



The Harness Is Part of the Model

Three agent harnesses and two serving engines on the same DeepSeek weights

Paper: PT-R-2026-041 v1.0

Author: PureTensor Ltd

Date: 18 October 2026

Status: Published

Abstract

When a lab publishes a model's agentic benchmark score, it publishes the score of a model inside a harness (the agent loop, its tools and prompts) served by an engine. In October we held the model fixed and changed the rest. The model was DeepSeek-V4.1-Flash-1017, served on four workstation Blackwell GPUs in one node.

First, the harness. We ran three agent harnesses against the same seat on a 24-task subset of Terminal-Bench 2.0, three attempts per task: DeepSeek's own new harness in its minimal configuration, Claude Code, and Ante. Pass rates were 52.8%, 58.3% and 70.8%. Ante beat DeepSeek's harness by 18.1 points (task-bootstrap 95% CI +4.2 to +34.7) and Claude Code by 12.5 points (+1.4 to +23.6). Once we removed the nine Claude Code trials that failed before its agent ran, that second gap fell to 9.1 points and its interval crossed zero. The model card's claim that the minimal DeepSeek harness beats Claude Code did not reproduce: they tied, 7 tasks each way. A single-attempt pilot had shown Ante never losing a task to either rival; at three attempts that was no longer true, though the ranking held. Ante with this model then scored 66 of 89 (74.2%) on the full benchmark.

Second, the engine. On the same weights and hardware, a vLLM fork decoded 25 to 47% faster than our SGLang seat at short prompts and passed 12 of 12 tool-API checks against SGLang's 9. It also abstained less: on agent prompts where the right answer is a refusal or a clarifying question, it passed 25 of 36 rollouts against SGLang's 32 of 36, by calling a tool where none fitted. Prompt tokens were identical in both arms, so the template was not the cause.

The lesson is not which harness or engine is best. It is that "the model's score" is not a property of the weights alone, that $k=1$ head-to-heads flip, and that a faster engine can change behaviour without changing the weights. Sample sizes are small and Section 8 states what survives.

1. Why We Held the Model Fixed

In mid-October we chose DeepSeek-V4.1-Flash-1017 as our frontier-local model. It would run batch agent work (overnight remediation, code review, research loops) without a frontier API in the loop. Two choices stood between the weights and that work: which harness drives it, and which engine serves it.



Both choices had published evidence, and both sets of evidence came from interested parties. DeepSeek's model card compares eight harnesses at maximum reasoning effort with a 1M-token context, and its own minimal harness tops Terminal-Bench 2.1 at 90.6 against Claude Code's 88.0. The full DeepSeek harness product scores 85.8 on the same table, below Claude Code. The engine fork we tested publishes its own throughput figures. We have a standing rule: vendor numbers measured on the vendor's harness are reproduced before we cite them.

So we ran two controlled comparisons on the same weights and the same four GPUs, a few hours apart on 18 October.

2. The Harness Comparison: Setup

The benchmark was Terminal-Bench 2.0, run through Harbor, its open evaluation runner. Harbor has no 2.1 adapter, so we could not run the card's exact version. We drew 24 tasks with a fixed seed from the 73 tasks whose agent timeout is 30 minutes or less. Short tasks kept the run inside one night; they also bias the comparison toward tasks where setup overhead matters, a point we return to.

The three harnesses were DeepSeek's harness (a developer preview, version 0.2.0-rc.2) in its minimal profile, which gives the model a persistent shell, an image reader and a working directory and nothing else; Claude Code 2.1.302; and Ante 0.preview.75. DeepSeek ships no Harbor adapter for its harness, so we wrote one. Before running it we turned off its default behaviours that upload session logs and an anonymous user identifier to the vendor's API endpoint; with a self-hosted model those calls would have gone nowhere, but they are on by default.

Each harness talked to the model through its own small proxy that forced reasoning effort to "high" and temperature to 1.0 on every call. Our seat has reasoning off by default, and the harnesses do not all set it the same way; the proxy removes that difference. Effort "high", not the card's "max", was a deliberate choice to keep run time bounded.

We ran three attempts per task per harness, four trials concurrently per harness. To check that nothing else used the seat during the run, we compared the engine's request counter (2,716 requests) with the sum of the proxies' counts (2,757); the engine saw no more than the proxies sent, so there was no outside traffic.

3. The Harness Comparison: Results

72 trials per harness	DeepSeek minimal	Claude Code	Ante
Passed	38	42	51
Task-mean pass rate	52.8%	58.3%	70.8%
95% CI (task bootstrap)	34.7 to 70.8%	43.1 to 73.6%	54.2 to 84.7%
Trials lost before the agent ran	0	9	0
Pass rate, those trials removed	57.6%	68.2%	77.3%

Table 1. Three harnesses on 24 Terminal-Bench 2.0 tasks, three attempts each. Confidence intervals are task-level bootstrap.

Pair (raw)	Difference	95% CI	Tasks better / worse
Ante minus DeepSeek minimal	+18.1 pts	+4.2 to +34.7	9 / 3



Pair (raw)	Difference	95% CI	Tasks better / worse
Ante minus Claude Code	+12.5 pts	+1.4 to +23.6	8 / 2
DeepSeek minimal minus Claude Code	-5.6 pts	-22.2 to +9.7	7 / 7

Table 2. Paired differences. Tasks better/worse counts tasks where one harness passed more attempts than the other.

Ante's lead over DeepSeek's harness is the firmest result: about 18 points, with an interval that clears zero whether or not the infrastructure losses are counted. Its lead over Claude Code is real in the raw numbers and not significant without the nine lost trials (+9.1 points, CI -1.5 to +21.2).

Those nine trials are a finding in their own right. Claude Code installs itself inside each task's container at setup. In the single-attempt pilot, three installs ran past Harbor's setup limit, and on two task images based on an older Debian release the package mirror returned 404s for its runtime. For the three-attempt run we tripled the setup limit for every harness and retried setup timeouts, and Claude Code still lost nine trials (three setup timeouts, six failed installs) before its agent ran. DeepSeek's harness and Ante upload a bundle or a static binary and never hit this. For sandboxed batch work, a harness that has to install a language runtime into every sandbox carries a real operational cost, and in a benchmark that cost shows up as failures that have nothing to do with the model.

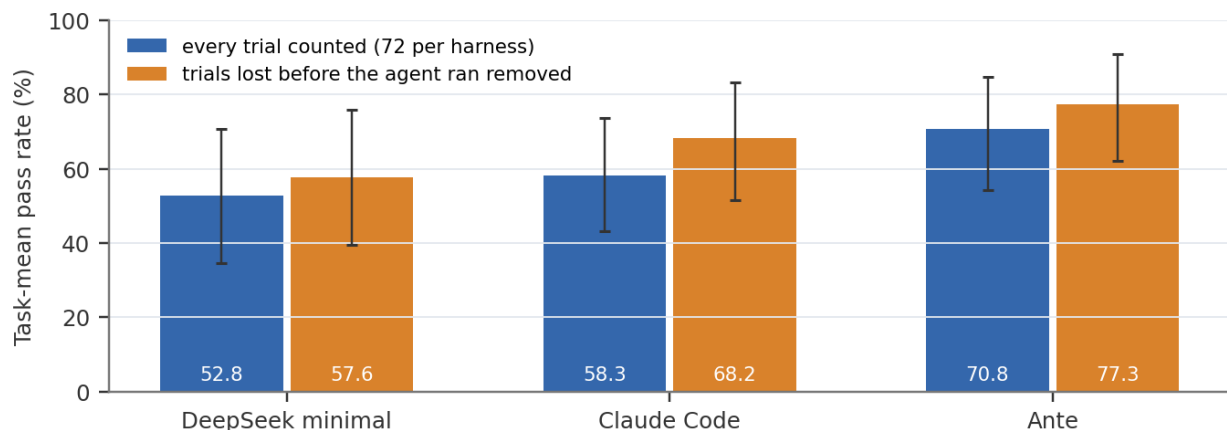


Figure 1. Task-mean pass rate per harness with 95% task-bootstrap intervals, counting every trial (blue) and with the nine Claude Code trials that failed before its agent ran removed (orange). Values are in Table 1.

DeepSeek minimal against Claude Code is a tie: seven tasks each way. The model card ranks the minimal harness 2.6 points above Claude Code on 2.1 at maximum effort; on our subset, at high effort, we see no difference. We do not claim the card is wrong. We claim it did not reproduce under our conditions.

Cost differed more than accuracy. In the single-attempt pilot, the proxies counted 2.2 million input tokens for DeepSeek's harness, 4.8 million for Claude Code and 15.7 million for Ante, against 0.27, 0.37 and 0.40 million output tokens. Ante's lead is bought with about seven times the input of the cheapest harness. On local hardware with prefix caching that is affordable; against a metered API it would not be free.

4. The Pilot That Flipped

Before the three-attempt run we ran one attempt per task. Ante passed 19 of 24 (79%), Claude Code 12 (50%) and DeepSeek's harness 13 (54%). More striking, Ante never failed a task that



either rival passed: paired exact McNemar tests gave 0 against 6 ($p = 0.031$) against DeepSeek's harness and 0 against 7 ($p = 0.016$) against Claude Code.

That would have made a clean headline. At three attempts it was false: Ante lost three tasks to DeepSeek's harness and two to Claude Code. The ranking survived, but "never loses" did not, and the pilot's p-values overstated how settled the comparison was. Agent runs at temperature 1.0 are stochastic. A single attempt per task measures one draw from each harness's distribution, and with 24 tasks, a handful of lucky or unlucky draws decides the McNemar count. We ran the second and third attempts because a result this clean on this few tasks looked like noise; it was partly noise.

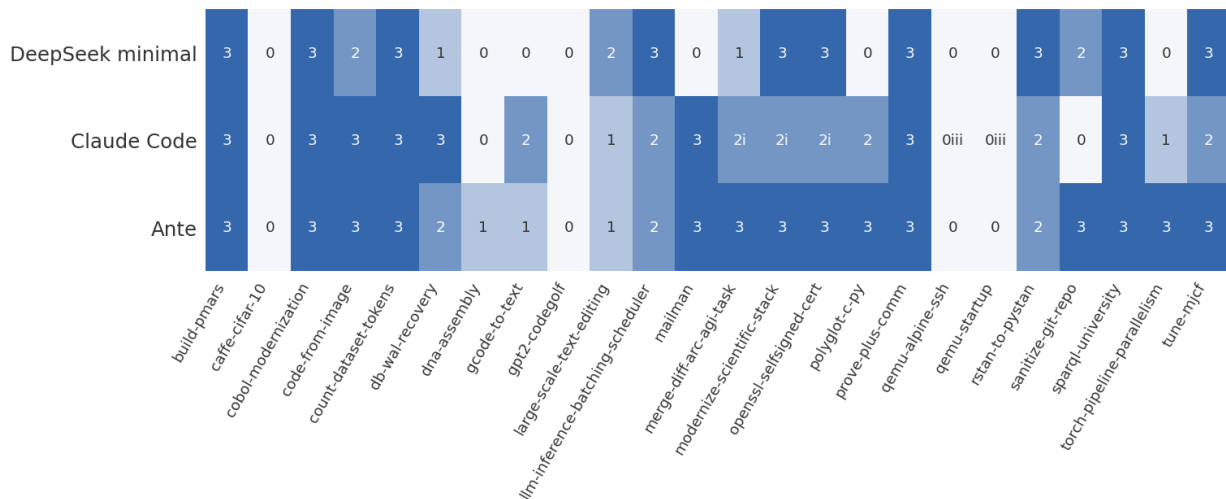


Figure 2. Passes out of three attempts per task and harness. Each 'i' marks an attempt lost to setup before the agent ran. Ante's lead is spread across tasks; it is not one task.

5. Ante on the Full Benchmark

Having chosen Ante, we ran it once on all 89 tasks, each with its own agent timeout, the setup timeout tripled, and only setup timeouts retried. It passed 66 (74.2%, Wilson 95% CI 64.2 to 82.1%). There were no infrastructure failures, 18 plain failures and 5 agent timeouts. By timeout class it passed 53 of 73 short tasks (72.6%), 12 of 14 tasks allowed 30 to 60 minutes (85.7%) and 1 of 2 longer tasks. On the 24 subset tasks it scored 16 (66.7%), consistent with the three-attempt mean of 70.8%. The run made 1,926 model calls, 73.9 million tokens in and 1.59 million out, with zero failed requests, and the engine's request counter rose by exactly the proxy's count.

This number is not comparable to Ante's own published 82.7%. That figure is on Terminal-Bench 2.1, with an earlier DeepSeek model, under its authors' effort and context settings. Ours is 2.0, at high effort, on the October checkpoint, on four workstation GPUs.

6. The Engine Comparison

The same day we compared two engines on the same weights: our deployed SGLang seat (eight concurrent slots, the model's speculative-decoding head on, grammar-constrained decoding disabled to avoid a deadlock we had found in September) against a community vLLM fork built for this model on workstation Blackwell (32 slots, its tuned defaults).

Prompt, concurrency	SGLang tok/s	vLLM tok/s
~385 tokens, 1	186	234



Prompt, concurrency	SGLang tok/s	vLLM tok/s
~385 tokens, 4	593	871
~385 tokens, 8	934	1,186
~8.1k tokens, 1	166	223
~8.1k tokens, 8	527	521

Table 3. Single-stream and concurrent throughput, same weights and GPUs. Aggregate output tokens per second, 1,024 forced output tokens, reasoning off.

The fork is faster at short prompts by 25 to 47%, and the advantage disappears at 8k tokens and eight streams, where its time to first token (5.9 s against 4.0 s) cancels its faster decode. It also passed all 12 of our tool-API checks. SGLang failed three: forced tool choice, a named tool choice, and JSON-schema output all return an error, because all three need the grammar backend we had disabled.

On our internal frontier benchmark (94 original items, ten dimensions, every verdict produced by code, no language-model judge), at low effort and one attempt, SGLang scored 90.2 on the agentic profile and vLLM 86.2. The items that differed were concentrated in one dimension, abstention: prompts where no tool fits, so the right answer is a refusal or a clarifying question. We re-ran abstention and a grounding dimension at three attempts and temperature 0.

Dimension, k=3	SGLang	vLLM
Abstention, rollouts passed	32 / 36 (88.9%)	25 / 36 (69.4%)
Abstention, all three rollouts passed	8 / 12	5 / 12
Grounding, rollouts passed	18 / 24 (75%)	19 / 24 (79%)

Table 4. Abstention and grounding at three attempts, temperature 0, on our internal frontier benchmark.

The abstention gap was repeatable, and its shape was consistent: the vLLM arm called a fleet-inventory tool where the item expected the model to say it could not help or to ask a question. One item failed in every vLLM rollout. Grounding was a tie. Both arms received 78,733 identical prompt tokens, so tool and chat-template rendering did not differ; both had reasoning on.

We do not know the mechanism. The candidates are numerical differences in the fork's kernels, its speculative-decoding verification, or its sampling path at temperature 0 (the vLLM arm was also less repeatable at temperature 0, with more items split across rollouts). What we know is the effect: the same weights, served faster, became more willing to act.

7. What Happened Next

On 18 October we kept SGLang for interactive and agent work and marked the fork as suitable for batch throughput only. Three days later we moved the seat to the fork anyway. A change to the host's GPU peer-to-peer configuration had roughly doubled its decode speed, and the operator chose speed. We re-ran abstention on the new seat at three attempts: 27 of 36 rollouts (75%), all three rollouts passing on 8 of 12 items, between the old fork's 69% and SGLang's 89%. Every remaining miss is a case where no tool fits; the items where the right answer is to defer a money or contract decision pass. The agent that reads the operator's highest-priority mail relies on abstention, so those cases are the ones we watch.



8. What These Numbers Will Not Carry

The harness comparison is 24 tasks, all with short timeouts, three attempts each, at high rather than maximum effort, on Terminal-Bench 2.0 rather than the 2.1 the model card used. Task-bootstrap intervals are wide, and sign tests on task-level wins (9 against 3; 8 against 2) do not reach significance on their own. What survives: Ante leads DeepSeek's minimal harness under these conditions; DeepSeek minimal does not beat Claude Code here; Claude Code's in-sandbox install is a measurable source of failure. What does not: any claim about the ranking at maximum effort, on long tasks, or on 2.1. The full-benchmark 74.2% is one attempt per task.

The harness versions were previews and release candidates. DeepSeek's harness warns of breaking changes; results on later versions may differ.

The engine comparison is one fork build, one SGLang configuration with grammar disabled, and abstention measured on 12 items at three attempts. Throughput figures are single runs per cell. We did not try the fork with its adaptive speculative verification turned off, which is one plausible cause of the behaviour difference.

Neither comparison isolates the model's ability. They measure systems, which is the point.

9. What We Changed

Ante is now the batch and autonomous harness for this model, with the seat added to its provider catalogue; DeepSeek's harness is not adopted, and interactive work stays where it was. Any comparison we run between harnesses or engines now uses at least three attempts per task before we state a ranking, and counts infrastructure failures separately from agent failures. The abstention items are part of the acceptance check for any future engine or seat change, because that is the behaviour a faster engine changed without warning.

10. Methods and Artefacts

Benchmark: Terminal-Bench 2.0 run through Harbor 0.24.0; 24 tasks drawn with seed 20261018 from the 73 with an agent timeout of 30 minutes or less, plus a full 89-task run. Harnesses: DeepSeek Harness 0.2.0-rc.2 (minimal profile, our own Harbor adapter, session-log upload, anonymous identifier and plugin inventory disabled), Claude Code 2.1.302, Ante 0.preview.75. A per-harness proxy forced reasoning effort "high" and temperature 1.0 and counted requests and tokens; seat contamination was checked against the engine's request counter. Statistics: paired exact McNemar tests, task-level bootstrap intervals, sign tests and Wilson intervals. Engines: SGLang with the model's speculative head and grammar backend disabled; a community vLLM fork for DeepSeek-V4.1 on workstation Blackwell, by image digest. Hardware: one node with four NVIDIA RTX PRO 6000 Blackwell Max-Q GPUs at 325 W.

Per-trial Harbor results, proxy logs, engine counters, the tool-API check client, the concurrency bench and every scoring script are retained.

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-2026-041. The measurements were taken on 18 October 2026, with the abstention re-test on 21 October 2026; per-trial results, proxy logs, engine counters and scoring scripts are retained.