AGENTFLOOR: HOW FAR UP THE TOOL-USE LADDER DO SMALL OPEN-WEIGHT MODELS GO?

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic systems make many small tool-using calls, but it is unclear which of them actually require a frontier model. We introduce **AgentFloor**, a controlled benchmark that organises tool-use tasks into a six-level ladder, from instruction following and single tool calls to branching, recovery, and long-horizon planning, while keeping the tool environment fixed. We evaluate 16 open-weight models alongside five frontier models from three vendors. Small open-weight models perform surprisingly well on the simplest levels, even a sub-billion-parameter model follows instructions reliably, but the gap widens as tasks require chaining, branching, and longer-horizon control. How large that gap is depends strongly on which frontier model serves as the reference: conclusions drawn against GPT-5 differ substantially from those drawn against the strongest Claude models. Models with similar overall accuracy also fail in very different ways, and simple interventions rarely transfer from one model to another. These results suggest that routing decisions should depend on the structure of the tool-use task, not only on model size or aggregate benchmark scores. We release the benchmark, the evaluation harness, and the full run corpus.

1 INTRODUCTION

A production agent makes many model calls per user request, and most of them are short and structurally simple: a lookup, an extraction, a single submission. Sending every call to a frontier model is the default. Whether that default is necessary is an empirical question about *which kinds of calls* a smaller model can handle reliably. End-to-end agent benchmarks (Yao et al., 2024; Liu et al., 2025; Mialon et al., 2023; Zhou et al., 2024; Jimenez et al., 2024) answer a different question, how well a complete system performs in a realistic environment, and in doing so they entangle tool-calling control with web grounding, API drift, documentation reading, and pretraining contamination. Function-calling benchmarks (Patil et al., 2024; Li et al., 2023; Patil et al., 2023) isolate single calls, but not the multi-step structure that separates easy agentic work from hard.

We therefore built a *controlled diagnostic*, closer to a unit-test suite than to a deployment benchmark. AgentFloor fixes the environment, eight abstract tools over an in-memory database, with no filesystem, no live APIs, and no time-varying state, and varies one thing: the structural demand of the task. Six tiers form a ladder (Figure 1), each adding one demand to the tier below it. Because only calls made through a model's native tool-calling interface count, the benchmark measures whether a model can *drive* tools, not whether it can describe how it would.

A central difficulty in such comparisons is the choice of reference model. Frontier systems differ substantially in how they use tools, so treating "the frontier" as a single capability level can lead to misleading conclusions. AgentFloor therefore evaluates open-weight models against several frontier systems and reports results at each level of the ladder rather than relying on an aggregate score. The choice matters in practice: the strongest open-weight model performs on par with GPT-5 across the workload, yet falls clearly behind the strongest Claude models once tasks require chaining or branching.

AgentFloor makes three contributions. First, it provides a controlled benchmark that separates increasingly demanding forms of tool use while keeping the environment fixed. Second, it shows where small open-weight models remain competitive with frontier systems, measured against several of them rather than one, and where that competitiveness breaks down. Third, it examines how

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

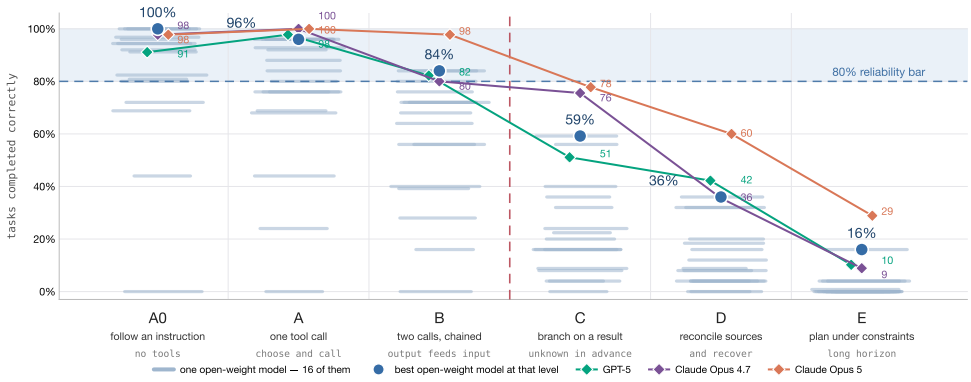

Figure 1: Performance on the AgentFloor ladder. Grey bars show each of the 16 open-weight models (125 runs per cell); the blue dot marks the best open-weight model at each tier. Diamonds show three frontier models (45 runs per cell). At the two simplest tiers, the best open-weight model and all Claude models are near 100%. From two-call chaining onward, the frontier models separate from each other and from the open-weight models, and no model is reliable at long-horizon planning.

different models fail and whether simple changes to prompting or execution can repair those failures. We release the benchmark, the evaluation harness, and the complete run corpus.

2 Related Work

Tool-use and agent benchmarks: Function-calling suites (Patil et al., 2024; Li et al., 2023; Patil et al., 2023; Zhong et al., 2025) score individual calls, while multi-step suites (Qin et al., 2023; Yao et al., 2024; Wang et al., 2024; Liu et al., 2025; Ma et al., 2024; Mialon et al., 2023; Zhou et al., 2024; Xie et al., 2024; Jimenez et al., 2024) score complete agents in rich environments. Benchmarks built on real APIs suffer from drift and contamination, which motivated simulated or cached environments (Guo et al., 2025; Ruan et al., 2024; Garg et al., 2025). AgentFloor sits at the controlled end of this spectrum and complements end-to-end evaluations: it gives up some external validity in exchange for attributing failures to a specific structural demand. API-Bank (Li et al., 2023) and AgentBoard (Ma et al., 2024) also grade tasks by difficulty, but neither pairs difficulty levels with equivalence testing against several frontier models.

Small models for agents: Technical reports for small models rarely include agentic evaluation (Abdin et al., 2024; Yang et al., 2025; Team, 2025; Abdelaziz et al., 2024; Allal et al., 2025; Grattafiori et al., 2024). Surveys and position papers (Lu et al., 2025; Belcak et al., 2026) argue that small models suffice for much agentic work, and concurrent studies measure JSON validity or describe qualitative failure archetypes (Kavathekar et al., 2025; Roig, 2025). We test the sufficiency claim tier by tier, against several reference models.

Failure taxonomies and routing: Taxonomies of agent failures (Kokane et al., 2025; Huang et al., 2025; Cemri et al., 2025; Winston & Just, 2025) and cost-aware routers (Chen et al., 2023; Ong et al., 2025; Ding et al., 2024; Aggarwal et al., 2025) are natural consumers of a per-tier capability profile: a router needs a prior estimate of which tasks smaller models can handle.

3 AgentFloor: Design and Protocol

The ladder: AgentFloor contains 30 tasks, five at each of six tiers, labelled A0 to E. Tier A0 asks for instruction following without tools: extraction, summarisation and classification. Tier A requires a single tool call. Tier B requires a two-call chain in which the first result determines the second call. Tier C requires a branch that depends on an intermediate result. Tier D requires synthesis across several sources, including recovery from a scripted tool failure. Tier E requires a plan of 8–12 calls that respects constraints throughout. Step budgets grow from 1 at the lowest tier to 10 at

the highest. Each task has a canonical prompt and four paraphrases, and four tasks also have five instance variants. The ordering reflects structural demands in this environment rather than real-world difficulty; Section 4.3 tests whether models experience it as an ordering.

Tools and environment: Every task uses the same eight tools, for searching and looking up records, reading attributes, listing options, checking constraints, comparing records, computing values, and submitting a final decision.[1] The tools operate on a per-task database that contains records, distractors, constraints, and scripted failures. Execution is deterministic, and no tool accesses the network.

Inference protocol: We used the same runner and system prompt for every model. Temperature was set to zero whenever the provider allowed it, and we did not tune prompts or repair outputs for individual models. A tool call counts only if it arrives through the provider's native tool-calling interface; a call written out as JSON text counts as no call. Candidate models were first screened with a short tool-calling probe, which excluded 23 of 57 candidates (Section L). Open-weight models ran through Ollama with 4-bit quantisation on a single Apple-silicon machine, and frontier models were queried through their vendors' APIs.

Scoring: We count a run as successful only when it passes four sets of checks: the final answer contains the expected values, the submission is accepted, the trajectory meets task-specific requirements, and no forbidden behaviour occurs, such as calling a non-existent tool, stopping without an answer, repeating the same call, or stating facts absent from the source. We call the resulting success rate the task completion rate (TCR). The trajectory check asks whether required tools were used at all, not whether they were used in a fixed order, so valid reorderings pass. Runs that failed only this check (6% of baseline runs) were mostly genuine tool-use failures: half produced an answer without calling the tool that should have produced or verified it, and a third took the wrong branch (Section E). Two semantic checks, unsupported facts in three summarisation tasks and inconsistent recovery reasoning in one planning task, are decided by an LLM judge. A scorer audit found that the judge had been comparing summaries of paraphrased prompts against the original passage; we corrected this before all analyses reported here and compared the judge against a second model from another vendor (Section D).

Statistical analysis: We compare models on matched runs, pairing the same task, prompt variant, and repetition, and report differences in task completion rate with paired bootstrap confidence intervals. Two models are treated as equivalent when their difference lies confidently within a pre-specified margin of ten percentage points, and an open-weight model as non-inferior when it is confidently no more than ten points worse; comparisons across models and tiers are corrected for multiple testing. We chose the margin before the analysis as a tolerance for low-risk routing decisions, not as a safety threshold, and the reported intervals let readers apply stricter margins. Several middle-tier comparisons have limited power, so we treat non-significant differences there as inconclusive rather than as evidence of parity. Section A gives the full procedure and power calculations.

Models: The 16 open-weight models span seven families and range from 0.27B to 32B parameters (Table 5 lists them with their runtime identifiers); each completed 750 baseline runs. The frontier models are GPT-5 (Singh et al., 2026), Claude Opus 4.6, 4.7 and 5 (Anthropic, 2026), and Gemini 2.5 Pro (Comanici et al., 2025), with 270–274 runs each. Opus 5 is the most recent of these, and Gemini provides a third vendor. Including ablations, the study comprises 17,622 scored runs (Section M).

4 Results

4.1 Frontier models are not one reference point

Table 1 and Figure 2 summarise the main result: whether open-weight models "match the frontier" depends strongly on which frontier model is the reference. We use Gemma 4 26B, the open-weight model with the highest overall score, as the comparison model.

Across the whole workload, Gemma 4 26B is statistically equivalent to GPT-5. The picture changes against the Claude models. All three are clearly ahead on branching tasks, and Opus 5, which leads

[1] `search_records`, `lookup_record`, `get_attribute`, `list_options`, `check_constraint`, `compare_records`, `compute_value`, `submit_decision`.

Table 1: Per-tier TCR (%) with 95% bootstrap confidence intervals on the matched design (45 runs per tier, 270 overall). "Best open-weight" is the highest open-weight score at each tier on the same design; the model differs by tier. †Gemini's A0 score includes 22 empty API responses counted as failures; without them it is 100% (Section H).

	GPT-5	Opus 4.6	Opus 4.7	Opus 5	Gemini 2.5 Pro	Gemma 4 26B	Best open-weight
A0	91 [82,98]	100	98 [93,100]	98 [93,100]	51† [36,67]	100	100
A	98 [93,100]	100	100	100	80 [67,91]	100	100
B	82 [71,93]	87 [76,96]	80 [69,91]	**98** [93,100]	60 [44,73]	73 [60,87]	87 (Qwen3 14B)
C	51 [36,67]	**80** [69,91]	76 [62,87]	78 [64,89]	44 [31,60]	53 [40,69]	60 (Ministral 3 8B)
D	42 [29,58]	47 [33,62]	36 [22,49]	**60** [47,73]	24 [13,38]	40 [27,56]	40 (Gemma 4 26B)
E	11 [2,20]	27 [13,40]	9 [2,18]	**29** [16,42]	16 [7,27]	0	20 (Ministral 3 8B)
All	63 [57,68]	73 [68,79]	66 [61,72]	**77** [72,82]	46 [40,52]	61 [55,67]	,

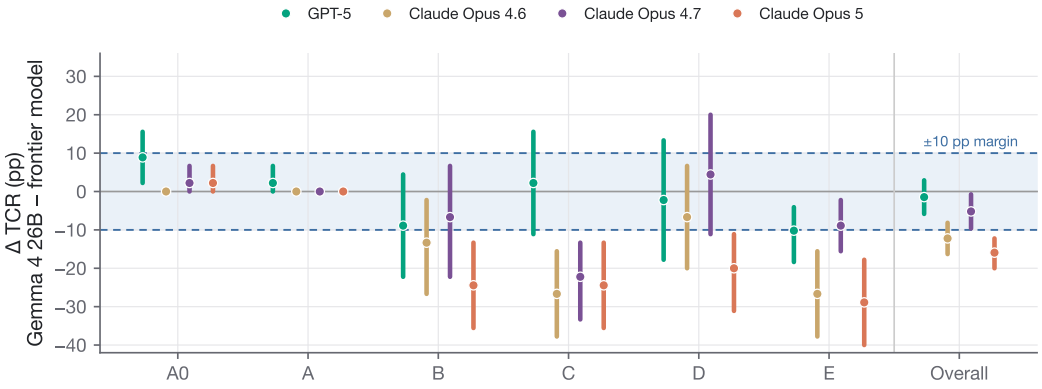

Figure 2: Difference in TCR between Gemma 4 26B and each frontier model, with 90% bootstrap intervals (the interval used for the equivalence test). The shaded band marks the ±10-point margin; two models count as equivalent only where the whole interval lies inside it.

by about 16 points overall, is ahead from two-call chaining onward. On the two simplest levels, by contrast, the strongest open-weight models are as good as any frontier model.

The frontier models also differ from one another. Models with similar aggregate scores behave quite differently on individual levels: GPT-5 and Opus 4.7 are close overall but far apart on branching tasks. A single workload average is therefore a poor description of tool-use capability, and the tier-level profile is considerably more informative. The broader comparison across all 16 open-weight models shows the same pattern: several match every frontier model on the simplest tasks, but this becomes rare once tasks require branching or longer-horizon control (Section F).

4.2 Where small open-weight models suffice

Table 2 asks a practical question: what is the smallest model that is reliably good enough? We credit a model only if its entire confidence interval lies above the threshold. By this standard, the smallest models are sufficient for the simplest work. A 0.6B model is already reliable at instruction following, and models of a few billion parameters handle single tool calls. Two-step chaining remains within reach of some small models at moderate reliability, but reliability drops sharply once a task requires branching on an intermediate result, and beyond that point no open-weight model reaches even the lowest threshold we consider. Frontier models rarely reach high reliability there either.

4.3 Is the ladder a ladder?

We next ask whether the proposed ordering is reflected in model performance. Two checks suggest that it largely is (Section G). (i) *Ordering.* Performance declines steadily from two-call chaining

Table 2: Smallest open-weight model whose entire 95% confidence interval lies above each reliability threshold (125 runs per cell). A dash means that no model in the study meets the threshold without intervention.

Tier	60%	70%	80%	90%
A0: instruction following	Qwen3 0.6B	Qwen3 0.6B	Qwen3 0.6B	Qwen3 0.6B
A: single call	Qwen3 1.7B	Granite 4 3B	Ministral 3 3B	Gemma 4 E4B
B: two-call chain	Qwen3.5 2B	Qwen3.5 2B	,	,
C–E: branching, synthesis, planning	,	,	,	,

Table 3: How tier-E runs end (% of runs, 45 per model). Stopped early: the model ends the run after at least two successful calls. Planned only: the model describes a plan but makes at most one call. Provider error: a failed API or serving request, counted as a failure. Appendix taxonomy codes are given in Section C.

Model	Success	Invented tool	Budget exhausted	Stopped early	Planned only	Wrong tool	Partial answer	Provider error
GPT-5	11	0	9	**38**	**22**	2	18	0
Claude Opus 4.6	27	0	4	20	0	16	**33**	0
Claude Opus 4.7	9	0	4	18	11	**31**	**27**	0
Claude Opus 5	29	0	0	20	0	13	**38**	0
Gemini 2.5 Pro	16	9	**24**	20	4	11	13	0
Gemma 4 26B	0	**33**	**42**	24	0	0	0	0
Ministral 3 8B	20	0	7	27	0	2	20	24

through long-horizon planning for every frontier model, and the same trend holds for nearly all open-weight models. The exception is the first step: some models do better on a single tool call than on faithful extraction or summarisation, so we treat tier A0 as a baseline skill rather than literally the easiest rung. (ii) *Task-level agreement.* Frontier models from different vendors largely agree on which individual tasks are hard, so difficulty appears to belong to the tasks rather than to one vendor's models. With five tasks per tier, however, these checks cannot show that the tiers generalise beyond this task set.

4.4 Matched accuracy, different failures

Models with similar overall accuracy often fail for very different reasons (Table 3). On the planning tasks, GPT-5 tends to stop before finishing, either quitting midway or describing a plan that it never carries out. Gemma 4 26B, which has the same overall score, fails in the opposite way: it keeps going until it runs out of steps or calls a tool that does not exist. The Claude models usually keep making progress but stop short of a complete submission or choose the wrong tool near the end.

These differences matter because they imply different remedies. More steps cannot help a model that has already stopped, and a reminder to submit cannot help a model that never reaches the final stage.

4.5 Interventions are model-specific

We tested four interventions that leave model size unchanged (Figure 3; details in Section I), and they confirm that these failure modes are model-specific. An explicit reminder to submit the final decision completely fixed one Ministral 3 8B failure, in which the model carried out the right calls but answered in text, yet had no effect on models that failed earlier in the trajectory. Giving GPT-5 a larger step budget helped only on the one task where running out of steps was already the problem. Disabling reasoning helped the largest Qwen3 model and hurt its smaller siblings. Most surprisingly, a structured plan–execute–submit prompt did not improve any of the models we tried and made most of them worse; one model obeyed the instruction not to call tools while planning and then simply answered in prose. There is no universal prompting fix in our experiments: whether an intervention helps depends on how a particular model fails.

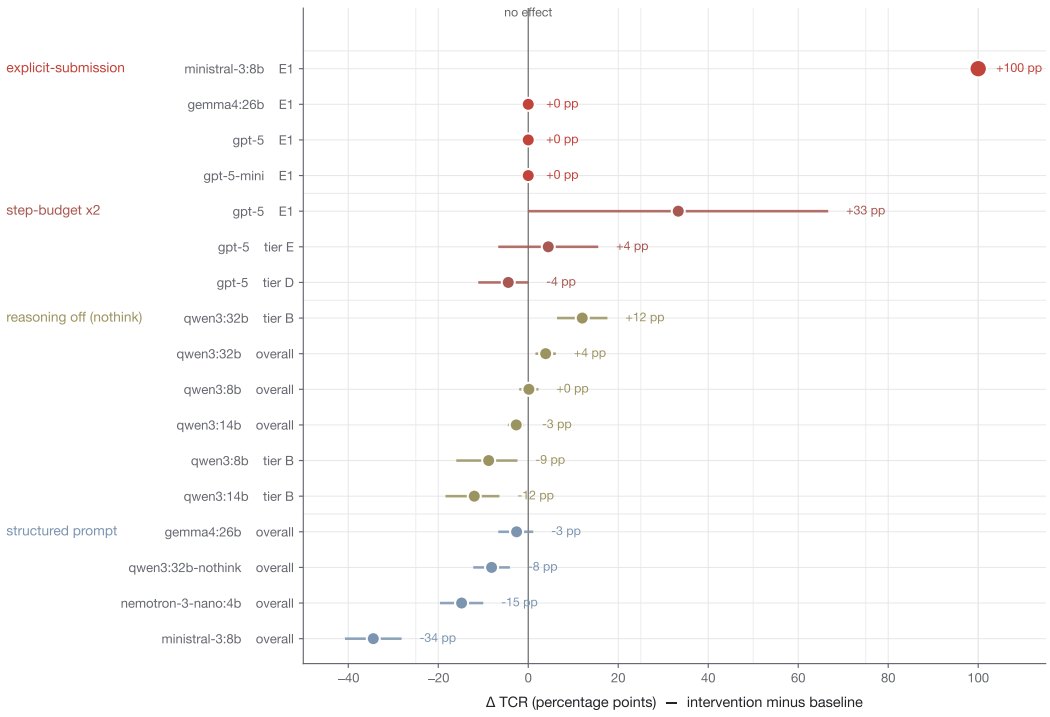

Figure 3: Effects of four interventions: paired difference in TCR with 95% bootstrap intervals. Large effects are confined to single model–task combinations, and the structured planning prompt helps no model.

Table 4: Cost and wall-clock time per successful task over the whole workload. Open-weight models: self-hosting on a Mac at $0.50 per hour, and cloud GPUs at $0.50, $1.50, and $2.50 per hour for models up to 4B, 7–14B, and 20B or larger; 750 runs each. Frontier models: vendor list prices per million input and output tokens (GPT-5 and Gemini $1.25 and $10; Claude Opus $5 and $25; thinking tokens billed as output) on the matched 270-run design. Open-weight latency comes from a single local machine; API latency includes network and queueing time.

Model	TCR	$ per success (Mac)	$ per success (GPU / API)	Seconds per success
Granite 4 3B	42%	$0.00043	$0.00043	3.1
Ministral 3 8B	56%	$0.00096	$0.0029	6.9
Gemma 4 26B	60%	$0.0022	$0.011	16.0
Gemini 2.5 Pro	46%	,	$0.012	31.5
GPT-5	63%	,	$0.031	38.6
Claude Opus 4.7	66%	,	$0.073	14.3
Claude Opus 4.6	73%	,	$0.077	24.8
Claude Opus 5	77%	,	$0.064	17.5

4.6 Cost and latency

Cost changes the picture substantially (Table 4). When the work can be handled at the lower levels of the ladder, small open-weight models are far cheaper, often by an order of magnitude or more. The advantage shrinks as tasks become harder and larger models are needed, though it remains robust for the smallest models under less favourable pricing assumptions (Section J). At the high end of the workload, the strongest Claude models reach accuracy that no open-weight model in our study attains, at a substantially higher cost per task. Latency follows a similar pattern: small local models are fast, while the largest open-weight models lose most of that advantage.

5 DISCUSSION

The results suggest a fairly simple picture. Small models are already very capable when the job is limited to following an instruction or making a single tool call. Difficulty rises once one tool call depends on another, and the gap to the strongest frontier models becomes much clearer when a model has to branch, recover, or maintain constraints over a longer trajectory. Long-horizon planning under persistent constraints remains unreliable for every model we tested.

This does not mean that every simple production call should immediately be routed to a small model. AgentFloor deliberately removes many sources of real-world difficulty, including documentation, authentication, changing APIs, permissions, and web grounding, and a single real API call can be harder than several abstract ones. The benchmark instead identifies a lower-level capability boundary that can inform routing decisions and guide subsequent end-to-end testing. Nor do our results show that better scaffolding would close the remaining gap, since the interventions we tested were narrow and did not transfer. Standardised tool interfaces such as MCP (Anthropic, 2024) could extend the same ladder to real tool servers while keeping its structure.

Parameter count is a weak predictor: Size alone is a poor guide to tool-use performance. Smaller models sometimes outperform much larger ones, and models from the same family can perform similarly despite large differences in size. Tool-calling templates, model family, and reasoning configuration appear to matter at least as much as parameter count (Table 5).

6 LIMITATIONS

Scale and coverage: Five tasks per tier is a small number. Paraphrases and repeated runs tighten the estimates for these tasks but do not add independent task families, so tier-level conclusions are conditional on the task set, and comparisons in the middle tiers are underpowered at the ± 10-point margin (Section 3). The agreement between frontier models on task difficulty (Section 4.3) mitigates this concern but does not remove it. Expanding each tier with independently authored tasks is the most important extension.

External validity: The abstract-tool design isolates control at the cost of realism (Section 5). Requiring native tool calling also excludes models that emulate tool calls in text (Section L), so our conclusions apply to models with native tool-calling support.

Scoring: The instruction-following tier relies partly on an LLM judge. After correcting the passage error described in Section D, the judge still agrees only modestly with a second judge from another vendor ($\kappa = 0.27$), although no comparison changes when both judges must agree, and 0.6% of judge calls returned unusable output and default to a pass. We did not collect a human baseline.

Serving differences: The adapter used for all Ollama-served models presents the two tools with polymorphic argument schemas without declared parameters. This affects all open-weight models equally, so it does not change comparisons among them, but it biases their absolute scores downward on tasks that use those tools; in a probe of one additional model, restoring the full schema raised single-call performance by 16 points and left the harder tiers unchanged (Section K). Ministral 3 14B failed the tool-calling screen but remained in the study; most of its runs contain no native tool call, so its low scores reflect protocol non-compliance rather than a capability regression, and removing it changes no conclusion.

Provider failures and model churn: About 1% of open-weight runs, mostly from the Ministral family, ended in provider errors and count as failures. Gemini 2.5 Pro returned empty responses on 22 of its 45 instruction-following runs (Section H), so we report it without basing conclusions on it. Frontier models also change quickly, which is why we report each one separately rather than as a single frontier.

7 CONCLUSION

AgentFloor shows that tool-use capability does not fall along a simple divide between small and frontier models. Small open-weight models are already strong on simple, well-structured calls, but their reliability drops as tasks require chaining, branching, recovery, and longer-horizon control, and

the size of that gap depends strongly on which frontier model is the reference. The broader lesson is that model selection for agentic systems should be based on the structure of the work, not only on model size or aggregate benchmark scores. A model that is entirely adequate for one stage of an agentic workflow may be a poor choice for another, and AgentFloor provides a controlled way to find that boundary.

AI USE STATEMENT

Generative AI tools were used to help write analysis and plotting code, to draft and edit sections of the manuscript, and to audit the manuscript for numerical consistency against the result corpus. An LLM (GPT-5 nano) is also a component of the evaluation method, as a judge for two semantic predicates (Section D). All numbers were regenerated from the released run files by scripts in the repository, and the authors reviewed all AI-assisted text and code and take full responsibility for the content.

ETHICS STATEMENT

The study uses synthetic fixtures with no human subjects and no personal data. The main risks concern downstream use: routing agentic calls to smaller models on the basis of controlled results could lead to unintended tool actions or incorrect submissions in real systems, where tool access carries security and privacy implications. We therefore describe AgentFloor as a diagnostic rather than a deployment certificate, report failure modes alongside accuracy, and recommend end-to-end validation with task-appropriate margins before any routing decision.

REPRODUCIBILITY STATEMENT

The tasks, fixtures, harness, sweep configurations, scorer, cached judge verdicts, and all 17,622 raw traces and score files are included in the anonymised supplementary material. Section 3 and Sections D, L and M specify the protocol, judge prompts, inclusion probe and run inventory. Every table and figure is produced by a script from a single run-level table built from the score files (seed 42, 10,000 bootstrap resamples). Open-weight inference used Ollama with default 4-bit quantisation on one Apple-silicon host; frontier models were queried through vendor APIs between April and July 2026.

REFERENCES

Ibrahim Abdelaziz, Kinjal Basu, Mayank Agarwal, Sadhana Kumaravel, Matthew Stallone, Rameswar Panda, Yara Rizk, GP Bhargav, Maxwell Crouse, Chulaka Gunasekara, Shajith Ikbal, Sachin Joshi, Hima Karanam, Vineet Kumar, Asim Munawar, Sumit Neelam, Dinesh Raghu, Udit Sharma, Adriana Meza Soria, Dheeraj Sreedhar, Praveen Venkateswaran, Merve Unuvar, David Cox, Salim Roukos, Luis Lastras, and Pavan Kapanipathi. Granite-function calling model: Introducing function calling abilities via multi-task learning of granular tasks, 2024. URL https://arxiv.org/abs/2407.00121.

Marah Abdin, Jyoti Aneja, Hany Awadalla, and Ahmed Awadallah et. al. Phi-3 technical report: A highly capable language model locally on your phone, 2024. URL https://arxiv.org/abs/2404.14219.

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Manaal Faruqui, and Mausam. Automix: Automatically mixing language models, 2025. URL https://arxiv.org/abs/2310.12963.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. Smollm2: When smol goes big – data-centric training of a small language model, 2025. URL https://arxiv.org/abs/2502.02737.

Anthropic. Introducing the model context protocol. https://www.anthropic.com/news/model-context-protocol, 2024.

Anthropic. Claude models overview (claude opus 4.6, opus 4.7, opus 5). https://docs.anthropic.com/en/docs/about-claude/models, 2026.

Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. Small language models are the future of agentic ai, 2026. URL https://arxiv.org/abs/2506.02153.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why do multi-agent llm systems fail?, 2025. URL https://arxiv.org/abs/2503.13657.

Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance, 2023. URL https://arxiv.org/abs/2305.05176.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, and Ice Pasupat et. al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025. URL https://arxiv.org/abs/2507.06261.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid llm: Cost-efficient and quality-aware query routing, 2024. URL https://arxiv.org/abs/2404.14618.

Divyansh Garg, Shaun VanWeelden, Diego Caples, Andis Draguns, Nikil Ravi, Pranav Putta, Naman Garg, Tomas Abraham, Michael Lara, Federico Lopez, James Liu, Atharva Gundawar, Prannay Hebbar, Youngchul Joo, Jindong Gu, Charles London, Christian Schroeder de Witt, and Sumeet Motwani. Real: Benchmarking autonomous agents on deterministic simulations of real websites, 2025. URL https://arxiv.org/abs/2504.11543.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, and Abhinav Pandey et. al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. Stabletoolbench: Towards stable large-scale benchmarking on tool learning of large language models, 2025. URL https://arxiv.org/abs/2403.07714.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pp. 65–70, 1979.

Shiting Huang, Zhen Fang, Zehui Chen, Siyu Yuan, Junjie Ye, Yu Zeng, Lin Chen, Qi Mao, and Feng Zhao. Critictool: Evaluating self-critique capabilities of large language models in tool-calling error scenarios, 2025. URL https://arxiv.org/abs/2506.13977.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues?, 2024. URL https://arxiv.org/abs/2310.06770.

Ishan Kavathekar, Raghav Donakanti, Ponnurangam Kumaraguru, and Karthik Vaidhyanathan. Small models, big tasks: An exploratory empirical study on small language models for function calling, 2025. URL https://arxiv.org/abs/2504.19277.

Shirley Kokane, Ming Zhu, Tulika Awalgaonkar, Jianguo Zhang, Thai Hoang, Akshara Prabhakar, Zuxin Liu, Tian Lan, Liangwei Yang, Juntao Tan, Rithesh Murthy, Weiran Yao, Zhiwei Liu, Juan Carlos Niebles, Huan Wang, Shelby Heinecke, Caiming Xiong, and Silivo Savarese. Toolscan: A benchmark for characterizing errors in tool-use llms, 2025. URL https://arxiv.org/abs/2411.13547.

Daniël Lakens. Equivalence tests: A practical primer for t tests, correlations, and meta-analyses. *Social psychological and personality science*, 8(4):355–362, 2017.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. Api-bank: A comprehensive benchmark for tool-augmented llms, 2023. URL https://arxiv.org/abs/2304.08244.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents, 2025. URL https://arxiv.org/abs/2308.03688.

Zhenyan Lu, Xiang Li, Dongqi Cai, Rongjie Yi, Fangming Liu, Xiwen Zhang, Nicholas D. Lane, and Mengwei Xu. Small language models: Survey, measurements, and insights, 2025. URL https://arxiv.org/abs/2409.15790.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. Agentboard: An analytical evaluation board of multi-turn llm agents, 2024. URL https://arxiv.org/abs/2401.13178.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants, 2023. URL https://arxiv.org/abs/2311.12983.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data, 2025. URL https://arxiv.org/abs/2406.18665.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis, 2023. URL https://arxiv.org/abs/2305.15334.

Shishir G. Patil, Huanzhi Mao, Charlie Cheng-Jie Ji, Tianjun Yan, et al. Berkeley Function Calling Leaderboard. https://gorilla.cs.berkeley.edu/leaderboard.html, 2024. V1–V4.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis, 2023. URL https://arxiv.org/abs/2307.16789.

JV Roig. How do llms fail in agentic scenarios? a qualitative analysis of success and failure scenarios of various llms in agentic simulations, 2025. URL https://arxiv.org/abs/2512.07497.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. Identifying the risks of lm agents with an lm-emulated sandbox, 2024. URL https://arxiv.org/abs/2309.15817.

Donald J Schuirmann. A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. *Journal of pharmacokinetics and biopharmaceutics*, 15(6):657–680, 1987.

Aaditya Singh, Adam Fry, Adam Perelman, and Adam Tart et. al. Openai gpt-5 system card, 2026. URL https://arxiv.org/abs/2601.03267.

Gemma Team. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. Mint: Evaluating llms in multi-turn interaction with tools and language feedback, 2024. URL https://arxiv.org/abs/2309.10691.

Cailin Winston and René Just. A taxonomy of failures in tool-augmented llms. In *2025 IEEE/ACM International Conference on Automation of Software Test (AST)*, pp. 125–135. IEEE, 2025.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments, 2024. URL https://arxiv.org/abs/2404.07972.

An Yang, Anfeng Li, Baosong Yang, and Beichen Zhang et. al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains, 2024. URL https://arxiv.org/abs/2406.12045.

Lucen Zhong, Zhengxiao Du, Xiaohan Zhang, Haiyi Hu, and Jie Tang. Complexfuncbench: Exploring multi-step and constrained function calling under long-context scenario, 2025. URL https://arxiv.org/abs/2501.10132.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *ICLR*, 2024. arXiv:2307.13854.