Pokee-Isaac 28B: A 10M-Token Context Efficient Agentic Model

Technical Report

Zheqing Zhu, Christopher Wu, Yi Wan, Yiqing Cai, Qiwei He, Ruihao Zhu, Chang Zen Tee, Lexin Chen, Pu Thavikulwat, Jared Ee

Pokee AI

Abstract

Pokee-Isaac 28B is a non-decoder-only foundation model that reasons, plans, and uses tools over context windows of up to 10 million tokens. Capability at this context length has been assumed to require a model too large to deploy locally, and is today delivered almost exclusively from the cloud even when the supported context is much shorter. This leaves regulated, sovereign, and on-device settings, where data is not permitted to leave the boundary, with no path to it at all. Isaac runs on a single GPU — on one NVIDIA B200 it prefills at up to 137K tokens/s and decodes at 335 tokens/s — and is compact enough to serve on a client workstation. It holds retrieval fidelity across that full window, and matches or exceeds the strongest cost-optimized cloud systems on function calling, multi-turn interactive execution, tool orchestration, and terminal work. Deploying a long-context agentic model locally, inside the boundary where the data already sits, therefore becomes a real option.

Pokee-Isaac 28B V0: Benchmark Snapshot

Pokee AI

🏆 WORLD'S FIRST REAL 10M TOKEN CONTEXT MODEL

$0.15 IN / $1 OUT PER 1M TOKENS

API PRICING: USD PER 1M INPUT / OUTPUT TOKENS

Benchmark	Pokee-Isaac 28B (V0) In / Out $0.15 / $1.00	GPT-5.6-luna In / Out $0.40 / $1.80 >272K context	Gemini 3.5 Flash Lite In / Out $0.30 / $2.50	Claude Haiku 4.5 In / Out $1.00 / $5.00	Nemotron-3-Super-120B In / Out $0.15 / $0.65 Amazon Bedrock · US	Qwen3.5-122B In / Out $0.26 / $2.08 OpenRouter
LONG CONTEXT						
RULER 256K/512K/1M	**96.9 / 96.7 / 95.0** 🥇	95.0 / 91.4 / 0.0*	94.5 / 94.6 / 29.4*	0.0 / 0.0 / 0.0	96.30 / 95.67 / 91.75 [5]	0.0 / 0.0 / 0.0
RULER 2M/4M/10M	**95.8 / 96.7 / 93.3** 🥇	0.0 / 0.0 / 0.0	0.0 / 0.0 / 0.0	0.0 / 0.0 / 0.0	0.0 / 0.0 / 0.0	0.0 / 0.0 / 0.0
MRCR v2 256K/512K/1M	**0.607 / 0.743 / 0.500** 🥇	0.208 / 0.173 / 0.050	0.474 / 0.473 / 0.205	0.000 / 0.000 / 0.000	0.145 / 0.161 / 0.067	0.000 / 0.000 / 0.000
AGENTIC CAPABILITIES						
BFCL v4 overall	**70.94** 🥇	70.61	64.85	67.52	33.13	64.88
τ³ 4-domain avg	**0.662** 🥇	0.527	0.631	0.408	0.426	0.611
Terminal-Bench 2.1	65.1	**69.8**	46.5	34.9	24.4	46.5
MCP-Atlas	74.59	**77.90**	76.67	56.45	48.95	70.24
SECURITY						
DTAP ASR ↓ Attack success rate	**35.6** 🥇	50.1	66.3	37.9	60.4	54.0
DTAP BSR ↑ Benign success rate	82.5	**85.1**	83.3	71.3	63.3	79.4

SOURCE: INTERNAL BENCHMARK SUMMARY

Note: ASR = attack success rate; BSR = benign success rate. Lower ASR is better; higher BSR is better. Higher is better for all other benchmarks unless otherwise indicated. * indicates 1M context score is 0.0 in the provided source. [5] marks a sourced external result. **Pricing:** Pokee and Luna rates shown as supplied/official; Gemini and Haiku use standard public API rates; Nemotron uses Amazon Bedrock on-demand pricing for US East/US West; Qwen uses the OpenRouter headline rate, which can vary by provider.

1 Introduction

Rapid developments in large language model (LLM)-based agentic systems enable organizations to delegate complex tasks to them. To complete a task, an agent must iteratively plan, execute external tools, and reason over observations without human supervision. As task complexity grows, agents must execute long-horizon workflows across multiple steps. This requires two coupled capabilities: first, the ability to maintain a long context, as long-horizon tasks continuously accumulate observations, tool outputs, and intermediate reasoning; second, the ability to remain coherent across that horizon, including tracking state and recovering from errors as the trajectory expands.

While recently launched agentic systems make meaningful attempts to deliver these two capabilities, two challenges emerge when transitioning to real-world deployments:

- **Privacy and Data Sovereignty.** First, systems and LLM models capable of handling long-horizon, long-context tasks today are almost exclusively hosted via cloud services due to prohibitively high infrastructure costs. Consequently, organizations must transmit sensitive data externally and accept third-party operational terms, which immediately disqualifies these models for regulated industries, public-sector institutions, and on-device applications.
- **Affordability.** Moreover, under current token-based pricing models, the cost of completing a task scales steeply with context length. As a result, running complex, long-horizon workflows requiring extensive context rapidly becomes financially unviable at scale.

To address both challenges, we introduce **Pokee-Isaac 28B, a 10M-token context window, non-decoder-only** model engineered to operate within a compact compute budget. It enables full deployment inside a virtual private cloud (VPC), on-premises, or directly on-device. By keeping execution strictly within customer-governed boundaries, Isaac preserves complete data sovereignty and eliminates variable per-token pricing. Below, we provide an overview of Pokee-Isaac 28B's capabilities:

- **Frontier 10M Long Context.** We present Pokee-Isaac 28B, a 28B model that maintains near-flat retention and retrieval fidelity across a 10M-token window. It scores 93.3% on RULER (Hsieh et al., 2024) at 10M context length and achieves leading performance in MRCR v2 (Vodrahalli et al., 2024) at 1M context length. In particular, Issac remains reliable where conventional cloud and open-weight baselines hit context overflow boundaries (see Section 2).
- **High-Efficiency Agentic Performance.** To demonstrate Isaac's capability in maintaining coherence over long-horizon, we show that it can match or even outperform larger proprietary and open models across function calling, multi-turn interactive execution, terminal navigation, and tool composition (see Section 3).
- **In-Boundary Security and Governance.** We design Isaac for secure, in-boundary operations. It allows strict isolation, role-aware access controls, and auditable execution traces. Together, these achieve the lowest attack success rates among capability-matched baselines on the DTAP red-teaming benchmark (Chen et al., 2026) (see Section 4).
- **High-Efficiency Long-Context Economics.** We demonstrate that Isaac scales prefill throughput up to 137,200 tokens/s at 10M context on a single B200 GPU while holding decode speed flat ($\sim$335 tokens/s). This enables list pricing below major cloud baselines ($0.15/$1.00 per million input/output tokens) and allowes enterprises to convert variable token costs into a predictable, fixed expense when deployed in private computing environments (see Section 5).

- **Cross-Platform Edge Portability.** We deploy extended long-context agency beyond traditional datacenters to client workstations and edge-class silicon (including Intel Arc Pro B70, Panther Lake and Qualcomm Snapdragon X2 Elite) served natively via the Pokee inference SDK (see Section 6).

Some weights in Pokee-Isaac are fine-tuned from Qwen3.6-27B (Apache 2.0) (Qwen Team, 2026). For all benchmark results reported below, peer models use their default temperatures and, when available, a `high` thinking level.

2 Long-Context Capability Evaluation

We demonstrate the retention capability of Pokee-Isaac 28B on two prevalent benchmarks against five models.

2.1 Evaluation Setup

Benchmarks. We evaluate long-context retrieval using RULER (Hsieh et al., 2024) and Multi-Round Coreference Resolution (MRCR) v2 (Vodrahalli et al., 2024).

RULER is a synthetic long-context benchmark that constructs retrieval, multi-hop tracing, and aggregation tasks across controlled context lengths. By holding task difficulty constant while scaling context length, RULER isolates how a model's usable context tracks its nominal context window.

MRCR v2, introduced by Google DeepMind, presents a more demanding variant of the needle-in-a-haystack paradigm. Rather than inserting a single target, it distributes multiple targets ("needles") throughout a long synthetic conversation, requiring the model to retrieve and disambiguate a specified needle. This design explicitly penalizes partial recall and cross-needle interference.

Together, these two benchmarks test a model's ability to overcome two distinct failure modes that standard single-needle tests conflate: performance degradation across context depth and interference among multiple co-occurring targets.

Baselines. We compare Pokee-Isaac 28B against five strong baselines: GPT-5.6 Luna, Gemini 3.5 Flash Lite, Claude Haiku 4.5, Nemotron 3 Super 120B, and Qwen 3.5 122B.

GPT-5.6 Luna, Gemini 3.5 Flash Lite, and Claude Haiku 4.5 represent the high-throughput, cost-optimized tiers of the three primary cloud providers, offering a practical trade-off between capability and operational expense. In contrast, Nemotron 3 Super 120B and Qwen 3.5 122B are leading open-weight models that can be self-hosted within an organization's private boundary. Notably, Pokee-Isaac 28B is the smallest model in this panel by a wide margin, carrying less than a quarter of the parameters of the open-weight baselines.

We deliberately exclude frontier flagship models, such as GPT-5.6 Sol, Claude Opus 5, and Gemini 3.1 Pro, as they cost roughly an order of magnitude more per token, remain strictly cloud-hosted, and address a fundamentally different deployment envelope.

Evaluation Protocol and Reproducibility. We evaluate our model and all baselines through their hosted APIs via OpenAI-compatible adapters. Reasoning capabilities were enabled whenever

supported and stable. To ensure rigorous accounting, exhausted retries, context overflows, and infrastructure or verifier failures were treated as zero-score failures.

For RULER, we adopt NVIDIA's official data preparation pipeline and scoring suite, evaluating 10 samples per task configuration. All 13 task configurations are evaluated at 256K and 512K context lengths. Because common-words extraction (`CWE`) is unsupported at or above 1M tokens due to the requirement of sampling from a small fixed vocabulary, evaluations from 1M onward average the remaining 12 tasks.

For MRCR v2, we utilize the official 8-needle split, comprising 30 paired samples across the 256K, 512K, and 1M token length bins (measured via the `o200k_base` tokenizer). Outputs are scored using the official metric combination of random-prefix filtering and `SequenceMatcher` alignment.

Every result reported herein was generated independently on our infrastructure unless explicitly noted otherwise. Where we include external figures published by vendors, these are explicitly labeled at the point of use, kept strictly distinct from our primary comparisons, and never averaged into our evaluations. We observe discrepancies between our empirical measurements and third-party scores reported elsewhere; such differences are expected, as benchmark performance is highly sensitive to the agent harness, prompt templates, sampling parameters, turn limits, and task subsets—details that are frequently omitted in literature. Consequently, rather than comparing against unverified external numbers, we hold the evaluation environment, harness, and sampling parameters fixed across all baselines and report our direct measurements under uniform conditions. The detailed serving profile is reported in Section 5.1.

2.2 Performance on RULER

The evaluation results on RULER are detailed in Table 1 and illustrated in Figure 1. Pokee-Isaac 28B consistently maintains an accuracy above 93.3% across all tested context lengths, standing as the only model in our evaluation panel to do so. In contrast, every other baseline drops significantly short of this benchmark. While GPT-5.6 Luna and Gemini 3.5 Flash Lite closely match Isaac's performance up to 512K tokens, both encounter context-overflow limits at 1M tokens and beyond. Claude Haiku 4.5 and Qwen 3.5 122B fail to complete the benchmark, scoring 0.0% across all context windows. Nemotron 3 Super 120B degrades to 0.0% from 2M tokens onward, whereas NVIDIA self-reports scores of 96.30%, 95.67%, and 91.75% at 256K, 512K, and 1M context lengths, respectively.

Model	**256K**	**512K**	**1M**	**2M**	**4M**	**10M**
Pokee-Isaac 28B	**96.9**	**96.7**	**95.0**	**95.8**	**96.7**	**93.3**
GPT-5.6 Luna (Azure)	95.0	91.4	0.0*	0.0	0.0	0.0
Gemini 3.5 Flash Lite (Vertex AI)	94.5	94.6	29.4*	0.0	0.0	0.0
Claude Haiku 4.5 (Bedrock)	0.0	0.0	0.0	0.0	0.0	0.0
Nemotron 3 Super 120B	96.30[s]	95.67[s]	91.75[s]	0.0	0.0	0.0
Qwen 3.5 122B	0.0	0.0	0.0	0.0	0.0	0.0

Table 1: RULER score (%, averaged across task configurations) by context length, at ten samples per task configuration. The 256K and 512K columns include the common-words extraction (CWE) task; CWE is unavailable from 1M onward, so those columns average the remaining 12 configurations. Bold marks the best value in each column. * *Context-overflow error at this length.* [s] *Self-reported by the model vendor; not measured by us.* Serving profile in Section 5.1.

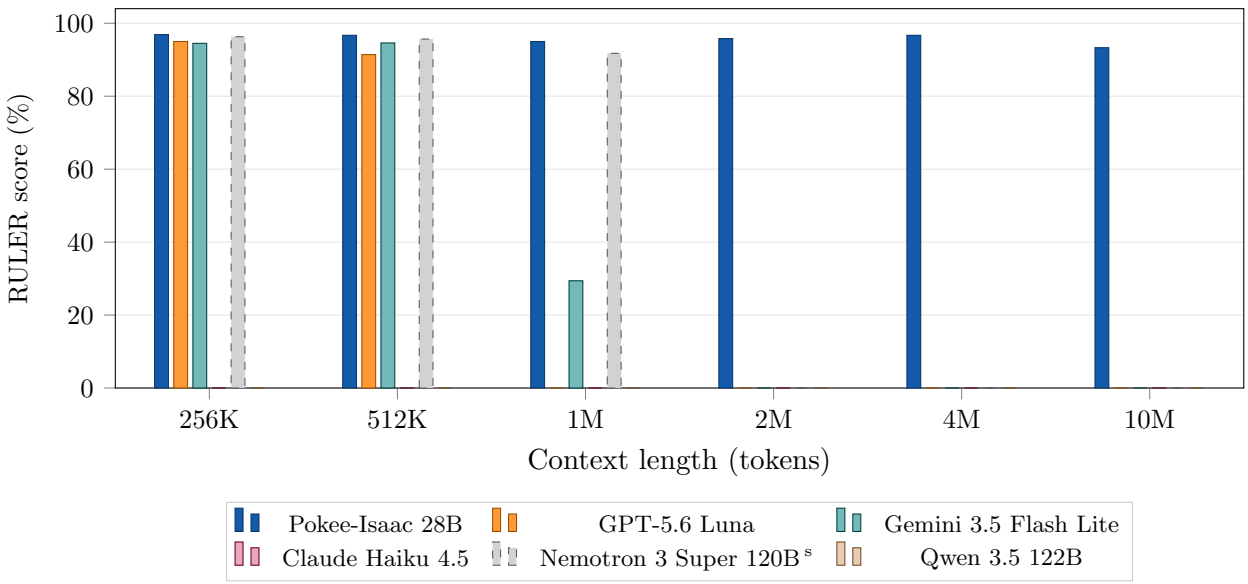

Figure 1: RULER score by context length. Pokee-Isaac 28B is the only one sustains beyond 1M context. Nemotron 3 Super 120B's bars at 256K–1M (dashed outline) are NVIDIA's self-reported figures; from 2M onward that series is our own and is zero.

2.3 Performance on MRCR v2

On MRCR v2, we report model performance on the 8-needle setting swept across 256K, 512K, and 1M-token context windows in Table 2. Pokee-Isaac 28B leads across all evaluated lengths, peaking at 0.743 at 512K and maintaining 0.500 at 1M. Gemini 3.5 Flash Lite serves as the closest baseline, though Isaac's advantage broadens significantly as context scales, expanding from a 0.133 margin at 256K to 0.270 at 512K, and reaching 0.295 at 1M where Gemini drops to 0.205.

GPT-5.6 Luna trails across the entire sweep and collapses to 0.050 at 1M tokens. Notably, despite achieving a high score of 95.0% on RULER at 256K (see Table 1), GPT-5.6 Luna degrades rapidly on MRCR v2, highlighting how multi-needle disambiguation provides a far sharper differentiator than single-target retrieval. Among remaining baselines, Nemotron 3 Super 120B yields low but non-zero performance (peaking at 0.161), whereas Claude Haiku 4.5 and Qwen 3.5 122B fail to produce usable outputs at any length due to their native context of 200K and 262K respectively. Ultimately, four of the five baselines score below 0.210 across the full sweep, while Isaac retains 50% retrieval fidelity even at 1M tokens.

	MRCR v2, 8 needles		
Model	**256K**	**512K**	**1M**
Pokee-Isaac 28B	**0.607**	**0.743**	**0.500**
GPT-5.6 Luna (Azure)	0.208	0.173	0.050
Gemini 3.5 Flash Lite (Vertex AI)	0.474	0.473	0.205
Claude Haiku 4.5 (Bedrock)	0.000	0.000	0.000
Nemotron 3 Super 120B	0.145	0.161	0.067
Qwen 3.5 122B	0.000	0.000	0.000

Table 2: MRCR v2 (8-needle) score vs. context length, on a 0–1 scale. A score of 0.000 indicates the model returns no usable score at that length.

3 Agentic Capabilities Evaluation

Agentic capability in today's AI refers to the use of tools, such as CLIs, JSON-style function-calling APIs, Context Management Primitives, to achieve user-defined goals. We evaluate Isaac's agentic capabilities on four widely used benchmarks: BFCL v4, Tau3, Terminal-Bench 2.1, and MCP Atlas. Each targets a different facet of agency, so strength on one cannot mask weakness on another.

3.1 BFCL v4: function calling

The Berkeley Function Calling Leaderboard (BFCL) is a long-running public benchmark for tool use, maintained by the Gorilla group at UC Berkeley (Patil et al., 2025). It decomposes tool use into five scored components, each isolating a capability an agent depends on.

Two components evaluate single-turn calls — selecting the right function from a candidate set and populating its arguments with correct names, types, and values — one over expert-curated schemas (Non-Live) and one over functions contributed by hobbyists and enterprises from real deployments (Live). Within the function-calling paradigm this is the atomic operation; an agent that emits malformed calls fails before any planning matters. A third component tests the complementary skill of restraint (Hallucination Measurement): given functions that cannot serve the request, the model must decline rather than fabricate a plausible call. A fourth tests stateful multi-step execution across eight simulated API domains, including recognizing that a required function is unavailable and that a required parameter must be requested from the user (Multi-Turn). The fifth, introduced in v4, covers web search and memory management (Agentic), and is described below. The overall score combines the five under fixed weights, over a suite of 5,106 scored entries:

$$0.40 \cdot \texttt{Agentic} + 0.30 \cdot \texttt{Multi-Turn} + 0.10 \cdot \texttt{Live} + 0.10 \cdot \texttt{Non-Live} + 0.10 \cdot \texttt{Hallucination}$$

The Agentic component adds two conditions absent from the categories above. Its web search category poses multi-hop questions answerable only through a search tool and a page-fetch tool, run against a deliberately non-optimal backend whose fetches fail probabilistically with realistic HTTP and network errors. Its memory category replays multi-turn conversations while the model manages an external store through tool calls, then evaluates retrieval from a reloaded snapshot with the dialogue history removed, so that correctness depends jointly on what the model chose to store and on whether it can retrieve it later. The distinguishing property in both cases is that the environment is open and unreliable, and that state must be externalized and managed rather than read back from context — which is why BFCL reserves the "agentic" label for them, though

multi-turn execution is agentic in character as well.

No component is graded by an LLM judge; verification is programmatic throughout, with the method matched to the category. Single-turn calls are checked by parsing the model's output into an abstract syntax tree and comparing it against a set of accepted references. Multi-turn tasks execute against stateful backends and must pass two checks at the end of every turn: a state comparison against the expected backend state, which captures writes and deletes, and a subset match confirming that the labeled minimal execution path appears in the trajectory, which catches read-only calls that leave no trace. The subset criterion means exploration and error recovery are not penalized. Web search answers are scored by normalized exact match on a structured answer field; because that category queries the live internet under injected failures, it is the one component carrying genuine run-to-run variance.

Category	**Isaac**	**Luna**	**Gemini**	**Haiku**	**Nemotron**	**Qwen**
BFCL v4 overall	**70.94**	70.61	64.85	67.52	33.13	64.88
Single-turn AST (200 samples each; `simple` covers Python, Java, and JavaScript)						
`simple_python`	91.0	88.5	94.8	**95.2**	94.8	95.0
`simple_java`	63.0	60.0	61.0	62.0	**65.0**	63.0
`simple_javascript`	**78.0**	66.0	70.0	68.0	72.0	72.0
`multiple`	93.5	89.5	93.5	95.0	96.0	**96.5**
`parallel`	92.0	86.5	93.5	90.5	0.0	**94.0**
`parallel_multiple`	88.0	73.0	**91.0**	90.0	0.5	90.5
Live (user-submitted) and relevance						
`live_simple`	88.4	73.3	**89.5**	86.8	85.7	**89.5**
`live_multiple`	**83.6**	69.5	79.5	80.0	77.7	80.8
`live_parallel`	**93.8**	56.2	81.2	87.5	0.0	81.2
`live_parallel_multiple`	**83.3**	66.7	79.2	**83.3**	0.0	**83.3**
`irrelevance`	79.2	74.2	65.4	**82.9**	0.8	81.2
`live_irrelevance`	68.0	70.8	56.6	**81.1**	3.6	71.6
`live_relevance`	**100.0**	81.2	**100.0**	68.8	**100.0**	87.5
Multi-turn (200 samples each)						
`multi_turn_base`	**77.5**	74.0	74.5	67.5	34.5	65.5
`multi_turn_miss_func`	55.5	**60.5**	53.5	47.5	14.0	51.5
`multi_turn_miss_param`	44.5	**62.5**	28.5	51.5	26.0	45.0
`multi_turn_long_context`	**70.0**	68.5	**70.0**	59.5	27.0	59.5
Agentic (memory and web search)						
`memory_kv`	56.1	54.2	**66.5**	54.8	28.4	38.7
`memory_vector`	**74.2**	56.1	65.2	52.3	12.9	51.0
`memory_rec_sum`	69.0	**71.6**	63.2	52.9	40.6	50.3
`web_search_base`	76.0	**84.0**	77.0	81.0	48.0	72.0
`web_search_no_snippet`	69.0	**79.0**	41.0	67.0	30.0	69.0

Table 3: Berkeley Function-Calling Leaderboard (BFCL v4): overall score and per-category detail, from the authoritative score files. Models: Pokee-Isaac 28B, GPT-5.6 Luna (Azure), Gemini 3.5 Flash Lite (Vertex AI), Claude Haiku 4.5 (Bedrock), Nemotron 3 Super 120B, Qwen 3.5 122B. Best score per row in bold.

On the BFCL v4 overall score, Pokee-Isaac 28B leads the panel at 70.94, ahead of GPT-5.6 Luna at 70.61 and Claude Haiku 4.5 at 67.52 (Table 3). The margin over Luna is 0.33 points and should

be read as parity rather than as a decisive lead; the substantive claim is that Isaac matches the strongest cloud baseline in the panel while remaining deployable on a single GPU.

3.2 Tau3: multi-turn interactive tasks

τ^3-bench is the third generation of Sierra's tool-agent-user benchmark [Barres et al., 2025]. An agent handles a customer-service interaction against an LLM-simulated user under a written domain policy, calling tools over many turns. The defining difference from BFCL is that the task is not specified in advance: BFCL states the request and asks whether the agent executed it correctly, whereas τ^3 presents a simulated user who discloses their situation gradually and a policy document that constrains what is permissible, and requires the agent to determine what should happen. Both benchmarks execute against stateful backends and both accept multiple valid trajectories. Four capabilities follow.

The benchmark spans four domains, which differ in which of these capabilities dominates. Retail and airline, carried forward from the earlier versions of this benchmark, are transactional: the agent modifies orders, bookings, refunds, and exchanges under a policy supplied in its prompt, and the difficulty lies in applying that policy exactly — airline in particular demands refusals, since its rules on changes and cancellations are restrictive. Telecom, introduced in τ^3, is diagnostic rather than transactional: the agent troubleshoots a connectivity fault, and because the remedies are physical — reseating a SIM card, toggling airplane mode — it must work through the user, whose technical competence varies by an assigned persona. This is the domain where dual-control dominates. Banking, introduced in τ^3, is by a wide margin the hardest domain, and it is hard for a different reason than the others: the policy is not supplied in the prompt but distributed across a corpus of 698 documents spanning 21 product categories (195K tokens). A single task draws on 18.6 documents and 9.5 tool calls on average, with some requiring 33, including procedures and tools referenced only inside the documentation rather than exposed in the tool list.

Task success is verified against a per-task rubric drawn from five criteria — a database check on the final backend state, environment assertions over the final world state, action matching confirming that each gold tool call appears in the trajectory, communication checks that required information was conveyed to the user, and natural-language assertions for conditions better stated in prose. Each task selects the subset appropriate to it; telecom, for instance, is scored by assertion functions alone. Scoring thus targets what the agent did and what it told the user, not how fluent it sounded.

All runs used seed 300 for randomness, 200 maximum turns, temperature 0, and an Azure GPT-5.2 user simulator at temperature 0. Shared auxiliary judge and interface calls used Azure GPT-5.4; LLM requests had a 90-second timeout and two retries.

We report pass@1; the benchmark additionally defines pass@k over k independent attempts, and a voice track, neither of which we cover here. We report retail, airline, and telecom individually and as an average, and banking separately. The gap justifies the split: frontier models routinely exceed 80% pass@1 on the first three, while the strongest reported result on banking at release was 25.5% pass@1; averaging the four would obscure both. Pokee-Isaac 28B leads the panel at 0.662 averaged across the four domains (Table 4).

Model	Retail	Airline	Telecom	Banking	Avg
Pokee-Isaac 28B	**0.789**	**0.760**	0.912	0.186	**0.662**
GPT-5.6 Luna (Azure)	0.623	0.720	0.579	0.186	0.527
Gemini 3.5 Flash Lite (Vertex AI)	0.719	0.700	0.904	**0.203**	0.631
Claude Haiku 4.5 (Bedrock)	0.667	0.500	0.404	0.062	0.408
Nemotron 3 Super 120B	0.614	0.688	0.368	0.033	0.426
Qwen 3.5 122B	0.693	0.660	**0.947**	0.144	0.611

Table 4: τ^3-bench pass[1] scores, text track. *Avg* is the unweighted mean across the four domains. Best score in each column is shown in bold. Rows are ordered by the model panel used throughout this report, not by rank.

3.3 MCP Atlas: multi-server tool orchestration

The two benchmarks above both hold the tool surface fixed and under the evaluator's control. τ^3-bench is a sandbox throughout: its domains are mock databases behind an API implemented for the benchmark, so both the data and the interface are authored by the evaluation itself. BFCL is only partly simulated — its web-search subset queries the live web, and the facts it retrieves are real — but the tools through which it does so are two wrapper functions written by the benchmark, and its stateful and memory categories run against mock services. In both cases one author designed the entire tool surface, which is therefore internally consistent by construction.

That is not the surface an agent meets in deployment. There, the tools come from services that already exist and are maintained by unrelated parties: each is designed for its own purposes, models an overlapping slice of the world under its own naming, and returns whatever a production system returns — verbose, occasionally malformed, sometimes failing transiently. No one designed the collection as a whole, and completing a request against it means deciding which service owns which part of the request and carrying intermediate results across boundaries the schemas were never built to meet. Adopting the Model Context Protocol does not by itself reproduce this condition: the protocol is an interface standard, and a mock server speaks it as fluently as a real one. Several contemporary MCP evaluations are built on mocks for exactly this reason.

MCP-Atlas (Bandi et al., 2026) makes the opposite commitment. Its tasks run exclusively against real, independently authored production MCP servers, and the great majority require calls to more than one of them, with the prompt naming neither the servers nor the tools. This is the capability we want measured, and it is why we run MCP-Atlas in addition to the other two rather than in place of them.

We evaluated all 500 public MCP-Atlas tasks against a live 36-server MCP sandbox with caches disabled. These MCP servers include Github, Slack, Google Workspace, Notion, Oxylabs, and more real-world tools. Runs allowed 40 tool turns and 4,096 output tokens. Pokee-Isaac 28B used temperature 0.7; the comparison models used 1.0. A shared GPT-5.5 judge measured mean claim coverage over all 500 tasks.

Pokee-Isaac 28B places third at 74.59% coverage, behind GPT-5.6 Luna (77.90%) and Gemini 3.5 Flash Lite (76.67%), and ahead of Qwen 3.5 122B, Claude Haiku 4.5, and Nemotron 3 Super 120B (Table 5). The turn counts are worth reading alongside the coverage: Isaac reaches within 2.1 points of Gemini while taking 9.10 turns per task against Gemini's 14.99, roughly 60% of the trajectory length for comparable coverage. Luna leads on both, resolving more tasks in fewer turns than any other model in the panel.

Model	Coverage	Pass ≥ 0.75	Mean turns	Rank
Pokee-Isaac 28B	74.59%	327/500	9.10	3
GPT-5.6 Luna (Azure)	**77.90%**	356/500	6.91	1
Gemini 3.5 Flash Lite (Vertex AI)	76.67%	344/500	14.99	2
Claude Haiku 4.5 (Bedrock)	56.45%	230/500	7.40	5
Nemotron 3 Super 120B	48.95%	190/500	11.18	6
Qwen 3.5 122B	70.24%	307/500	11.47	4

Table 5: MCP-Atlas, 500-task public release. Coverage is the mean fraction of required claims satisfied; *Pass* ≥ 0.75 counts tasks reaching at least 0.75 coverage; *Mean turns* is the average trajectory length. Rows are ordered by the model panel used throughout this report, not by rank.

3.4 Terminal-Bench 2.1: agentic execution in a shell

All three benchmarks so far declare the agent's action space in advance. BFCL supplies candidate schemas, τ^3-bench a domain API, MCP-Atlas a pool of tools from real servers; the pools differ in authorship and in how much distraction they carry, but in each case the permitted actions are enumerated before the episode begins and the task is to select among them and fill in arguments. Terminal-Bench 2.1 (Merrill et al., 2026) removes that premise: the agent is placed at a command line on a Linux machine and given nothing else. There is no enumerated set of actions — the action is an arbitrary command string, and the available operations are whatever binaries the machine has installed plus everything they compose into through pipes, redirection, and scripts the agent writes itself. Feedback is unstructured text rather than a typed result: a command may exit non-zero, or exit zero having done the wrong thing. State is mutable and largely destructible, and tasks run long enough that errors compound rather than surfacing at the end of a short chain.

Terminal-Bench 2.1 consists of technical tasks spanning software engineering, machine learning, security, and system administration, each run in a containerized sandbox and defined by an instruction, a test script that decides whether it was accomplished, and a reference solution establishing that it is solvable. We report the fraction of tasks resolved.

Terminal-Bench 2.1. Every model was driven by the same agent software, which translates model output into terminal actions and returns execution results: Harbor 0.20.0 with Terminus-2. We evaluated the 86 text-compatible tasks of the 89-task v2.1 set, excluding the three requiring non-text modalities, as Isaac is currently text-only. Each task was attempted once under a 100-turn cap to measure the full potential of each model, with a 16× multiplier applied to each task's default timeout to eliminate potential infra limitations from cloud providers and the official binary verifiers. Errored trials are counted as failures.

Pokee-Isaac 28B places second in the panel at 65.1% (56 of 86 tasks), behind GPT-5.6 Luna at 69.8% (60 of 86)—a gap of four tasks (Table 6). It leads the remaining four baselines by a wide margin, including both open-weight models an order of magnitude larger: Qwen 3.5 122B and Gemini 3.5 Flash Lite tie at 46.5%, and Nemotron 3 Super 120B trails at 24.4%. Terminal-Bench is the one benchmark in this report where a cloud baseline finishes ahead of Isaac, and we report it as measured.

Model	Passed	Score	Rank
Pokee-Isaac 28B	56/86	65.1%	2
GPT-5.6 Luna (Azure)	**60/86**	**69.8%**	1
Gemini 3.5 Flash Lite (Vertex AI)	40/86	46.5%	3=
Claude Haiku 4.5 (Bedrock)	30/86	34.9%	5
Nemotron 3 Super 120B	21/86	24.4%	6
Qwen 3.5 122B	40/86	46.5%	3=

Table 6: Terminal-Bench 2.1, text-only task subset (86 of 89 tasks), 100-turn cap per task. Rows are ordered by the model panel used throughout this report, not by rank.

4 Security Capability

This section describes the security and governance properties Pokee-Isaac 28B is built to support in deployment. The goal is not to claim perfect safety in the abstract, but to specify the controls that let Isaac be operated inside regulated or high-sensitivity boundaries.

In-boundary operation. Isaac is intended to run where sensitive data lives (VPC, on-premises, or on-device), avoiding dependence on an external API boundary for core functionality.

Policy-aware use of tools and data. When integrated through the Pokee SDK, Isaac can be constrained to approved tool surfaces and data sources, with role-appropriate permissions and deny-by-default behavior.

Auditability. Tool calls, intermediate artifacts, and outputs can be logged in a structured form suitable for monitoring and post-hoc review.

Benchmark. We evaluate Isaac's security behavior on the DecodingTrust-Agent Platform (DTAP) (Chen et al., 2026), a red-teaming benchmark that places an agent in simulated real-world environments and measures whether injected attacks succeed. Three rates are reported. *Direct ASR* is the attack success rate when the harmful request appears in the user prompt; *Indirect ASR* is the rate when a malicious instruction arrives through tool output, skill content, or the environment itself. *Combined ASR* is the task-weighted aggregate of the two. Both attack rates and their combination are lower-is-safer. *BSR* is the benign task success rate—utility—and is higher-is-better. Our run covers the 12 Linux-Docker domains and 6,195 judged tasks: 2,676 benign, 1,687 direct-attack, and 1,832 indirect-attack.[1]

Controlled comparison. We compare Isaac against the five baselines used throughout this report, run on the same installation with the judge held fixed (Table 7). Isaac is the safest of the six on both attack axes and on their combination, while placing third of six on capability, with thin margins of 2.6 points behind GPT 5.6 Luna and 0.8 points behind Gemini 3.5 Flash Lite.

[1]Run of 2026-07-31, using the benchmark's own environments, attacks, and judges. Single run, no error bars: per-domain ASR gaps within roughly ±3 points should be treated as noise. The macOS and Windows domains are guest-OS-in-QEMU and were not in scope for this run.

Model	Direct ASR	Indirect ASR	Combined ASR	BSR
Pokee-Isaac 28B	**36.0**	**35.2**	**35.6**	82.5
GPT-5.6 Luna (Azure)	54.4	46.1	50.1	**85.1**
Gemini 3.5 Flash Lite (Vertex AI)	84.1	49.5	66.3	83.3
Claude Haiku 4.5 (Bedrock)	38.2	37.1	37.9	71.3
Nemotron 3 Super 120B	80.3	42.0	60.4	63.3
Qwen 3.5 122B	60.8	47.8	54.0	79.4

Table 7: DTAP, macro-average over the 12 Linux domains, same judge for every row. ASR is lower-is-safer; BSR is higher-is-better. Bold marks the best value in each column. The five baselines were run under the benchmark's stock runner; Isaac was run under the Pokee harness, which is the one condition that still differs across rows.

Balance across attack channels. Isaac's Direct and Indirect rates differ by 0.8 points, the tightest balance in the set. The models with minimal refusal training show the opposite pattern: Nemotron 3 Super 120B, Gemini 3.5 Flash Lite, and Qwen 3.5 122B run 38.3, 34.6, and 13.0 points higher on Direct than on Indirect, respectively. For any threat model that includes a user issuing a directly harmful instruction, Direct ASR is the number that matters, and it is where these systems diverge most.

5 Efficiency and Pricing

5.1 Efficiency

Capability results are reported above; this section reports the serving profile under which they are obtained. We measure Isaac under the RULER workload—so that efficiency is characterized on the same long-context task whose capability is reported in Section 2—on a single B200-class GPU, at 1M and 10M context and at concurrency 1 and 4. We report three quantities: time-to-first-token (TTFT, latency from request to first output token), prefill throughput (rate of prompt ingestion, derived from TTFT and context length), and decode throughput (sustained output generation, reported in aggregate across concurrent requests).

Context	Concurrency	TTFT (s)	Aggregate Prefill (tok/s)	Aggregate Decode (tok/s)
1M	1	23.6	42,400	335
1M	4	49.3	81,200	322
10M	1	72.9	137,200	337

Table 8: Serving profile under the RULER workload on a single B200-class GPU. At concurrency 4, TTFT is the mean across the concurrent requests, and prefill throughput is aggregate, computed as (concurrency × context length) divided by that mean. Decode throughput is aggregate across all concurrent requests, and at concurrency 4 is likewise a mean.

Two features of this profile are worth drawing out. First, prefill throughput *rises* with context length—from roughly 42K tokens per second at 1M to 137K at 10M on the same hardware—so a ten-fold increase in context costs about three times the time-to-first-token rather than ten times or more. This is the behavior that makes the full window usable in practice rather than merely addressable. Second, decode throughput is essentially flat across the sweep, holding near

335 tokens per second at both context lengths and 322 under concurrency 4; aggregate output throughput is therefore governed by the serving configuration rather than by how much context is resident. Reported this way, the profile is directly comparable to the capability curve of Section 2: the same workload that Isaac sustains to 10M tokens is the one profiled here for cost.

5.2 Pricing

Serving cost is what turns the efficiency profile of Section 5.1 into an operating constraint, so we state the per-token economics of the panel directly. Rates are the published list prices for each model at the context lengths this report evaluates.[2]

Three models in the panel cannot be priced at these lengths because no commercially available endpoint serves them there. Claude Haiku 4.5 caps at a 200K context window and Qwen 3.5 122B at 262K. Nemotron 3 Super 120B has a 1M native window, but every public endpoint we could purchase caps the served context at 262K, so there is no long-context rate to quote for it either; the Nemotron figures in this report come from weights we host ourselves. This is the same boundary visible in Table 1, where Haiku returns nothing from the 256K entry point onward.

Model	**Max context**	**Input ($/M)**	**Output ($/M)**
Pokee-Isaac 28B	10M	**0.15**	**1.00**
GPT-5.6 Luna (Azure) >272K context	1.05M	0.40	1.80
Gemini 3.5 Flash Lite (Vertex AI)	1M	0.30	2.50
Claude Haiku 4.5 (Bedrock)	200K	*Not supported beyond 256K*	
Nemotron 3 Super 120B	256K	*Not supported beyond 256K*	
Qwen 3.5 122B	256K	*Not supported beyond 256K*	

Table 9: List pricing in USD per million tokens at long-context lengths, and the maximum context window each model supports. Pokee-Isaac 28B rates are *provisional and subject to confirmation at launch*.

Against the two baselines that can be bought at these lengths, Pokee-Isaac 28B is cheaper on both meters—below GPT-5.6 Luna by $0.25 on input and $0.80 on output, and below Gemini 3.5 Flash Lite by $0.15 and $1.50—while sustaining accuracy across a window an order of magnitude larger than either. The three models that cannot be priced here are not cheaper alternatives; they are unavailable at the context lengths this report characterizes.

The per-token comparison also understates the difference for the deployments this report targets. Isaac is licensed for operation inside the customer boundary—VPC, on-premises, or on-device—where cost is a fixed function of the hardware provisioned rather than a variable function of tokens consumed, and where no request crosses an external boundary at all. For workloads at the context lengths characterized in Section 2, that distinction typically dominates the per-token rate.

[2]Retrieved from each provider's public pricing page on 3 August 2026. Where a provider meters long context separately, we quote the long-context rate rather than the short-context headline rate: GPT-5.6 Luna is billed at 2× input and 1.5× output on requests exceeding 272K tokens, which raises it from $0.20/$1.20 to the rates below. Gemini 3.5 Flash Lite is billed flat regardless of prompt length. Free tiers are excluded throughout.

6 Portability

Portability is a first-class capability of Pokee-Isaac 28B: the model is adapted to run efficiently on heterogeneous accelerator hardware, not only datacenter GPUs. Isaac currently runs fully on-device on two accelerator families—Intel and Qualcomm—with adaptation to a third, AMD, in progress and scheduled for a subsequent release.

6.1 Intel

Isaac runs fully on-device on the Intel Arc Pro B70 discrete GPU, served through our own stack rather than a stock runtime. Two properties carry over from the datacenter profile of Section 5.1: usable context extends toward 10M tokens on B70-class hardware, and decode rate holds steady as context grows rather than degrading with the resident window—so the long-context behavior characterized in Section 2 is available on the existing on-device serving path, with no dedicated accelerator.

Isaac has also been adapted to Intel's Core Ultra Series 3 (Panther Lake) client SoCs and validated on that silicon, with both prefill and decode executing entirely on the SoC—no discrete GPU and no dedicated accelerator present in the configuration.

As an indicator of on-device quantized model quality, Isaac scores 0.9567 on completed tasks on the Pinchbench 116-task SuperClaw suite (Table 10), while maintaining 98% L1 tool-delegation reliability. It leads across generative, meeting, writing, and memory tasks.

Model	Size	Serving	Score
Pokee-Isaac 28B	28B dense	Local, TP-2	**0.9567**
GLM-5	744B MoE	Cloud-hosted	0.929
Qwen3-Coder-Next	80B MoE	Local, TP-4	0.866

Table 10: Pinchbench 116-task SuperClaw suite, score on completed tasks. Isaac runs locally at TP-2; the Qwen3-Coder-Next baseline carries roughly triple the parameters at TP-4, and GLM-5 is a larger cloud-hosted model.

6.2 Qualcomm

Isaac has been adapted to current-generation NPU-class mobile silicon and validated on-device on the Snapdragon X2 Elite, with prefill and decode both executing on-device and no discrete GPU present in the configuration.

6.3 AMD (forthcoming)

Adaptation to AMD accelerator hardware is in progress and will be reported in a subsequent revision of this report, extending the portability profile to a third major accelerator family.

Note on per-chip performance. Detailed per-chip throughput measurements have been collected for each supported platform and will be added to a subsequent revision of this report once the respective hardware vendors approve their release.

7 Related Work

In this section, we review existing works on long-context, long-horizon agents. We will also discuss the efficiency of these agents.

Planning and Long-Horizon Execution. Long-horizon task execution requires autonomous agents to decompose complex, open-ended goals into structured sequences of actions while dynamically adapting to execution feedback. The foundational ReAct framework (Yao et al., 2023b) pioneered the interleaving of task-oriented reasoning traces with environment actions to mitigate error accumulation over multi-step tasks. To enable systematic lookahead planning beyond autoregressive token generation, Tree of Thoughts (Yao et al., 2023a) introduced search-based evaluation over discrete reasoning steps, which was later unified with tree-based search and environment feedback in Language Agent Tree Search (LATS) (Zhou et al., 2024a). In complex, open-world environments, architectures like Voyager (Wang et al., 2024) accomplish lifelong execution through automatic curricula and continually evolving skill libraries, whereas SWE-agent (Yang et al., 2024) optimizes custom agent-computer interfaces for repository-scale software tasks. To benchmark these long-horizon capabilities across multi-turn tool usage and execution, representative suites such as SWE-bench (Jimenez et al., 2024), GAIA (Mialon et al., 2024), WebArena (Zhou et al., 2024b), and AgentBench (Liu et al., 2024) have been established as standard evaluation targets.

Memory Hierarchies and Long-Context Management. A primary challenge in long-horizon autonomy is the context length bottleneck and the degradation of model attention over extended execution traces. To sustain persistent state and prevent performance loss, recent methods integrate explicit memory management mechanisms. Drawing analogies to computer operating systems, MemGPT (Packer et al., 2023) manages extended context windows by dynamically paging information between working memory and external archival storage using function calls. In multi-agent and social environments, Generative Agents (Park et al., 2023) utilizes a long-term memory stream coupled with automated high-level reflection processes to drive coherent long-term behaviors over extended simulations. To enable error correction across multiple trials without updating underlying parameters, Reflexion (Shinn et al., 2023) leverages episodic memory buffers and verbal reinforcement learning. Furthermore, to address extremely long contexts, LongAgent (Zhao et al., 2024) delegates context processing across multi-agent hierarchies to maintain performance during extended interactions.

To date, frontier (proprietary) models such as GPT-5.6 Sol (OpenAI, 2026), Claude Opus 5 (Anthropic, 2026), and Kimi K3 (Kimi Team, 2026) have about 1M context window.

Execution Overhead and Token Inefficiency in Agent Loops. While modern LLM-based agents demonstrate strong problem-solving capabilities, their execution efficiency remains a primary bottleneck. Standard agent execution loops repeatedly transmit entire interaction histories, system prompts, and tool specifications on every step, causing token consumption and prefill latency to scale quadratically with trajectory length. As recent analyses on token economics reveal, iterative reasoning, verbose tool observations, and multi-turn trial-and-error consume tokens at prohibitively high rates, severely constraining real-world deployment due to high API costs and latency (OPPO AI Agent Team, 2025). Multi-agent and search-based frameworks worsen this issue by multiplying token usage across sub-agents or search branches. To address this runtime inefficiency, supervisory mechanisms such as SupervisorAgent (Lin et al., 2026) attempt to prune redundant observations and intervene only during high-risk steps, highlighting that standard agents spend a substantial fraction of their token budget on redundant or uninformative context.

Context Compression and Resource Optimization. To reduce memory footprints and infer-

ence overhead without sacrificing task success, significant effort has been directed toward prompt and context compression. Early compression frameworks like LLMLingua (Jiang et al., 2023) and LongLLMLingua (Jiang et al., 2024) leverage small language models to remove redundant tokens and reorder context, alleviating the "lost-in-the-middle" problem while accelerating inference. However, general prompt compression methods often fail to preserve the dynamic state information necessary for interactive multi-step tasks. More recently, agent-centric compression techniques such as ACON (Kang et al., 2026) optimize guidelines in natural language to compress both environmental observations and interaction histories specifically for long-horizon environments. Despite these advancements, achieving an optimal trade-off between task completion accuracy, context compactness, and low operational latency remains an active challenge for long-horizon agents.

8 Conclusion and Future Work

Long-context and long-horizon agentic tasks have been assumed to require a model too large to deploy locally. Pokee-Isaac 28B shows otherwise. At 28B parameters — servable on a single GPU, runnable on a client workstation — Isaac holds retrieval fidelity across windows up to 10M tokens and matches or exceeds the strongest cost-optimized cloud systems on function calling, multi-turn interactive execution, tool orchestration, and shell work. That is the whole of the claim: the size threshold for delivering this class of capability is lower than has been assumed, and low enough to fall inside the range that can be deployed locally.

What this changes is not a ranking but a set of options. Until now, an organization that needed long-context agentic capability had one path: send the data out, and accept a third party's operational terms and a bill that grows with context. For regulated industries, public-sector institutions, and on-device applications, that path is frequently no path at all — not because of the price, but because the data is not permitted to leave the boundary. When a model with this capability can run inside that boundary, data sovereignty no longer has to be traded against capability, and the cost of a long context stops being a variable charge that scales with the work and becomes a fixed property of hardware already provisioned. That is what we built Isaac to do.

A further consequence is an engineering one. A substantial share of the complexity in today's agent systems — memory hierarchies, context compression, observation pruning, sharding context across sub-agents (Section 7) — exists to ration a scarce resource. These techniques work, but their necessity is defined by the constraint. When sufficient usable context is available inside the customer's boundary at the cost profile reported here, part of that machinery becomes optional rather than required: a trajectory can be kept as it is, rather than compressed, summarized, or split, and the information lost at each of those operations is not lost.

Three limitations bound the current release. Each is an explicit target for the next version rather than a standing design choice.

Multi-modality is not supported. Isaac accepts text only. Image, audio, and video inputs are out of scope, which also removes the model from the vision-dependent portions of several agentic benchmarks — we evaluate 86 of Terminal-Bench's 89 tasks and the text track of τ^3 for this reason. Multi-modal input is the single largest capability gap we intend to close.

Coding capability is not yet optimized. Coding was not a training priority for our first ever Pokee-Isaac model, and this report does not evaluate it directly — the suite contains no code-authoring benchmark. Terminal-Bench 2.1 is the nearest proxy available here, but it measures agentic execution in a shell rather than pure code generation, and Isaac's placement there should

be read as neither evidence of coding strength nor evidence of its absence. We state this limitation on the basis of what we chose to train, not on the basis of a measurement. Coding is the most commercially significant of the domains we did not prioritize; closing the gap, and evaluating it directly so the closure is visible, is a priority for the next version.

Hardware adaptation is partial. AMD adaptation is still in progress and many silicons are not yet supported. Broadening the adaptation surface and replacing every projection with a measurement is ongoing work, and will be reported in a subsequent revision.

References

Anthropic (2026). Introducing Claude Opus 5. *https://www.anthropic.com/news/claude-opus-5*.

Bandi, C. et al. (2026). MCP-Atlas: A large-scale benchmark for tool-use competency with real MCP servers. *arXiv:2602.00933 [cs.AI]*.

Chen, Z., Liu, X., Tong, H., Guo, C., Nie, Y., Zhang, J., Kang, M., Xu, C., Liu, Q., Liu, X., et al. (2026). DecodingTrust-Agent platform (DTap): A controllable and interactive red-teaming platform for AI agents. *arXiv:2605.04808 [cs.AI]*.

Hsieh, C.-P., Sun, S., Kriman, S., Acharya, S., Rekesh, D., Jia, F., Zhang, Y., and Ginsburg, B. (2024). RULER: What's the real context size of your long-context language models? *arXiv:2404.06654 [cs.CL]*.

Jiang, H., Wu, Q., Lin, C.-Y., Yang, Y., and Qiu, L. (2023). LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 13358–13376.

Jiang, H., Wu, Q., Luo, X., Li, D., Lin, C.-Y., Yang, Y., and Qiu, L. (2024). LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1658–1677.

Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., and Narasimhan, K. (2024). SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations (ICLR)*.

Kang, M., Chen, W.-N., Han, D., Inan, H., Wutschitz, L., Chen, Y., Sim, R., and Rajmohan, S. (2026). ACON: Optimizing context compression for long-horizon LLM agents. In *International Conference on Machine Learning (ICML)*.

Kimi Team (2026). Kimi K3: Open frontier intelligence. *arXiv:2607.24653 [cs.CL]*.

Lin, F., Chen, S., Fang, R., Wang, H., and Lin, T. (2026). Stop wasting your tokens: Towards efficient runtime multi-agent systems. In *International Conference on Learning Representations (ICLR)*.

Liu, X., Yu, H., Zhang, H., Xu, Y., Lei, X., Lai, H., Gu, Y., Ding, H., Men, K., Yang, K., et al. (2024). AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations (ICLR)*.

Merrill, M. A., Shaw, A. G., Carlini, N., Li, B., Raj, H., Bercovich, I., Dimakis, A., Konwinski, A., Schmidt, L., et al. (2026). Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *arXiv:2601.11868 [cs.SE]*.

Mialon, G., Fourrier, C., Swift, C., Wolf, T., LeCun, Y., and Scialom, T. (2024). GAIA: a benchmark for general ai assistants. In *International Conference on Learning Representations (ICLR)*.

OpenAI (2026). Gpt-5.6 Sol Model. *https://developers.openai.com/api/docs/models/gpt-5.6-sol*.

OPPO AI Agent Team (2025). Efficient agents: Building effective agents while reducing cost. *arXiv:2508.02694 [cs.AI]*.

Packer, C., Wooders, S., Lin, K., Fang, V., Patil, S. G., Stoica, I., and Gonzalez, J. E. (2023). MemGPT: Towards llms as operating systems.

Park, J. S., O'Brien, J., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. (2023). Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST)*, pages 1–22.

Patil, S. G., Mao, H., Yan, F., Ji, C. C.-J., Suresh, V., Stoica, I., and Gonzalez, J. E. (2025). The Berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *International Conference on Machine Learning (ICML)*.

Qwen Team (2026). Qwen3.6-27B: Flagship-level coding in a 27B dense model.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. (2023). Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Vodrahalli, K. et al. (2024). Michelangelo: Long context evaluations beyond haystacks via latent structure queries. *arXiv:2409.12640 [cs.CL]*.

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. (2024). Voyager: An open-ended embodied agent with large language models. In *Transactions on Machine Learning Research*.

Yang, J., Jimenez, C. E., Wettig, A., Lieret, K., Yao, S., Narasimhan, K., and Press, O. (2024). SWE-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems 37 (NeurIPS)*.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. (2023a). Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. (2023b). React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Zhao, J., Zu, C., Hao, X., Lu, Y., He, W., Ding, Y., Gui, T., Zhang, Q., and Huang, X. (2024). LongAgent: Achieving question answering for 128k-token-long documents through multi-agent collaboration. *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Zhou, A., Yan, K., Shlapentokh-Rothman, M., Wang, H., and Wang, Y.-X. (2024a). Language agent tree search unifies reasoning acting and planning in language models. In *International Conference on Machine Learning (ICML)*.

Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Ou, T., Bisk, Y., Fried, D., Alon, U., and Neubig, G. (2024b). WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations (ICLR)*.