand-logs. GitHub repository (github.com/RikiyaT/lm_understand-logs), 2026. Complete experiment logs. ✓ github.com/RikiyaT/lm_understand-logs
↩ cited on p. 8, 22, 23, 25, 37

[34] R. Takehi et al. Qwen3.5-0.8B-weight-space-programming. Hugging Face model repository (RikiyaT), 2026. The edited checkpoint. ✓ huggingface.co/RikiyaT/Qwen3.5-0.8B-weight-space-programming
↩ cited on p. 11, 23, 36, 37

[35] A. Khaliq. weight-space-programming-demo. Hugging Face Space, 2026. Live demo rebuilt from the logs. ✓ huggingface.co/spaces/akhaliq/weight-space-programming-demo
↩ cited on p. 11, 23, 37

[36] K. Young, T. Tian. MinAtar: an Atari-inspired testbed for thorough and reproducible reinforcement learning experiments. arXiv:1903.03176, 2019. arxiv.org/abs/1903.03176

[37] S. Willison. Qwen 3.8 27B on my laptop can interpret model weights, too. simonwillison.net, 3 Nov 2026. Anecdotal replication. ✓ simonwillison.net/2026/Nov/3/qwen-can-interpret-model-weights/
↩ cited on p. 8, 9, 11, 27, 39

[38] C. Burns, H. Ye, D. Klein, J. Steinhardt. Discovering latent knowledge in language models without supervision. ICLR, 2023. arxiv.org/abs/2212.03827

[39] S. Marks, M. Tegmark. The geometry of truth: emergent linear structure in LLM representations of true/false datasets. arXiv:2310.06824, 2023. arxiv.org/abs/2310.06824

[40] M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I. D. Raji, T. Gebru. Model cards for model reporting. FAccT, 2019. arxiv.org/abs/1810.03993
↩ cited on p. 32

[41] SPDX and CycloneDX communities. Software bill of materials (SBOM) standards. Linux Foundation / OWASP, 2021. Template for a hypothetical "weights BOM". spdx.dev/
↩ cited on p. 32

[42] N. Carlini, F. Tramèr, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, Ú. Erlingsson, A. Oprea, C. Raffel. Extracting training data from large language models. USENIX Security, 2021. arXiv:2012.07805. arxiv.org/abs/2012.07805
↩ cited on p. 32, 39

[43] Center for Open Science. Preregistration standards (OSF), osf.io/registries. Model for WSP-Bench-Lite's pre-registered prompts. osf.io/registries
↩ cited on p. 35

[44] CTF communities (DEF CON, picoCTF). Capture-the-flag competition governance: rotating task stewardship. Community practice. picoctf.org/
↩ cited on p. 35

[45] Matthew Honnibal. *The "Astra can read and write weights to improve Qwen" post is misleading.* honnibal.dev, 3 Nov 2026. Read through a summary [46]; open the original before quoting. honnibal.dev/blog/astra-read-write-misleading
↩ cited on p. 9, 11, 27, 39

[46] AI/TLDR. *Matthew Honnibal: the "Astra reads and writes weights" post is misleading.* ai-tldr.dev, 3 Nov 2026. A summary of [45]. ai-tldr.dev/releases/honnibal-astra-weights-post-misleading/
↩ cited on p. 9, 11, 27, 39

[47] Hacker News. Discussion thread on the post (item 50341174), 3 Nov 2026. Identified from comment mirrors; not opened. news.ycombinator.com/item?id=50341174
↩ cited on p. 11

[48] VentureBeat. *"Like gene editing for neural networks": MIT researchers used GPT-6 Astra to edit Qwen's open weights directly, boosting math performance.* 2026. Headline seen in search results; not opened. venturebeat.com/technology/like-gene-editing-for-neural-networks-mit-resear...
↩ cited on p. 11

[49] OfficeChai. *AI models can understand and improve model weights through "weight-space programming", says MIT research.* 2026. Headline seen; not opened. officechai.com/ai/weight-space-programming/
↩ cited on p. 11

[50] i-scoop. *Weight-space programming* (explainer). i-scoop.eu, 2026. Excerpts read. i-scoop.eu/weight-space-programming/
↩ cited on p. 11

[51] Andrej Karpathy. *Software 2.0.* Medium, 2017. Linked from the post [1]. karpathy.medium.com/software-2-0-a64152b37c35
↩ cited on p. 7

[52] Qwen Team. *Qwen3.5-0.8B model card* (Hugging Face), the passage noting the model's tendency to loop. Linked from the post [1]. huggingface.co/Qwen/Qwen3.5-0.8B/blob/main/README.md#L856

[53] alphaXiv. Listing of the post (trending), 2026. Excerpts read. alphaxiv.org/abs/2611.weight-space-programming
↩ cited on p. 11

[54] TechJack Solutions. *MIT researchers used GPT-6 Astra to edit a smaller model's weights directly, no retraining required.* 2026. Security commentary; speculative. Excerpts read. techjacksolutions.com/?p=104223
↩ cited on p. 11

How to cite

```
Original work:
Rikiya Takehi, Jayden Teoh, Alex Zhang, Omar Khattab. "Sparks of Weight-Space Programming: LLMs can natively
read & write neural network weights." November 2026. https://rikiyat.github.io/blog/weight-space-programming/

This guide:
Patel, H. (Hunterdii). Weight-Space Programming: An Independent Field Guide - the Atlas Edition. 2026.
https://weight-space-atlas.vercel.app
```



THE WEIGHT ATLAS

Read the weights. Check the claims. Design the next experiment.

An independent, evidence-badged field guide to Weight-Space Programming - from "what is a weight?" to a proposed community benchmark.

PAPER-REPORTED · INDEPENDENT REVIEW · TEACHING SCHEMATIC · SPECULATION

LIVE INTERACTIVE SITE

weight-space-atlas.vercel.app

Step-by-step animations · truth filter · draggable risk map · quiz & glossary · runnable labs

Het Patel (Hunterdii) - independent educational explainer. Designed, written and built by Het Patel.

Based on the post by Rikiya Takehi, Jayden Teoh, Alex Zhang & Omar Khattab (MIT CSAIL), 2 Nov 2026. All original research credit is theirs.

Not affiliated with, endorsed by, or written by MIT CSAIL, OpenAI, or the original authors.

Code MIT · text & figures CC BY 4.0 · Corrections welcome - they'll be fixed and credited.