

# Kimi K2.5: Visual Agentic Intelligence

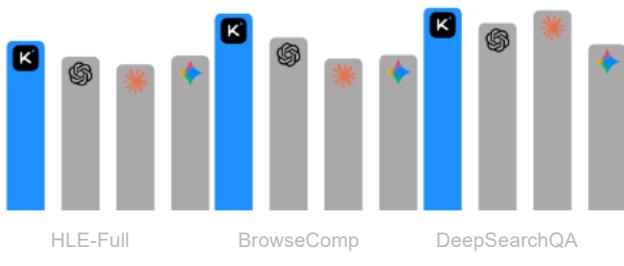
Today, we are introducing Kimi K2.5, the most powerful open-source model to date.

Kimi K2.5 builds on Kimi K2 with continued pretraining over approximately 15T mixed visual and text tokens. Built as a native multimodal model, K2.5 delivers state-of-the-art **coding and vision** capabilities and a self-directed **agent swarm** paradigm.

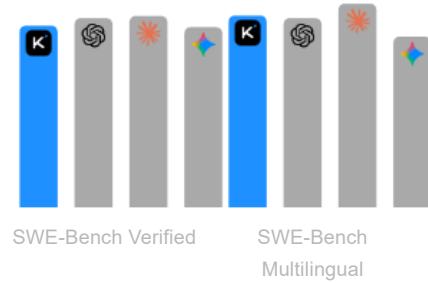
For complex tasks, Kimi K2.5 can self-direct an **agent swarm** with up to 100 sub-agents, executing parallel workflows across up to **1,500 tool calls**. Compared with a single-agent setup, this reduces execution time by up to **4.5x**. The agent swarm is automatically created and orchestrated by Kimi K2.5 without any predefined subagents or workflow.

Kimi K2.5 is available via [Kimi.com](#), the [Kimi App](#), the [API](#), and [Kimi Code](#). Kimi.com & Kimi App now supports 4 modes: K2.5 Instant, K2.5 Thinking, K2.5 Agent, and K2.5 Agent Swarm (Beta). Agent Swarm is currently in beta on Kimi.com, with free credits available for high-tier paid users.

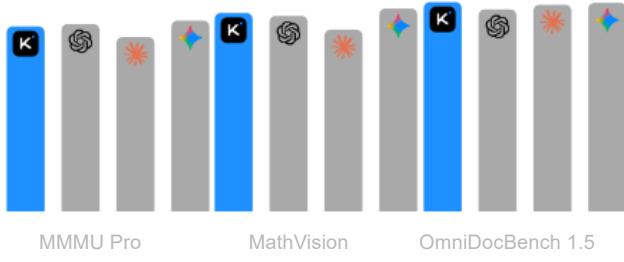
**Agents**



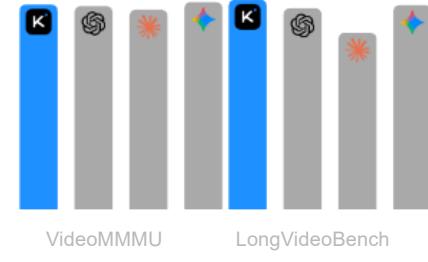
**Coding**

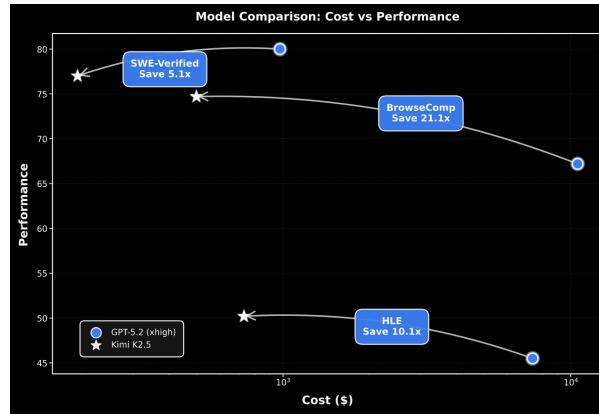


**Image**



**Video**





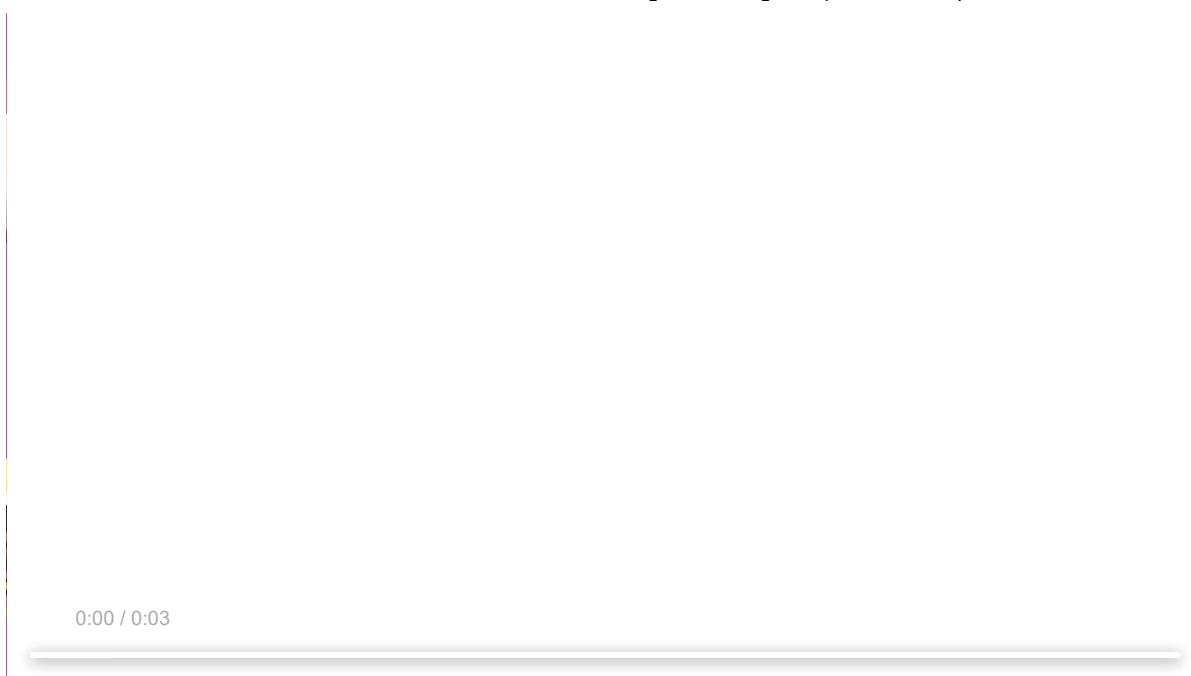
Across three agentic benchmarks—HLE, BrowseComp, and SWE-Verified—Kimi K2.5 delivers strong performance at a fraction of the cost.

## 1. Coding with Vision

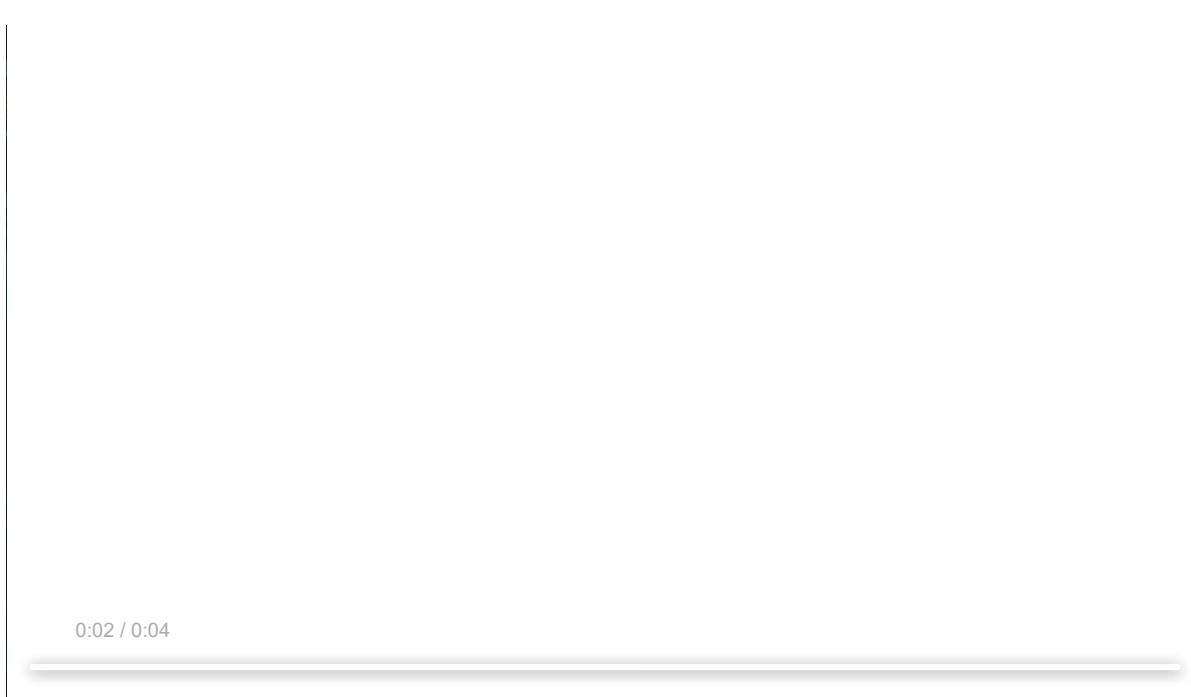
Kimi K2.5 is the strongest open-source model to date for coding, with particularly strong capabilities in front-end development.

K2.5 can turn simple conversations into complete front-end interfaces, implementing **interactive layouts** and **rich animations such as scroll-triggered effects**. Below are examples generated by K2.5 from a single prompt with image-gen tool:

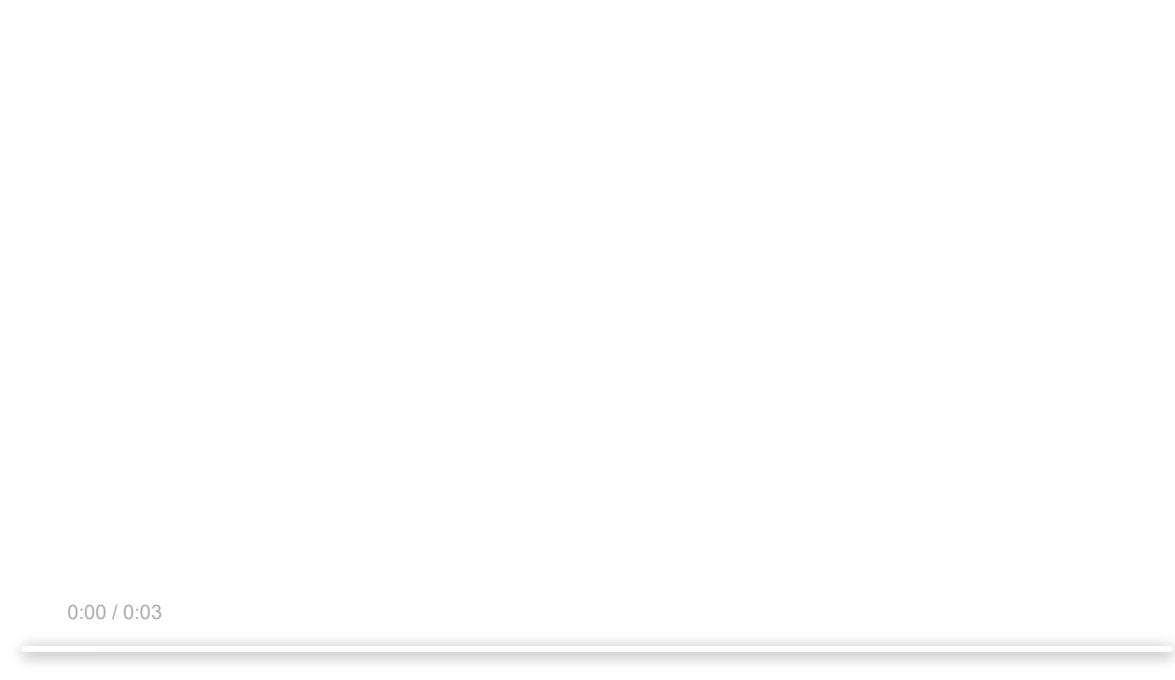
0:00 / 0:09



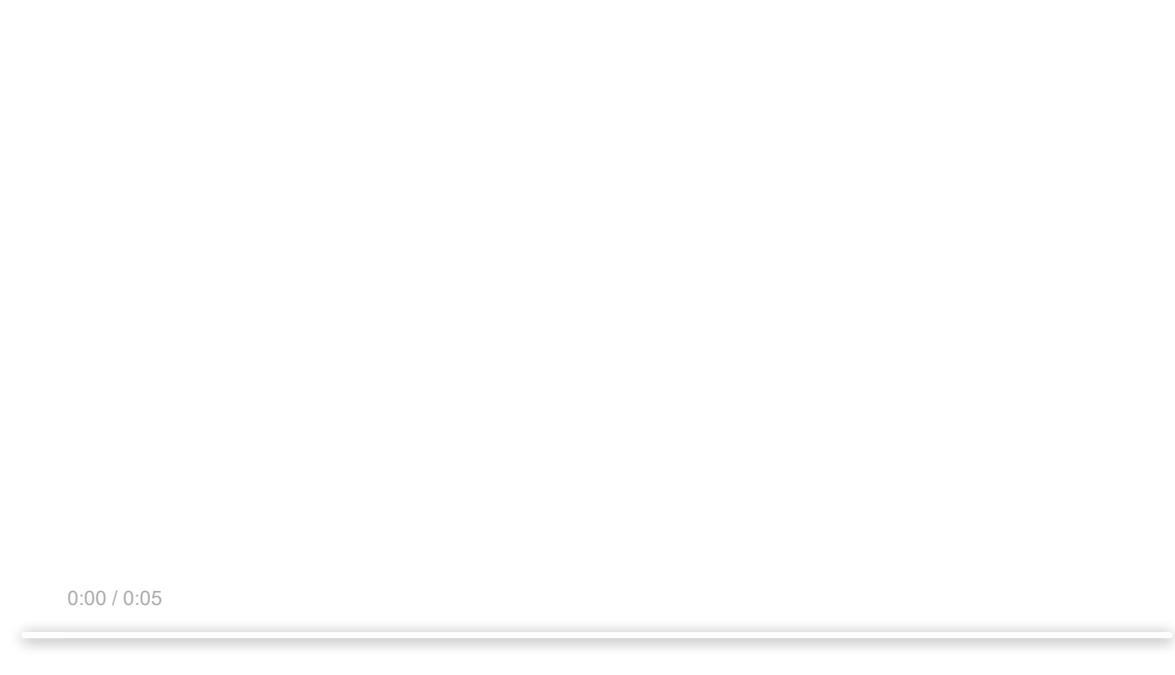
0:00 / 0:03



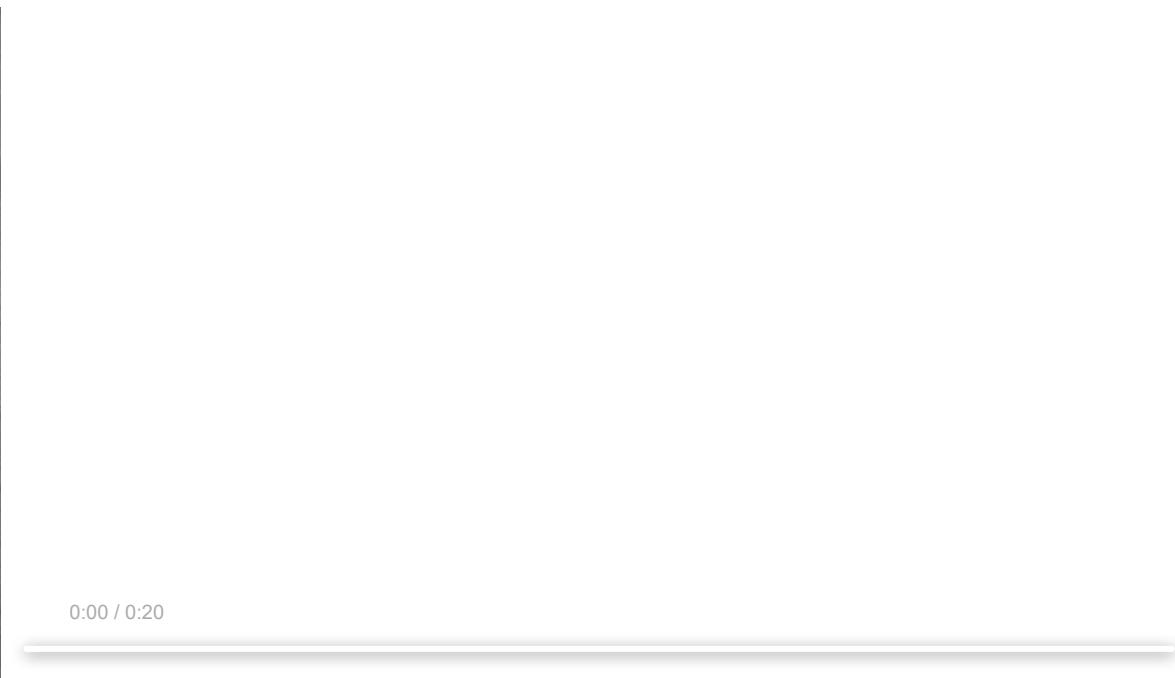
0:02 / 0:04



0:00 / 0:03



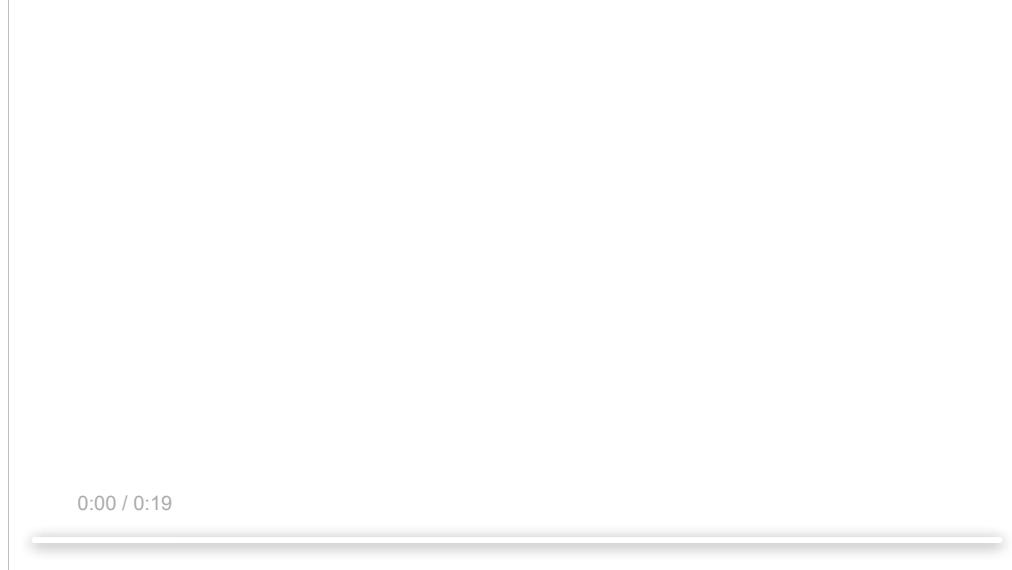
0:00 / 0:05

A video player interface with a black background. At the top left, it shows "0:00 / 0:20". Below this is a horizontal progress bar consisting of a thin black line with a small grey segment to its left.

0:00 / 0:20

Beyond text prompts, K2.5 excels at **coding with vision**. By reasoning over images and video, K2.5 improves image/video-to-code generation and visual debugging, lowering the barrier for users to express intent visually.

Here is an example of K2.5 reconstructing a website from video:

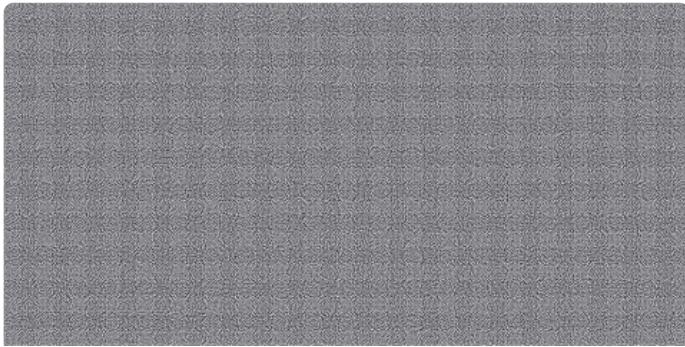
A video player interface with a black background. At the top left, it shows "0:00 / 0:19". Below this is a horizontal progress bar consisting of a thin black line with a small grey segment to its left.

0:00 / 0:19

This capability stems from massive-scale vision-text joint pre-training. At scale, the trade-off between vision and text capabilities disappears — they improve in unison.

Below is an example of K2.5 reasoning over a puzzle and marking the shortest path using code:

Chat



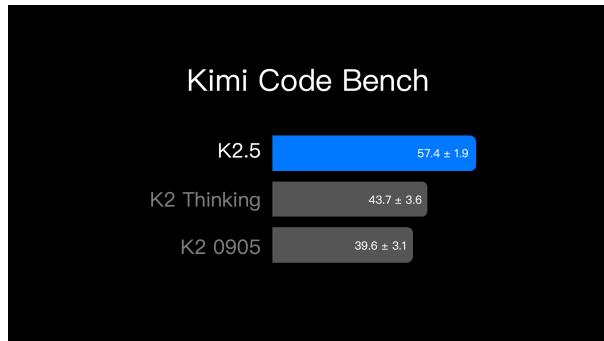
Find the shortest path from the top-left corner (green dot) to the bottom-right corner (red dot), where black represents the road.



Find Shortest Path in Image Using BFS or A\*

I'll help you find the shortest path through this maze. Let me analyze the image and implement a pathfinding algorithm.

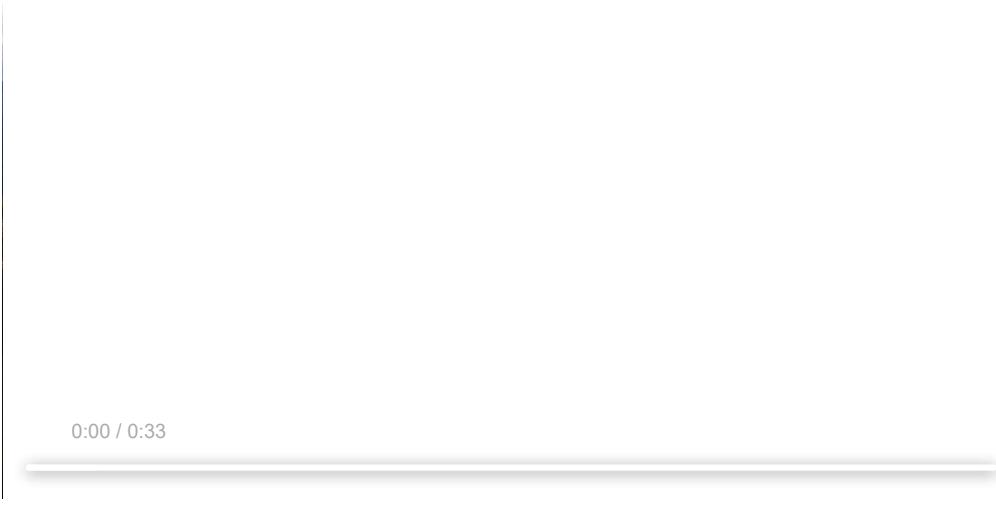
**K2.5 excels in real-world software engineering tasks.** We evaluate it using **Kimi Code Bench**, our internal coding benchmark covering diverse end-to-end tasks — from building to debugging, refactoring, testing, and more. On this benchmark, **K2.5 shows competitive performance across task types.**



To try out K2.5's agentic coding capabilities, [K2.5 Agent](#) offers a set of preconfigured tools for immediate, hands-on experiences. For software engineering use cases, we recommend pairing Kimi K2.5 with our new coding product, [Kimi Code](#).

[Kimi Code](#) works in your terminal and can be integrated with various IDEs including VSCode, Cursor, Zed, etc. Kimi Code is open-sourced and supports images and videos as inputs. It also automatically discovers and migrates existing skills and MCPs into your working environment in Kimi Code.

Here's an example using [Kimi Code](#) to translate the aesthetic of Matisse's *La Danse* into the Kimi App. This demo highlights a breakthrough in **autonomous visual debugging**. Using visual inputs and documentation lookup, K2.5 visually inspects its own output and iterates on it autonomously. It creates an art-inspired webpage created end to end:

A video player interface with a progress bar at the bottom. The text "0:00 / 0:33" is displayed in the center of the bar.

0:00 / 0:33

## 2. Agent Swarm

**Scaling Out, Not Just Up.** We release [K2.5 Agent Swarm](#) as a research preview, marking a shift from single-agent scaling to self-directed, coordinated swarm-like execution.

Trained with Parallel-Agent Reinforcement Learning (PARL), K2.5 learns to self-direct an **agent swarm** of up to **100 sub-agents**, executing parallel workflows across **up to 1,500 coordinated steps**, without predefined roles or hand-crafted workflows.

PARL uses a **trainable orchestrator agent** to decompose tasks into parallelizable subtasks, each executed by dynamically instantiated, **frozen subagents**. Running these subtasks concurrently significantly reduces end-to-end latency compared to sequential agent execution.

Training a reliable parallel orchestrator is challenging due to delayed, sparse, and non-stationary feedback from independently running subagents. A common failure mode is **serial collapse**, where the orchestrator defaults to single-agent execution despite having parallel capacity. To address this, PARL employs *staged reward shaping* that encourages parallelism early in training and gradually shifts focus toward task success.

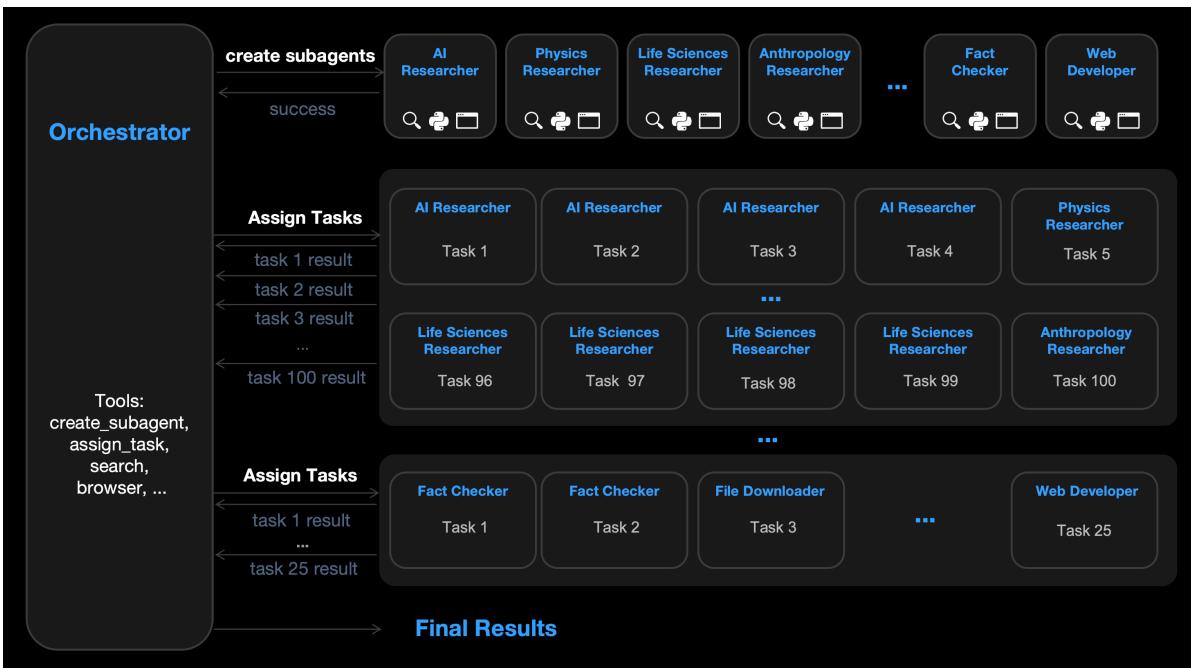
We define the reward as

where  $\alpha$  anneals from 0 to 1 over training. Early on, the auxiliary reward incentivizes subagent instantiation and concurrent execution, promoting exploration of the parallel scheduling space. As training progresses, optimization shifts toward end-to-end task quality, preventing degenerate solutions where parallelism is enabled in name only.

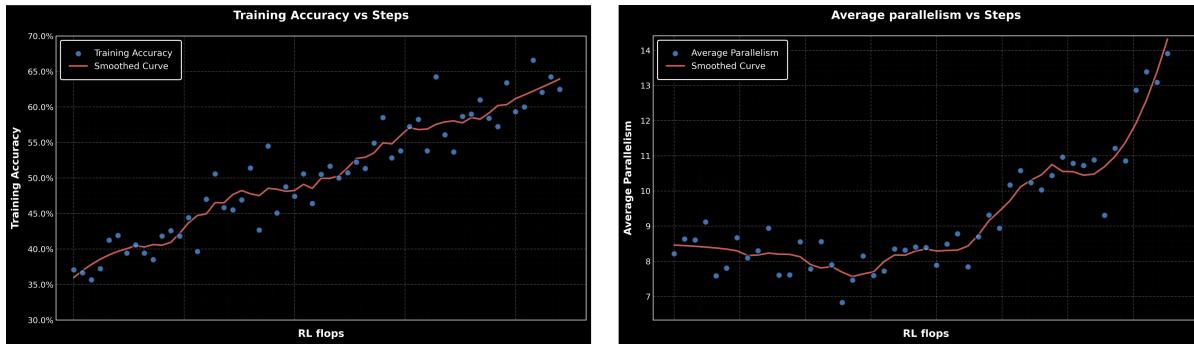
To further force parallel strategies to emerge, we introduce a computational bottleneck that makes sequential execution impractical. Instead of counting total steps, we evaluate performance using **Critical Steps**, a latency-oriented metric inspired by the critical path in parallel computation:

$\text{Critical Steps} = \max_{\text{stage}} (\text{slowest subagent})$

captures orchestration overhead, while  $\text{slowest subagent}$  reflects the slowest subagent at each stage. Under this metric, spawning more subtasks only helps if it shortens the critical path.

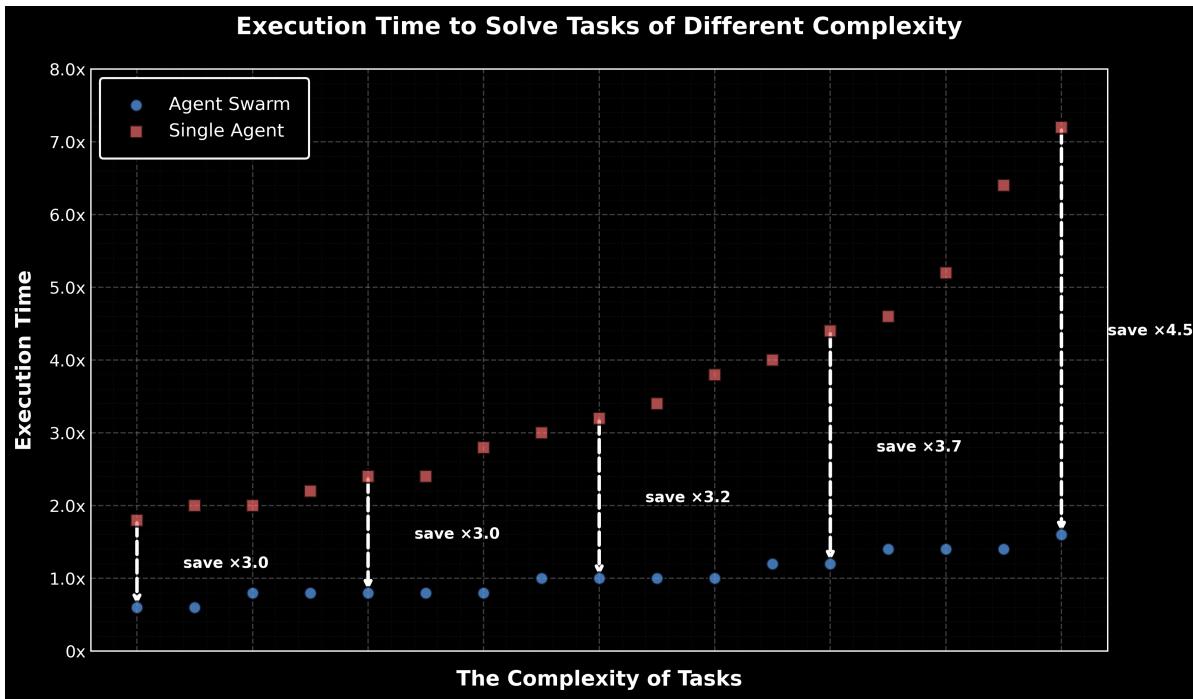
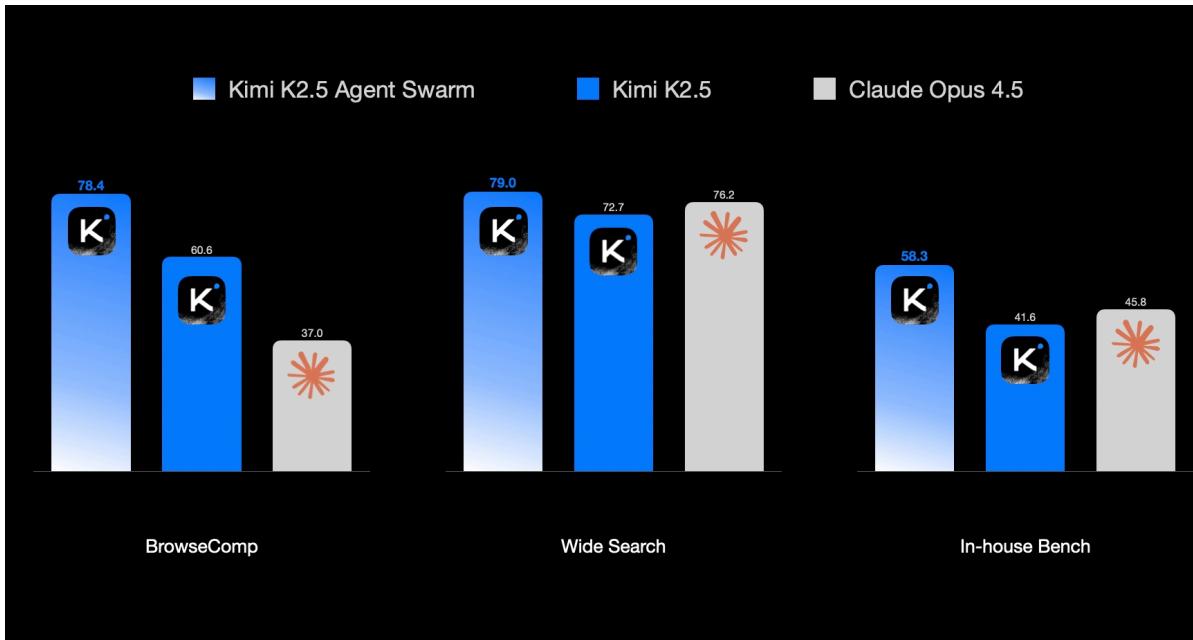


An agent swarm has an orchestrator that dynamically creates specialized subagents (e.g., AI Researcher, Physics Researcher, Fact Checker) and decomposes complex tasks into parallelizable subtasks for efficient distributed execution.



In our parallel-agent reinforcement learning environment, the reward increases smoothly as training progresses. At the same time, the level of parallelism during training also gradually increases.

**K2.5 Agent Swarm** improves performance on complex tasks through parallel, specialized execution. In our internal evaluations, it leads to an 80% reduction in end-to-end runtime while enabling more complex, long-horizon workloads, as shown below.



Agent Swarm reduces the minimum critical steps required to achieve target performance by 3×–4.5× compared to single-agent execution in wide search scenario, with savings scaling as targets rise—translating to up to 4.5× wall-clock time reduction via parallelization.

Here are representative **trajectories** demonstrating [K2.5 Agent Swarm](#) in action:

Parallel at Scale    Tool Use at Scale    Output at Scale    Download at Scale

Parallel at Scale  
100 Sub-agents Hunting for Creators

0:00 / 0:12

The task is to identify the top three YouTube creators across **100 niche domains**.

K2.5 Agent Swarm first researches and defines each domain, then autonomously **creates 100 sub-agents to conduct parallel searches**.

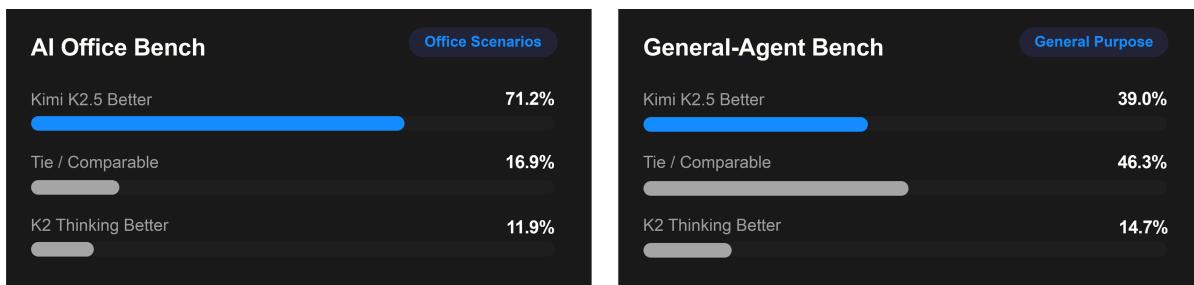
Each sub-agent identifies leading creators within its assigned niche, and the results—300 YouTuber profiles—are aggregated into a structured spreadsheet.

### 3. Office Productivity

Kimi K2.5 brings agentic intelligence into **real-world knowledge work**.

**K2.5 Agent** can handle high-density, large-scale office work end to end. It reasons over large, high-density inputs, coordinates multi-step tool use, and delivers expert-level outputs: documents, spreadsheets, PDFs, and slide decks—directly through conversation.

With a focus on real-world professional tasks, we design **two internal expert productivity benchmarks**. The **AI Office Benchmark** evaluates end-to-end Office output quality, while the **General Agent Benchmark** measures multi-step, production-grade workflows against human expert performance. Across both benchmarks, **K2.5 shows 59.3% and 24.3% improvements over K2 Thinking**, reflecting stronger end-to-end performance on real-world tasks.



Internal Expert Productivity Bench (AI Office)

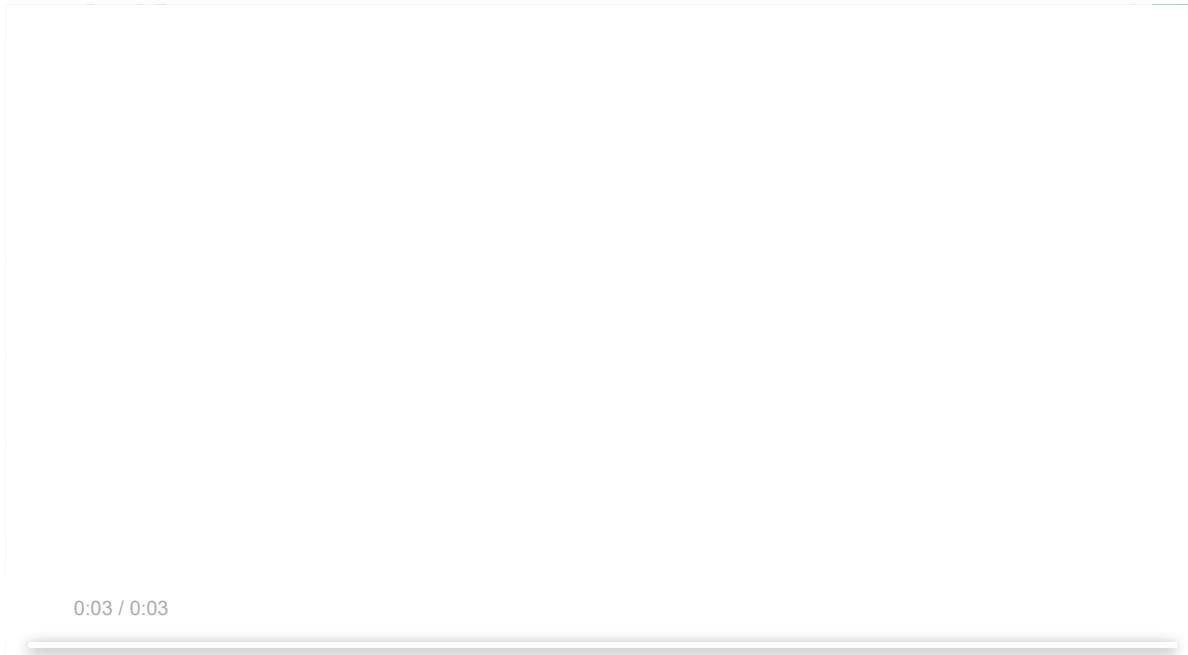
K2.5 agent supports advanced tasks such as **adding annotations in Word, constructing financial models with Pivot Tables, and writing LaTeX equations in PDFs**, while scaling to long-form outputs like **10,000-word papers or 100-page documents**.

Tasks that once took hours or days now complete in minutes. Here are some examples:

[Spreadsheet](#)    [Doc](#)    [PDF](#)    [Slide](#)

### Spreadsheet

100-Shot Storyboard in Spreadsheet with Images



## 4. Conclusion

Grounded in advances in coding with vision, agent swarms, and office productivity, Kimi K2.5 represents a meaningful step toward AGI for the open-source community, demonstrating strong capability on real-world tasks under real-world constraints. Looking ahead, we will push further into the frontier of agentic intelligence, redefining the boundaries of AI in knowledge work.

## Appendix

### Benchmark table

Benchmark	Kimi K2.5 (Thinking)	GPT-5.2 (xhigh)	Claude 4.5 Opus (Extend Thinking)	Gemini 3 Pro (High Thinking Level)	DeepSeek V3.2 (Thinking)	Qwen3-VL-235B-A22B (Thinking)
-----------	-------------------------	--------------------	--------------------------------------	---------------------------------------	-----------------------------	----------------------------------

Reasoning & Knowledge

Benchmark	Kimi K2.5 (Thinking)	GPT-5.2 (xhigh)	Claude 4.5 Opus (Extend Thinking)	Gemini 3 Pro (High Thinking Level)	DeepSeek V3.2 (Thinking)	Qwen3-VL-235B-A22B (Thinking)
HLE-Full	30.1	34.5	30.8	<b>37.5</b>	25.1*	—
HLE-Full w/ tools	<b>50.2</b>	45.5	43.2	45.8	40.8*	—
AIME 2025	96.1	<b>100.0</b>	92.8	95.0	93.1	—
HMMT 2025 (Feb)	95.4	<b>99.4</b>	92.9*	97.3*	92.5	—
IMO-AnswerBench	81.8	<b>86.3</b>	78.5*	83.1*	78.3	—
GPQA-Diamond	87.6	<b>92.4</b>	87.0	91.9	82.4	—
MMLU-Pro	87.1	86.7*	89.3*	<b>90.1</b>	85.0	—
Image & Video						
MMMU-Pro	78.5	79.5	74.0	<b>81.0</b>	—	69.3
CharXiv (RQ)	77.5	<b>82.1</b>	67.2*	81.4	—	66.1
MathVision	84.2	83.0	77.1*	<b>86.1</b>	—	74.6
MathVista (mini)	<b>90.1</b>	82.8*	80.2*	89.8*	—	85.8
ZeroBench	<b>9.0</b>	<b>9.0*</b>	3.0*	8.0*	—	4.0*
ZeroBench w/ tools	11.0	7.0*	9.0*	<b>12.0*</b>	—	3.0*
OCRBench	<b>92.3</b>	80.7*	86.5*	90.3*	—	87.5
OmniDocBench 1.5	<b>88.8</b>	85.7	87.7*	88.5	—	82.0*
InfoVQA (test)	<b>92.6</b>	84.0*	76.9*	57.2*	—	89.5
SimpleVQA	<b>71.2</b>	55.8*	69.7*	69.7*	—	56.8*
WorldVQA	46.3	28.0	36.8	<b>47.4</b>	—	23.5
VideoMMU	86.6	85.9	84.4*	<b>87.6</b>	—	80.0
MMVU	80.4	<b>80.8*</b>	77.3	77.5	—	71.1
MotionBench	<b>70.4</b>	64.8	60.3	70.3	—	—
VideoMME	87.4	86.0	—	<b>88.4*</b>	—	79.0
LongVideoBench	<b>79.8</b>	76.5	67.2	77.7*	—	65.6*
LVBenchmark	<b>75.9</b>	—	—	73.5*	—	63.6
Coding						
SWE-Bench Verified	76.8	80.0	<b>80.9</b>	76.2	73.1	—
SWE-Bench Pro	50.7	<b>55.6</b>	55.4*	—	—	—
SWE-Bench Multilingual	73.0	72.0	<b>77.5</b>	65.0	70.2	—
Terminal-Bench 2.0	50.8	54.0	<b>59.3</b>	54.2	46.4	—
PaperBench	63.5	63.7*	<b>72.9*</b>	—	47.1	—
CyberGym	41.3	—	<b>50.6</b>	39.9*	17.3*	—
SciCode	48.7	52.1	49.5	<b>56.1</b>	38.9	—

Benchmark	Kimi K2.5 (Thinking)	GPT-5.2 (xhigh)	Claude 4.5 Opus (Extend Thinking)	Gemini 3 Pro (High Thinking Level)	DeepSeek V3.2 (Thinking)	Qwen3-VL-235B-A22B (Thinking)
OJ Bench (cpp)	57.4	—	54.6*	<b>68.5*</b>	54.7*	—
LiveCodeBench (v6)	85.0	—	82.2*	<b>87.4*</b>	83.3	—
Long Context						
Longbench v2	61.0	54.5*	64.4*	<b>68.2*</b>	59.8*	—
AA-LCR	70.0	<b>72.3*</b>	71.3*	65.3*	64.3*	—
Agentic Search						
BrowseComp	<b>60.6</b>	—	37.0	37.8	51.4	—
BrowseComp (w/ctx mgm)	<b>74.9</b>	65.8	57.8	59.2	67.6	—
BrowseComp (Agent Swarm)	<b>78.4</b>	—	—	—	—	—
WideSearch (item-f1)	72.7	—	<b>76.2*</b>	57.0	32.5*	—
WideSearch (item-f1) (Agent Swarm)	<b>79.0</b>	—	—	—	—	—
DeepSearchQA	<b>77.1</b>	71.3*	76.1*	63.2*	60.9*	—
FinSearchCompT2&T3	<b>67.8</b>	—	66.2*	49.9	59.1*	—
Seal-0	<b>57.4</b>	45.0	47.7*	45.5*	49.5*	—

To reproduce official **Kimi-K2.5** benchmark results, we recommend using the [official API](#). For third-party providers, refer to **Kimi Vendor Verifier (KVV)** to choose high-accuracy services. Details: <https://kimi.com/blog/kimi-vendor-verifier.html>

## Footnotes

### 1. General Testing Details

- We report results for Kimi K2.5 and DeepSeek-V3.2 with thinking mode enabled, Claude Opus 4.5 with extended thinking mode, GPT-5.2 with xhigh reasoning effort, and Gemini 3 Pro with a high thinking level. For vision benchmarks, we additionally report results for Qwen3-VL-235B-A22B-Thinking.
- Unless otherwise specified, all Kimi K2.5 experiments were conducted with temperature = 1.0, top-p = 0.95, and a context length of 256k tokens.
- Benchmarks without publicly available scores were re-evaluated under the same conditions used for Kimi K2.5 and are marked with an asterisk (\*).
- We could not evaluate GPT-5.2 xhigh on all benchmarks due to service stability issues. For benchmarks that were not tested, we mark them as "-".

### 2. Text and Reasoning

- HLE, AIME 2025, HMMT 2025 (Feb), GPQA-Diamond and IMO-AnswerBench were evaluated with a maximum completion budget of 96k tokens.
- Results for AIME and HMMT are averaged over 32 runs (avg@32); GPQA-Diamond over 8 runs (avg@8).
- For HLE, we report scores on the full set (text & image). Kimi K2.5 scores 31.5 (text) and 21.3 (image) without tools, and 51.8 (text) and 39.8 (image) with tools. The DeepSeek-V3.2 score corresponds to its text-only subset (marked with †). Hugging Face access was blocked to prevent potential data leakage. HLE with tools uses simple context management: once the context exceeds a threshold, only the latest round of tool messages is retained.

### 3. Tool-Augmented / Agentic Search

- Kimi K2.5 was equipped with search, code-interpreter, and web-browsing tools for HLE with tools and all agentic search benchmarks.
- Except for BrowseComp (where K2.5 and DeepSeek-V3.2 used the discard-all strategy), no context management was applied, and tasks exceeding the supported context length were directly counted as failed.
- The test system prompts emphasize deep and proactive tool use, instructing models to reason carefully, leverage tools, and verify uncertain information. Full prompts will be provided in the technical report.
- Results for Seal-0 and WideSearch are averaged over four runs (avg@4).

### 4. Vision Benchmarks

- Max-tokens = 64k, averaged over three runs (avg@3).
- ZeroBench (w/ tools) uses max-tokens-per-step = 24k and max-steps = 30 for multi-step reasoning.
- MMMU-Pro follows the official protocol, preserving input order and prepending images.
- GPT-5.2-xhigh had ~10% failure rate (no output despite 3 retries), treated as incorrect; reported scores likely underestimate true performance.
- WorldVQA, a benchmark designed to evaluate atomic vision-centric world knowledge. Access WorldVQA at <https://github.com/MoonshotAI/WorldVQA>.
- OmniDocBench Score is computed as  $(1 - \text{normalized Levenshtein distance}) \times 100$ , where a higher score denotes superior accuracy.

### 5. Coding Tasks

- Terminal-Bench 2.0 scores were obtained with the default agent framework (Terminus-2) and the provided JSON parser. In our implementation, we evaluated Terminal-Bench 2.0 under non-thinking mode. This choice was made because our current context management strategy for the thinking mode is incompatible with Terminus-2.
- For the SWE-Bench series of evaluations (including verified, multilingual, and pro), we used an internally developed evaluation framework. This framework includes a minimal set of tools—bash tool, createfile tool, insert tool, view tool, strreplace tool, and submit tool—along with tailored system prompts designed for the tasks. The highest scores were achieved under non-thinking mode.
- The score of Claude Opus 4.5 on CyberGym is reported under the non-thinking setting.

- All reported scores of coding tasks are averaged over 5 independent runs.

## 6. Long-Context Benchmarks

- AA-LCR: scores averaged over three runs (avg@3).
- LongBench-V2: identical prompts and input contexts standardized to ~128k tokens.

## 7. Agent Swarm

- BrowseComp (Swarm Mode): main agent max 15 steps; sub-agents max 100 steps.
- WideSearch (Swarm Mode): main and sub-agents max 100 steps.