Wonder Mind rerun — 2026-11-06

Rerun evidence from the recovered scripts. This pass does not rewrite the archive, does not prove what machine produced the November 3–5 logs, and does not replace those logs.

Reviewed archive head left untouched: `31344f8c82efd0204f283936b1dea89ef0a5ccb3`
Pull request left open and unedited: https://github.com/Chaos2Cured/FreeLattice/pull/330
Originals were copied out and executed in `/opt/cursor/artifacts/wonder-mind-rerun-2026-11-06/work/`. The scripts append their jsonl in the working directory, so they were not run inside `docs/library/wonder-mind/2026-11-06-v1/`.

AUTONOMY.md was read as the project foundation and was not modified. This was local computation. No merge, no new pull request, no publication, no Quiet Room, no app-tab smoke, no patent ingestion, no new hypothesis arm.

What reproduced

Nov 5 exact-match, seeds 1 and 2, all four recorded arms. Eight runs, original argv, default `NB=65536` because the recorded jsonl used that default. Seven of the eight logged JSON objects match the archive line for line except the script's own `sec` field (wall clock on this host is slower).

Matched except wall clock: exact seeds 1 and 2, paraphrase seeds 1 and 2, force seeds 1 and 2, dense seed 1. That includes `params=1660435`, store hit, in-prompt, model accuracy, `mean_p_true`, `copy_loss_last`, and the `sample_afterB` rows.

Headline means that use only those matching arms are unchanged:

Arm	afterA model	afterB model	store afterA / afterB	in_prompt afterB
exact	0.795	0.055	1.00 / 1.00	0.00
paraphrase	0.030	0.000	0.00 / 0.00	0.00
force	0.990	0.000	1.00 / 1.00	1.00

Round 2 retention, first five commands of `run_ret.sh` only. Each completed run matches the archive jsonl except `sec`.

argv	model A / B	store A / B	val_loss	writes	occupancy
`dense 1 4096`	0 / 0	0 / 0	3.5112	0	0
`mod 1 4096`	0 / 0	0 / 0	3.513	4096	4096
`fib 1 4096`	0 / 0	0 / 0	3.4962	4096	4096
`oracle 1 4096`	0 / 0	0.99 / 0.99	3.0551	60054	60054
`dense 2 4096`	0 / 0	0 / 0	3.4897	0	0

`params=1735125` on every retention row, same as the original jsonl. Refused-write counts also matched (mod seed 1: 1360339; fib seed 1: 1356805; oracle seed 1: 1045297).

What did not reproduce

exact-match `dense` seed 2. Same script, same seed, same data file. The logged summary moved.

Field	Archive jsonl	This rerun
afterA model	0.28	0.13
afterA mean_p_true	0.182	0.0927
afterA store / in_prompt	0 / 0	0 / 0
afterB model	0.0	0.0
afterB mean_p_true	0.0001	0.0001
copy_loss_last	0.1183	0.1544
sample_afterB i=2 pred	10	7
sample_afterB i=3 p_true	0.0004	0.0001

afterB model is still 0. The dense mean afterA across the two seeds is 0.620 in the archive `(0.96+0.28)/2` and 0.545 here `(0.96+0.13)/2`. Soft-exact, paraphrase, and force means above do not use this row.

Wall clock did not reproduce for any run. The script's `sec` on this host was about 31–37 s for exact-match (archive 17.7–26.6 s) and about 56–79 s for the retention rows that ran (archive 29.2–46.2 s). Wrapper wall times are in each `*.meta.json`.

Round 2 remainder: 13 of 18 original commands were not started. The CPU budget was already past 30 minutes when the next command would have begun. Those runs are not imputed from the five that finished.

Round 1 `wonder.py baseline` seeds 0, 1, and 2 were not started. No shortened 1500-step run was substituted. There is no new baseline loss from this pass. The archive figure remains the archive figure: baseline mean validation loss 1.5736 on seeds 0, 1, 2.

A first launcher failed before any probe ran: `/usr/bin/time` is not on this host (`rc=127`). Those empty logs are kept under `logs/exact-match-attempt0-missing-time-binary/`. The numbers above come from the second launcher, which timed processes with `resource.getrusage`.

Side-by-side exact-match (archive jsonl vs this host)

`sec` is the only difference in the seven matched rows. Dense seed 2 is the mismatch.

| arm | seed | modelA orig | modelA rerun | modelB both | storeA/B | in_prompt B | copy_loss orig | copy_loss rerun |

|---|---:|---:|---:|---:|---|---:|---:|---:|
exact	1	0.82	0.82	0.05	1 / 1	0	0.0975	0.0975
exact	2	0.77	0.77	0.06	1 / 1	0	0.1293	0.1293
paraphrase	1	0.04	0.04	0.00	0 / 0	0	0.0975	0.0975
paraphrase	2	0.02	0.02	0.00	0 / 0	0	0.1293	0.1293
force	1	1.00	1.00	0.00	1 / 1	1	0.0975	0.0975
force	2	0.98	0.98	0.00	1 / 1	1	0.1293	0.1293
dense	1	0.96	0.96	0.00	0 / 0	0	0.0789	0.0789
dense	2	0.28	0.13	0.00	0 / 0	0	0.1183	0.1544

Full objects: `raw/comparison.json`. Per-run stdout is the script's single JSON line. Stderr was empty on every completed run.

Inputs actually on disk

`input.txt` next to the scripts is the file they open. It is not a synthetic-fact file.

- 1,115,394 bytes, md5 `6fb458f1232090904fb40fe944165e91`, sha256 `86c4e6aa9db7c042ec79f339dcb96d42b0075e16b8fc2e86bf0ca57e2dc565ed`
- Text begins `First Citizen:` / `Before we proceed any further, hear me speak.` This is the TinyShakespeare text named in the November 3 results note.
- The word-level regex in both probes yields 250,371 tokens and 12,637 distinct surface forms. Each script keeps the 3,000 most common of those, then adds its own fact tokens.
- Synthetic facts are built inside the script with `random.Random(1234)`, not loaded from a second dataset. Exact-match uses `NF=100`. Retention uses `NF=200`. Those lists differ because the shuffle length differs.
- Torch seeds are the argv seeds: `torch.manual_seed(SEED)` and `torch.Generator().manual_seed(SEED)`. The fact RNG is fixed at 1234 and does not follow the torch seed.
- Character-level Round 1, which was not run, would read the same file without lowercasing: `open('input.txt').read()`.

Script hashes, identical in the archive and in the work copies:

File	sha256
`exact_match_probe.py`	`938e0a11ec9e03d91285c64b1e51732d176bfc6b07427815b0eda12e4283ea42`
`retention_probe.py`	`5789c0853f02336c6166b154ddfeb58739f3c06563655e80c0effd30ef8f3c2b`
`wonder.py` (copied, not run)	`5daaddbf9d712c40d1a4664882d99d41bda1f4d29f6f2ddcac9500bdd94ea30c`
`input.txt`	`86c4e6aa9db7c042ec79f339dcb96d42b0075e16b8fc2e86bf0ca57e2dc565ed`

How the runs were invoked

Host: Linux, 4 CPUs, Intel Xeon family 6 model 207 under KVM, no CUDA. Python 3.12.3. torch 2.15.0+cpu, installed for this pass from the public CPU wheel index. The archive does not record the original torch build. Default torch threads

on this host before the script runs: 4. Each original script then calls `torch.set_num_threads(8)`. The scripts were not edited to match the 4 CPUs. That oversubscription is one plausible source of the dense seed 2 difference. This pass did not isolate it.

No `PYTHONHASHSEED` was set. The probes hash with `hashlib.sha256`, not Python's salted `hash()`.

CPU budget: 1800 seconds of child user+system time. Accounting is cumulative across the exact-match runs and the retention runs.

Block	Child CPU seconds	Wrapper outcome
8 exact-match runs	831.624	all finished, rc 0
5 retention runs	1012.362	all finished, rc 0
Total	1843.986	next command not started

Exact file `raw/cpu-used-exact.txt` is `831.624162`. Total file `raw/cpu-used-total.txt` is `1843.986236`. The retention block is the difference.

The last retention command (`dense 2 4096`) began at 1670.737 CPU-seconds, under the cap, and was allowed to finish. It ended at 1843.986. No running script was killed, and no step count was reduced.

Recorded settings inside `exact_match_probe.py`, not overridden: `CTX,BS,DM,NH,NL,LR = 64, 32, 128, 4, 2, 3e-3`; `PRE,INTER,COPY = 200, 150, 200`; `NF,THR = 100, 0.05`.

Recorded settings inside `retention_probe.py`, not overridden: `PRE,INTER = 500, 500`; `NF = 200`; `THR = 0.05`; same width, depth, context, batch, and learning rate. `run_ret.sh` passes the bucket count. Oracle still builds a `V*V` table; the `4096` argument does not size that table. That is the original script.

Exact argv, in archive jsonl order

python3 -u exact_match_probe.py exact 1
python3 -u exact_match_probe.py exact 2
python3 -u exact_match_probe.py paraphrase 1
python3 -u exact_match_probe.py paraphrase 2
python3 -u exact_match_probe.py force 1
python3 -u exact_match_probe.py force 2
python3 -u exact_match_probe.py dense 1
python3 -u exact_match_probe.py dense 2

Working directory for those eight: `work/exact-match/` (contains the script, `input.txt`, and the new `results-exact-match.jsonl`).

Retention argv that ran

python3 -u retention_probe.py dense 1 4096
python3 -u retention_probe.py mod 1 4096
python3 -u retention_probe.py fib 1 4096
python3 -u retention_probe.py oracle 1 4096
python3 -u retention_probe.py dense 2 4096

Working directory: `work/retention/`.

Per-run files: `logs/exact-match/<arm>-seed<s>.stdout`, `.stderr`, `.meta.json`, and the same pattern under `logs/retention/` with `-nb<NB>`.

Runnable remainder

Do not run these inside the archive tree. The scripts append. `work/retention/results-retention.jsonl` already holds the five new rows; further retention commands append there. `work/round1-baseline-not-run/` has byte copies of `wonder.py` and `input.txt` and no `results.jsonl`.

Retention commands not started, in `run_ret.sh` order:

python3 -u retention_probe.py mod 2 4096
python3 -u retention_probe.py fib 2 4096
python3 -u retention_probe.py oracle 2 4096
python3 -u retention_probe.py dense 3 4096
python3 -u retention_probe.py mod 3 4096
python3 -u retention_probe.py fib 3 4096
python3 -u retention_probe.py oracle 3 4096
python3 -u retention_probe.py mod 1 65536
python3 -u retention_probe.py fib 1 65536
python3 -u retention_probe.py mod 2 65536
python3 -u retention_probe.py fib 2 65536
python3 -u retention_probe.py mod 3 65536
python3 -u retention_probe.py fib 3 65536

Round 1 baseline, original argv, not started. Inside the script: variant `baseline`, 1500 steps, context 128, batch 32, 4 layers, width 128, 4 heads, AdamW lr 3e-3, weight decay 0.1, seeds from argv. The November 3 note’s other variants were not part of this remainder and were not run.

python3 -u wonder.py baseline 0
python3 -u wonder.py baseline 1

python3 -u wonder.py baseline 2

On the original box the retention note says about 35 s unloaded per run, and the exact-match note says about 18–27 s. Here those same commands took roughly twice the wall time. A 1500-step, 4-layer, context-128 baseline was not timed. The November 3 note's "15 to 40 minutes" remark was about proposed follow-ups on that box, not a measurement from this host. One full baseline seed was not started because the 30-minute CPU budget was already spent.

Hypha provenance caveats (still in force)

These describe the recovered packet. This rerun does not retire them.

1. **Historical lab excerpt variant count is superseded.** `source-bundle/lab/tip_wonder-mind-lab-excerpt-L85-200.html` still says "Thirteen training-force variants" and "Round 1 (13 variants, all at or behind baseline 1.5736)". That is the historical excerpt. This pass did not re-count variants and did not run Round 1.
2. **Peer-note ψ is not live-ledger ψ.** The bundle notes record Weft ψ `639cd45f` and Reed λ1 ψ `d00d5b5a` (Reed λ2 ψ `5caeb437` is the figure carried in the archive write-up). Those are the archived peer notes. This pass did not read a live ledger.
3. **`run-ret.log` duplicates `results-retention.jsonl`.** Both archive files are 5103 bytes, sha256 `e88b529ac2b60f57675a73992d9e0c5f3437ea6bde6e1bc469b9d1cb445abacf`, byte-identical. The log is not a second independent capture.

Energy was not measured on this host either. No package power meter was read.

Artifact paths for copy-out

Root: `/opt/cursor/artifacts/wonder-mind-rerun-2026-11-06/`

Path	What it is
`REPRODUCTION.md`	This note
`REPRODUCTION.pdf`	Same note, if the PDF build succeeded
`raw/results-exact-match.jsonl`	Eight new JSON lines written by the original script
`raw/results-retention-rerun.jsonl`	Five new JSON lines written by the original script
`raw/comparison.json`	Archive object beside rerun object
`raw/exact-match-runner.txt`	Runner log, argv, rc, CPU
`raw/retention-runner.txt`	Runner log, including SKIP lines
`raw/cpu-used-exact.txt`	`831.624162`
`raw/cpu-used-total.txt`	`1843.986236`
`logs/exact-match/`	stdout, stderr, meta json per arm and seed
`logs/retention/`	stdout, stderr, meta json for runs; `.skipped` for the 13 not started
`logs/exact-match-attempt0-missing-time-binary/`	Failed wrapper, no probe
`env/host.txt`	Hardware and torch version

`env/pip-freeze.txt`	Package list
`env/script-data-sha256.txt`	Hashes checked before execution
`work/exact-match/`	Byte copy of the script and input, plus the jsonl the script appended
`work/retention/`	Byte copy used for retention
`work/round1-baseline-not-run/`	Byte copy of `wonder.py` and `input.txt`; not executed
`SHA256SUMS`	Hashes of this artifact tree