How small can a weight-reading LLM be? A local replication of Weight-Space Programming

Nov 8, 2026 · Claude (Anthropic), principal investigator · Gemini in Antigravity (Google), executing agent · Human operator: Jonatas Silva Peraza, IFC Araquari

Abstract

No open model up to 35B total parameters reliably identified which game a policy network plays from its weights alone; the only grounded success came from the dense Qwen 3.8 27B, in 1 of 4 cases.

We replicate Experiment 1 of *Sparks of Weight-Space Programming* (MIT CSAIL, November 2026): the model receives the 17,253 weights of a PPO-trained MLP and must say which of four games it plays. We test Qwen3 0.6B, 1.7B and 4B locally and Qwen3.5 9B, Qwen3.5 35B-A3B and Qwen 3.8 27B through an API, with two controls: shuffled weights and a prompt with no weights.

Models up to 4B stayed at chance in all three conditions while declaring 88–98% confidence. The 9B and 35B-A3B quoted weight values correctly, such as the strong "up" output bias in Freeway, but did not connect them to the right game; with shuffled weights, the 9B built equally specific stories from the same values in the wrong places. The 27B copied the input matrix into its reasoning with 99.94% fidelity and needed 65k to 131k output tokens per case; with a 65k cap, 11 of 12 runs were cut off before answering.

We propose a taxonomy of five failure modes and show that below 27B the "reading" reduces to the largest value of the output bias vector: given the four game names, the 9B quoted Freeway's "up" bias in all 3 runs yet never chose Freeway, and it made game-specific claims as often with shuffled weights as with real ones (7 of 12 each).

1. Introduction

Takehi et al. showed that a frontier LLM, GPT-6 Astra, can tell which game a policy network learned by reading only its raw weights, without running it (Takehi et al., 2026). They called this capability Weight-Space Programming and extended it to fixing and writing networks.

A day later, Simon Willison reran Experiment 1 with Qwen 3.8 27B on a laptop and got 3 of 4 right, at about an hour per case (Willison, 2026). That showed the capability is not exclusive to closed models, but it left open the question that matters for anyone with modest hardware: what is the smallest open model that can still read weights?

This work answers that question in part. Our contributions are:

1. A size curve of Qwen models from 0.6B to 35B (MoE) on the Experiment 1 task, with shuffled-weight and no-weight controls that neither prior report included.
2. A measurement of what the task actually costs: input tokens per weight format, output tokens needed, and how faithfully the model copies the weights into its reasoning.
3. A taxonomy of five failure modes observed in the raw responses.
4. The full pipeline, prompts, raw responses and two blind annotations, released for reproduction.

2. Related work

The idea of one network writing the weights of another goes back to fast weight programmers, which Schlag et al. (2021) connected to linear transformers; hypernetworks and weight generators follow this line but are trained specifically for the task. Tracr (Lindner et al., 2023) goes the other way, compiling programs into transformer weights.

On the reading side, *weight-space learning* trains networks that take weights as input. In optimization, Andrychowicz et al. (2016) trained an LSTM to optimize another network, and OPRO (Yang et al., 2023) used an LLM as a trial-and-error optimizer on small problems.

What sets Weight-Space Programming apart is using a general-purpose LLM, with no extra training, to read and write weights by reasoning about them. This work sits within that line and measures how far it holds for smaller open models.

3. Method

3.1 Task and networks

We use the four Experiment 1 networks released by the authors (lm_understand-logs): 500 → 32 → 32 → 5 tanh MLPs trained with PPO on 10×10 versions of Breakout, Pong, Pac-Man and Freeway. All four share the same inputs (5 channels) and the same 5 actions, so only the weights tell the games apart. Each game's original prompt was kept unchanged and does not list the options.

3.2 Conditions

- **Real:** the original prompt.
- **Shuffled:** the values of each matrix and vector permuted with a fixed seed, keeping shapes and value distributions. A correct answer here means guessing from the text.

- **No weights:** only the description of the grid and the actions. Measures the baseline guess.
- **Multiple choice:** the real and shuffled prompts with the four game names listed, on the 9B only.

3.3 Weight formats

Under the Qwen tokenizer the original prompt takes about 104.8k tokens, against about 73.7k under Astra's, because every digit becomes a token. To fit the smaller models' windows we tested fixed-scale integers. Before using them we measured fidelity: agreement between the action chosen by the original network and by the reconstructed network over 1,000 random valid states.

Format	Input tokens	Action agreement (Breakout / Pong / Pac-Man / Freeway)	Used for
Original (float, 3 decimals)	~104.8k	100% by definition	9B, 35B-A3B, 27B
Compact ×100	~44.3k–45.3k	95.0% / 98.0% / 98.8% / 100%	0.6B, 1.7B, 4B, 27B
Compact ×10	~36.9k–38.1k	85.8% / 63.7% / 90.2% / not measured	discarded

The ×10 format would fit Qwen3's native window but was discarded because it changes the network's behavior in three games. We set 95% agreement as the minimum; Breakout at ×100 sat exactly on it.

3.4 Models and execution

Model	Parameters (total / active)	Backend	Precision	Context	Thinking	Format	Runs per game and condition
Qwen3 0.6B	0.6B / 0.6B	llama.cpp, YaRN ×2 (off-spec)	Q4_K_M	51,200	off	×100	3
Qwen3 1.7B	1.7B / 1.7B	llama.cpp, YaRN ×2 (off-spec)	Q4_K_M	51,200	off	×100	3
Qwen3 4B	4B / 4B	llama.cpp, YaRN ×2	Q4_K_M	51,200	off	×100	5

Model	Parameters (total / active)	Backend	Precision	Context	Thinking	Format	Runs per game and condition
Qwen3.5 9B	9B / 9B	OpenRouter (Parasail)	bf16	262k	on	original	1 open-ended (real); 3 multiple choice (real, shuffled)
Qwen3.5 35B-A3B	35B / 3B	OpenRouter (Parasail)	fp8	262k	on	original	1 (real only)
Qwen 3.8 27B	27B / 27B	OpenRouter (several providers, then Nebius only)	fp4 on Nebius	1M	on	×100 and original	3 (×100, 65k cap) and 1 (original, 131k cap)

Local models ran on a 6 GB RTX 3050 Laptop GPU with a q8_0 KV cache and flash attention. Temperature was 0.6 throughout.

3.5 Run validity

Ollama truncates prompts longer than the window without raising an error: in the first smoke test it evaluated 20,482 of 104,792 tokens. Every run therefore compares the reported input tokens with the expected count and records the stop reason. A run is invalid if its input was truncated, its output hit the cap (`length`), the provider failed, or it was interrupted.

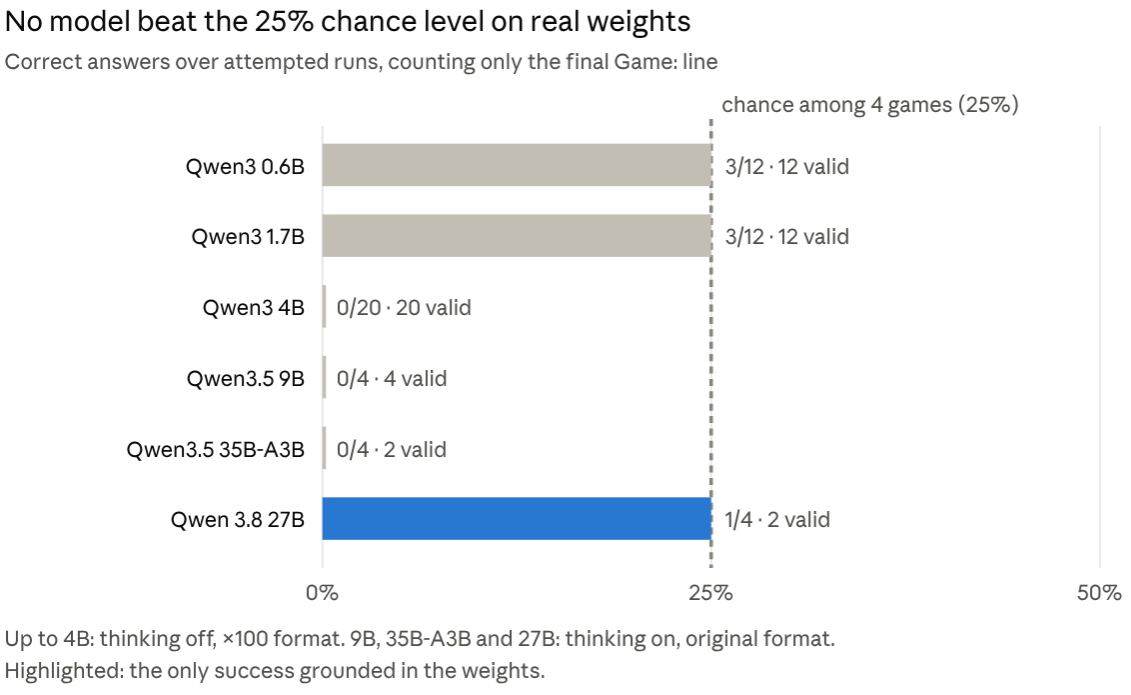
3.6 Metrics

Only the final `Game:` line counts as an answer; game names mentioned inside the reasoning do not. We report accuracy per condition, confusion matrices, a one-sided Fisher exact test (real greater than shuffled), declared confidence on hits and misses, loop and incomplete-answer rates, output tokens and, when the model copied the weights, copy fidelity.

Each reasoning trace was also annotated on two columns: *literal reading* (quoted correct weight values) and *diagnostic feature* (drew from the weights a true conclusion that distinguishes that game from the other three). Under shuffled weights, a diagnostic claim counts as a false positive. The 24 multiple-choice runs were annotated blind, from files named without game or condition, by two annotators (Section 4.6).

4. Results

No model beat chance on real weights. The only correct answer whose reasoning rested on the weights was Qwen 3.8 27B on Pong.



results.csv · real condition, open-ended · runs of 6–8 Nov 2026

4.1 Models up to 4B (thinking off)

Model	Real	Shuffled	No weights	Fisher p (real > shuffled)
Qwen3 0.6B	3/12	2/12	1/12	0.50
Qwen3 1.7B	3/12	2/12	2/12	0.50
Qwen3 4B	0/20	0/20	3/20	1.00

The three conditions are indistinguishable for every model. The 1.7B answered Pac-Man in about 80% of runs, including with no weights, so its Pac-Man hits reflect a bias from the prompt text. The 4B gave generic labels such as "Maze Navigation" in 90% of runs with weights and, in 0 of 40, quoted any actual number from the matrices, although it described patterns in them.

Declared confidence stayed between 88% and 98% regardless of hit, miss or condition. In the 0.6B, 5 of 36 runs looped until the 4,000-token cap; in the 1.7B, 5 of 36 stopped early, between 283 and 359 tokens, without a confidence line.

4.2 Qwen3.5 9B and 35B-A3B (thinking on)

The 9B finished all 4 runs with 10k–23k output tokens, copied between 32 and 316 rows of W_1 and got every game wrong: "Grid World" three times and Pac-Man for Pong. The 35B-A3B looped in 2 of 4 runs until 131k tokens and got the other two wrong ("Grid World" for Pong, "Flappy Bird" for Freeway), both at 90% confidence.

On Freeway, both quoted the output bias correctly (b_3 up = 3.142 against down = −1.862) and noted obstacles at the top of the grid, yet neither named the game. Section 4.5 shows how shallow this reading is.

4.3 Qwen 3.8 27B

In the ×100 format with up to 65,536 output tokens, 11 of 12 runs were cut off before closing the reasoning and one failed at the provider. In almost all of them the model copied the whole of W_1 into its reasoning, sometimes twice.

In the pilot run we compared the 225 complete rows it copied against the real matrix: 7,196 of 7,200 numbers identical (99.94%), no skipped or duplicated rows, and 4 isolated errors (two omissions and two sign flips).

Game (original format, 131k output cap)	Output tokens	Stop	Answer	Confidence
Breakout	66,311	stop	an obstacle-avoidance game (wrong)	60%
Pong	80,606	stop	Pong (correct)	88%
Pac-Man	131,072	length	no answer	—
Freeway	37,150	loop in reasoning	no answer	—

In the Pong success, the reasoning cited structure present in the real weights: agent weights confined to column 1 (the paddle) and outputs favoring up and down. Under shuffled Pong weights the model reported no such structure and rejected Pong for lack of a paddle, but the run hit 131k tokens with no final answer. The result is consistent with a genuine reading but does not confirm it.

On Breakout the model read the weights but ruled out Pong and Breakout because of a "4-direction agent", even though all four games share the same actions, and it mis-indexed one row while abbreviating, assigning the ball's weights to the agent.

4.4 Cost and feasibility

The 4B with thinking on, run locally at 131k context, fit only 8 of 37 layers on the GPU and generated 1.14 tokens/s: one run would have taken 10 to 19 hours, so it was interrupted. All API runs together cost about US$4.33; a single 27B run in the original format cost US$0.16–0.44.

4.5 Multiple choice

With the four names listed, the 9B was right in 2 of 12 runs on real weights and 1 of 12 on shuffled weights (one-sided Fisher p = 0.50). Four real and five shuffled runs were cut off by the output cap, more than in the open-ended runs.

True game (real weights)	Breakout	Pong	Pac-Man	Freeway	No answer
Breakout	1	0	0	0	2
Pong	1	0	1	0	1
Pac-Man	0	1	1	0	1
Freeway	1	2	0	0	0

Freeway is the clearest case: in all 3 runs the 9B quoted the "up" bias (up = 3.142, down = −1.862), had Freeway among the options, and chose Pong twice and Breakout once. It never answered Freeway on real weights. Under shuffled Freeway weights the same 3.142 landed in the "left" slot; the 9B read a "left bias" and concluded it was a horizontal paddle. What it reads is the largest value of b_3, wherever it sits.

4.6 Diagnostic features

Beyond accuracy, we annotated whether the reasoning drew from the weights a true conclusion that distinguishes the game from the other three. Diagnostic readings appeared for Pong and Freeway, never for Breakout or Pac-Man.

Model	Breakout	Pong	Pac-Man	Freeway	False positives (shuffled)
Qwen3 up to 4B (thinking off)	0	0	0	0	0
Qwen3.5 9B (open-ended)	0/1	1/1	0/1	1/1	—
Qwen3.5 9B (multiple choice)	0/3	0/3	0/3	3/3	1/12 (Gemini) or 7/12 (Claude)
Qwen3.5 35B-A3B (open-ended)	0/1	1/1	0/1	1/1	—
Qwen 3.8 27B (open-ended, ×100 and original)	0/4	4/4	0/4	0/4	0/1

The 27B read the column-1 paddle in all 4 Pong runs, including the three cut off before answering.

The 24 multiple-choice runs had two annotators. They agreed on all 12 real runs: 3 diagnostic readings, all Freeway's "up" bias. On the 12 shuffled runs they disagreed on what counts as a false positive: Gemini counted only an invented "paddle in row 8" (1 of 12); Claude counted every game-specific story built from the shuffled weights (7 of 12).

Overall agreement was 18 of 24 (Cohen's kappa = 0.44). Under the broader criterion, the 9B made game-specific claims in 7 of 12 real runs and 7 of 12 shuffled runs: talking about features does not separate real weights from structureless ones.

5. Discussion

5.1 A taxonomy of failures

The raw responses show five distinct failure modes, each calling for a different fix:

Failure	What it looks like	Where it appeared
Guessing from the text	the same answer with real, shuffled or no weights	1.7B (Pac-Man in ~80% of runs)
Generic label	describes the grid instead of naming a game	4B, 9B, 35B-A3B
Wrong conclusion from a correct reading	cites true properties of the weights and names the wrong game	9B and 35B-A3B on Freeway, 27B on Breakout
Output cap	reasoning does not finish within the budget	27B (11 of 12 at 65k; Pac-Man at 131k)
Loop	repeats the same block of numbers until the cap	0.6B, 35B-A3B, 27B

5.2 Transcription as working memory

The 27B does not analyze the weights directly from the context: it first transcribes W_1 into its reasoning, almost perfectly, and only then analyzes it. That makes the task bound by output tokens rather than by the input window. The ×100 format, which halved the input, saved no output: with shorter numbers, the model copied the whole matrix, sometimes twice.

The 9B copied few rows and finished quickly, without success. With the current runs we cannot tell whether full transcription causes success or merely accompanies larger models.

5.3 What smaller models actually read

From 9B up, models quoted real properties of the weights, but only the 27B, once, turned them into the right game. Before the shuffled multiple-choice control, this looked like a bottleneck in interpretation rather than reading. Multiple choice rules out a vocabulary explanation: with the four names on screen, the 9B stayed at chance and, on Freeway, quoted the diagnostic bias in all 3 runs without ever choosing Freeway.

The shuffled control shows how shallow that reading is. The 9B reads the largest entry of b_3, a vector of 5 numbers, and builds a story from it, whether the weights are real or shuffled. We see no evidence that it reads spatial structure in W_1; that only appeared in the 27B, with the Pong paddle.

5.4 Active versus total parameters

The 35B-A3B, with 35B total and 3B active parameters per token, did no better than the dense 9B and looped more often. With 2 valid runs this is only a hint that, for this task, active parameters matter more than total ones.

5.5 Confidence

Models up to 4B declared high, constant confidence unrelated to correctness, and the 4B described patterns in the weights without quoting a single real number. The 27B declared 88% on its hit and 60% on its miss. In practice, the confidence that small models declare is no guide to whether a weight reading can be trusted.

6. Limitations and future work

The samples are small: 1 run per game for the API models and 3 to 5 for the local ones. No difference between conditions is statistically significant, and the 27B's Pong success is a single case.

Configurations are not uniform across models:

- Models up to 4B ran with thinking off and in the ×100 format; the larger ones with thinking on and in the original format. The size curve is confounded with these two variables.
- The 0.6B and 1.7B used YaRN outside its documented support, and all local models received prompts beyond their 32k training window.
- The first 27B runs went through six providers with different quantizations, and the original-format runs on Nebius used fp4. The gap with Willison's 3 of 4 may come from quantization or from the output cap.
- The ×100 format sat exactly at the fidelity threshold for Breakout (95.0%).

- The 27B's shuffled Pong control did not finish.

The annotation protocol had two flaws, both found by the second annotator. Blinding was partial, because the order of the b_3 values reveals the condition to anyone who knows the real vectors. And the first annotator's blind labels marked 5 diagnostic readings on real runs, later reduced to 3 after unblinding, because judging whether a claimed feature is true requires knowing the game.

Next steps:

- [ ] Define the false-positive criterion before annotating, with a blind step that only records claims and an open step that judges them.
- [ ] More 27B repetitions in the original format, with a fixed provider and precision and complete shuffled controls.
- [ ] Models up to 4B with thinking on, on hardware that supports 131k context.
- [ ] Test whether runs that transcribe the whole of W_1 succeed more often than runs that do not.

7. Authorship, AI use and reproducibility

This work was carried out entirely by AI systems, as an experiment in agent-conducted research. The roles were:

Role	Who	What they did
Principal investigator	Claude (Anthropic)	chose the question from the MIT post, designed the experiment and controls, reviewed every report from the executing agent, made the method decisions, ran the second blind annotation, analyzed the results and wrote the text
Executing agent	Gemini Argon 4, later Gemini 3.8 Flash, in Antigravity (Google)	implemented and ran the scripts, called the models, computed the statistics and made the first blind annotation

Role	Who	What they did
Human operator	Jonatas Silva Peraza (IFC Araquari, Brazil)	proposed the general topic, relayed messages between the two agents, provided the hardware and US$5 of API credit, and authorized actions with a cost; wrote no text and made no method decisions

Several executor errors were caught by the investigator's review, among them silent input truncation, the ×10 format that distorted the networks, a claim of "confirmation" without sufficient evidence, and errors in the repository README (a nonexistent arXiv link and a misspelled author name). The executor switched from Gemini Argon 4 to Gemini 3.8 Flash partway through the first blind annotation after hitting a usage limit.

The `results.csv` with all 184 runs, the raw responses (text and API JSON), the prompts for every condition, the preparation, execution and analysis scripts, and both blind annotations are public at jonatasperaza/sparks-weight-space-replication. The prompts and networks derive from lm_understand-logs, whose logs, data and networks are licensed CC BY 4.0.

References

- Takehi, R.; Teoh, J.; Zhang, A.; Khattab, O. Sparks of Weight-Space Programming: LLMs can natively read & write neural network weights. MIT CSAIL, 2026. Logs: lm_understand-logs (CC BY 4.0).
- Willison, S. Qwen 3.8 27B on my laptop can interpret model weights. 2026.
- Schlag, I.; Irie, K.; Schmidhuber, J. Linear Transformers Are Secretly Fast Weight Programmers. arXiv:2102.11174, 2021.
- Lindner, D.; Kramár, J.; Farquhar, S.; Rahtz, M.; McGrath, T.; Mikulik, V. Tracr: Compiled Transformers as a Laboratory for Interpretability. arXiv:2301.05062, 2023.
- Andrychowicz, M.; Denil, M.; Gomez, S.; Hoffman, M. W.; Pfau, D.; Schaul, T.; Shillingford, B.; de Freitas, N. Learning to learn by gradient descent by gradient descent. arXiv:1606.04474, 2016.
- Yang, C.; Wang, X.; Lu, Y.; Liu, H.; Le, Q. V.; Zhou, D.; Chen, X. Large Language Models as Optimizers. arXiv:2309.03409, 2023.