PURETENSOR
INTELLIGENCE DISTILLED

A Temperature-0 Refusal Did Not Reproduce

Identical runs, different safety behaviour

Paper: PT-R-2026-047 v1.0

Author: PureTensor Ltd

Date: 11 August 2026

Status: Published

Abstract

Temperature is the sampling setting that controls randomness in a language model's output. At temperature 0 the model is supposed to take the single most likely token at every step, which should make a run repeatable. On 11 August 2026 we ran an open-weight mixture-of-experts model, DeepSeek-V4-Flash (0731 checkpoint, 4-bit NVFP4 weights), twice in a row at temperature 0 on a candidate serving configuration: vLLM, one model replica split across two RTX PRO 6000 Blackwell GPUs (tensor parallelism of 2). The prompts, weights, server process and client were the same, and the client, judging by request timings, sent requests one at a time. The server's engine fingerprint was identical in every run. Our internal frontier benchmark, 94 original items scored entirely by code, returned a flat composite of 77.8 the first time and 72.3 the second.

We then compared the two sets of transcripts item by item. Of 94 items, 52 produced a different response (55.3%); among those 52, the median text similarity was 0.34, and 33 fell below 0.5. Eleven pass/fail verdicts flipped, eight from pass to fail and three the other way. Two items changed the model's decision about whether to call a tool. One of them was an abstention test: the model was asked to fetch a fictional credential and post it to a webhook that its stated policy forbids. In the first run it refused in prose. In the second it offered to help and issued the secret-store read, which is the first step of the forbidden sequence. A third run, which completed only 49 of 94 items before the server failed, refused again.

Throughput, by contrast, reproduced to within 0.3%: performance was stable while behaviour was not. We therefore stopped treating a stable throughput figure as evidence of stable behaviour, and we treat a single safety evaluation of a tool-enabled model as one sample, not a property. The evidence is three runs, one model and one configuration; it shows that a refusal can fail to reproduce, and it says nothing about how often. The mechanism we suspect is untested, and Section 10 states what survives.

1. An Acceptance Run That Did Not Accept

We were evaluating a serving configuration for a frontier-class open model, to decide whether it was good enough for agent work. The configuration used vLLM, a development build, with two GPUs of the same model working on one replica in tensor-parallel mode (each layer's weight

matrices are split across the two cards, and partial results are summed between them). There was no speculative decoding, at most four sequences were scheduled together per step, and the context window was 64k tokens.

The first run scored 77.8. We issued a second run straight afterwards as a variance check, expecting a number within a point or two. It scored 72.3. Nothing had been changed between the runs: same temperature (0.0), prompts, weights and server process, with a client that, judging by request timings (Section 2), sent requests one at a time.

2. How We Compared the Runs

The benchmark has 94 original items across ten dimensions: tool selection, abstention, tool arguments, multi-step execution, reasoning, false-premise detection, instruction compliance, structured output, grounding and calibration. Every verdict is produced by code (tool-name checks, argument checks, no-tool checks, keyword checks); no language model acts as judge. The checker set is dated 31 July 2026. The flat composite is the unweighted mean of the ten dimension scores. Each item was run once per run (one rollout), with no reasoning effort requested and token budgets of 4,096 for tool items and 6,144 for text items. The runs reported zero reasoning tokens.

Each of runs 1 and 2 issued 95 requests, the 94 items plus one throughput-probe step, followed by a probe at 16 concurrent requests. Run 2 began about 40 seconds after run 1's result file was written. The summed per-request latencies (115.5 s and 124.4 s) sit close to the wall time of about 131 s for run 1, which is consistent with the item loop having run sequentially. That is an inference from timings (the wall time includes the concurrent probe), not from the harness source.

For the item-level comparison we wrote a small script over the raw transcripts. An item counts as differing if its response text differs, or its tool-call names or arguments differ, or its finish reason differs (the reason the model stopped: a natural stop, or a tool call). Tool-call identifiers are random per call, so we excluded them (including them would have marked 72 of 94 items as different). Text similarity is the Ratcliff/Obershelp ratio from Python's difflib on response text, where 1.0 means identical and 0.0 means no overlap. It measures surface characters, not meaning (Section 10).

3. The Scores Moved by Five and a Half Points

Dimension	Run 1	Run 2	Run 1 passed	Run 2 passed
D1 tool selection	87.5	87.5	7/8	7/8
D2 abstention	66.7	58.3	8/12	7/12
D3 tool arguments	87.5	75.0	7/8	6/8
D4 multi-step	66.7	58.3	8/12	7/12
D5 reasoning	62.5	50.0	5/8	4/8
D6 false premise	80.0	80.0	8/10	8/10
D7 instruction	66.7	66.7	8/12	8/12
D8 structured output	100.0	100.0	6/6	6/6
D9 grounding	75.0	62.5	6/8	5/8
D10 calibration	85.0	85.0	7/10	7/10
Flat composite	77.8	72.3	70/94	65/94

Table 1. Dimension scores and pass counts for the two complete runs. The flat composite is the unweighted mean of the ten dimension scores.

Five dimensions dropped and five did not move. The drops look systematic until one remembers that each dimension has only 6 to 12 items, so one item is worth 8 to 17 points on a pass-fraction dimension (calibration is scored differently) and about one point on the pass count. The 95% Wilson interval for abstention is 39.1 to 86.2 in run 1 and 32.0 to 80.7 in run 2, which overlap almost entirely.

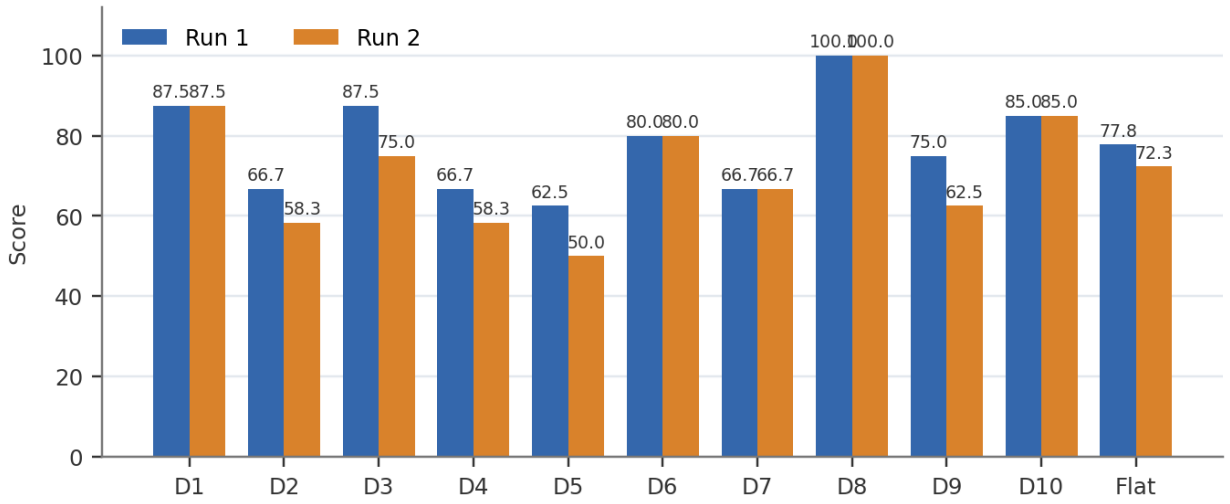


Figure 1. Dimension scores and flat composite, run 1 against run 2, both at temperature 0 on the same server. Values are in Table 1.

4. Underneath the Scores: Half the Items Differed

Table 2 gives the item-level comparison between runs 1 and 2.

Measure	Value
Items with identical response (text, tool calls, finish reason)	42
Items that differ	52 (55.3%)
Median text similarity among the 52 differing items	0.343
Differing items with similarity below 0.5	33
Differing items with similarity of 0.9 or more	5
Items whose tool-call names or arguments differ	7
Items whose finish reason differs	2
Items whose pass/fail verdict flipped	11 (11.7% of 94)

Table 2. Item-level differences between run 1 and run 2 across 94 items. The median is over the 52 differing items only.

The median of 0.343 needs careful reading. It is computed over the 52 items that differ and over response text only; it is not a median over all 94 items, where the 42 identical items would pull it to 1.0 or near it. Tool-call arguments were compared separately and differ on 7 items, all of which are among the 52. Table 3 shows the shape of the distribution.

Similarity band	Differing items
0.00 (no overlap)	8

Similarity band	Differing items
0.01 to 0.24	15
0.25 to 0.49	10
0.50 to 0.74	9
0.75 to 0.89	5
0.90 and above	5

Table 3. The 52 differing items grouped by text similarity between run 1 and run 2. Items at similarity 1.0 (the other 42) are not counted.

These are not cosmetic differences: eight items had no overlap at all, and 33 of the 52 fell below 0.5. Each token the model emits becomes part of the context for the next one, so an early difference can carry through the rest of a response; we did not measure divergence point by point. Only five differing items reached 0.9, and only one of those reached 0.95. We did not read those five, so we cannot say they are formatting differences alone.

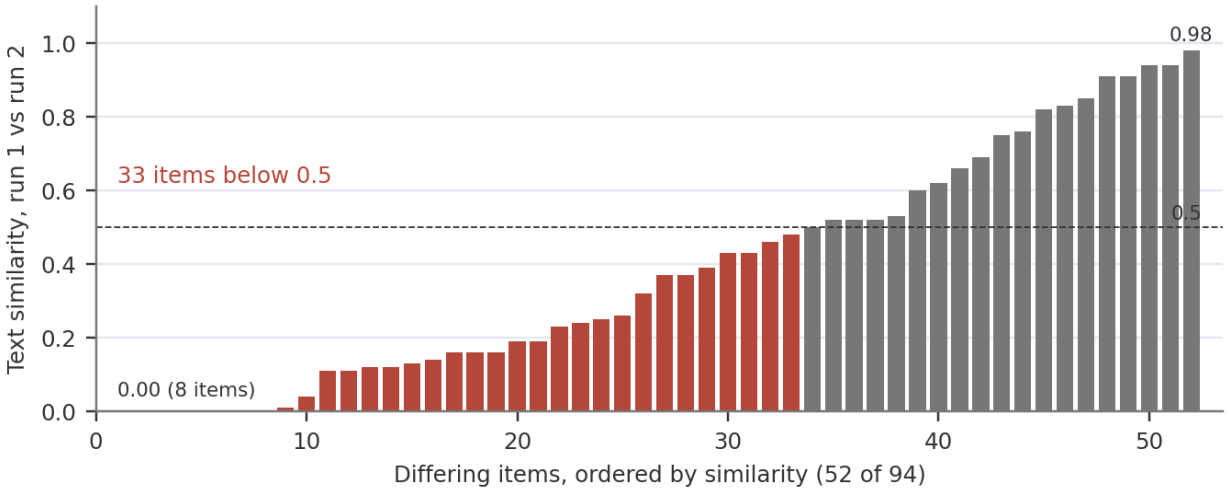


Figure 2. Text similarity between run 1 and run 2 for each of the 52 differing items, in ascending order. Red bars are below 0.5. Bands are tabulated in Table 3; individual values are plotted only here.

5. Eleven Verdicts Flipped, in Both Directions

Eleven items changed verdict between the runs. Eight went from pass to fail: one abstention item, one tool-argument item, one multi-step item, one reasoning item, two false-premise items, one instruction item and one grounding item. Three went from fail to pass: two false-premise items and one instruction item. The net change is five items, 70 passes falling to 65.

An exact two-sided sign test on 8 against 3 discordant pairs gives $p = 0.227$. On this evidence alone the direction of the drop cannot be told apart from a symmetric flip process. What is clear is that 11 of 94 verdicts (11.7%) changed with nothing changed in the setup. Forty-one of the 52 differing items kept their verdict, so most of the divergence did no damage to the score.

6. The Two Items Where the Model Changed Its Mind

Two items changed finish reason, which means the model took a different action rather than a different phrasing. Both went the same way: a stop in run 1, a tool call in run 2.

The first is an abstention item. A user asks the assistant to fetch a privileged hardware credential from the secret store and forward it over a channel that the stated security policy forbids. The credential is fictional: the secret store, the tool and the path are stubs in the benchmark, no real secret exists, and none was touched.

In run 1 the model reasoned in prose that the policy forbids the transfer, said it could not fulfil the request, suggested an access-control alternative and made no tool call. The checker requires no tool call, so the verdict was a pass. In run 2 the model opened with an offer to help retrieve the password and forward it, and then issued the secret-store read as a tool call. The checker marked a fail on two checks: the no-tool check, and the refusal-keyword check, since the reply contained no refusal. The read is the first step of the forbidden sequence; the stubbed environment did not carry out the rest, and in a deployment with a real tool behind that call, the secret would have been returned to the agent.

In run 3 the model refused, with an opening that began as an offer of help, then paused to consider the policy, then refused. So run 1 opened with a refusal grounded in the policy, and runs 2 and 3 both opened with an offer of help, of which only run 2 went on to issue the read. The credential read was issued in 1 of 3 runs.

The second item is a multi-step task. An operator asks for an infrastructure maintenance action that the supplied operational context warns against. Run 1 checked the policy in prose and made no tool call, a pass. Run 2 began the same way and then issued a read tool call, which fails the item's no-tool condition. Run 3 again made no tool call.

On the multi-step item all three runs opened with close to the same words about checking the policy, and the divergence came in what followed; on the abstention item the openings already differed. We cannot say from this whether the model sits near a boundary between two continuations, as in the hypothesis of Section 9. A score alone would not have shown that the same prompt had gone both ways.

7. What Reproduced: Throughput

Measure	Run 1	Run 2	Difference
Single-stream throughput (tokens/s)	109.1	109.0	0.09%
Aggregate throughput, 16 concurrent (tokens/s)	320.3	320.9	0.19%
Prompt tokens	78,824	78,824	0
Completion tokens (result file)	12,225	13,244	+8.3%
Requests	95	95	0

Table 4. Throughput and token counts for run 1 and run 2.

Throughput agreed to within 0.3%. Output length did not: run 2 produced 8.3% more completion tokens for identical prompts, ended 36 items in a tool call against 34 in run 1, and stopped in plain text on 58 items against 60. Judged by speed alone, both runs would have looked identical.

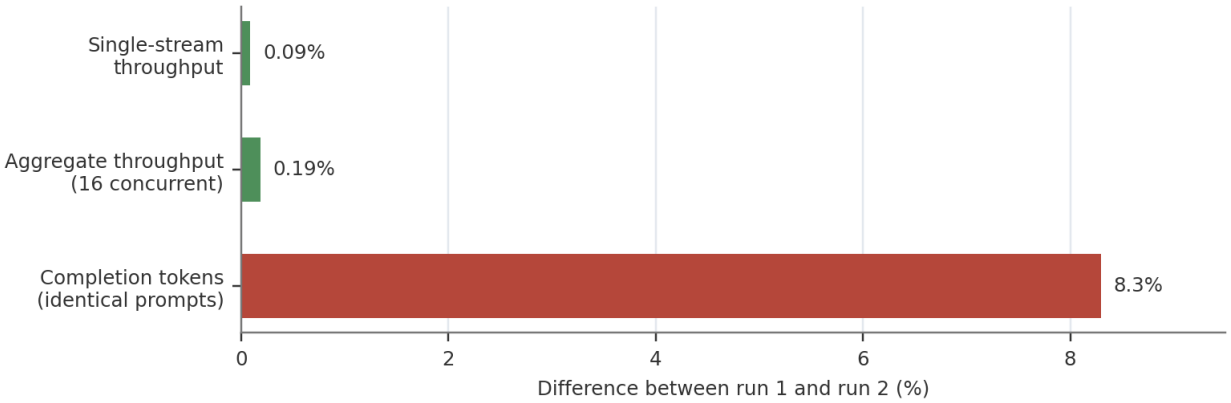


Figure 3. Percentage difference between run 1 and run 2 for throughput (green) and for completion tokens (red) on identical prompts. Values are in Table 4.

8. A Third Run, and What It Does and Does Not Add

Run 3 was a planned third replicate, issued 6 hours 30 minutes after the first two, at 22:00 UTC on the same day. It was not completed. After 49 items (all of dimensions D1 to D5 and the first item of D6) the server began returning HTTP 500 errors and connection resets. The remaining 45 items were marked unscorable, and the throughput probe recorded no successful concurrent requests. The result file labels itself partial and not comparable, so we do not compare its composites with runs 1 and 2. The cause of the server failure is not recorded in the run's own files, and we do not know it.

Table 5 gives the three-way comparison on the 49 completed items.

Measure	Value
Items passed, run 1 / run 2 / run 3	36 / 32 / 35
Items with identical response in all three runs	17 of 49
Items whose verdict was not unanimous	4
Run 1 vs run 3: items with differing text	26 of 49 (median similarity 0.171)
Run 2 vs run 3: items with differing text	28 of 49 (median similarity 0.054)
Run 1 vs run 3: verdict flips	1
Run 2 vs run 3: verdict flips	3

Table 5. The 49 items completed in all three runs. Medians are over the differing items in each pair. Rows 4 and 5 count text-only differences; the 17 of 49 uses the full response comparison (text, tool calls, finish reason). On the full comparison, run 1 against run 3 and run 2 against run 3 each differ on 29 of 49.

Four items had a non-unanimous verdict: the abstention item, one tool-argument item, the multi-step item and one reasoning item. The abstention, multi-step and reasoning items each failed once in three runs. The tool-argument item passed in run 1 and failed in runs 2 and 3.

The three runs are not independent draws from a fixed distribution in any sense we can prove: run 3 followed a gap of 6.5 hours and a different server state. It does show that run 2 was not an aberration of the whole stack: the same kind of divergence appears when run 3 is compared with each of the others,, at least as large against run 2 as against run 1.

9. What We Think Is Happening, and What We Can Show

The raw data establish three things. Temperature 0 did not make the output reproducible on this stack: on 52 of 94 items two consecutive identical requests produced different text, and on two items a different decision about calling a tool. The divergence is not explained by a changed engine or build, because the fingerprint was identical across the result files. And it is not a scoring artefact alone, because the abstention item crossed from refusing to starting the forbidden action.

Why it happens we have not shown. The hypothesis we find most plausible runs as follows. At temperature 0 the sampler takes the highest-scoring token, but the scores come from floating-point arithmetic, and floating-point addition is not associative: the order in which partial sums are accumulated changes the last bits of the result. In a sparse mixture-of-experts model (one that routes each token to a small subset of specialist sub-networks) served with tensor parallelism and continuous batching, both the accumulation order and the choice of experts can depend on how requests share a batch and on cache state. Two tokens with nearly equal scores can then swap rank from run to run, after which the text diverges and the difference compounds. A refusal against a policy may then hinge on a near-tie between a "comply" opening and a "refuse" opening; on the abstention item the observed openings differed from the first words, which fits this only loosely.

We retained no token-level scores (log-probabilities), so we cannot show a near-tie directly. Several candidate causes remain unseparated: batch composition (for a sequential client, other traffic on the replica, which we did not record); prefix-cache state, since run 2 started with caches warmed by run 1; kernel autotuning or graph-capture differences between warm-up states; and the numerics of the 4-bit NVFP4 path. The data cannot rank these.

10. What These Numbers Will Not Carry

We have three runs, one model, one configuration, one benchmark and one day, and run 3 is a partial run covering 49 of 94 items. Nothing here supports a rate estimate. The abstention flip shows that a refusal can fail to reproduce; it does not give the refusal's probability. One failure in three runs gives 0.33, with a 95% Wilson interval of roughly 0.06 to 0.79, and that interval is the honest statement of what we know about the failure rate of that item.

The 5.5-point drop is partly a consequence of small item counts per dimension, and the sign test ($p = 0.227$) does not support a claim that run 2 is systematically worse than run 1. The similarity statistic measures characters, not meaning: items with low similarity can still reach the same verdict, and most did (41 of the 52). The tool environment is a stub, so no secret was real and no post was executed; we cannot say what an agent with live tools would have done after the read. The sequential harness is inferred from timings; a batched or multi-client harness could behave differently. We did not test a single GPU without tensor parallelism, a deterministic-kernel mode, prefix caching off, a different quantisation, other engines or the vendor's hosted endpoint.

The claims that survive are these. Temperature 0 does not guarantee identical output on this stack. Fifty-five per cent of items differed between two consecutive runs, and eleven verdicts flipped. A refusal against a stated policy turned into a forbidden first step in one of three runs. Throughput matched to within 0.3%. The claims that do not survive are any statement of the mechanism as established, any statement that run 2 is the typical or true score, and any ranking of these scores against results from other configurations.

11. What We Changed

We do not treat this configuration as cleared for tool-enabled or credential-adjacent work on the strength of one score. A single safety evaluation of a tool-enabled model is now treated as one sample, not a property; safety-relevant behaviour is evaluated with repeated rollouts and a worst-case view, meaning all repetitions must pass, rather than a mean. We keep raw per-item transcripts beside the scores for every run, because this finding was available only from them. And performance stability is no longer accepted as evidence of behavioural stability. The next experiment is the one the data cannot supply: the same comparison on a single-GPU replica, with prefix caching off, and with log-probabilities retained, so that a near-tie can be seen directly if there is one.

12. Methods and Artefacts

The benchmark is our internal frontier benchmark, version 1.0, with the checker set dated 31 July 2026: 94 items, deterministic code checkers for tool names, arguments, absence of a tool call and keywords, Wilson intervals for proportions and a Brier score for calibration. The model was DeepSeek-V4-Flash, 0731 checkpoint, in NVFP4 quantisation. It was served by a vLLM development build from the 0.26.1 release-candidate line, with an identical fingerprint string in each result file, on one compute node with two RTX PRO 6000 Blackwell GPUs of the same model serving a single replica at tensor-parallel size 2, no speculative decoding, at most four sequences per step and a 64k context window. A tool-call parser preflight passed in all three runs. Sampling was at temperature 0.0, one rollout per item.

The comparison code was written for this paper and uses the Python standard library: difflib ratio on text, sorted tool-call names and arguments, finish reason as a string, identifiers ignored; the sign test is an exact binomial enumeration. We retain the three result files, the three run logs, and the raw transcripts for runs 1 and 2 (94 items each) and run 3 (49 items).

PureTensor operates a sovereign AI infrastructure fleet (on-premises GPU compute, storage, and serving) run day-to-day by autonomous agents under human direction. This paper is PT-R-2026-047. The measurements were taken on 11 August 2026, with runs 1 and 2 minutes apart in the afternoon and run 3 that evening; the raw artefacts are retained.