ANUSANDHAN · FINDINGS REPORT

Deep research by a team of four small language models: measured, step by step

Findings from 15 measured Anusandhan research runs, two controlled reader tests and two weeks of tuning: accuracy, speed, cost and the lessons that got it to a steady state

15 research runs · 579 model calls · 2,300 claims

Hiren · www.hirenonline.com
3 October 2026 · version 1.1
Figures frozen 3 Oct 2026 from the per-phase measurements of every run; nothing in this report leaves Hiren's AI Lab to be measured

Summary

Anusandhan (Sanskrit for investigation) is a deep-research tool that runs entirely on small language models (SLMs) in **Hiren's AI Lab**, a single home server with no GPU. It is given a research brief. It then plans searches, ranks what the web returns, reads the pages, writes a cited report and critiques that report, using four local models for the six steps. The only things that leave the lab are the search queries and the page fetches. No cloud model is involved at any step.

This report covers the **15 completed research runs** that exist today, run between 2 and 3 October 2026: three topics, each at three depths, with two choices of reading model. It also covers **two controlled A/B tests** of the reading model, and the **two weeks of measurement and tuning** (21 September to 3 October 2026) that it took to reach the current, steady configuration. Every model call in every run was measured: time, tokens, speed, truncation, CPU time, memory and energy. Every phase also has a mechanical effectiveness score, and no model grades another model.

You can read about the lab and see Anusandhan's architecture at **www.hirenonline.com/hiren-on-ai#anusandhan**. All published reports are at **www.hirenonline.com/reports**.

The findings

1. **Small models can do grounded research if every claim is checked mechanically.** Across all 15 runs, 233 pages were read and 2,300 claims extracted. **2,041 (88.7 %)** passed two code-based checks: the quoted evidence is on the page, and every number in the claim is in or near that quote. A claim that fails either check never reaches the report.
2. **The reading model decides accuracy.** With the same checks in place, llama3.2 (3B) verified **84.7 %** of its claims over nine runs, and gemma4 (8B) verified **94.6 %** over six. Rejected claims fell from 15.3 % to 5.4 %, and quotes not found on the page fell from 10.5 % of claims to 2.9 %.
3. **The bigger reader was not much slower.** In the controlled tests, gemma4 took +4 % and +37 % longer to read the same pages. In whole runs, read time rose only **+16 %** and total run time **+9 %**, because llama3.2 spends much of its speed on claims the checks then throw away. Per verified figure, gemma4 is cheaper: **67 s** against **86 s**.
4. **Restraint matters more than volume.** On error pages, menus and abstract-only pages, llama3.2 filled its claim allowance with invented or useless statements. gemma4 extracted nothing from such pages. Claims from pages under 500 words verified at **64 %** with llama3.2 and **99 %** with gemma4.
5. **More verified figures make better reports.** gemma4 runs carried **3.38 verified figures per page**, against 2.11 for llama3.2. Their reports were 18 % longer, with 29 % more data-point rows. They were judged better in 4 of 6 paired comparisons and equal in the other 2.
6. **The new weak point is relevance, not truth.** A faithful reader extracts accurately from whatever it is given, including off-topic pages: a wellness-magazine page in a haptics report, a factory touch-panel catalogue in a space report. Verification cannot catch a claim that is true and irrelevant.
7. **The critic is the weakest stage.** Only **19 %** of the qwen3.5 critic's flags agree with the mechanical checks. It occasionally makes a catch no rule can, but most of its flags are noise. The checks, not the critic, are what protect accuracy.

8. **Reading dominates time and cost.** The read phase takes **52–71 % of a run's CPU time**. A deep run takes **1 h 56 min – 2 h 6 min** with gemma4 reading, a standard run 70–80 min, and a quick run 36–39 min. All 15 runs together used 117 CPU-hours and an estimated **0.56 kWh**, about 37 Wh per run.
9. **Most problems were found by measuring, not by reading the output.** The tool reached its current state through 20+ measured changes. Among them: a report that was the single word "Microsoft", a critic that never finished thinking, a prompt limit the model ignored, and a reader that invented statistics on error pages. Each was found from a number in the run records.

Recommendations in one table

If you deploy small models for…	Do this	Because (from this report)
Extracting facts from documents	Use the strongest model you can afford as the **reader**, and verify every quote and number in code	Accuracy came from the reader (84.7 % → 94.6 %); the checks caught errors but could not create good claims
Keeping cost down	Compare models by **time per verified result**, not time per call	The 8B model was cheaper per verified figure than the 3B one
Structured output	Put limits in the **schema** (`maxItems`, enums), not the prompt	"The 10 most important claims" was ignored on 9 of 24 pages; a schema limit was obeyed every time
Long inputs	Give each phase the **context window it needs**, and detect when context, not output length, stopped a call	One report was a single word because its prompt filled the context and left one token to answer
Self-review	Do not rely on a small model to judge another; use **mechanical checks** first	Critic precision was 19 %; the quote and figure checks did the real work
Deciding between models	**A/B on the same pages, then confirm on whole runs and a second topic**	One early result reversed within the hour on a second topic

What Anusandhan is

The idea

Anusandhan answers a research question the way a careful analyst would. It works out what to look for, finds and ranks sources, reads them, extracts claims with evidence, writes a cited report and reviews it. It does this on hardware that fits under a desk, with models small enough to run on a CPU. It is Hiren's private tool, sitting behind his password with Gotia, his private assistant; findings can be filed into Gotia's memory and later removed again (unlearned). This report is about the research itself, not the filing. No run described here was filed.

Hiren's AI Lab

Everything runs in **Hiren's AI Lab**: one home server with a 12-thread CPU, 30 GB of RAM and **no GPU**. It reaches the internet only through an outbound tunnel and has no open inbound ports. The

same machine hosts the other projects on www.hirenonline.com: Ramu, the strict retrieval assistant at www.hirenonline.com/ramu; Gotia, the private assistant; and Shastrarth, where small models debate at shastrarth.hirenonline.com. Its models are served locally by Ollama, all 4-bit quantised. The lab runs one research job at a time and pauses it whenever a visitor starts a Shastrarth debate, so visitors always come first.

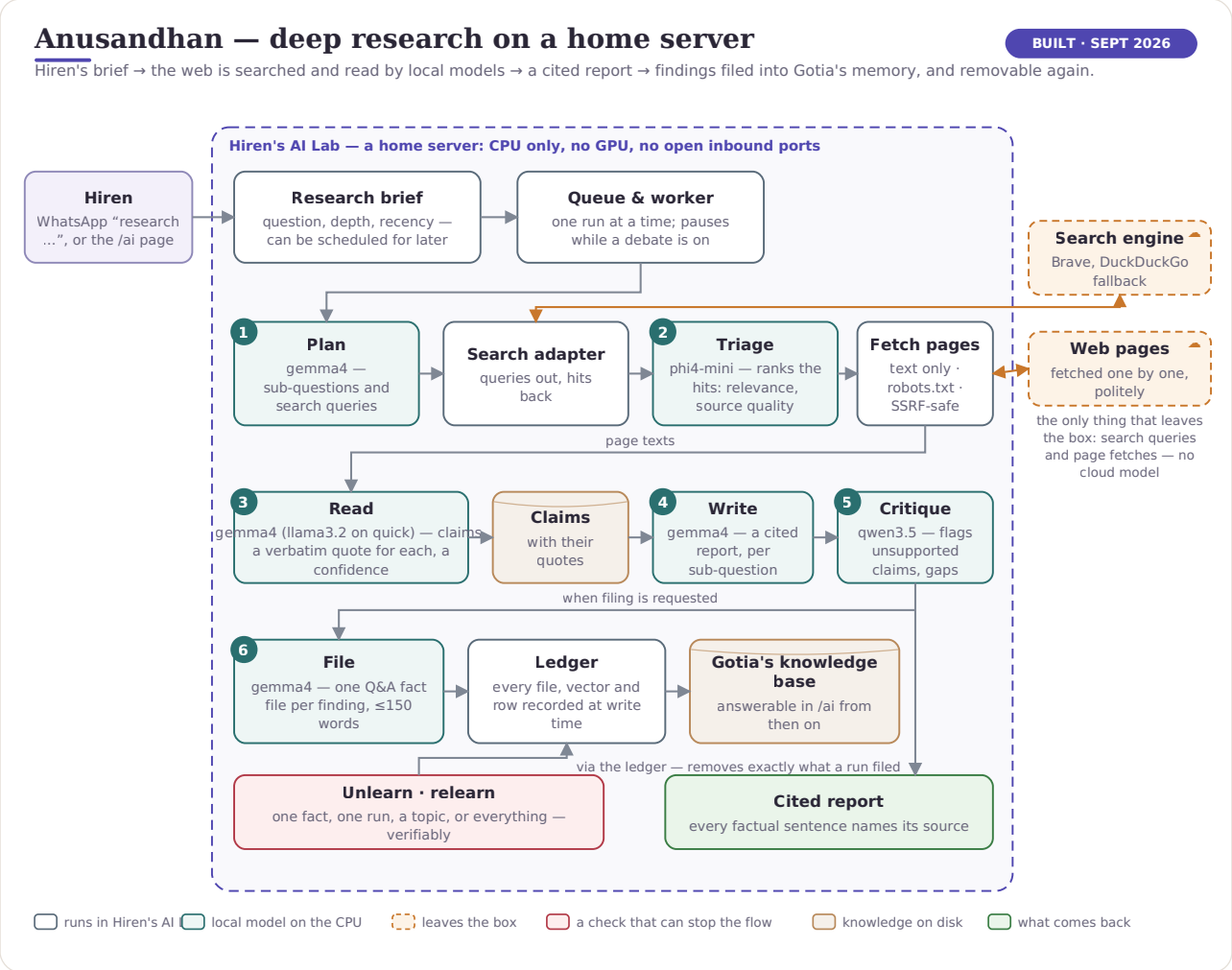


Figure 1. Anusandhan's architecture, as published at www.hirenonline.com/hiren-on-ai#arch-anusandhan, updated to the current reading model. Only search queries and page fetches leave the lab.

The six phases and the model behind each

Phase	Model (size)	What it does	Output
1 Plan	gemma4:e4b (8B)	Turns the brief into 4 specific sub-questions, with 3 search queries each	sub-questions, queries
2 Triage	phi4-mini (3.8B)	Scores every search hit 0–3 for relevance and source quality, and picks the shortlist	6 / 12 / 24 pages
— Fetch	(no model)	Downloads page text politely: robots.txt, rate-limited, SSRF-safe. Detects block pages; replaces unreadable pages with the next-ranked hit	page texts
3 Read	**gemma4:e4b** (deep, standard) · llama3.2 (3.2B, quick)	Per page: up to 14 claims, each with a verbatim quote and a date	claims with quotes
— Verify	(no model)	**Quote check:** is the quote on the page (near-misses repaired to the page's own words)? **Figure check:** is every number in the claim in or within 500 characters of its quote?	verified claims
4 Write	gemma4:e4b	One section per sub-question, 600–900 words. Every factual sentence cites its source; code builds a table of every verified figure	cited report
5 Critique	qwen3.5:4b (4.7B)	A different model reviews each section for claims their quote does not support and sentences no claim backs	review notes
6 File	gemma4:e4b	Drafts short question-and-answer fact files that can be filed into Gotia's memory	fact drafts

The fifth model in the lab, **mistral** (7.2B), has no role. In the Shastrarth benchmark it was the slowest and was beaten by gemma4 on every measure.

Why split the work across models? Because reading dominates. A deep run reads ~24 pages of up to 5,000 words each. That is about four times the reading of all the other phases combined, and the planner and writer, where quality matters most, take only a few calls each. Splitting roles lets each step use the cheapest model that does it well. It also means the critic is never the same model as the writer: Shastrarth showed that a model judging its own kind is biased.

The three presets

Preset	Pages	Reader	Critique	Time budget (cap)	Measured run time
Quick	6	llama3.2	no	40 min	36–39 min
Standard	12	gemma4	yes	100 min	70–80 min (gemma4), 60–74 min (llama3.2)
Deep	24 (+ a gap round if evidence is thin)	gemma4	yes	180 min	116–126 min (gemma4), 90–124 min (llama3.2)

A run that passes its cap does not stop. It skips its critique, so the cap protects the schedule, not the accuracy.

How everything was measured

Every model call is recorded: its model, tokens in and out, wall time, and Ollama's own split of load time, prompt reading and generation. It also records whether it stopped normally or ran out of room (truncated) and whether its structured output could be parsed. Every phase also records CPU time, peak memory and energy, read from the model server's own resource counters, plus three counts that measure **effectiveness**: items in, items out, and items good.

No model grades another model. "Good" is decided by code wherever possible, by the critic only in phase 5 (and then checked by code), and by Hiren in phase 6.

The Anusandhan statistics table, column by column

This is the per-model table that Hiren's monitoring page shows for Anusandhan, and that every run's report carries for its own run. One row per phase and model.

Column	What it is	How to read it
Phase	The pipeline step (Plan, Triage, Read, Write, Critique, File)	A model working in two phases has two rows, so a good planner cannot hide a poor writer
Model	The SLM that did the work	gemma4 appears in four phases
Runs	Runs contributing to the row	Only in this report's aggregate tables
Calls	Model calls made	Reading is one call per page; writing is one per section plus one for the summary
Prompt tok/s	Tokens of input the model reads per second, for the **median** call	Reading speed. Long prompts dominate this workload, so this often matters more than generation speed
Gen tok/s	Tokens of output generated per second, median call	Writing speed. On a CPU these models generate 9–12 tokens a second
Cold loads	Calls that first had to load the model into memory (load > 1 s)	The cost of switching models. Here about one per phase per run, which is expected
Median s/call	Wall-clock seconds of the median call	What a user waits per step. The median ignores one-off outliers
Failed	Calls that returned an error	Zero across all 15 runs
Trunc %	Share of calls that ended because they ran out of output or context room	Above ~5 % means a budget is wrong. A truncated answer looks like a short answer unless counted
Parse-fail %	Share of structured (JSON) answers nothing could be recovered from	A rising value is what a silent model update looks like
Effectiveness	good ÷ out, with a named measure per phase (next table)	The quality number. Each phase's measure is different and not comparable across phases
CPU-s/call	CPU seconds per call, across all 12 threads	~1,000 CPU-s for a 3–4 minute call means the model keeps about 5 of 12 threads busy
J/call	Energy per call, in joules (whole CPU package)	An upper bound: it includes the idle machine. Estimated where the energy counter wrapped (see Limitations)
Human edits	Changes Hiren made to a phase's output before it continued	Zero in these runs: none was edited by hand

Effectiveness, phase by phase

Phase	Measure	good ÷ out means	All 15 runs	What the number can and cannot tell you
Plan	**Sub-question yield**	sub-questions that ended with at least one verified claim ÷ sub-questions planned	**100 %** (60 / 60)	Every planned question found evidence. It does not say the questions were the best ones to ask
Triage	**Triage precision**	shortlisted pages that were fetched and yielded a claim ÷ shortlisted	**96.6 %** (226 / 234)	Good at finding readable, on-topic-looking pages. It judges from title and snippet, so it cannot see that a page is a vendor brochure
Read	**Quote fidelity**	claims passing the quote **and** figure checks ÷ claims extracted	**88.7 %**: llama3.2 84.7 %, gemma4 94.6 %	The key accuracy number. It measures faithfulness to the page, not relevance to the question
Write	**Citation coverage**	factual sentences carrying a resolvable source tag ÷ factual sentences	**85.2 %** (1,262 / 1,482)	Most of the uncited sentences are the Limits paragraphs, which carry no tag by design. It does not check that the cited source says what the sentence says (the critic's job)
Critique	**Critic precision**	flags the mechanical checks agree with ÷ flags raised	**19.2 %** (15 / 78)	Low. Part of this is by design, since a critic should also see what rules cannot, but most of it is noise
File	**Survival**	facts still filed 30 days later ÷ facts filed	pending	Nothing from these runs was filed into Gotia's memory, so there is nothing to survive yet

Read-quality measures used in this report

Measure	Definition
Claims	Statements the reader extracted, each with a quote, capped at 14 per page and scaled down on short pages (1 claim per 25 words, minimum 3)
Verified	Claims passing both checks. Verified % = verified ÷ claims
Repaired	Verified claims whose quote differed from the page by a word or two (similarity ≥ 0.92) and was replaced with the page's exact text
Not on page	Rejected: the quote is not on the page, even approximately. A misquote or an invention
Figure check failed	Rejected: the claim carries a number that is not in its quote or within 500 characters of it
No quote	Rejected: the reader gave no evidence at all
Verified figures	Verified claims containing at least one number. Figures are what reports put forward, so this measures how useful the reading was
Pages all rejected	Pages where every claim failed: the signature of a reader padding a page that had nothing to give
Pages with no claims	Pages from which the reader extracted nothing. With a careful reader this is often the correct answer

The road to a steady state: lessons learnt

Anusandhan was designed and approved on 21 September 2026, built by 26 September, and has been measured and tuned since. The measurements were wiped twice, on 27 September and 1 October, so that each configuration was judged on clean data. The run before those wipes (run 1, deleted on 2 October) and the runs before the first wipe survive only as the figures quoted below. This section is the story of what the numbers showed and what was changed.

1. The first estimates were wrong by a factor of two (21–23 September)

The design estimated a standard run at **19 minutes**. The first measured standard run took **44 minutes**. Reading alone was 2.6× the estimate, because the published speed figures assumed shorter contexts than real web pages need. After 16 runs, the measured medians were **7.8 min** (quick), **35.1 min** (standard) and **51 min** (deep). Time caps were then set from the worst observed run × 1.3, not from the median. A cap set near the median would quietly skip the critique on ordinary runs.

> **Lesson:** budget from measurements, and size a safety limit from the worst case, not the average.

2. The critic never finished thinking (23 September)

The critic (qwen3.5) was the one phase given a "thinking" pass, on the theory that a reviewer should reason. **Half its calls ended with an empty answer.** The model reasoned until its token

budget ran out and never reached its findings. Doubling the context only made the reasoning longer (16,481 characters). Turning thinking off produced usable reviews in 28–201 s instead of 534 s with nothing.

> **Lesson:** a small model's "thinking" mode can consume the whole budget. Measure `done_reason`, not just whether a reply arrived.

3. The first reader test, and the second topic that reversed it (23 September)

On ten pages about one technical topic, **llama3.2 verified 87 %** of its claims at 68 s per page. **qwen3.5 verified 57 %** at 131 s per page. Both produced the same number of verified claims, and llama3.2 did it in 71 % of the time, so llama3.2 became the reader. Within the hour, a run on a second subject saw llama3.2 verify only **24 %**.

> **Lesson:** quote fidelity is a property of **reader × corpus**, not of the reader alone. Test on more than one topic before generalising.

4. A deep run whose whole report was "Microsoft" (28 September)

A deep run read 21 pages and verified 156 claims. Its report was one word. The writer's prompt had filled the entire 8,192-token context, so the model had **one token** left to answer with. The monitoring alert blamed the output limit, but no output limit could have helped. The fix sizes the evidence to fit the context with room to answer, and recognises a call stopped by its context rather than its output cap.

> **Lesson:** there are two ways to run out of room, and they look identical in the output. Detect which one happened.

5. The reader invented sub-question numbers (29 September)

A 4-sub-question plan came back with claims filed under sub-questions 5 to 19. The fix was to make the read schema list the plan's real sub-question numbers as the only allowed values. The model's structured output can then only choose from them.

6. Reports were too thin, and five stages were not pulling their weight (1 October)

After two deep runs, Hiren found the reports high level. The numbers agreed: **156 verified claims became a 437-word report**, and the writer never saw the quotes that held the figures. 21 "claims" were sub-question headings copied back. Triage precision on one run was 13 / 24. The critic flagged sub-question numbers that did not exist. What changed:

- The writer now writes **one section per sub-question** with all of its evidence, and code appends a table of every verified figure. In a replay of the same run, report length went from **408 to 2,609 words** and factual sentences from 10 to 71.

- Echoed headings are dropped. The planner must ask questions evidence can answer, not topic headings. Unscored search hits are re-scored. Domains that repeatedly cannot be read are ranked last.
- The critic reviews each section. Code, not the critic, decides when evidence is "thin", which triggers a gap round in deep runs.
- Report conventions moved into a file Hiren edits. Nine recommendations were adopted the same day: units and as-of dates, a paragraph wherever sources disagree, an evidence line per section, a "thin evidence" note, a closing section on what the findings mean for the brief's purpose, and source dates and classes.

7. A limit in the prompt is a request; a limit in the schema is a rule (2 October)

Run 1 of the second clean slate verified only **50 %** of its claims, and **9 of 24** page reads were truncated. Three causes, three fixes:

- Asked for "the 10 most important claims", llama3.2 began 12–16 on dense pages and ran out of room. Putting the limit into the JSON schema as `maxItems` was obeyed every time it was tested.
- **44 of 111 rejected quotes** differed from the page by a word ("projected" for "expected"). Quote repair replaces such a quote with the page's own text, so a verified quote is still verbatim. It recovered 45 claims.
- **11 of 34 verified claims with a number carried a number their quote did not contain.** The figure check now rejects them. The critic had caught 2 of the 11.

Re-processed, run 1 rose from 50 % to 67 % verified. Run afresh under the new code, it reached **63 % verified with 1 of 24 reads truncated**, 70 % more checked figures, and finished 22 minutes sooner.

> **Lesson:** measure the denominator that matters. Quote fidelity was high while a third of the figure claims were unsupported, because the metric only looked at the quotes.

8. A plausible fix that lost (2 October)

Making the reader copy the quote before writing the claim should, in theory, reduce misquotes. On the same six pages it did the opposite. **Exact quotes fell from 41 to 11**, verified figure claims from 13 to 5, and the reader leaned on repair. It was faster, but worse at what reports are for, so it was not adopted.

> **Lesson:** test changes to a model's task on the same pages before adopting them, however good the rationale.

9. The reader's own context window, block pages and short pages (2 October)

- Pages that triage rated most relevant are read in a doubled window. With the prompt they took ~7,000 of 8,192 tokens, leaving the reader ~1,100 tokens to answer. **The reader now**

has its own 12,288-token context. On the test page this went from 12 claims with 8 verified to **14 claims, all 14 verified**.

- Three "pages" in one run were a medical database's Access Denied notice, and the reader made 40 claims from them, all rightly rejected. Block, captcha and cookie-wall pages are now detected and replaced. PubMed abstracts are fetched through the official API instead of the website.
- The claim cap now scales with page length, so a 200-word page cannot be asked for 14 claims.

10. Choosing the reader: two A/B tests and a whole-run confirmation (2–3 October)

This was the last open question, and the subject of the reader decision recorded in this report. It is set out in full under The reader decision below. In short: gemma4 won both controlled tests, on two different topics, on accuracy and on restraint. Nine production runs with llama3.2 then set the baseline. Six re-runs with gemma4 confirmed the gain on whole runs at about half the time cost projected.

Ten learnings, kept so they are not re-derived

1. **Measure the denominator that matters.** A metric can be high while what it protects is failing.
2. **A prompt limit is a request; a schema limit is a rule.** Where the model server's grammar can enforce a shape (enum, maxItems, property order), enforce it.
3. **There are two ways to truncate, and they look the same.** Running out of context is not running out of output.
4. **Near-misses are not fabrications.** Repairing a one-word misquote to the page's own text loses nothing; rejecting real misquotes is still right.
5. **The critic's value is judgement, not volume.** Few of its flags agree with the rules, but some catch what no rule can, such as a 2016 study presented as a recent breakthrough.
6. **A plausible fix can lose.** Test it on the same pages first.
7. **Run long tests where the session cannot kill them.** Detached processes that write their results to a file.
8. **Re-processing has limits.** A phase that is not re-run keeps its old measurements.
9. **Look where the losses are, not at the average.** Six thin or blocked pages produced three quarters of one run's rejected claims.
10. **The model that writes less can be the faster one.** Compare models by time per verified result, not per page.

Results across all 15 runs

Fifteen runs completed between **2 October 07:50 and 3 October 14:12 JST**: three topics (kinesthetic haptic feedback, space teleoperation, and digital twins for VR, AI and automation simulation), each at all three depths with llama3.2 reading (runs 2–10, pipeline 1.30). Then the deep and standard runs were repeated from the same briefs with gemma4 reading (runs 11–16,

pipeline 1.31). Every brief asked for a report for Hiren himself, with a stated purpose, preferring primary sources. Run 1 was deleted; its figures appear only in the history above.

The runs at a glance

Run	Topic	Preset	Reader	Started (JST)	Minutes	Cap	Pages read	Claims	Verified	Figures	Report words	Citations
2	Haptics	deep	llama3.2	10-02 07:50	89.5	180	24 of 29	214	78.5%	33	4,459	180
3	Space teleop.	deep	llama3.2	10-02 10:30	110.7	180	24 of 30	268	83.6%	57	5,762	212
4	Digital twins	deep	llama3.2	10-02 15:03	124.0	180	24 of 33	273	90.1%	51	5,767	219
5	Haptics	standard	llama3.2	10-02 17:22	71.1	100	12 of 17	117	91.5%	29	4,006	148
6	Haptics	quick	llama3.2	10-02 18:38	38.2	40	6 of 7	55	90.9%	19	3,334	118
7	Space teleop.	standard	llama3.2	10-02 19:17	60.1	100	12 of 13	138	85.5%	18	3,632	138
8	Space teleop.	quick	llama3.2	10-02 20:18	36.4	40	6 of 6	62	80.6%	14	3,073	135
9	Digital twins	standard	llama3.2	10-02 20:55	73.9	100	12 of 15	158	76.6%	23	4,143	166
10	Digital twins	quick	llama3.2	10-02 22:12	39.2	40	6 of 8	71	90.1%	22	3,686	120
11	Haptics	deep	gemma4	10-03 04:13	116.0	180	23 of 31	172	94.2%	72	6,224	240
12	Space teleop.	deep	gemma4	10-03 06:11	124.3	180	24 of 28	229	95.6%	81	7,086	256
13	Digital twins	deep	gemma4	10-03 08:17	126.2	180	24 of 27	222	92.8%	54	6,368	250
14	Haptics	standard	gemma4	10-03 10:26	69.8	100	12 of 13	106	95.3%	44	4,529	169
15	Space teleop.	standard	gemma4	10-03 11:37	72.5	100	12 of 15	119	95.0%	50	4,190	159
16	Digital twins	standard	gemma4	10-03 12:51	80.2	100	12 of 13	96	95.8%	37	4,366	181

Minutes = the sum of the six phases' wall time, excluding any pause for a visitor's debate. Pages read = pages fetched of pages tried; the difference is pages that were blocked, paywalled or timed out and were replaced. Figures = verified claims carrying a number. Report words exclude the sources list and the review.

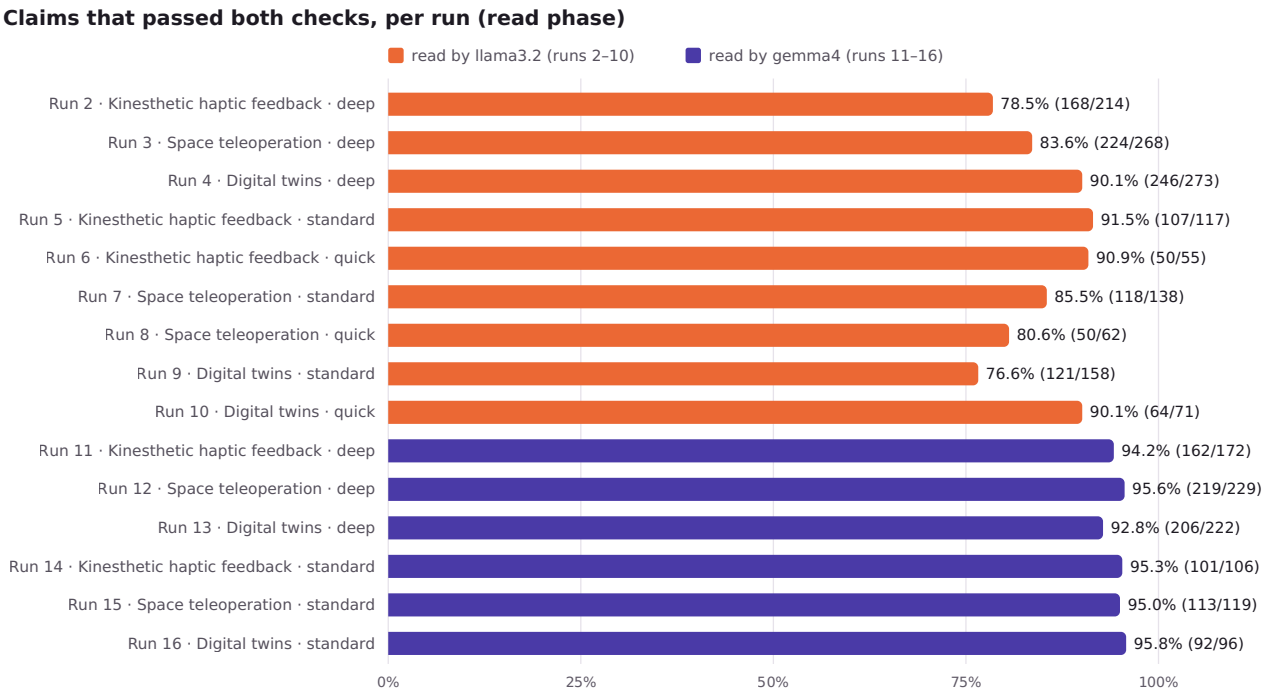


Figure 2. Every gemma4 run verified more than 92 % of its claims. The llama3.2 runs ranged from 76.6 % to 91.5 %.

Totals

	llama3.2 reading (runs 2–10)	gemma4 reading (runs 11–16)	All 15 runs
Runs (deep / standard / quick)	3 / 3 / 3	3 / 3 / 0	6 / 6 / 3
Total run time	10 h 43 min	9 h 49 min	20 h 32 min
Pages fetched of tried	126 of 158	107 of 127	233 of 285
Words read	380,520	288,859	669,379
Claims extracted	1,356	944	2,300
Claims verified	1,148 (**84.7 %**)	893 (**94.6 %**)	2,041 (88.7 %)
Rejected: quote not on page · figure check · no quote	143 · 34 · 31	27 · 24 · 0	170 · 58 · 31
Quotes repaired	265	54	319
Verified figure claims (per page with claims)	266 (2.11)	338 (3.38)	604 (2.67)
Read seconds per verified claim · per verified figure	19.9 · 86.1	25.4 · 67.1	22.3 · 75.4
Report words · citations · data-point rows	37,862 · 1,436 · 335	32,763 · 1,255 · 361	70,625 · 2,691 · 696
Tokens in · out	0.90 M · 0.18 M	0.73 M · 0.16 M	1.64 M · 0.35 M
CPU hours · energy (est.)	60.6 · 0.29 kWh	56.6 · 0.27 kWh	117.3 · 0.56 kWh

The statistics table: all 15 runs

Phase	Model	Runs	Calls	Prompt tok/s	Gen tok/s	Cold loads	Median s/call	Failed	Trunc %	Parse-fail %	Effectiveness	CPU-s/call	J/call (est.)	Human edits
Plan	gemma4	15	15	53.4	10.7	4	58.4	0	0.0	0.0	100% sub-question yield	255	1,440	0
Triage	phi4-mini	15	130	61.5	12.2	15	23.4	0	0.0	0.0	96.6% triage precision	124	603	0
Read	llama3.2	9	126	57.0	9.4	6	185.8	0	0.0	0.0	84.7% quote fidelity	1,076	5,038	0
Read	gemma4	6	107	46.3	9.1	6	217.0	0	0.9	0.9	94.6% quote fidelity	1,238	5,822	0
Write	gemma4	15	90	46.4	9.3	15	173.3	0	0.0	0.0	85.2% citation coverage	947	4,516	0
Critique	qwen3.5	12	96	49.9	10.3	12	68.4	0	1.0	0.0	19.2% critic precision	367	1,764	0
File	gemma4	15	15	46.0	9.2	0	155.8	0	0.0	0.0	pending (survival)	926	4,349	0

How to interpret it.

- **Reliability is not the problem.** 579 model calls produced no errors. Only one read and one critique call in the whole set were truncated, and one structured answer could not be parsed: a single 38 KB market-report page that filled gemma4's 12,288-token reading window, so the page gave nothing.
- **Speed is similar across the four models.** It is 46–62 tokens a second reading a prompt and 9–12 a second generating. phi4-mini is the fastest of the four, which is why it does triage, the phase with the most calls. llama3.2 reads prompts 23 % faster than gemma4 (57.0 against 46.3 tok/s), and generates at the same speed here because it reads with a long context.
- **Wall time follows the work, not the model.** A read call takes 3–4 minutes with either reader. A plan call takes a minute. Most of a run's time is the number of calls in the read phase multiplied by those 3–4 minutes.
- **Cold loads are about one per phase per run.** This is the expected cost of moving between four models. Since gemma4 took over reading, the writer usually starts with gemma4 already loaded.
- **The effectiveness column ranks the phases by how much they can be trusted.** Planning (100 %) and triage (96.6 %) almost always succeed at what they are asked. Reading is good and became very good with gemma4. Writing cites 85 % of its factual sentences. The critique, at 19 %, is the weakest link.
- **Cost per call** ranges from ~2 CPU-minutes for triage to ~20 CPU-minutes for a gemma4 page read. At about 4.9 J per CPU-second, a gemma4 page read costs ~5.8 kJ, roughly what a 100 W bulb uses in a minute.

The same table split by reading model, so each row compares like with like:

Runs	Phase	Model	Calls	Prompt tok/s	Gen tok/s	Median s/ call	Trunc %	Effectiveness	CPU-s/ call	J/call (est.)
2–10	Plan	gemma4	9	53.1	10.7	61.0	0.0	100%	261	1,500
2–10	Triage	phi4-mini	76	61.6	12.3	23.4	0.0	100%	121	589
2–10	Read	llama3.2	126	57.0	9.4	185.8	0.0	84.7%	1,076	5,038
2–10	Write	gemma4	54	46.8	9.4	171.0	0.0	85.1%	877	4,213
2–10	Critique	qwen3.5	48	50.1	10.7	59.8	0.0	27.3%	323	1,570
2–10	File	gemma4	9	46.3	9.3	155.3	0.0	pending	924	4,330
11–16	Plan	gemma4	6	53.9	10.7	57.9	0.0	100%	248	1,349
11–16	Triage	phi4-mini	54	61.4	12.2	27.0	0.0	92.6%	127	622
11–16	Read	gemma4	107	46.3	9.1	217.0	0.9	94.6%	1,238	5,822
11–16	Write	gemma4	36	46.3	9.2	195.8	0.0	85.2%	1,051	4,970
11–16	Critique	qwen3.5	48	49.6	10.2	74.9	2.1	13.3%	411	1,957
11–16	File	gemma4	6	46.0	9.2	157.2	0.0	pending	928	4,376

Triage precision in runs 11–16 (92.6 %) is lower than in runs 2–10 (100 %) mostly for a reason that flatters the new reader: gemma4 correctly extracted nothing from 7 shortlisted pages (generic market and selection-guide pages), and a page with no claim counts against triage. Write time per call rose with gemma4 reading because there was more evidence per section to write from.

Where the time goes

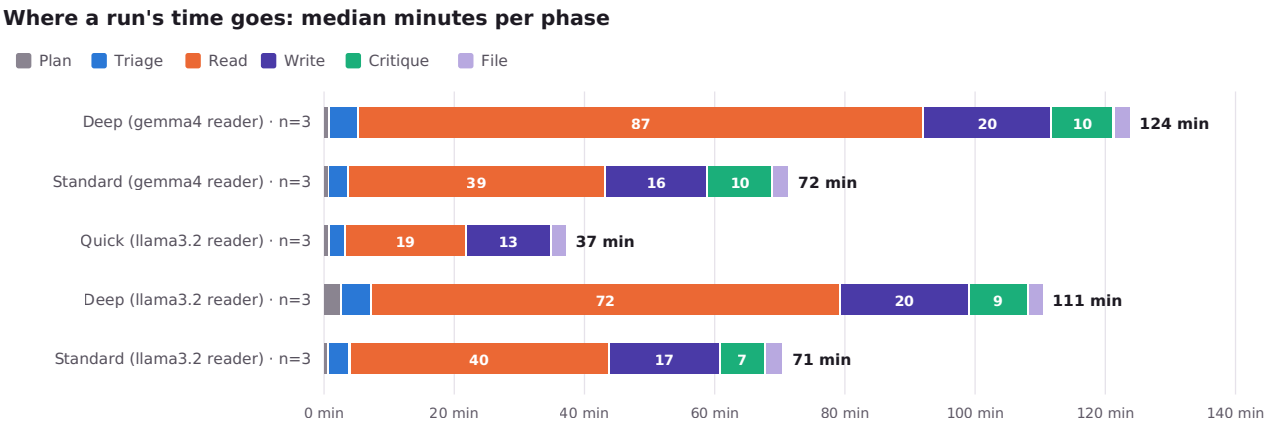


Figure 3. Reading takes 49–70 % of every run's time. Planning and triage together take under 6 minutes, whatever the preset.

Run	Preset	Plan	Triage	Read	Write	Critique	File	Total	Elapsed	Model loading	% of cap
2	deep	1.5	3.6	55.6	17.9	8.0	2.9	89.5	93.6	3.3	50%
3	deep	2.8	4.6	72.0	19.8	9.1	2.4	110.7	112.7	5.2	62%
4	deep	2.9	4.7	84.0	20.2	9.9	2.3	124.0	126.0	5.9	69%
5	standard	1.0	2.8	39.8	17.4	7.5	2.6	71.1	76.1	3.8	71%
6	quick	1.0	2.4	18.6	13.3	—	2.9	38.2	38.7	2.7	96%
7	standard	0.8	3.4	29.1	17.1	6.9	2.8	60.1	60.8	2.7	60%
8	quick	0.8	2.4	17.8	13.0	—	2.5	36.4	36.9	2.5	91%
9	standard	0.8	3.3	44.6	15.4	7.0	2.7	73.9	77.2	2.9	74%
10	quick	1.0	2.7	20.2	13.1	—	2.3	39.2	40.1	2.4	98%
11	deep	1.2	4.0	75.4	19.6	13.5	2.4	116.0	118.0	2.3	64%
12	deep	0.9	4.7	86.8	21.8	7.4	2.6	124.3	125.8	2.5	69%
13	deep	1.0	4.5	88.6	19.6	9.6	2.9	126.2	128.6	2.3	70%
14	standard	1.0	2.9	39.5	17.3	6.7	2.4	69.8	70.6	3.6	70%
15	standard	0.9	3.0	38.4	15.7	11.8	2.6	72.5	73.4	3.9	72%
16	standard	0.7	3.3	49.0	14.6	10.0	2.7	80.2	81.2	3.4	80%

All times in minutes. Elapsed = first phase start to last phase end, including the time between phases. Model loading = time spent loading models into memory, included in the phase times. Fetching is inside Read. Quick runs have no critique by design.

Interpretation. Deep runs take 1.5–2 hours, standard runs just over an hour, and quick runs about 40 minutes. Read time grows with pages × page length: run 4's 24 pages held 88,937 words and took 84 minutes, while run 2's 57,203 words took 56. With gemma4 reading, a deep run's read phase rose by 4.6–19.8 minutes and a standard one by −0.3 to +9.3 minutes. Write time is steady at 15–22 minutes because its size is fixed: four sections plus a summary. **Quick runs used 91–98 % of their 40-minute cap**, which leaves too little margin (see Recommendations).

Accuracy in detail

Run	Reader	Pages with claims	Claims	Verified	Repaired	Not on page	Figure check failed	No quote	Verified figures	Figures per page	Pages all rejected	Pages with no claims
2	llama3.2	24	214	168 (78.5%)	16	31	5	10	33	1.38	3	0
3	llama3.2	24	268	224 (83.6%)	40	28	8	8	57	2.38	2	0
4	llama3.2	24	273	246 (90.1%)	66	15	5	7	51	2.12	0	0
5	llama3.2	12	117	107 (91.5%)	28	10	0	0	29	2.42	0	0
6	llama3.2	6	55	50 (90.9%)	8	0	1	4	19	3.17	0	0
7	llama3.2	12	138	118 (85.5%)	36	16	4	0	18	1.50	0	0
8	llama3.2	6	62	50 (80.6%)	12	11	0	1	14	2.33	0	0
9	llama3.2	12	158	121 (76.6%)	41	25	11	1	23	1.92	1	0
10	llama3.2	6	71	64 (90.1%)	18	7	0	0	22	3.67	1	0
11	gemma4	19	172	162 (94.2%)	5	4	6	0	72	3.79	0	4
12	gemma4	24	229	219 (95.6%)	11	8	2	0	81	3.38	0	0
13	gemma4	24	222	206 (92.8%)	21	11	5	0	54	2.25	0	0
14	gemma4	11	106	101 (95.3%)	1	0	5	0	44	4.00	0	1
15	gemma4	12	119	113 (95.0%)	5	2	4	0	50	4.17	0	0
16	gemma4	10	96	92 (95.8%)	11	2	2	0	37	3.70	0	2

Interpretation.

- **llama3.2's errors are mostly misquotes** (143 quotes not on the page). It also relies on repair: 265 of its verified claims needed a word fixed, against 54 for gemma4.
- **gemma4's remaining errors are mostly numbers** (24 figure-check failures, about half of its rejections). When it errs, it misquotes a figure rather than inventing a passage.
- **Pages all rejected** is llama3.2's padding signature: seven pages lost every claim, including a PubMed abstract, an arXiv abstract page, a staff biography and a consultancy menu. gemma4 has none. Its **pages with no claims** (7) are the same kind of page, correctly left alone. One of them, in run 16, was a failure rather than restraint (the context overflow above).
- **Topic matters less than the reader.** The digital-twin topic gave llama3.2 one of its best runs (90.1 %) and its worst (76.6 %). gemma4 stayed between 92.8 % and 95.8 % on every topic.

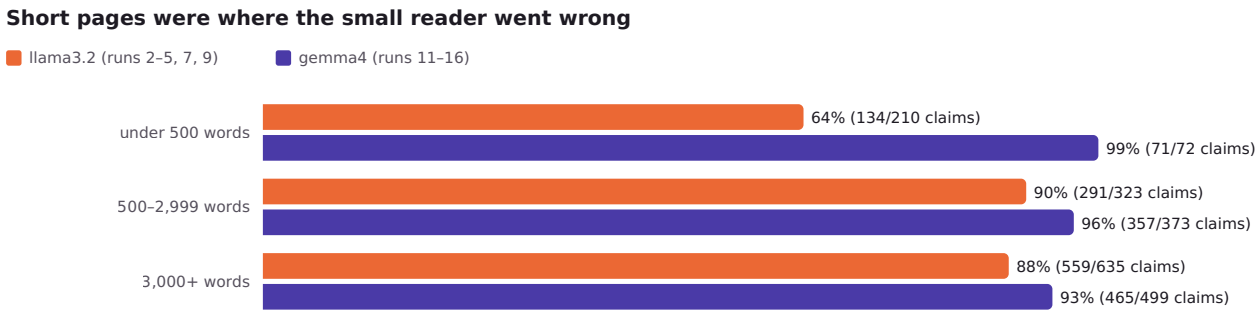


Figure 4. Deep and standard runs only, so both readers cover the same presets. Short pages separated the two readers; on long pages both did well.

Writing and review in detail

Run	Words	Citations	Sources cited	Data-point rows	Fact sentences cited	Critic flags	Flags agreeing with the checks	Unbacked sentences	Fact drafts
2	4,459	180	20	37	91 of 103 (88%)	5	5	1	6
3	5,762	212	22	81	90 of 112 (80%)	5	2	0	5
4	5,767	219	23	74	93 of 109 (85%)	5	1	6	5
5	4,006	148	12	33	71 of 87 (82%)	8	0	0	5
6	3,334	118	6	17	72 of 79 (91%)	—	—	—	6
7	3,632	138	12	18	85 of 97 (88%)	4	0	2	6
8	3,073	135	6	11	70 of 89 (79%)	—	—	—	5
9	4,143	166	11	36	85 of 95 (89%)	6	1	3	5
10	3,686	120	5	28	75 of 89 (84%)	—	—	—	6
11	6,224	240	19	79	83 of 104 (80%)	16	2	0	5
12	7,086	256	24	91	107 of 123 (87%)	4	0	0	5
13	6,368	250	24	67	102 of 124 (82%)	11	1	4	5
14	4,529	169	11	40	82 of 91 (90%)	3	0	0	5
15	4,190	159	12	41	74 of 92 (80%)	3	2	2	5
16	4,366	181	10	43	82 of 88 (93%)	8	1	0	5

Data-point rows: the code-built tables listing every verified figure in a section; no model can add a row. Flags agreeing with the checks: critic flags on claims the mechanical checks also rejected. Unbacked sentences: sentences the critic said no claim supports. Quick runs have no critique.

Interpretation. Report length tracks evidence. Deep reports with gemma4 reading run 6,200–7,100 words, against 4,500–5,800 with llama3.2. The writer cites 80–93 % of its factual sentences in every run, whichever model read. The critic is erratic: 16 flags on run 11 (seven of them near-duplicates about one paraphrase), 3 on runs 14 and 15. Its agreement with the checks

fell from 27 % to 13 % once gemma4 was reading, because there were fewer bad claims left to find.

Resources

Run	Preset	Tokens in	Tokens out	CPU hours	Energy (Wh, est.)	Read share of CPU
2	deep	136,647	26,477	8.5	40	64%
3	deep	153,809	32,397	10.5	49	68%
4	deep	179,468	32,685	11.7	55	71%
5	standard	101,150	18,258	6.7	32	58%
6	quick	49,797	11,185	3.5	17	52%
7	standard	81,814	19,334	5.7	27	51%
8	quick	48,008	10,796	3.4	16	52%
9	standard	102,919	20,722	7.0	33	63%
10	quick	51,300	12,050	3.6	17	55%
11	deep	153,567	30,710	11.3	53	66%
12	deep	147,834	36,439	12.1	57	71%
13	deep	161,890	34,565	12.3	58	71%
14	standard	85,375	19,664	6.6	31	57%
15	standard	75,804	23,694	6.8	32	53%
16	standard	110,043	19,787	7.6	36	62%

Interpretation. A deep run reads ~150,000 tokens and writes ~33,000. Here it costs 11–12 CPU-hours and ~55 Wh, about what a laptop uses in a working morning. The lab's case is privacy and control, not speed: no page text, claim or report ever leaves the machine. CPU-hours are summed across 12 threads, so 12 CPU-hours is about 2 hours of the machine at half load.

How each SLM performed

Model	Size	Role here	Measured strengths	Measured weaknesses	Verdict
gemma4:e4b	8.0B	Plan, Read (deep/ standard), Write, File	100 % sub-question yield; **94.6 %** verified reading; restraint on empty pages; 3.38 verified figures per page; 0 truncation in planning and writing	19 % slower prompt reading than llama3.2; occasional misquoted figures; one context overflow on a 38 KB page; as writer, invents years and turns "as of" dates into event dates	The workhorse. The best model in the lab for any step where quality matters
llama3.2	3.2B	Read (quick; deep and standard until 2 October)	Fastest reader per page; 87 % verified on quick runs; good on long, dense pages (87–94 %)	**Pads short pages** with invented claims (64 % verified under 500 words); misquotes (143 quotes not on the page); not deterministic (the same page went 0/14 then 3/9)	Good for speed when pages are long and the checks are strict; not for unattended accuracy
phi4-mini	3.8B	Triage	Fastest of the four (61.5 tok/s prompt, 12.2 gen); 96.6 % of its shortlisted pages yielded claims; 0 truncation, 0 parse failures in 130 calls	Judges by title and snippet, so it rated a vendor blog "primary / technical" and cannot see off-topic content	Ideal for many short classification calls
qwen3.5:4b	4.7B	Critique	Fast per call (68 s median); occasionally catches what no rule can (a 2016 study presented as recent)	**19 % precision**; near-duplicate flags; malformed output lines; in thinking mode it never finished (23 Sep)	Keep for its rare real catches, but do not trust its volume
mistral	7.2B	none	—	Slowest in the lab; beaten by gemma4 on every Shastrarth measure	Not used

These results agree with Hiren's earlier Shastrarth study of the same models, How five small language models argue (www.hirenonline.com/reports#shastrarth-findings-report). There, gemma4 argued best and held its position most firmly, and llama3.2 was the fastest. Size again predicts neither speed nor quality: the 8B gemma4 was cheaper per verified figure than the 3.2B llama3.2.

The reader decision

The question: should the 8B gemma4 replace the 3B llama3.2 as the reader, given that reading is two thirds of every run's time and cost? The decision was made in three gated steps, each with its pass mark set before the data came in.

Step 1 — A/B on six pages of one topic (2 October)

Same six pages of a haptics run, same prompt, same checks, same limits, one model after the other. The pages were chosen to include an Access Denied block page, a thin methods section, a long page and three ordinary pages.

	llama3.2	**gemma4**
Claims · verified	58 · 44 (**76 %**)	50 · 50 (**100 %**)
Exact quote · repaired	34 · 10	43 · 7
Quote not on page · figure check failed	6 · 3	0 · 0
Claims from the block page	1	0
Thin page	1 claim, rejected	4 claims, all verified
Verified figure claims	28	25
Read time	23.2 min	24.0 min (+4 %)

Decision: keep llama3.2 and judge after five more runs. One topic and six pages are not enough, and a reader result had already reversed once on a second topic (lesson 3).

Step 2 — nine production runs with llama3.2 (2 October), then a second A/B on another topic (3 October)

The nine runs verified **84.7 %**, under the 20 % rejection level set in advance as the case for switching. Their losses were concentrated on short pages (64 % verified under 500 words). So the A/B was repeated on six pages of the worst run, a market-heavy digital-twin topic. The pages included a consultancy menu, a dense short page (to see whether gemma4 would lose real claims), a figure-heavy market report and a control page llama3.2 had read perfectly.

	llama3.2	**gemma4**
Claims · verified	78 · 61 (**78 %**)	55 · 54 (**98 %**)
Exact quote · repaired	50 · 11	52 · 2
Quote not on page · figure check failed	15 · 2	1 · 0
Menu page (357 words)	14 claims: 4 invented percentages, 5 true but useless	**0 claims**
Dense short page (346 words)	9 of 12 verified	**10 of 10**
Verified figure claims	11	**20**
Read time	**17.0 min**	23.3 min (+37 %)
Seconds per verified claim · per verified figure	**16.7** · 93	25.8 · **70**

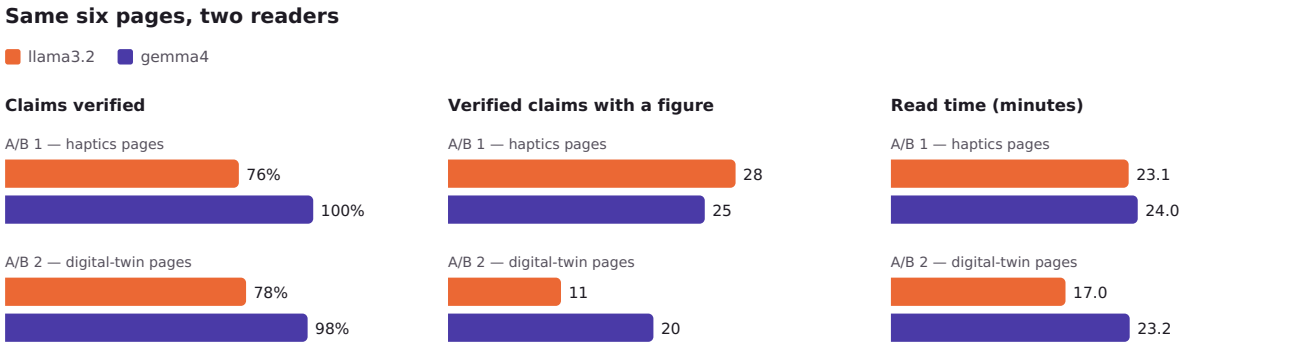


Figure 5. The accuracy result replicated on a second topic, and the small figure deficit of the first test reversed.

The pass mark, "switch only if gemma4 again verifies markedly more with no loss of figures", was met: +20 points verified, +9 figures. **Decision (3 October): gemma4 reads for deep and standard. Quick keeps llama3.2**, because quick exists for speed and has not been tested with gemma4.

Step 3 — six whole-run re-runs with gemma4 (3 October)

The six deep and standard briefs were run again from scratch. A fresh run searches again, so each pair shared only 1–4 of its 12 or 24 pages, and the comparison mixes the reader's effect with search luck.

Criterion set in advance	Result	Verdict
Verified share near the A/Bs' 98 %	94.6 % overall, 92.8–95.8 % per run; each run up ≥ 2.7 points on its pair	Met in substance
Verified figures per page ≥ llama3.2's 2.1	3.38 (lowest run 2.25)	Met
Read time near the projected +37 %	**+16 %** overall (deep +18.5 %, standard +11.9 %); total run time +9.4 %	Better than projected
Every run under its time cap, every critique run	Deep 116–126 of 180 min; standard 70–80 of 100; all 6 critiques ran	Met
Model-switching cost	Read-phase loading rose from 2 to 7 minutes over six runs, while writer loading fell from 9 to 3 minutes, because gemma4 was already resident	Net slightly lower

Report quality, judged by reading each pair: the gemma4 report was better in 4 pairs and equivalent in 2, and the llama3.2 report was not better in any pair. The gemma4 reports name more systems, studies and measured results. Their weakness is relevance: faithful extraction from off-topic pages, which verification cannot catch.

Recommendations from the reader review

1. **Keep gemma4 as the reader for deep and standard.** Every measurable criterion was met or beaten, and no paired report favoured llama3.2.
2. **Leave quick on llama3.2 for now.** Its own rejection rate (12.8 %) is the lowest of the three presets. A quick A/B is worth doing only if quick reports are seen to carry invented figures.
3. **Do not raise the standard time cap yet.** The worst standard run took 81 of 100 minutes. Re-check after the next 5–10 runs, and raise the cap to about 110 if any run passes 90 minutes. Do not cut pages.
4. **Keep the claim cap (14 per page) and the page-length scaling** either way. With llama3.2 they bound padding; with gemma4 they are headroom it does not abuse.
5. **Do not move other phases to a bigger model on this evidence.** Only reading was tested.
6. **Still open:** a deep run that needs a gap round, a check that Gotia's answers are not slowed while a run reads with the same model, and gemma4's run-to-run variation, which has not been measured.

Run by run

Each card gives the brief, the timing, the run's statistics and its highlights. Highlights are **the run's own key takeaways, as written by the models**: every one is backed by a verified quote, but this report has not re-checked the original pages. Where a reviewer read the reports side by side (all deep and standard runs), the main strengths and errors found are noted.

Topic 1 — Kinesthetic haptic feedback

Brief: "Discoveries and innovations on Kinesthetic Haptic Feedback systems". Purpose: understand the latest developments and industry demands. Recency: 6 months. Prefer primary sources; ieee.org allowed.

Run 2 · deep · llama3.2 reader

When: 2 Oct 07:50–09:23 JST · **89.5 min** of a 180-min cap (plan 1.5, triage 3.6, read 55.6, write 17.9, critique 8.0, file 2.9).

Statistics: 24 of 29 pages · 214 claims, **78.5 %** verified (31 not on page, 5 figure, 10 no quote) · 33 verified figures (1.38 a page) · 4,459 words, 180 citations of 20 sources · critic 5 flags, all 5 agreeing with the checks · 8.5 CPU-h, ~40 Wh.

Highlights: end-to-end latency must stay below 1 ms for critical tasks, with ~20 ms the general expectation · orthopaedic surgery is the most mature haptic application (e.g. Mako) · high-voltage electrostatic actuators are emerging as fast, silent alternatives.

Reviewer's note: mostly general statements and market sizes, with two contradictory market figures side by side; one section strays into consumer gaming hardware. The weakest of the haptics reports.

Run 11 · deep · gemma4 reader (re-run of run 2)

When: 3 Oct 04:13–06:11 JST · **116.0 min** of 180 (plan 1.2, triage 4.0, read 75.4, write 19.6, critique 13.5, file 2.4).

Statistics: 23 of 31 pages, 4 rightly left empty · 172 claims, **94.2 %** verified · **72 verified figures (3.79 a page)**, the most of any haptics run · 6,224 words, 240 citations of 19 sources · critic 16 flags, 2 agreeing · 11.3 CPU-h, ~53 Wh.

Highlights: HapMorph wearable interface stiffness 0.12–0.46 N/mm · a 5G teleoperation demonstration with 40 % higher force-feedback accuracy than 4G · an exoskeleton controller cutting interaction-torque error from up to 0.16 to ~0.07 Nm/kg.

Reviewer's note: full of named devices with measured results (37 → 79 data-point rows against run 2). Errors: one figure misattributed and carried into the conclusion, plus some off-topic padding (a patent lawsuit, exoskeleton products). **Better than run 2.**

Run 5 · standard · llama3.2 reader

When: 2 Oct 17:22–18:38 JST · **71.1 min** of 100 (plan 1.0, triage 2.8, read 39.8, write 17.4, critique 7.5, file 2.6).

Statistics: 12 of 17 pages · 117 claims, **91.5 %** verified · 29 verified figures (2.42 a page) · 4,006 words, 148 citations · critic 8 flags, 0 agreeing · 6.7 CPU-h, ~32 Wh.

Highlights: a multimodal device rendering 3.5–8.1 N of force and 10–55 °C · combined force-and-touch feedback identified stiffness with 92 % accuracy against 62 % for force alone.

Reviewer's note: good primary results but thin middle sections. It misframed a teleoperation test as "surgical simulation" and padded with VR-healthcare figures that are not about haptics.

Run 14 · standard · gemma4 reader (re-run of run 5)

When: 3 Oct 10:26–11:37 JST · **69.8 min** of 100 (plan 1.0, triage 2.9, read 39.5, write 17.3, critique 6.7, file 2.4).

Statistics: 12 of 13 pages, 1 rightly left empty · 106 claims, **95.3 %** verified · 44 verified figures (**4.00 a page**) · 4,529 words, 169 citations · 6.6 CPU-h, ~31 Wh.

Highlights: haptic bandwidths up to 275 Hz · surgical robotics as the main demand driver · force and kinesthetic feedback projected to grow at 13.4 % a year.

Reviewer's note: broader, kinesthetic-specific results (haptic thimble, force-regulation error, commercial devices with specifications). Its flaw is a wellness-magazine page that appears in all four sections. **Better than run 5**, and faster.

Run 6 · quick · llama3.2 reader

When: 2 Oct 18:38–19:17 JST · **38.2 min** of 40 (plan 1.0, triage 2.4, read 18.6, write 13.3, file 2.9; no critique).

Statistics: 6 of 7 pages · 55 claims, **90.9 %** verified · 19 verified figures (3.17 a page) · 3,334 words, 118 citations of 6 sources · 3.5 CPU-h, ~17 Wh.

Highlights: a shift from replicating touch towards conveying abstract "semantic" signals · a haptic ring producing up to 1.75 N · the haptic technology market projected at USD 9.54 bn by 2031.

Topic 2 — Space teleoperation

Brief: "Recent works on Space Teleoperation". Purpose: understand innovations and research in progress. Recency: 6 months. Prefer primary sources. Run 3 was the first run started by the scheduler.

Run 3 · deep · llama3.2 reader

When: 2 Oct 10:30–12:22 JST · **110.7 min** of 180 (plan 2.8, triage 4.6, read 72.0, write 19.8, critique 9.1, file 2.4).

Statistics: 24 of 30 pages · 268 claims, **83.6 %** verified · 57 verified figures (2.38 a page) · 5,762 words, 212 citations of 22 sources · critic 5 flags, 2 agreeing · 10.5 CPU-h, ~49 Wh.

Highlights: NASA's deep-space optical link experiments · the Open X-Embodiment dataset (over 1 million robot episodes, 22+ robot types) · teleoperation under 100 ms round trip indistinguishable from local control for pick-and-place.

Reviewer's note: strongest on deep-space communications. Padding: bibliography entries listed as data points; one car study cited as a space benchmark.

Run 12 · deep · gemma4 reader (re-run of run 3)

When: 3 Oct 06:11–08:17 JST · **124.3 min** of 180 (plan 0.9, triage 4.7, read 86.8, write 21.8, critique 7.4, file 2.6).

Statistics: 24 of 28 pages · 229 claims, **95.6 %** verified · **81 verified figures**, the most of any run · 7,086 words (the longest report), 256 citations of 24 sources · 12.1 CPU-h, ~57 Wh.

Highlights: predictive video interfaces cutting task time by up to 26 % under high latency · leader–follower arms at 3–8 ms against 15–40 ms for VR-headset control · latency near 300 ms degrading closed-loop human control.

Reviewer's note: strongest on latency and the space-robot inventory. Errors: it dated two rover drives to a year that has not happened yet, and it repeated a citation three times. **Equivalent to run 3**, each with its own strengths.

Run 7 · standard · llama3.2 reader

When: 2 Oct 19:17–20:18 JST · **60.1 min** of 100 (plan 0.8, triage 3.4, read 29.1, write 17.1, critique 6.9, file 2.8).

Statistics: 12 of 13 pages · 138 claims, **85.5 %** verified · 18 verified figures (1.50 a page) · 3,632 words, 138 citations · 5.7 CPU-h, ~27 Wh.

Highlights: hybrid tactile and kinesthetic feedback cutting applied force by ~50 % · the Lunokhod-1 lunar rover driven under a 2.5-second light-speed delay · a hybrid of classical aerospace control and AI prediction as the realistic path.

Reviewer's note: good material on telesurgery delays and autonomous driving on Mars, but its section on deep-space latency is hollow (built largely from a general control-theory article).

Run 15 · standard · gemma4 reader (re-run of run 7)

When: 3 Oct 11:37–12:50 JST · **72.5 min** of 100 (plan 0.9, triage 3.0, read 38.4, write 15.7, critique 11.8, file 2.6).

Statistics: 12 of 15 pages · 119 claims, **95.0 %** verified · 50 verified figures (**4.17 a page**, the highest of any run) · 4,190 words, 159 citations · 6.8 CPU-h, ~32 Wh.

Highlights: a review of 52 studies of AI in space robotics showing a sharp rise after 2022 · simulation platforms and datasets for testing space robots before flight · exoskeleton gloves at USD 8,000–20,000 a pair.

Reviewer's note: more space-specific than run 7, but its section on human–machine interfaces is the worst in all twelve deep and standard reports: the model read "HMI" as a factory touch panel. It also called a mission that had already flown "scheduled". **Equivalent to run 7.**

Run 8 · quick · llama3.2 reader

When: 2 Oct 20:18–20:55 JST · **36.4 min** of 40 (plan 0.8, triage 2.4, read 17.8, write 13.0, file 2.5).

Statistics: 6 of 6 pages · 62 claims, **80.6 %** verified (the lowest quick run) · 14 verified figures · 3,073 words, 135 citations · 3.4 CPU-h, ~16 Wh.

Highlights: AI-for-space-robotics studies increasing after 2022 · a lunar rendezvous demonstration targeted for 2027 · a GPS receiver weighing 3.5 pounds for lunar navigation.

Topic 3 — Digital twins with VR, AI and automation simulation

Brief: "Recent developments on Digital twin, VR, AI & Automation Simulation". Purpose: innovations and industry developments. Any date. Prefer primary sources.

Run 4 · deep · llama3.2 reader

When: 2 Oct 15:03–17:09 JST · **124.0 min** of 180 (plan 2.9, triage 4.7, read 84.0, write 20.2, critique 9.9, file 2.3).

Statistics: 24 of 33 pages, 88,937 words read (the most of any run) · 273 claims, **90.1 %** verified (llama3.2's best deep run) · 51 verified figures · 5,767 words, 219 citations of 23 sources · 11.7 CPU-h, ~55 Wh.

Highlights: overall equipment effectiveness raised from 65 % to 82 % in one deployment · an edge-AI and federated-learning framework cutting latency by 35 % · NASA's spacecraft and mission twins for "test as you fly".

Reviewer's note: leans on market-size forecasts; one solid case study; some padding from a reference list presented as current work.

Run 13 · deep · gemma4 reader (re-run of run 4)

When: 3 Oct 08:17–10:26 JST · **126.2 min** of 180, the longest run (plan 1.0, triage 4.5, read 88.6, write 19.6, critique 9.6, file 2.9).

Statistics: 24 of 27 pages · 222 claims, **92.8 %** verified · 54 verified figures · 6,368 words, 250 citations of 24 sources · 12.3 CPU-h, ~58 Wh.

Highlights: a physics-informed twin predicting power-plant corrosion · a fuel-cell design reached in under an hour within 5 % of full CFD simulation · the ISO/IEC 30173 framework for digital-twin interoperability.

Reviewer's note: more named cases with measured results. **One serious flaw:** the same case study, read from three copies (web page, PDF and mirror), was presented as corroboration "across multiple sectors". A reader cannot see that; source selection should. **Better than run 4, with that caveat.**

Run 9 · standard · llama3.2 reader

When: 2 Oct 20:55–22:12 JST · **73.9 min** of 100 (plan 0.8, triage 3.3, read 44.6, write 15.4, critique 7.0, file 2.7).

Statistics: 12 of 15 pages · 158 claims, **76.6 %** verified, the lowest of all 15 runs (25 not on page, 11 figure) · 23 verified figures · 4,143 words, 166 citations · 7.0 CPU-h, ~33 Wh.

Highlights: unplanned downtime reduced by 10–25 % in reported deployments · ISO 23247 and the FMI 3.0 standard for exchanging simulation models · the Asset Administration Shell for interoperability.

Reviewer's note: the thinnest of the older reports. A consultancy menu page gave 14 claims, four of them invented percentages, all caught by the checks. Its six pages became the second A/B test.

Run 16 · standard · gemma4 reader (re-run of run 9)

When: 3 Oct 12:51–14:12 JST · **80.2 min** of 100, the closest any deep or standard run came to its cap (plan 0.7, triage 3.3, read 49.0, write 14.6, critique 10.0, file 2.7).

Statistics: 12 of 13 pages, 2 empty (one by restraint, one a context overflow) · 96 claims, **95.8 %** verified (the highest of all 15) · 37 verified figures (3.70 a page) · 4,366 words, 181 citations · 7.6 CPU-h, ~36 Wh.

Highlights: named case results such as a 23 % throughput gain at an appliance maker, a clinic's waits cut by 37 %, and a call centre's compliance raised from 76 % to 94 % · ISO 23247 and OPC UA for interoperability.

Reviewer's note: answers its "what improvements were measured?" question directly, though much of the return-on-investment evidence comes from vendor blogs. Raw evidence text leaked into one section through a case-sensitive clean-up step. **Better than run 9.**

> **Run 10 · quick · llama3.2 reader**
>
> **When:** 2 Oct 22:12–22:52 JST · **39.2 min** of 40, the closest any run came to its cap (plan 1.0, triage 2.7, read 20.2, write 13.1, file 2.3).
>
> **Statistics:** 6 of 8 pages · 71 claims, **90.1 %** verified · 22 verified figures (**3.67 a page**) · 3,686 words, 120 citations · 3.6 CPU-h, ~17 Wh.
>
> **Highlights:** a distribution company cutting order fulfilment time by 24 % with a process twin and no capital investment · an automotive case with 54 % lower costs and 37 % less downtime · a utility improving defect detection by 67 % with synthetic 3D images.

Limitations

Of the evidence

- **Small samples.** 15 runs on three topics, with one run per configuration. gemma4's run-to-run variation is unmeasured; llama3.2's is large (on the same six pages it verified 53 of 81 in production and 61 of 78 in the A/B).
- **Re-runs read different pages.** A fresh run searches again, so paired runs shared only 1–4 pages. Paired report comparisons mix the reader's effect with search luck; the per-claim statistics are much less affected.
- **Report quality was judged by one careful reading of each pair**, not by a blind panel or a fixed rubric. The source pages behind the highlights were not re-opened.
- **No quick run with gemma4**, no deep run that needed a gap round, and no measurement of whether Gotia's answers slow down while a run reads with the same model.
- **English only, three technical or industry topics.** A topic where sources paraphrase each other heavily cut llama3.2's quote fidelity to 24 % in September, and no gemma4 run has faced one.

Of the measurements

- **"Verified" means faithful to the page, not true and not relevant.** A verified claim quotes its page correctly; the page itself may be a vendor blog, out of date or off-topic. Verification cannot catch irrelevance, the main weakness of the gemma4 reports.
- **Effectiveness scores are not comparable across phases.** Each phase has its own measure, and some, like citation coverage, count form (a tag is present) rather than substance.
- **Critic precision undercounts real catches.** A correct flag the mechanical checks cannot confirm counts against the critic.
- **Energy is the whole CPU package**, idle machine included, so it is an upper bound on what the models used. **The energy counter wraps every 65.5 kJ**, and a phase that used more than that, which includes every long read phase, was recorded too low by whole multiples of the wrap. This report shows an estimate: the raw reading plus the number of whole wraps that brings the phase in line with the median of 4.86 J per CPU-second measured on phases too short to wrap. 9 of 87 phase records were corrected this way. CPU time is measured directly and needs no correction.

- **CPU and memory are the model server's**, not one run's. That is exact here, because nothing else was using the models during these runs (zero contention was recorded), but it would not be under load.

Of the tool

- **Relevance.** The reader extracts faithfully from off-topic pages; triage judges by title and snippet.
- **Duplicates.** Three copies of one paper were counted as three corroborating sources.
- **Dates.** The writer turns "as of" dates into event dates and once invented a year.
- **Formatting.** LaTeX fragments and doubled citations appear in reports, and one clean-up step is case-sensitive.
- **Context edge.** One very dense 38 KB page overflowed gemma4's 12,288-token reading window.
- **Time budgets.** Quick runs use 91–98 % of their cap, and the standard cap has 19 minutes of margin at worst.

Recommendations for deploying small language models

These come from what was measured here, and apply beyond this tool.

1. **Put a mechanical check between every model and the user.** For extraction, require a verbatim quote and check it, and check every number against the quote. This was the single most effective control in the system: it turned a 76 %-accurate reader into a pipeline that put no unverified claim into a report.
2. **Spend model size where accuracy is decided.** Here that was the reader, not the writer or the planner. The 8B reader was worth it; a bigger triage model would not have been (96.6 % already).
3. **Measure cost per verified result.** Per call, the 3B model looked cheaper. Per verified figure, it was 28 % more expensive. Any model comparison that counts output rather than checked output will favour the model that pads.
4. **Prefer a model that declines.** On empty, blocked or menu pages, the best behaviour is to extract nothing. Test candidate models on those pages specifically.
5. **Enforce structure with the grammar, not the prompt.** Use JSON schemas with enums and `maxItems`. Prompt limits were ignored a third of the time; schema limits never were.
6. **Budget context per step, and detect context exhaustion separately from output truncation.** Give the reading step its own, larger context. Never let a prompt fill the window.
7. **Turn "thinking" off for small models on bounded tasks, or cap it hard.** The 4B critic's thinking mode never reached an answer.
8. **Do not let a small model mark its own homework, or another's, unchecked.** Use a different model from the writer for review, but treat its output as suggestions; at 19 % precision it is not a gate.
9. **Decide model swaps with pass marks set in advance:** an A/B on the same inputs, then a second topic, then whole production runs. This sequence caught a result that would have reversed, and confirmed one that held.

10. **Set time limits from the worst observed case × 1.3, re-derive them after any model change, and make an over-limit run degrade visibly, not silently.**
11. **Next for Anusandhan:** a relevance gate between reading and writing (the biggest remaining lever), de-duplication of mirrored sources, a date-handling rule for the writer, raising the quick cap or trimming quick's pages, and an A/B for quick only if its reports show invented figures.

Appendix

Where to find more

Hiren's website	www.hirenonline.com
How Anusandhan, Gotia, Ramu and Shastrarth work, with diagrams	www.hirenonline.com/hiren-on-ai (Anusandhan: #anusandhan)
All of Hiren's published reports, including this one	www.hirenonline.com/reports
The companion study of the same SLMs as debaters	www.hirenonline.com/reports#shastrarth-findings-report
Watch small models debate, live	shastrarth.hirenonline.com
Ask Ramu about Hiren	www.hirenonline.com/ramu
Hiren on LinkedIn	www.linkedin.com/in/hiren-joshi-b085342

Method notes

- **Sources of the numbers:** each run's per-phase records (model, calls, tokens, wall and load time, CPU, energy, truncation, parse failures, items in/out/good); the token log of every model call (speed, load time); and each run's saved claims, sources, report and review. Frozen on 3 October 2026; the PDF is built from that snapshot so it cannot drift from its own text.
- **The A/B tests** used copies of the run folders, an in-memory database and the real models, one model after the other on an otherwise idle machine. Their model calls are not in the per-run tables above.
- **Medians, not means,** for speeds and per-call times, so one call whose prompt was already cached cannot distort a row.
- **Hardware:** one CPU-only home server, 12 threads, 30 GB RAM; models served by Ollama, 4-bit quantised. Reading context 12,288 tokens, other phases 8,192.
- **What was not used:** no cloud language model at any step; no human edits to any phase's output; nothing filed into Gotia's memory.