KIMI K2.5: VISUAL AGENTIC INTELLIGENCE

TECHNICAL REPORT OF KIMI K2.5

Kimi Team

ABSTRACT

We introduce Kimi K2.5, an open-source multimodal agentic model designed to advance general agentic intelligence. K2.5 emphasizes the joint optimization of text and vision so that two modalities enhance each other. This includes a series of techniques such as joint text-vision pre-training, zero-vision SFT, and joint text-vision reinforcement learning. Building on this multimodal foundation, K2.5 introduces Agent Swarm, a self-directed parallel agent orchestration framework that dynamically decomposes complex tasks into heterogeneous sub-problems and executes them concurrently. Extensive evaluations show that Kimi K2.5 achieves state-of-the-art results across various domains including coding, vision, reasoning, and agentic tasks. Agent Swarm also reduces latency by up to 4.5× over single-agent baselines. We release the post-trained Kimi K2.5 model checkpoint[1] to facilitate future research and real-world applications of agentic intelligence.

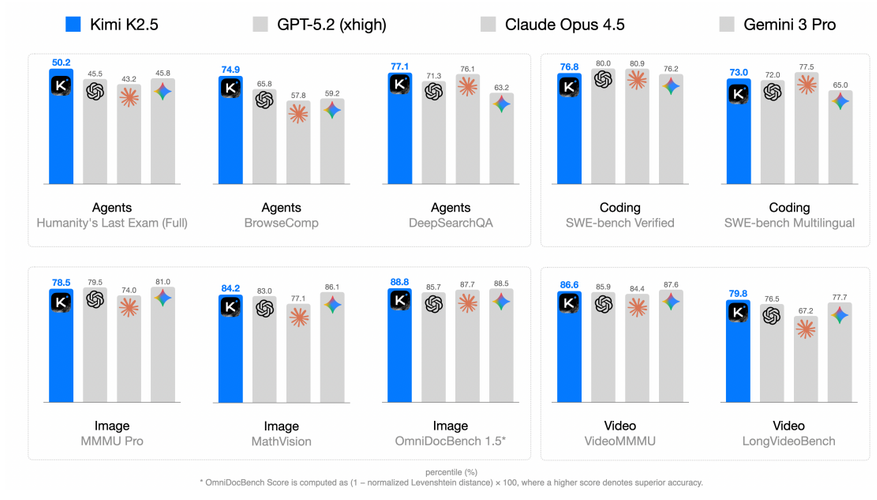


Figure 1: Kimi K2.5 main results.

1 Introduction

Large Language Models (LLMs) are rapidly evolving toward agentic intelligence. Recent advances, such as GPT-5.2 [41], Claude Opus 4.5 [6], Gemini 3 Pro [20], and Kimi K2-Thinking [1], demonstrate substantial progress in agentic capabilities, particularly in tool calling and reasoning. These models increasingly exhibit the ability to decompose complex problems into multi-step plans and to execute long sequences of interleaved reasoning and actions.

[1] https://huggingface.co/moonshotai/Kimi-K2.5

KIMI K2.5：视觉自主智能

KIMI K2.5技术报告

Kimi团队

摘要

我们介绍了Kimi K2.5，这是一个开源的多模态自主模型，旨在推进通用智能体智能。K2.5强调文本和视觉的联合优化，使两种模态相互增强。这包括一系列技术，如联合文本-视觉预训练、无视觉SFT以及联合文本-视觉强化学习。在此多模态基础上，K2.5引入了Agent Swarm，这是一个自导向并行代理编排框架，能够动态地将复杂任务分解为异构子问题并并行执行。广泛的评估表明，Kimi K2.5在编码、视觉、推理和自主任务等多个领域均取得了最先进的结果。Agent Swarm通过高达4.5× 超过单代理基线[1]的效率降低了延迟。我们发布了预训练的KIMI K2.5模型检查点，以促进自主智能的未来研究和实际应用。

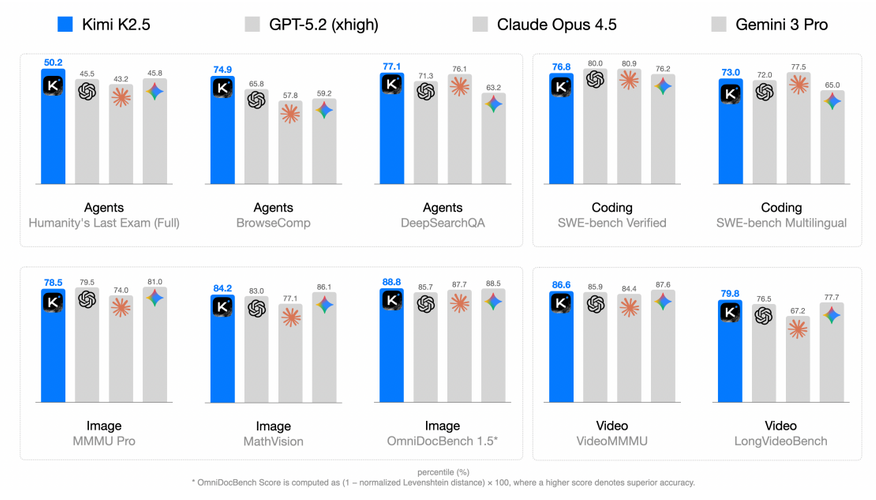


图1：Kimi K2.5主要结果。

1 引言

大语言模型（LLMs）正迅速向自主智能演进。近期进展，如GPT-5.2 [41], Claude Opus 4.5 [6], Gemini 3 Pro [20], 和Kimi K2-Thinking [1], ，在自主能力方面展现了显著进步，特别是在工具调用和推理方面。这些模型越来越多地展现出将复杂问题分解为多步计划，并执行长序列交替推理和行动的能力。

[1] https://huggingface.co/moonshotai/Kimi-K2.5

In this report, we introduce the training methods and evaluation results of Kimi K2.5. Concretely, we improve the training of K2.5 over previous models in the following two key aspects.

Joint Optimization of Text and Vision. A key insight from the practice of K2.5 is that joint optimization of text and vision enhances both modalities and avoids the conflict. Specifically, we devise a set of techniques for this purpose. During pre-training, in contrast to conventional approaches that add visual tokens to a text backbone at a late stage [8, 21], we find early vision fusion with lower ratios tends to yield better results given the fixed total vision-text tokens. Therefore, K2.5 mixes text and vision tokens with a constant ratio throughout the entire training process.

Architecturally, Kimi K2.5 employs MoonViT-3D, a native-resolution vision encoder incorporating the NaViT packing strategy [15], enabling variable-resolution image inputs. For video understanding, we introduce a lightweight 3D ViT compression mechanism: consecutive frames are grouped in fours, processed through the shared MoonViT encoder, and temporally averaged at the patch level. This design allows Kimi K2.5 to process videos up to 4 $\times$ longer within the same context window while maintaining complete weight sharing between image and video encoders.

During post-training, we introduce zero-vision SFT—text-only SFT alone activates visual reasoning and tool use. We find that adding human-designed visual trajectories at this stage hurts generalization. In contrast, text-only SFT performs better—likely because joint pretraining already establishes strong vision-text alignment, enabling capabilities to generalize naturally across modalities. We then apply joint RL on both text and vision tasks. Crucially, we find visual RL enhances textual performance rather than degrading it, with improvements on MMLU-Pro and GPQA-Diamond. This bidirectional enhancement—text bootstraps vision, vision refines text—represents superior cross-modal alignment in joint training.

Agent Swarm: Parallel Agent Orchestration. Most existing agentic models rely on sequential execution of tool calls. Even systems capable of hundreds of reasoning steps, such as Kimi K2-Thinking [1], suffer from linear scaling of inference time, leading to unacceptable latency and limiting task complexity. As agentic workloads grow in scope and heterogeneity—e.g., building a complex project that involves massive-scale research, design, and development—the sequential paradigm becomes increasingly inefficient.

To overcome the latency and scalability limits of sequential agent execution, Kimi K2.5 introduces *Agent Swarm*, a dynamic framework for parallel agent orchestration. We propose a Parallel-Agent Reinforcement Learning (PARL) paradigm that departs from traditional agentic RL [2]. In addition to optimizing tool execution via verifiable rewards, the model is equipped with interfaces for sub-agent creation and task delegation. During training, sub-agents are frozen and their execution trajectories are excluded from the optimization objective; only the orchestrator is updated via reinforcement learning. This decoupling circumvents two challenges of end-to-end co-optimization: credit assignment ambiguity and training instability. Agent Swarm enables complex tasks to be decomposed into heterogeneous sub-problems executed concurrently by domain-specialized agents, transforming task complexity from linear scaling to parallel processing. In wide-search scenarios, Agent Swarm reduces inference latency by up to 4.5$\times$ while improving item-level F1 from 72.8% to 79.0% compared to single-agent baselines.

Kimi K2.5 represents a unified architecture for general-purpose agentic intelligence, integrating vision and language, thinking and instant modes, chats and agents. It achieves strong performance across a broad range of agentic and frontier benchmarks, including state-of-the-art results in visual-to-code generation (image/video-to-code) and real-world software engineering in our internal evaluations, while scaling both the diversity of specialized agents and the degree of parallelism. To accelerate community progress toward General Agentic Intelligence, we open-source our post-trained checkpoints of Kimi K2.5, enabling researchers and developers to explore, refine, and deploy scalable agentic intelligence.

2 Joint Optimization of Text and Vision

Kimi K2.5 is a native multimodal model built upon Kimi K2 through large-scale joint pre-training on approximately 15 trillion mixed visual and text tokens. Unlike vision-adapted models that compromise either linguistic or visual capabilities, our joint pre-training paradigm enhances both modalities simultaneously. This section describes the multimodal joint optimization methodology that extends Kimi K2 to Kimi K2.5.

2.1 Native Multimodal Pre-Training

A key design question for multimodal pre-training is: Given a fixed vision-text token budget, what is the optimal vision-text joint-training strategy. Conventional wisdom [8, 21] suggests introducing vision tokens predominantly in the later stages of LLM training at high ratios (e.g., 50% or higher) should accelerate multimodal capability acquisition, treating multimodal capability as a post-hoc add-on to linguistic competence.

在本报告中，我们介绍了Kimi K2.5的训练方法和评估结果。具体而言，我们在以下两个关键方面改进了K2.5的训练。

文本和视觉的联合优化。 K2.5实践中的一个关键洞察是，文本和视觉的联合优化能够提升两种模态的表现并避免冲突。具体来说，我们为此设计了一套技术。在预训练阶段，与将视觉标记在晚期添加到文本主干的传统方法不同 [8,21], 我们发现早期视觉融合且比例较低时，在固定的总视觉-文本标记下往往能获得更好的结果。因此，K2.5在整个训练过程中始终以恒定比例混合文本和视觉标记。
在架构上，Kimi K2.5采用了MoonViT-3D，这是一种原生分辨率视觉编码器，集成了NaViT打包策略 [15], , 支持可变分辨率的图像输入。对于视频理解，我们引入了一种轻量级的3D ViT压缩机制：连续的帧被分为四组，通过共享的MoonViT编码器处理，并在块级别进行时间平均。这种设计使Kimi K2.5能够在相同的上下文窗口内处理长达 4 $\times$ 更长的视频，同时保持图像和视频编码器之间的完全权重共享。
在训练后，我们引入了无视觉SFT——仅文本SFT就能激活视觉推理和工具使用。我们发现，在这个阶段添加人工设计的视觉轨迹会损害泛化能力。相比之下，仅文本SFT表现更好——可能是因为联合预训练已经建立了强大的视觉-文本对齐，从而使得能力能够自然地泛化到不同模态。然后，我们在文本和视觉任务上都应用联合强化学习。关键在于，我们发现视觉强化学习增强了文本性能而不是降低它，并在MMLU-Pro和GPQA- Diamond上取得了改进。这种双向增强——文本引导视觉，视觉优化文本——代表了联合训练中更优越的跨模态对齐。

自主代理式：并行代理编排。 大多数现有的自主代理模型依赖于工具调用的顺序执行。即使是能够进行数百步推理的系统，如Kimi K2-Thinking [1], , 也受到推理时间线性扩展的影响，导致不可接受的延迟并限制任务复杂度。随着自主代理工作负载在范围和异质性上的增长——例如，构建一个涉及大规模研究、设计和开发的复杂项目——顺序范式变得越来越低效。
为克服顺序代理执行的延迟和可扩展性限制，Kimi K2.5 引入了 代理群，这是一个用于并行代理编排的动态框架。我们提出了一种并行代理强化学习（PARL）范式，该范式与传统代理式强化学习 [2] 不同。除了通过可验证的奖励优化工具执行外，该模型还配备了用于子代理创建和任务委派的接口。在训练过程中，子代理被冻结，其执行轨迹被排除在优化目标之外；只有编排器通过强化学习进行更新。这种解耦规避了端到端协同优化的两个挑战：信用分配模糊性和训练不稳定性。代理群使复杂任务能够分解为异构子问题，由领域专业化的代理并发执行，将任务复杂度从线性扩展转变为并行处理。在广泛搜索场景中，与单代理基线相比，代理群将推理延迟降低高达 4.5$\times$ ，同时将项目级 F1 从 72.8% 提升至 79.0%。

Kimi K2.5 代表了一种面向通用智能体智能的统一架构，集成了视觉与语言、思考与即时模式、对话与代理。它在广泛的智能体和前沿基准测试中展现出强大性能，包括在内部分析中达到视觉到代码生成（图像/视频到代码）的顶尖结果和真实的软件工程应用，同时扩展了专业代理的多样性及并行程度。为加速社区向通用智能体智能的进展，我们开源了 Kimi K2.5 的预训练检查点，使研究人员和开发者能够探索、优化和部署可扩展的智能体智能。

2 文本与视觉联合优化

Kimi K2.5 是基于 Kimi K2 构建的原生多模态模型，通过在约 150 万亿混合视觉和文本标记上进行大规模联合预训练而成。不同于牺牲语言或视觉能力的视觉适配模型，我们的联合预训练范式实现了两种模态的同步增强。本节将介绍将 Kimi K2 扩展至 Kimi K2.5 的多模态联合优化方法。

2.1 本地多模态预训练

多模态预训练的一个关键设计问题是：在固定的视觉-文本标记预算下，最优的视觉-文本联合训练策略是什么。传统观点 [8, 21] 建议在LLM训练的晚期阶段以高比例（例如50%或更高）引入视觉标记，应该能加速多模态能力获取，将多模态能力视为语言能力的后期附加功能。

Table 1: Performance comparison across different vision-text joint-training strategies. Early fusion with a lower vision ratio yields better results given a fixed total vision-text token budget.

	Vision Injection Timing	Vision-Text Ratio	Vision Knowledge	Vision Reasoning	OCR	Text Knowledge	Text Reasoning	Code
Early	0%	10%:90%	**25.8**	**43.8**	**65.7**	**45.5**	58.5	**24.8**
Mid	50%	20%:80%	25.0	40.7	64.1	43.9	**58.6**	24.0
Late	80%	50%:50%	24.2	39.0	61.5	43.1	57.8	24.0

However, our experiments (as shown in Table 1 Figure 9) reveal a different story. We conducted ablation studies varying the vision ratio and vision injection timing while keeping the total vision and text token budgets fixed. To strictly meet the targets for different ratios, we pre-trained the model with text-only tokens for a specifically calculated number of tokens before introducing vision data. Surprisingly, we found that the vision ratio has minimal impact on final multimodal performance. In fact, **early fusion with a lower vision ratio yields better results given a fixed total vision-text token budget**. This motivates our native multimodal pre-training strategy: rather than aggressive vision-heavy training concentrated at the end, we adopt a moderate vision ratio integrated early in the training process, allowing the model to naturally develop balanced multimodal representations while benefiting from extended co-optimization of both modalities.

2.2 Zero-Vision SFT

Pretrained VLMs do not naturally perform vision-based tool-calling, which poses a cold-start problem for multimodal RL. Conventional approaches address this issue through manually annotated or prompt-engineered chain-of-thought (CoT) data [8], but such methods are limited in diversity, often restricting visual reasoning to simple diagrams and primitive tool manipulations (`crop`, `rotate`, `flip`).

An observation is that high-quality text SFT data are relatively abundant and diverse. We propose a novel approach, zero-vision SFT, that uses only text SFT data to activate the visual, agentic capabilities during post-training. In this approach, all image manipulations are proxied through programmatic operations in `IPython`, effectively serving as a generalization of traditional vision tool-use. This "zero-vision" activation enables diverse reasoning behaviors, including pixel-level operations such as object size estimation via binarization and counting, and generalizes to visually grounded tasks such as object localization, counting, and OCR.

Figure 2 illustrates the RL training curves, where the starting points are obtained from zero-vision SFT. The results show that zero-vision SFT is sufficient for activating vision capabilities while ensuring generalization across modalities. This phenomenon is likely due to the joint pretraining of text and vision data as described in Section 2.1. Compared to zero-vision SFT, our preliminary experiments show that text-vision SFT yields much worse performance on visual, agentic tasks, possibly because of the lack of high-quality vision data.

2.3 Joint Multimodal Reinforcement Learning (RL)

In this section, we describe the methodology implemented in K2.5 that enables effective multimodal RL, from outcome-based visual RL to emergent cross-modal transfer that enhances textual performance.

Outcome-Based Visual RL Following the zero-vision SFT, the model requires further refinement to reliably incorporate visual inputs into reasoning. Text-initiated activation alone exhibits notable failure modes: visual inputs are sometimes ignored, and images may not be attended to when necessary. We employ outcome-based RL on tasks that explicitly require visual comprehension for correct solutions. We categorize these tasks into three domains:

- **Visual grounding and counting:** Accurate localization and enumeration of objects within images;
- **Chart and document understanding:** Interpretation of structured visual information and text extraction;
- **Vision-critical STEM problems:** Mathematical and scientific questions filtered to require visual inputs.

Outcome-based RL on these tasks improves both basic visual capabilities and more complex agentic behaviors. Extracting these trajectories for rejection-sampling fine-tuning (RFT) enables a self-improving data pipeline, allowing subsequent joint RL stages to leverage richer multimodal reasoning traces.

表1：不同视觉-文本联合训练策略的性能比较。早期融合在固定的视觉-文本总token预算下，使用较低的视觉比例能获得更好的结果。

	视觉注入时间	视觉-文本比率	视觉知识	视觉推理	OCR	Text 知识	Text 推理	Code
早期	0%	10%:90%	**25.8**	**43.8**	**65.7**	**45.5**	58.5	**24.8**
Mid	50%	20%:80%	25.0	40.7	64.1	43.9	**58.6**	24.0
Late	80%	50%:50%	24.2	39.0	61.5	43.1	57.8	24.0

然而，我们的实验（如表1 图9所示）揭示了一个不同的故事。我们进行了消融研究，在保持总视觉和文本标记预算固定的情况下，改变了视觉比例和视觉注入时间。为了严格满足不同比例的目标，我们在引入视觉数据之前，用文本标记对模型进行了预训练，预训练的标记数量是专门计算的。令人惊讶的是，我们发现视觉比例对最终多模态性能的影响很小。事实上，**在固定的总视觉-文本标记预算下，早期融合较低的视觉比例可以获得更好的结果**。这促使我们采用了本地的多模态预训练策略：我们不是在训练后期进行激进的视觉主导训练，而是采用适中的视觉比例，在训练过程中早期整合，让模型能够自然地发展平衡的多模态表示，同时从两种模态的扩展协同优化中受益。

2.2 无视觉SFT

预训练的视觉语言模型（VLMs）无法自然地执行基于视觉的工具调用，这对多模态强化学习（RL）提出了冷启动问题。传统方法通过手动标注或提示工程链式思维（CoT）数据 [8], 来解决这个问题，但这类方法多样性有限，通常将视觉推理限制在简单的图表和原始的工具操作（裁剪、旋转、翻转）上。

一个观察是高质量文本SFT数据相对丰富且多样。我们提出了一种新方法——无视觉SFT，它仅使用文本SFT数据在后训练期间激活视觉、自主代理式能力。在这种方法中，所有图像操作都通过IPython中的程序化操作进行代理，有效地作为传统视觉工具使用的泛化。这种“无视觉”激活能够实现多样化的推理行为，包括通过二值化和计数进行对象大小估计等像素级操作，并能泛化到基于视觉的任务，如对象定位、计数和OCR。

图 2 展示了RL训练曲线，其中起点来自无视觉SFT。结果表明，无视觉SFT足以激活视觉能力，同时确保跨模态泛化。这种现象可能是由于第2.1节中描述的文本和视觉数据的联合预训练所致。与无视觉SFT相比，我们初步实验表明，文本-视觉SFT在视觉、自主代理式任务上的性能要差得多，这可能是由于缺乏高质量视觉数据。

2.3 联合多模态强化学习 (RL)

在本节中，我们描述了K2.5中实现的、能够有效进行多模态RL的方法论，从基于结果的视觉强化学习到增强文本性能的跨模态迁移涌现现象。

基于结果的视觉强化学习 在无视觉SFT之后，模型需要进一步优化，以便可靠地将视觉输入整合到推理过程中。仅靠文本触发的激活表现出显著的失效模式：有时会忽略视觉输入，并且在必要时可能不会关注图像。我们在明确需要视觉理解才能得出正确解的任务上应用基于结果的RL。我们将这些任务分为三个领域：

- **视觉 grounding 和计数：** 在图像中精确定位和枚举物体；
- **图表和文档理解：** 解释结构化视觉信息并提取文本；
- Vision-cr **关键 STEM 问题：** 数学和科学问题被筛选，需要视觉输入 ts.

基于结果的强化学习在这些任务上提升了基本视觉能力以及更复杂的自主代理式行为。提取这些轨迹用于拒绝采样微调（RFT）能够实现自我改进的数据管道，使后续的联合强化学习阶段能够利用更丰富的多模态推理轨迹。

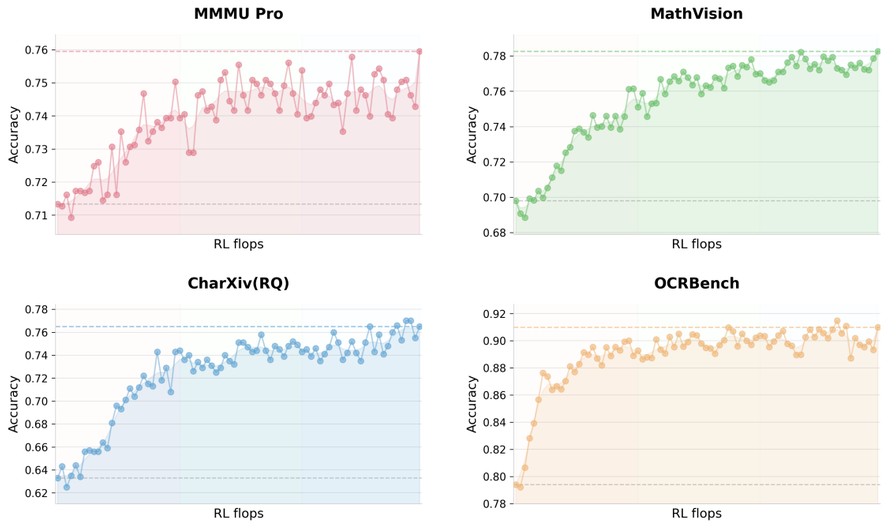


Figure 2: Vision RL training curves on vision benchmarks starting from minimal zero-vision SFT. By scaling vision RL FLOPs, the performance continues to improve, demonstrating that zero-vision activation paired with long-running RL is sufficient for acquiring robust visual capabilities.

Table 2: Cross-Modal Transfer: Vision RL Improves Textual Knowledge

Benchmark	Before Vision-RL	After Vision-RL	Improvement
MMLU-Pro	84.7	86.4	+1.7
GPQA-Diamond	84.3	86.4	+2.1
LongBench v2	56.7	58.9	+2.2

Visual RL Improves Text Performance To investigate potential trade-offs between visual and textual performance, we evaluated text-only benchmarks before and after visual RL. Surprisingly, outcome-based visual RL produced measurable improvements in textual tasks, including MMLU-Pro (84.7% $\rightarrow$ 86.4%), GPQA-Diamond (84.3% $\rightarrow$ 86.4%), and LongBench v2 (56.7% $\rightarrow$ 58.9%) (Table 2). Analysis suggests that visual RL enhances calibration in areas requiring structured information extraction, reducing uncertainty on queries that resemble visually grounded reasoning (e.g., counting, OCR). These findings indicate that visual RL can contribute to cross-modal generalization, improving textual reasoning without observable degradation of language capabilities.

Joint Multimodal RL Motivated by the finding that robust visual capabilities can emerge from zero-vision SFT paired with vision RL—which further enhances general text abilities—we adopt a joint multimodal RL paradigm during Kimi K2.5's post-training. Departing from conventional modality-specific expert divisions, we organize RL domains not by input modality but by abilities—knowledge, reasoning, coding, agentic, etc. These domain experts jointly learn from both pure-text and multimodal queries, while the Generative Reward Model (GRM) similarly optimizes across heterogeneous traces without modality barriers. This pardaigm ensures that capability improvements acquired through either textual or visual inputs inherently generalize to enhance related abilities across the alternate modality, thereby maximizing cross-modal capability transfer.

3 Agent Swarm

The primary challenge of existing agent-based systems lies in their reliance on sequential execution of reasoning and tool-calling steps. While this structure may be effective for simpler, short-horizon tasks, it becomes inadequate as the complexity of the task increases and the accumulated context grows. As tasks evolve to contain broad information gathering and intricate, multi-branch reasoning, sequential systems often encounter significant bottlenecks [5, 6, 7].

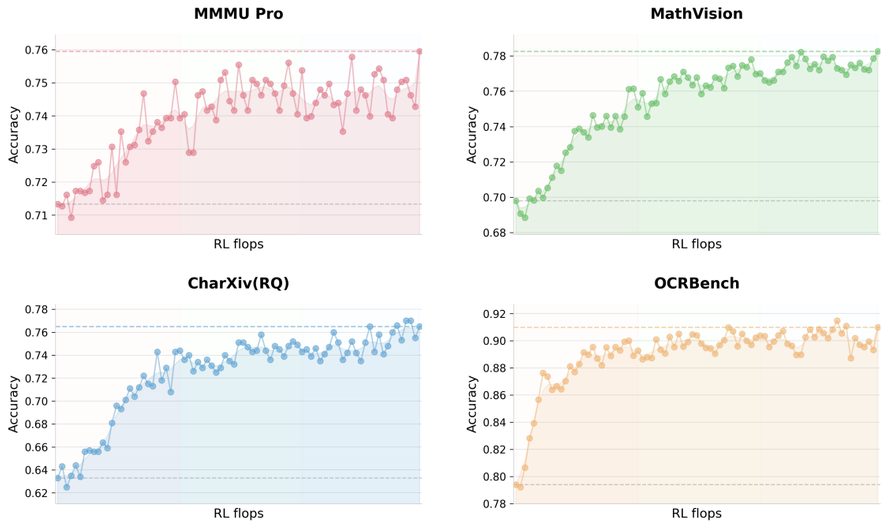


图2：从无视觉SFT开始，在视觉基准上进行视觉强化学习训练的曲线。通过扩展视觉RLFLOPs，性能持续提升，证明无视觉激活与长时间运行的强化学习足以获取稳健的视觉能力。

表2：跨模态迁移：视觉强化学习提升文本知识

基准测试	视觉强化学习前	在视觉强化学习之后	改进
MMLU-Pro	84.7	86.4	+1.7
GPQA-Diamond	84.3	86.4	+2.1
LongBench v2	56.7	58.9	+2.2

视觉强化学习提升文本性能为了研究视觉和文本性能之间的潜在权衡，我们在视觉强化学习前后评估了纯文本基准测试。令人惊讶的是，基于结果的视觉强化学习在文本任务中产生了可衡量的改进，包括MMLU-Pro（84.7% $\rightarrow$ 86.4%）、GPQA-Diamond（84.3% $\rightarrow$ 86.4%）和LongBench v2（56.7% $\rightarrow$ 58.9%）（表 2）。分析表明，视觉强化学习增强了需要结构化信息提取的领域的校准能力，减少了类似于视觉基础推理的查询的不确定性（例如，计数、OCR）。这些发现表明，视觉强化学习可以促进跨模态泛化，在语言能力没有明显退化的情况下提升文本推理能力。

联合多模态强化学习 受限于发现稳健的视觉能力可以从无视觉SFT与视觉强化学习相结合中产生——这进一步增强了通用文本能力——我们在Kimi K2.5的后期训练中采用了联合多模态强化学习范式。不同于传统的模态特定专家划分，我们不是按输入模态而是按能力组织RL域——知识、推理、编程、自主代理式等。这些领域专家共同从纯文本和多模态查询中学习，而生成式奖励模型（GRM）也类似地跨异构轨迹优化，不受模态限制。该范式确保通过文本或视觉输入获得的能力改进能够自然地泛化到增强交替模态的相关能力，从而最大限度地提高跨模态能力迁移。

3 Agent Swarm

现有基于代理的系统的核心挑战在于其依赖推理和工具调用步骤的顺序执行。虽然这种结构对于简单的短视任务可能有效，但随着任务复杂性的增加和累积上下文的增长，它变得不足够。当任务演变为包含广泛信息收集和复杂的多分支推理时，顺序系统往往会遇到显著的瓶颈 [5, 6, 7]。

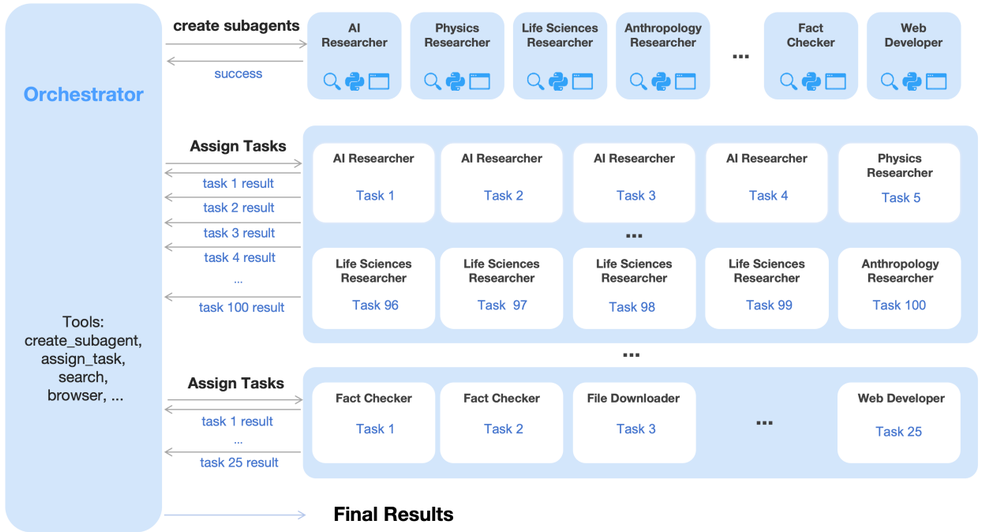


Figure 3: An agent swarm has a trainable orchestrator that dynamically creates specialized frozen subagents and decomposes complex tasks into parallelizable subtasks for efficient distributed execution.

The limited capacity of a single agent working through each step one by one can lead to the exhaustion of practical reasoning depth and tool-call budgets, ultimately hindering the system's ability to handle more complex scenarios. To address this, we introduce **Agent Swarm** and **Parallel Agent Reinforcement Learning (PARL)**. Instead of executing a task as a reasoning chain or relying on pre-specified parallelization heuristics, K2.5 initiates an Agent Swarm through dynamic task decomposition, subagent instantiation, and parallel subtask scheduling. Importantly, parallelism is not presumed to be inherently advantageous; decisions regarding whether, when, and how to parallelize are explicitly learned through environmental feedback and RL-driven exploration. As shown in Figure 4, the progression of performance demonstrates this adaptive capability, with the cumulative reward increasing smoothly as the orchestrator optimizes its parallelization strategy throughout training.

Architecture and Learning Setup The PARL framework adopts a decoupled architecture comprising a trainable orchestrator and frozen subagents instantiated from fixed intermediate policy checkpoints. This design deliberately avoids end-to-end co-optimization to circumvent two fundamental challenges: credit assignment ambiguity and training instability. In this multi-agent setting, outcome-based rewards are inherently sparse and noisy; a correct final answer does not guarantee flawless subagent execution, just as a failure does not imply universal subagent error. By freezing the subagents and treating their outputs as environmental observations rather than differentiable decision points, we disentangle high-level coordination logic from low-level execution proficiency, leading to more robust convergence. To improve efficiency, we first train the orchestrator using small-size subagents before transitioning to larger models. Our RL framework also supports dynamically adjusting the inference instance ratios between subagents and the orchestrator, thereby maximizing the resource usage across the cluster.

PARL Reward Training a reliable parallel orchestrator is challenging due to the delayed, sparse, and non-stationary feedback inherent in independent subagent execution. To address this, we define the PARL reward as:

$$r_{\text{PARL}}(x,y) = \lambda_1 \cdot \underbrace{r_{\text{parallel}}}_{\text{instantiation reward}} + \lambda_2 \cdot \underbrace{r_{\text{finish}}}_{\text{sub-agent finish rate}} + \underbrace{r_{\text{perf}}(x,y)}_{\text{task-level outcome}} .$$

The performance reward r_{perf} evaluates the overall success and quality of the solution y for a given task x. This is augmented by two auxiliary rewards, each addressing a distinct challenge in learning parallel orchestration. The reward r_{parallel} is introduced to mitigate *serial collapse*—a local optimum where the orchestrator defaults to single-agent execution. By incentivizing subagent instantiation, this term encourages the exploration of concurrent scheduling

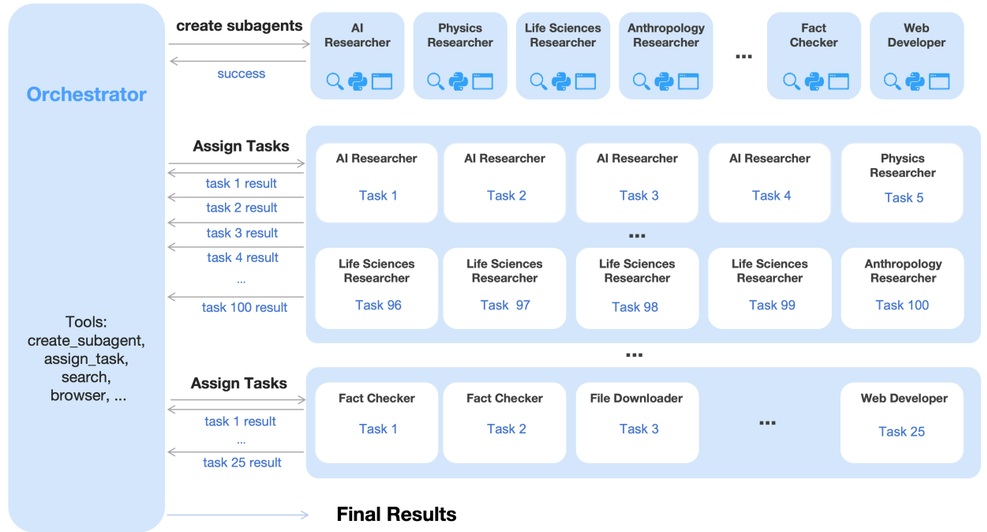


图3：一个代理群拥有可训练的编排器，该编排器动态创建专门化的冻结子代理，并将复杂任务分解为可并行化的子任务，以实现高效的分布式执行。

单个代理逐一处理每个步骤的有限容量可能导致实际推理深度和工具调用预算的耗尽，最终阻碍系统处理更复杂场景的能力。为解决这一问题，我们引入 **代理群** 和 **并行代理强化学习（PARL）**。K2.5不是将任务作为推理链执行，也不依赖预定义的并行化启发式算法，而是通过动态任务分解、子代理实例化和并行子任务调度来启动代理群。重要的是，并行化并非天然具有优势；关于是否、何时以及如何并行化的决策，是通过环境反馈和RL驱动的探索明确学习的。如图4所示，性能的进步展示了这种自适应能力，随着编排器在训练过程中优化其并行化策略，累积奖励平滑增加。

架构和学习设置 PARL框架采用一种解耦架构，由可训练的编排器和从固定的中间策略检查点实例化的冻结子代理组成。这种设计有意避免端到端协同优化，以规避两个基本挑战：信用分配模糊性和训练不稳定性。在这种多代理设置中，基于结果的奖励本质上稀疏且嘈杂；一个正确的最终答案并不能保证子代理执行的完美，就像失败也不意味着所有子代理都出错一样。通过冻结子代理并将它们的输出视为环境观察值而不是可微分的决策点，我们将高级协调逻辑与低级执行能力解耦，从而实现更鲁棒的收敛。为了提高效率，我们首先使用小尺寸子代理训练编排器，然后再过渡到更大的模型。我们的RL框架还支持动态调整子代理和编排器之间的推理实例比例，从而最大化集群的资源利用率。

PARLReward 训练一个可靠的并行编排器具有挑战性，这是由于独立子代理执行中固有的延迟、稀疏和非平稳反馈所致。为此，我们定义了PARL奖励：

$$r_{\text{PARL}}(x,y) = \lambda_1 \cdot \underbrace{r_{\text{parallel}}}_{\text{instantiation reward}} + \lambda_2 \cdot \underbrace{r_{\text{finish}}}_{\text{sub-agent finish rate}} + \underbrace{r_{\text{perf}}(x,y)}_{\text{task-level outcome}} .$$

性能奖励 r^{perf} 评估了针对给定任务 x 的解决方案 y 的整体成功和质量。这通过两个辅助奖励得到增强，每个奖励都针对学习并行编排中的不同挑战。奖励 r^{parallel} 被引入以缓解 串行崩溃——这是一种局部最优，其中编排器默认为单代理执行。通过激励子代理实例化，这一项鼓励对并发调度的探索

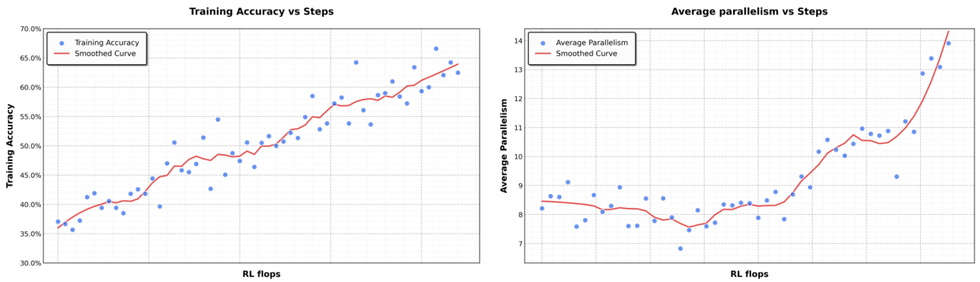


Figure 4: In our parallel-agent reinforcement learning environment, the training accuracy increases smoothly as training progresses. At the same time, the level of parallelism during training also gradually increases.

spaces. The r_{finish} reward focuses on the successful completion of assigned subtasks. It is used to prevent *spurious parallelism*, a reward-hacking behavior in which the orchestrator increases parallel metrics dramatically by spawning many subagents without meaningful task decomposition. By rewarding completed subtasks, r_{finish} enforces feasibility and guides the policy toward valid and effective decompositions.

To ensure the final policy optimizes for the primary objective, the hyperparameters λ_1 and λ_2 are annealed to zero over the course of training.

Critical Steps as Resource Constraint To measure computational time cost in a parallel-agent setting, we define *critical steps* by analogy to the *critical path* in a computation graph. We model an episode as a sequence of execution stages indexed by $t = 1, \ldots, T$. In each stage, the main agent executes an action, which corresponds to either direct tool invocation or the instantiation of a group of subagents running in parallel. Let $S^{(t)}_{\text{main}}$ denote the number of steps taken by the main agent in stage t (typically $S^{(t)}_{\text{main}} = 1$), and $S^{(t)}_{\text{sub},i}$ denote the number of steps taken by the i-th subagent in that parallel group. The duration of stage t is governed by the longest-running subagent within that cohort. Consequently, the total critical steps for an episode are defined as

$$\text{CriticalSteps} = \sum_{t=1}^{T} \left(S^{(t)}_{\text{main}} + \max_i S^{(t)}_{\text{sub},i} \right).$$

By constraining training and evaluation using critical steps rather than total steps, the framework explicitly incentivizes effective parallelization. Excessive subtask creation that does not reduce the maximum execution time of parallel groups yields little benefit under this metric, while well-balanced task decomposition that shortens the longest parallel branch directly reduces critical steps. As a result, the orchestrator is encouraged to allocate work across subagents in a way that minimizes end-to-end latency, rather than merely maximizing concurrency or total work performed.

Prompt Construction for Parallel-agent Capability Induction To incentivize the orchestrator to leverage the advantages of parallelization, we construct a suite of synthetic prompts designed to stress the limits of sequential agentic execution. These prompts emphasize either *wide search*, requiring simultaneous exploration of many independent information sources, or *deep search*, requiring multiple reasoning branches with delayed aggregation. We additionally include tasks inspired by real-world workloads, such as long-context document analysis and large-scale file downloading. When executed sequentially, these tasks are difficult to complete within fixed reasoning-step and tool-call budgets. By construction, they encourage the orchestrator to allocate subtasks in parallel, enabling completion within fewer critical steps than would be feasible for a single sequential agent. Importantly, the prompts do not explicitly instruct the model to parallelize. Instead, they shape the task distribution such that parallel decomposition and scheduling strategies are naturally favored.

4 Method Overview

4.1 Foundation: Kimi K2 Base Model

The foundation of Kimi K2.5 is Kimi K2 [53], a trillion-parameter mixture-of-experts (MoE) transformer [59] model pre-trained on 15 trillion high-quality text tokens. Kimi K2 employs the token-efficient MuonClip optimizer [30,

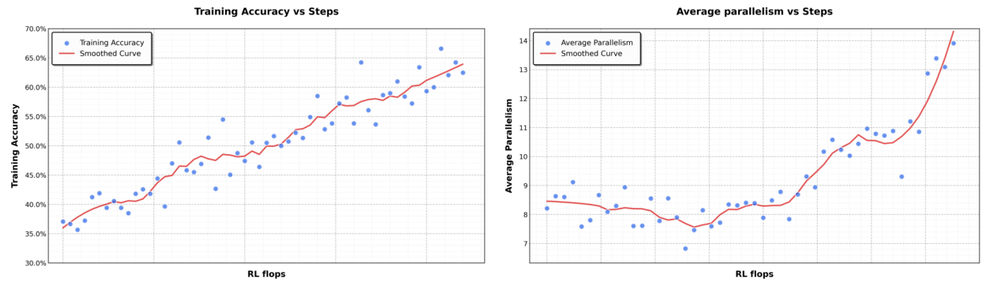


图4：在我们的并行代理强化学习环境中，随着训练的进行，训练准确率平滑增长。同时，训练过程中的并行度也逐渐提高。

空间。奖励 r_{finish} 侧重于分配的子任务的成功完成。它用于防止 虚假并行，这是一种奖励攻击行为，其中编排器通过大量生成子代理而大幅增加并行指标，而没有进行有意义的任务分解。通过奖励完成的子任务，r_{finish} 强制可行性，并引导策略朝向有效且合理的分解。

为确保最终策略优化主要目标，超参数 λ_1 和 λ_2 在整个训练过程中退火至零。

资源约束下的关键步骤 为了测量并行代理环境中的计算时间成本，我们通过类比计算图中的 关键路径 来定义 关键步骤 。我们将一个回合建模为一系列以 $t = 1, \ldots, T$索引的执行阶段。在每个阶段，主代理执行一个动作，该动作对应于直接工具调用或并行运行的一组子代理的实例化。令 $S^{(t)}\,\text{main}$ 表示主代理在阶段 t （通常 $S^{(t)}\,\text{main} = 1$）中执行的操作数量，而 $S^{(t)}\,\text{sub}_i$ 表示该并行组中第 i 个子代理执行的操作数量。阶段 t 的持续时间由该群体中运行时间最长的子代理决定。因此，一个回合的总关键步骤定义为

$$\text{CriticalSteps} = \sum_{t=1}^{T} \left(S^{(t)}_{\text{main}} + \max_i S^{(t)}_{\text{sub},i} \right).$$

通过使用关键步骤而非总步骤来约束训练和评估，该框架明确激励了有效的并行化。那些不减少并行组最大执行时间的过度子任务创建，在此指标下收益甚微，而均衡的任务分解则能直接缩短最长并行分支，从而减少关键步骤。因此，编排器被鼓励以最小化端到端延迟的方式分配工作给子代理，而非仅仅最大化并发性或总工作量。

并行代理能力诱导的提示构造 为激励编排器利用并行化的优势，我们构建了一套旨在测试顺序代理执行极限的合成提示。这些提示强调要么 广度搜索，要求同时探索许多独立的信息源，要么 深度搜索，要求多个推理分支并延迟聚合。我们额外包含受真实工作负载启发的任务，如长上下文文档分析和大规模文件下载。当顺序执行时，这些任务难以在固定的推理步数和工具调用预算内完成。按设计，它们鼓励编排器并行分配子任务，从而在少于单个顺序代理可能实现的更少关键步骤内完成。重要的是，这些提示并未明确指示模型并行化。相反，它们塑造了任务分配方式，使得并行分解和调度策略自然更受青睐。

4 方法概述

4.1 基础：Kimi K2 基础模型

Kimi K2.5 的基础是 Kimi K2 [53], 一个万亿参数专家混合（MoE）转换器 [59] 模型，在 150 万亿个高质量文本标记上进行预训练。Kimi K2 采用 token 效率的 MuonClip 优化器 [30,

Table 3: Overview of training stages: data composition, token volumes, sequence lengths, and trainable components.

Stages	ViT Training	Joint Pre-training	Joint Long-context Mid-training
Data	Alt text Synthesis Caption Grounding, OCR, Video	+ Text, Knowledge Interleaving Video, OS Screenshot	+ High-quality Text & Multimodal Long Text, Long Video Reasoning, Long-CoT
Sequence length	4096	4096	32768→262144
Tokens	1T	15T	500B→200B
Training	ViT	ViT & LLM	ViT & LLM

34] with QK-Clip for training stability. The model comprises 1.04 trillion total parameters with 32 billion activated parameters, utilizing 384 experts with 8 activated per token (sparsity of 48). For detailed descriptions of MuonClip, architecture design, and training infrastructure, we refer to the Kimi K2 technical report [53].

4.2 Model Architecture

The multimodal architecture of Kimi K2.5 consists of three components: a three-dimensional native-resolution vision encoder (MoonViT-3D), an MLP projector, and the Kimi K2 MoE language model, following the design principles established in Kimi-VL [54].

MoonViT-3D: Shared Embedding Space for Images and Videos In Kimi-VL, we employ MoonViT to natively process images at their original resolutions, eliminating the need for complex sub-image splitting and splicing operations. Initialized from SigLIP-SO-400M [78], MoonViT incorporates the patch packing strategy from NaViT [15], where single images are divided into patches, flattened, and sequentially concatenated into 1D sequences, thereby enabling efficient simultaneous training on images at varying resolutions.

To maximize the transfer of image understanding capabilities to video, we introduce **MoonViT-3D** with a unified architecture, fully shared parameters, and a consistent embedding space. By generalizing the "patch n' pack" philosophy to the temporal dimension, up to four consecutive frames are treated as a spatiotemporal volume: 2D patches from these frames are jointly flattened and packed into a single 1D sequence, allowing the identical attention mechanism to operate seamlessly across both space and time. While the extra temporal attention improves understanding on high-speed motions and visual effects, the sharing maximizes knowledge generalization from static images to dynamic videos, achieving strong video understanding performance (see in Tab. 4) without requiring specialized video modules or architectural bifurcation. Prior to the MLP projector, lightweight temporal pooling aggregates patches within each temporal chunk, yielding 4× temporal compression to significantly extend feasible video length. The result is a unified pipeline where knowledge and ability obtained from image pretraining transfers holistically to videos through one shared parameter space and feature representation.

4.3 Pre-training Pipeline

As illustrated in Table 3, Kimi K2.5's pre-training builds upon the Kimi K2 language model checkpoint and processes approximately 15T tokens across three stages: first, standalone ViT training to establish a robust native-resolution visual encoder; second, joint pre-training to simultaneously enhance language and multimodal capabilities; and third, mid-training on high-quality data and long-context activation to refine capabilities and extend context windows.

ViT Training Stage The MoonViT-3D is trained on image-text and video-text pairs, where the text components consist of a variety of targets: image alt texts, synthetic captions of images and videos, grounding bboxes, and OCR texts. The training incorporates two objectives following CoCa [75]: a SigLIP [78] loss L_{siglip} (a variant of contrastive loss) and a cross-entropy loss $L_{caption}$ for caption generation conditioned on input images. We adopt a two-stage alignment strategy. In the first stage, we optimize solely the captioning loss L_{caption} to align MoonViT-3D with Moonlight-16B-A3B [34], consuming 1T tokens, in which stage the ViT weights will be updated. A very short second stage follows, updating only the MLP projector to bridge the ViT with the 1T LLM for smoother joint pre-training.

表 3：训练阶段概述：数据组成、token 数量、序列长度和可训练组件。

阶段	ViT 训练	联合预训练	联合长上下文中期训练
Data	替代文本 合成字幕 定位、OCR、视频	+ 文本、知识 交织 视频，操作系统截图	+ 高质量文本 & 多模态 长文本，长视频 推理，长思维链
序列长度	4096	4096	32768→262144
标记	1T	15T	500B→200B
训练	ViT	ViT & 大语言模型	ViT & 大语言模型

34] 配合 QK-Clip 以确保训练稳定性。该模型包含 1.04 万亿个总参数，其中 320 亿个参数被激活，利用 384 个专家，每个 token 激活 8 个（稀疏性为 48%）。关于 MuonClip、架构设计和训练基础设施的详细描述，我们参考了 Kimi K2 技术报告 [53]。

4.2 模型架构

Kimi K2.5的多模态架构由三个组件构成：一个三维原生分辨率视觉编码器（MoonViT-3D）、一个MLP投影器，以及Kimi K2 MoE语言模型，遵循Kimi-VL [54]中确立的设计原则。

MoonViT-3D：图像和视频共享嵌入空间在Kimi-VL中，我们采用MoonViT原生处理原始分辨率的图像，无需复杂的子图像分割和拼接操作。从SigLIP-SO-400M [78], 初始化的MoonViT集成了来自NaViT [15], 的分块打包策略，将单张图像分割成块，展平，并顺序连接成1D序列，从而能够对分辨率不同的图像进行高效的同步训练。
为了最大化将图像理解能力迁移到视频，我们引入 **MoonViT-3D** ，它采用统一架构、完全共享的参数以及一致的嵌入空间。通过将“Patch n’ Pack”哲学推广到时间维度，最多四个连续帧被视为一个时空体积：这些帧的2D patch被联合展平并打包成一个单一的1D序列，使得相同的注意力机制能够无缝地作用于空间和时间。虽然额外的注意力机制提高了对高速运动和视觉效果的理解，共享性最大化了从静态图像到动态视频的知识泛化，实现了强大的视频理解性能（参见Tab.4），而无需专门的视频模块或架构分支。在MLP投影器之前，轻量级时间池化聚合每个时间块内的patch，产生 4× 时间压缩，显著延长了可行视频长度。结果是形成一个统一流程，从图像预训练中获得的知识和能力通过一个共享的参数空间和特征表示整体地迁移到视频。

4.3 预训练流程

如表3所示，Kimi K2.5 的预训练基于 Kimi K2 语言模型检查点，处理约 15T 标记，分为三个阶段：首先，进行独立的 ViT 训练以建立鲁棒的原始分辨率视觉编码器；其次，进行联合预训练以同时提升语言和多模态能力；最后，在高质量数据和长上下文激活的中期训练中，进一步优化能力并扩展上下文窗口。

ViT训练阶段 MoonViT-3D 在图像-文本和视频-文本对上进行训练，其中文本组件包括多种目标：图像替代文本、图像和视频的合成字幕、锚框以及OCR文本。训练结合了两个遵循CoCa [75]的目标：SigLIP [78] 损失 L^{siglip} （对比损失的变体）和条件于输入图像的生成字幕的交叉熵损失 $L^{caption}$ 。我们采用两阶段对齐策略。在第一阶段，我们仅优化字幕损失 L^{caption} ，以将MoonViT-3D与Moonlight-16B- A3B [34], 对齐，该阶段消耗1T标记，其中ViT权重将被更新。一个非常短的第二阶段随后进行，仅更新MLP投影器，以将ViT与1T LLM连接起来，以便进行更平滑的联合预训练。

Joint Training Stages The joint pre-training stage continues from a near-end Kimi K2 checkpoint over additional 15T vision-text tokens at 4K sequence length. The data recipe extends Kimi K2's pre-training distribution by introducing unique tokens, adjusting data proportions with increased weight on coding-related content, and controlling maximum epochs per data source. The third stage performs long-context activation with integrated higher-quality mid-training data, sequentially extending context length via YaRN [44] interpolation. This yields significant generalization improvements in long-context text understanding and long video comprehension.

4.4 Post-Training

4.4.1 Supervised Fine-Tuning

Following the SFT pipeline established by Kimi K2 [53], we developed K2.5 by synthesizing high-quality candidate responses from K2, K2 Thinking and a suite of proprietary in-house expert models. Our data generation strategy employs specialized pipelines tailored to specific domains, integrating human annotation with advanced prompt engineering and multi-stage verification. This methodology produced a large-scale instruction-tuning dataset featuring diverse prompts and intricate reasoning trajectories, ultimately training the model to prioritize interactive reasoning and precise tool-calling for complex, real-world applications.

4.4.2 Reinforcement Learning

Reinforcement learning constitutes a crucial phase of our post-training. To facilitate joint optimization across text and vision modalities, as well as to enable PARL for agent swarm, we develop a Unified Agentic Reinforcement Learning Environment (Appendix D) and optimize the RL algorithms. Both text-vision joint RL and PARL are built upon the algorithms described in this section.

Policy Optimization For each problem x sampled from a dataset $\mathcal{D}$, K responses $\{y_1, \ldots, y_K\}$ are generated using the previous policy π_{old}. We optimize the model π_θ with respect to the following objective:

$$L_{\text{RL}}(\theta) = \mathbb{E}_{x\sim\mathcal{D}}\left[\frac{1}{N}\sum_{j=1}^{K}\sum_{i=1}^{|y_j|}\text{Clip}\left(\frac{\pi_\theta(y_j^i|x,y_j^{0:i})}{\pi_{\text{old}}(y_j^i|x,y_j^{0:i})},\alpha,\beta\right)(r(x,y_j)-\bar{r}(x)) - \tau\left(\log\frac{\pi_\theta(y_j^i|x,y_j^{0:i})}{\pi_{\text{old}}(y_j^i|x,y_j^{0:i})}\right)^2\right]. \quad (1)$$

Here $\alpha,\beta,\tau>0$ are hyperparameters, $y_{0:i}^j$ is the prefix up to the i-th token of the j-th response, $N=\sum_{i=1}^{K}|y_i|$ is the total number of generated tokens in a batch, $\bar{r}(x)=\frac{1}{K}\sum_{j=1}^{K}r(x,y_j)$ is the mean reward of all generated responses.

This loss function departs from the policy optimization algorithm used in K1.5 [31] by introducing a token-level clipping mechanism designed to mitigate the off-policy divergence amplified by discrepancies between training and inference frameworks. The mechanism functions as a simple gradient masking scheme: policy gradients are computed normally for tokens with log-ratios within the interval $[\alpha,\beta]$, while gradients for tokens falling outside this range are zeroed out. Notably, a key distinction from standard PPO clipping [50] is that our method relies strictly on the log-ratio to explicitly bound off-policy drift, regardless of the sign of the advantages. This approach aligns with recent strategies proposed to stabilize large-scale RL training [74, 79]. Empirically, we find this mechanism essential for maintaining training stability in complex domains requiring long-horizon, multi-step tool-use reasoning. We employ the MuonClip optimizer [30, 34] to minimize this objective.

Reward Function We apply a rule-based outcome reward for tasks with verifiable solutions, such as reasoning and agentic tasks. To optimize resource consumption, we also incorporate a budget-control reward aimed at enhancing token efficiency. For general-purpose tasks, we employ Generative Reward Models (GRMs) that provide granular evaluations aligned with Kimi's internal value criteria. In addition, for visual tasks, we design task-specific reward functions to provide fine-grained supervision. For visual grounding and point localization tasks, we employ an F1-based reward with soft matching: grounding tasks derive soft matches from Intersection over Union (IoU) and point tasks derive soft matches from Gaussian-weighted distances under optimal matching. For polygon segmentation tasks, we rasterize the predicted polygon into a binary mask and compute the segmentation IoU against the ground-truth mask to assign the reward. For OCR tasks, we adopt normalized edit distance to quantify character-level alignment between predictions and ground-truth. For counting tasks, rewards are assigned based on the absolute difference between predictions and ground-truth. Furthermore, we synthesize complex visual puzzle problems and utilize an LLM verifier (Kimi K2) to provide feedback.

Generative Reward Models Kimi K2 leverages a self-critique rubric reward for open-ended generation [53], and K2.5 extends this line of work by systematically deploying *Generative Reward Models (GRMs)* across a broad range

联合训练阶段 联合预训练阶段从接近结束的Kimi K2检查点开始，在4K序列长度下处理额外的15T视觉-文本标记。数据配方通过引入独特标记、调整数据比例并增加与编程相关内容的权重来扩展Kimi K2的预训练分布，并控制每个数据源的最大轮数。第三阶段执行长上下文激活，并使用集成的高质量中期训练数据，通过YaRN [44]插值顺序扩展上下文长度。这显著提高了长上下文文本理解和长视频理解的泛化能力。

4.4 训练后

4.4.1 有监督微调

遵循 Kimi K2 [53]，建立的 SFT 流程，我们通过从 K2、K2 思考和一套专有的内部专家模型中合成高质量候选响应，开发了 K2.5。我们的数据生成策略采用针对特定领域的专用流程，将人工标注与先进的提示工程和多阶段验证相结合。这种方法论产生了一个大规模的指令调优数据集，其中包含多样化的提示和复杂的推理轨迹，最终训练模型以优先考虑交互式推理和精确的工具调用，用于复杂的现实世界应用。

4.4.2 强化学习

强化学习是我们训练后阶段的关键环节。为了促进文本和视觉模态的联合优化，以及使PARL支持代理群，我们开发了一个统一的自主代理式强化学习环境（附录D）并优化了RL算法。文本-视觉联合RL和PARL都基于本节所述的算法构建。

策略优化对于从数据集 $\mathcal{D}$中采样的每个问题x，使用先前的策略 π_{old}生成K个响应 $\{y1, ..., y_K\}$。我们针对以下目标优化模型 π_θ：

$$L_{\text{RL}}(\theta) = \mathbb{E}_{x\sim\mathcal{D}}\left[\frac{1}{N}\sum_{j=1}^{K}\sum_{i=1}^{|y_j|}\text{Clip}\left(\frac{\pi_\theta(y_j^i|x,y_j^{0:i})}{\pi_{\text{old}}(y_j^i|x,y_j^{0:i})},\alpha,\beta\right)(r(x,y_j)-\bar{r}(x)) - \tau\left(\log\frac{\pi_\theta(y_j^i|x,y_j^{0:i})}{\pi_{\text{old}}(y_j^i|x,y_j^{0:i})}\right)^2\right]. \quad (1)$$

其中α、β、$\tau>0$是超参数，$y_{0:i}^j$是第j个响应的前缀，直到第i个标记，$N=\sum_{i=1}^{K}|y_i|$是批次中生成的总标记数，$\bar{r}(x)=\frac{1}{K}\sum_{j=1}^{K}r(x,y_j)$是所有生成响应的平均奖励。

该损失函数与K1.5 [31]中使用的策略优化算法不同，通过引入一种token级别的裁剪机制，旨在减轻由训练和推理框架之间的差异所放大的离策略发散。该机制作为一个简单的梯度掩码方案工作：对于log-ratio在区间 $[\alpha,\beta]$内的token，正常计算策略梯度，而对于落在此范围之外的token，梯度将被置零。值得注意的是，与标准PPO裁剪 [50] 的关键区别在于，我们的方法严格依赖于log-ratio来显式地界定离策略漂移，而不管优势的符号如何。这种方法与最近提出的用于稳定大规模RL训练 [74, 79]的策略保持一致。经验上，我们发现该机制对于在需要长时程、多步工具使用推理的复杂领域中维持训练稳定性至关重要。我们采用MuonClip优化器[30, 34] 来最小化此目标。

奖励函数 对于具有可验证解决方案的任务（如推理和自主任务），我们应用基于规则的成果奖励。为了优化资源消耗，我们还结合了预算控制奖励，旨在提高token效率。对于通用任务，我们采用生成式奖励模型（GRMs），它们提供与Kimi的内部价值标准相一致的粒度评估。此外，对于视觉任务，我们设计了特定任务的奖励函数以提供细粒度监督。对于视觉定位和点定位任务，我们采用基于F1的奖励与软匹配：定位任务从交并比（IoU）中获得软匹配，点任务从在最佳匹配下的高斯加权距离中获得软匹配。对于多边形分割任务，我们将预测的多边形光栅化为二值掩码，并计算分割IoU以与真实掩码进行比较以分配奖励。对于OCR任务，我们采用归一化编辑距离来量化预测与真实掩码之间的字符级对齐。对于计数任务，奖励基于预测与真实掩码之间的绝对差值分配。此外，我们合成复杂视觉拼图问题，并利用大语言模型验证器（Kimi K2）提供反馈。

生成式奖励模型Kimi K2 利用自我批评评分奖励进行开放式生成 [53]，，而 K2.5 则通过系统性地部署 生成式奖励模型 *(GRM)* 来扩展这一工作线。

of agentic behaviors and multimodal trajectories. Rather than limiting reward modeling to conversational outputs, we apply GRMs on top of verified reward signals in diverse environments, including chat assistants, coding agents, search agents, and artifact-generating agents. Notably, GRMs function not as binary adjudicators, but as fine-grained evaluators aligned with Kimi's values that are critical to user experiences, such as helpfulness, response readiness, contextual relevance, appropriate level of detail, aesthetic quality of generated artifacts, and strict instruction following. This design allows the reward signal to capture nuanced preference gradients that are difficult to encode with purely rule-based or task-specific verifiers. To mitigate reward hacking and overfitting to a single preference signal, we employ multiple alternative GRM rubrics tailored to different task contexts.

Token Efficient Reinforcement Learning Token efficiency is central to LLMs with test-time scaling. While test-time scaling inherently trades computation for reasoning quality, practical gains require algorithmic innovations that actively navigate this trade-off. Our previous findings indicate that imposing a problem-dependent budget effectively constrains inference-time compute, incentivizing the model to generate more concise chain of thought reasoning patterns without unnecessary token expansion [31, 53]. However, we also observe a *length-overfitting phenomenon*: models trained under rigid budget constraints often fail to generalize to higher compute scales. Consequently, they cannot effectively leverage additional inference-time tokens to solve complex problems, instead defaulting to truncated reasoning patterns.

To this end, we propose *Toggle*, a training heuristic that alternates between inference-time scaling and budget-constrained optimization: for learning iteration t, the reward function is defined by

$$\tilde{r}(x,y) = \begin{cases} r(x,y)\cdot \mathbb{I}\left\{\frac{1}{K}\sum_{i=1}^{K} r(x,y_i) < \lambda \text{ or } |y_i| \le \text{budget}(x)\right\} & \text{if } \lfloor t/m \rfloor \pmod 2 = 0 \text{ (Phase0)} \\ r(x,y) & \text{if } \lfloor t/m \rfloor \pmod 2 = 1 \text{ (Phase1)} \end{cases}.$$

where λ and m are hyper-parameters of the algorithm and K is the number of rollouts per problem. Specifically, the algorithm alternates between two optimization phases every m iterations:

- Phase0 (*budget limited phase*): The model is trained to solve the problem within a task-dependent token budget. To prevent a premature sacrifice of quality for efficiency, this constraint is conditionally applied: it is only enforced when the model's mean accuracy for a given problem exceeds the threshold λ.
- Phase1 (*standard scaling phase*): The model generates responses up to the maximum token limit, encouraging the model to leverage computation for better inference-time scaling.

The problem-dependent budget is estimated from the ρ-th percentile of token lengths among the subset of correct responses:

$$\text{budget}(x) = \text{Percentile}\left(\{|y_j| \mid r(x,y_i) = 1, i = 1,\ldots,K\}, \rho\right). \tag{2}$$

This budget is estimated once at the beginning of training and remains fixed thereafter. Notably, Toggle functions as a stochastic alternating optimization for a bi-objective problem. It is specifically designed to reconcile reasoning capabilities with computational efficiency.

We evaluate the effectiveness of Toggle on K2 Thinking [1]. As shown in Figure 5, we observe a consistent reduction in output length across nearly all benchmarks. On average, Toggle decreases output tokens by 25∼30% with a negligible impact on performance. We also observe that redundant patterns in the chain-of-thought, such as repeated verifications and mechanical calculations, decrease substantially. Furthermore, Toggle shows strong domain generalization. For example, when trained exclusively on mathematics and programming tasks, the model still achieves consistent token reductions on GPQA and MMLU-Pro with only marginal degradation in performance (Figure 5).

4.5 Training Infrastructure

Kimi K2.5 inherits the training infrastructure from Kimi K2 [53] with minimal modifications. For multimodal training, we propose Decoupled Encoder Process, where the vision encoder is incorporated into the existing pipeline with negligible additional overhead.

4.5.1 Decoupled Encoder Process (DEP)

In a typical multimodal training paradigm utilizing Pipeline Parallelism (PP), the vision encoder and text embedding are co-located in the first stage of the pipeline (Stage-0). However, due to the inherent variations of multimodal input size (e.g., image counts and resolutions), Stage-0 suffers from drastic fluctuations in both computational load and memory usage. This forces existing solutions to adopt custom PP configurations for vision-language models — for instance, [54] manually adjusts the number of text decoder layers in Stage-0 to reserve memory. While this

自主代理式行为和多模态轨迹。与其将奖励建模局限于对话输出，我们将在多样化环境中（包括聊天助手、编程代理、搜索代理和生成工件代理）应用GRM，基于经过验证的奖励信号。值得注意的是，GRM并非作为二元裁决者，而是作为与Kimi价值观相一致的细粒度评估者，这些价值观对用户体验至关重要，例如助益性、响应就绪性、上下文相关性、适当的细节级别、生成工件的美学质量以及严格遵循指令。这种设计允许奖励信号捕捉难以用纯规则或特定任务验证器编码的细微偏好梯度。为缓解奖励攻击和过度拟合单一偏好信号，我们采用针对不同任务环境的多重替代GRM评估标准。

Token Efficient Reinforcement Learning Token效率对具有测试时扩展的大型语言模型至关重要。虽然测试时扩展本质上是用计算能力换取推理质量，但实际收益需要能够主动应对这种权衡的算法创新。我们之前的发现表明，施加与问题相关的预算可以有效地约束推理时的计算量，从而激励模型生成更简洁的思维链推理模式，而不会出现不必要的token扩展 [31, 53]。然而，我们也观察到一种长度过拟合现象：在严格的预算约束下训练的模型往往无法泛化到更高的计算规模。因此，它们无法有效利用额外的推理时token来解决复杂问题，而是默认采用截断的推理模式。

为此，我们提出了*Toggle*，一种在推理时扩展和预算约束优化之间交替的训练启发式方法：对于学习迭代t，奖励函数定义为

$$\tilde{r}(x,y) = \begin{cases} r(x,y)\cdot \mathbb{I}\left\{\frac{1}{K}\sum_{i=1}^{K} r(x,y_i) < \lambda \text{ or } |y_i| \le \text{budget}(x)\right\} & \text{if } \lfloor t/m \rfloor \pmod 2 = 0 \text{ (Phase0)} \\ r(x,y) & \text{if } \lfloor t/m \rfloor \pmod 2 = 1 \text{ (Phase1)} \end{cases}.$$

其中 λ和m 是算法的超参数，而K 是每个问题中的展开次数。具体来说，算法每隔m 次迭代就在两个优化阶段之间交替：

- Phase0 (预算有限阶段): 该模型被训练在任务相关的令牌预算内解决问题。为了防止过早地为了效率而牺牲质量，此约束是条件性地应用的：仅在模型的给定问题的平均准确率超过阈值 λ 时才强制执行。
- 第一阶段（标准缩放阶段）：模型生成响应直至最大token限制，鼓励模型利用计算进行更好的推理时缩放。

问题相关的预算是根据正确响应子集中的标记长度中位数估计的。

$$\text{budget}(x) = \text{Percentile}\left(\{|y_j| \mid r(x,y_i) = 1, i = 1,\ldots,K\}, \rho\right). \tag{2}$$

此预算在训练开始时估计一次，之后保持固定。值得注意的是，Toggle充当一个用于双目标问题的随机交替优化。它专门设计用于协调推理能力与计算效率。

我们评估了 Toggle 在 K2 思考上的有效性 [1]。如图 5所示，我们观察到在几乎所有基准测试中输出长度都呈现一致减少。平均而言，Toggle 使输出标记减少了 25∼30%，且对性能的影响可以忽略不计。我们还观察到，思维链中的冗余模式，例如重复验证和机械计算，显著减少。此外，Toggle 表现出很强的领域泛化能力。例如，当仅在对数学和编程任务进行训练时，模型在 GPQA 和 MMLU-Pro 上仍实现了持续标记减少，且性能仅略有下降（图 5）。

4.5 训练基础设施

Kimi K2.5 继承了 Kimi 2 [53] 的训练基础设施，仅做了少量修改。对于多模态训练，我们提出了解耦编码器流程（Decoupled Encoder Process），将视觉编码器集成到现有流程中，且额外开销可以忽略不计。

4.5.1 解耦编码器流程（DEP）

在采用流水线并行（PP）的典型多模态训练范式中，视觉编码器和文本嵌入位于流水线的第一阶段（Stage-0）。然而，由于多模态输入大小的固有差异（例如图像数量和分辨率），Stage-0 在计算负载和内存使用上都存在剧烈波动。这迫使现有解决方案为视觉语言模型采用定制的PP配置——例如， [54]手动调整Stage-0中文本解码器层的数量以保留内存。虽然这种

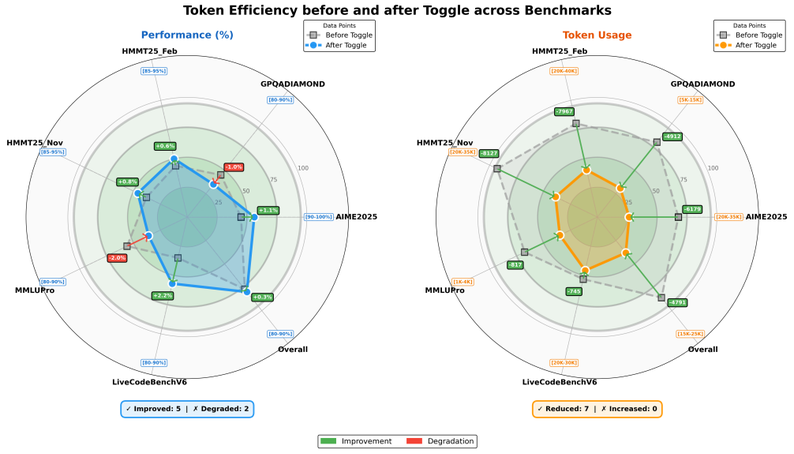


Figure 5: Comparison of model performance and token usage for Kimi K2 Thinking following token-efficient RL.

compromise alleviates memory pressure, it does not fundamentally resolve the load imbalance caused by multimodal input sizes. More critically, it precludes the direct reuse of parallel strategies that have been highly optimized for text-only training.

Leveraging the unique topological position of the visual encoder within the computation graph — specifically, its role as the start of the forward pass and the end of the backward pass — our training uses **Decoupled Encoder Process (DEP)**, which is composed of three stages in each training step:

- **Balanced Vision Forward:** We first execute the forward pass for all visual data in the global batch. Because the vision encoder is small, we replicate it on all GPUs regardless of other parallelism strategies. During this phase, the forward computational workload is evenly distributed across all GPUs based on load metrics (e.g., image or patch counts). This eliminates load-imbalance caused by PP and visual token counts. To minimize peak memory usage, we discard all intermediate activations, retaining only the final output activations. The results are gathered back to PP Stage-0;
- **Backbone Training:** This phase performs the forward and backward passes for the main transformer backbone. By discarding intermediate activations in the preceding phase, we can now fully leverage any efficient parallel strategies validated in pure text training. After this phase, gradients are accumulated at the visual encoder output;
- **Vision Recomputation & Backward:** We re-compute the vision encoder forward pass, followed by a backward pass to compute gradients for parameters in the vision encoder;

DEP not only achieves load-balance, but also decouples the optimization strategy of the vision encoder and the main backbone. K2.5 seamlessly inherits the parallel strategy of K2, achieving a multimodal training efficiency of 90% relative to text-only training. We note a concurrent work, LongCat-Flash-Omni [55], shares a similar design philosophy.

5 Evaluations

5.1 Main Results

5.1.1 Evaluation Settings

Benchmarks We evaluate Kimi K2.5 on a comprehensive benchmark suite spanning text-based reasoning, competitive and agentic coding, multimodal understanding (image and video), autonomous agentic execution, and computer use. Our benchmark taxonomy is organized along the following capability axes:

- **Reasoning & General**: Humanity's Last Exam (HLE) [46], AIME 2025 [4], HMMT 2025 (Feb) [58], IMO-AnswerBench [37], GPQA-Diamond [47], MMLU-Pro [64], SimpleQA Verified [22], AdvancedIF [23], and LongBench v2 [9].

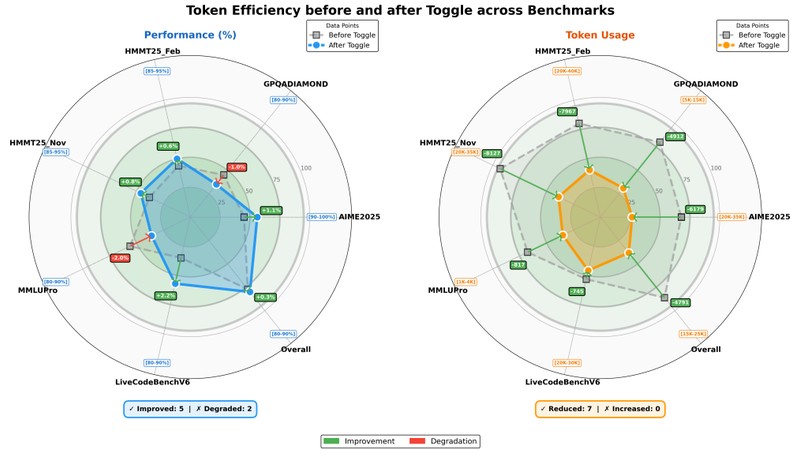


图 5：比较 Kimi K2 T 的性能和 token 使用情况 采用 token 高效强化学习进行思考。

折衷方案缓解了内存压力，但它并未从根本上解决由多模态输入大小引起的负载不平衡问题。更关键的是，它禁止了直接复用那些为纯文本训练高度优化的并行策略。

利用视觉编码器在计算图中的独特拓扑位置——具体来说，它作为前向传递的起点和反向传递的终点——我们的训练使用 **解耦编码器过程（DEP）**，该过程在每个训练步骤中由三个阶段组成：

- **平衡视觉前向：**我们首先对全局批次中的所有视觉数据进行前向传递。由于视觉编码器较小，我们将其复制到所有GPU上，无论其他并行策略如何。在此阶段，基于负载指标（例如图像或补丁数量），前向计算工作负载均匀分布在所有GPU上。这消除了由PP和视觉标记计数引起的负载不平衡。为了最小化峰值内存使用，我们丢弃所有中间激活，仅保留最终输出激活。结果被收集回PP Stage-0；
- **主干训练：**此阶段对主Transformer主干执行前向和反向传递。通过在前一阶段丢弃中间激活，我们现在可以充分利用在纯文本训练中验证的任何高效并行策略。在此阶段后，梯度在视觉编码器输出处累积；
- **视觉重新计算与反向传播：**我们重新计算视觉编码器的前向传递，然后进行反向传播以计算视觉编码器参数的梯度；

DEP 不仅实现了负载均衡，还解耦了视觉编码器优化策略和主干网络。K2.5 无缝继承了 K2 的并行策略，在多模态训练效率上比纯文本训练高 90%。我们注意到一项并发工作，LongCat-Flash-Omni [55], 采用了相似的设计理念。

5 评估

5.1 主要结果

5.1.1 评估设置

基准测试 我们在一个涵盖基于文本的推理、竞争和代理式编程、多模态理解（图像和视频）、自主代理式执行以及计算机使用的综合基准测试套件上评估了 Kimi K2.5。我们的基准测试分类沿着以下能力轴组织：

- **推理 & 综合**: 人类最后考试 (HLE) [46], AIME 2025 [4], HMMT 2025 (Feb) [58], IMO- AnswerBench [37], GPQA-Diamond [47], MMLU-Pro [64], SimpleQA Verified [22], AdvancedIF [23], 和 LongBench v2 [9].

- **Coding**: SWE-Bench Verified [29], SWE-Bench Pro (public) [16], SWE-Bench Multilingual [29], Terminal Bench 2.0 [39], PaperBench (CodeDev) [52], CyberGym [66], SciCode [56], OJBench (cpp) [65], and Live-CodeBench (v6) [28].
- **Agentic Capabilities**: BrowseComp [68], WideSearch [69],DeepSearchQA [60], FinSearchComp (T2&T3) [26], Seal-0 [45], GDPVal [43].
- **Image Understanding**: *(math & reasoning)* MMMU-Pro [76], MMMU (val) [77], CharXiv (RQ) [67], Math-Vision [61] and MathVista (mini) [36]; *(vision knowledge)* SimpleVQA [13] and WorldVQA [2]; *(perception)* ZeroBench (w/ and w/o tools) [48], BabyVision [12], BLINK [18] and MMVP [57]; *(OCR & document)* OCR-Bench [35], OmniDocBench 1.5 [42] and InfoVQA [38].
- **Video Understanding**: VideoMMMU [25], MMVU [80], MotionBench [24], Video-MME [17] (*with subtitles*), LongVideoBench [70], and LVBench [62].
- **Computer Use**: OSWorld-Verified [72, 73], and WebArena [81].

Baselines We benchmark against state-of-the-art proprietary and open-source models. For proprietary models, we compare against Claude Opus 4.5 (with extended thinking) [6], GPT-5.2 (with xhigh reasoning effort) [41], and Gemini 3 Pro (with high reasoning-level) [20]. For open-source models, we include DeepSeek-V3.2 (with thinking mode enabled) [14] for text benchmarks, while vision benchmarks report Qwen3-VL-235B-A22B-Thinking [8] instead.

Evaluation Configurations Unless otherwise specified, all Kimi K2.5 evaluations use temperature = 1.0, top-p = 0.95, and a context length of 256k tokens. Benchmarks without publicly available scores were re-evaluated under identical conditions and marked with an asterisk (*). The full evaluation settings can be found in appendix E.

5.1.2 Evaluation Results

Comprehensive results comparing Kimi K2.5 against proprietary and open-source baselines are presented in Table 4. We highlight key observations across core capability domains:

Reasoning and General Kimi K2.5 achieves competitive performance with top-tier proprietary models on rigorous STEM benchmarks. On Math tasks, AIME 2025, K2.5 scores 96.1%, approaching GPT-5.2's perfect score while outperforming Claude Opus 4.5 (92.8%) and Gemini 3 Pro (95.0%). This high-level performance extends to the HMMT 2025 (95.4%) and IMO-AnswerBench (81.8%), demonstrating K2.5's superior reasoning depth. Kimi K2.5 also exhibits remarkable knowledge and scientific reasoning capabilities, scoring 36.9% on SimpleQA Verified, 87.1% on MMLU-Pro and 87.6% on GPQA. Notably, on HLE without the use of tools, K2.5 achieves an HLE-Full score of 30.1%, with component-wise scores of 31.5% on text subset and 21.3% on image subset. When tool-use is enabled, K2.5' s HLE-Full score rises to 50.2%, with 51.8% (text) and 39.8% (image), significantly outperforming Gemini 3 Pro (45.8%) and GPT-5.2 (45.5%). In addition to reasoning and knowledge, K2.5 shows strong instruction-following performance (75.6% on AdvancedIF) and competitive long-context abilities, achieving 61.0% on LongBench v2 compared to both proprietary and open-source models.

Complex Coding and Software Engineering Kimi K2.5 exhibits strong software engineering capabilities, especially on realistic coding and maintenance tasks. It achieves 76.8% on SWE-Bench Verified and 73.0% on SWE-Bench Multilingual, outperforming Gemini 3 Pro while remaining competitive with Claude Opus 4.5 and GPT5.2. On LiveCodeBench v6, Kimi K2.5 reaches 85.0%, surpassing DeepSeekV3.2 (83.3%) and Claude Opus 4.5 (82.2%), highlighting its robustness on live, continuously updated coding challenges. On TerminalBench 2.0, PaperBench, and SciCode, it scores 50.8%, 63.5%, and 48.7% respectively, demonstrating stable competitionlevel performance in automated software engineering and problem solving across diverse domains. In addition, K2.5 attains a score of 41.3 on CyberGym, on the task of finding previously discovered vulnerabilities in real opensource software projects given only a highlevel description of the weakness, further underscoring its effectiveness in securityoriented software analysis.

Agentic Capabilities Kimi K2.5 establishes new state-of-the-art performance on complex agentic search and browsing tasks. On BrowseComp, K2.5 achieves 60.6% without context management techniques, 74.9% with Discard-all context management [14] —substantially outperforming GPT-5.2's reported 65.8%, Claude Opus 4.5 (37.0%) and Gemini 3 Pro (37.8%). Similarly, WideSearch reaches 72.7% on item-f1. On DeepSearchQA (77.1%), FinSearchCompT2&T3 (67.8%) and Seal-0 (57.4%), K2.5 leads all evaluated models, demonstrating superior capacity for agentic deep research, information synthesis, and multi-step tool orchestration.

[2]https://github.com/MoonshotAI/WorldVQA

- **编程**: SWE-Bench Verified [29], SWE-Bench Pro (public) [16], SWE-Bench 多语言 [29], 终端基准 2。0 [39], PaperBench (代码开发) [52], CyberGym [66], SciCode [56], OJBench (cpp) [65], 和 Live- 代码基准 (v6) [28]。
- **自主代理式能力**: BrowseComp [68], WideSearch [69],DeepSearchQA [60], FinSearchComp (T2&T3) [26], Seal-0 [45], GDPVal [43].
- **图像理解**: *(数学 & 推理)* MMMU-Pro [76], MMMU (val) [77], CharXiv (RQ) [67], Math- 视觉 [61] 和 MathVista (mini) [36]; *(视觉知识)* SimpleVQA [13] 和 WorldVQA [2]; *(感知)*ZeroBench (带和不带工具) [48], BabyVision [12], BLINK [18] 和 MMVP [57]; *(OCR & 文档)* OCR- Bench [35],OmniDocBench 1.5 [42] 和 InfoVQA [38].
- **视频理解**: VideoMMMU [25], MMVU [80], MotionBench [24], Video-MME [17] (带字幕), LongVideoBench [70],和 LVBench [62].
- **计算机使用**: OSWorld-Verified [72, 73],and WebArena [81].

基准测试我们与最先进的专有和开源模型进行基准测试。对于专有模型，我们比较Claude Opus 4.5（扩展思考） [6], GPT-5.2（xhigh推理努力） [41], 和Gemini 3 Pro（高推理级别） [20]。对于开源模型，我们包含DeepSeek-V3.2（启用思考模式） [14]进行文本基准测试，而视觉基准测试则报告Qwen3-VL-235B-A22B-Thinking [8] 。

评估配置 除非另有说明，所有Kimi K2.5评估使用温度 = 1.0，top-p =0.95，以及256k标记的上下文长度。没有公开可用分数的基准测试在相同条件下重新评估，并用星号(*)标记。完整的评估设置可以在附录 E.

5.1.2 评估结果

综合结果 将Kimi K2.5与专有和开源基线进行比较的结果展示在表4 .
我们强调关键观察 观测结果跨越核心能力领域:

推理和通用在严格的STEM基准测试上，Kimi K2.5与顶级专有模型实现了具有竞争力的性能。在数学任务方面，AIME 2025，K2.5得分96.1%，接近GPT-5.2的完美分数，同时优于Claude Opus 4.5（92.8%）和Gemini 3 Pro（95.0%）。这种高性能水平扩展到HMMT 2025（95.4%）和IMO-AnswerBench（81.8%），展示了K2.5卓越的推理深度。Kimi K2.5还表现出惊人的知识力和科学推理能力，在SimpleQA Verified上得分36.9%，在MMLU-Pro上得分87.1%，在GPQA上得分87.6%。值得注意的是，在未使用工具的HLE测试中，K2.5的HLE-Full得分为30.1%，其中文本子集得分为31.5%，图像子集得分为21.3%。当启用工具使用时，K2.5的HLE-Full得分上升到50.2%，其中文本得分为51.8%，图像得分为39.8%，显著优于Gemini 3 Pro（45.8%）和GPT-5.2（45.5%）。除了推理和知识，K2.5显示出强大的指令跟随性能（在AdvancedIF上得分75.6%）和具有竞争力的长上下文能力，在LongBench v2上得分61.0%，与专有和开源模型相比都具有竞争力。

复杂的编程和软件工程Kimi K2.5展现出强大的软件工程能力，特别是在真实的编程和维护任务上。它在SWE-Bench Verified上达到76.8%，在SWE-Bench Multilingual上达到73.0%，优于Gemini 3 Pro，同时与Claude Opus 4.5和GPT5.2保持竞争力。在LiveCodeBench v6上，Kimi K2.5达到85.0%，超过DeepSeekV3.2（83.3%）和Claude Opus 4.5（82.2%），突显其在实时、持续更新的编程挑战中的稳健性。在TerminalBench 2.0、PaperBench和SciCode上，它分别得分50.8%、63.5%和48.7%，在自动化软件工程和跨不同领域的解决问题方面展现出稳定的竞争水平。此外，K2.5在CyberGym上获得41.3分，在仅给出弱点的高层次描述的情况下，寻找真实开源软件项目中先前发现的漏洞的任务上，进一步强调了其在安全导向软件分析方面的有效性。

自主代理式能力 Kimi K2.5 在复杂的代理式搜索和浏览任务上建立了新的顶尖性能。在 BrowseComp 上，K2.5 在没有上下文管理技术的情况下达到 60.6%，在使用 Discard-all 上下文管理 [14] 的情况下达到 74.9%—大幅超越了 GPT-5.2 报告的 65.8%、Claude Opus 4.5 (37.0%) 和 Gemini 3 Pro (37.8%)。同样，WideSearch 在 item-f1 上达到 72.7%。在 DeepSearchQA (77.1%)、FinSearch-CompT2&T3 (67.8%) 和 Seal-0 (57.4%) 上，K2.5 领先所有评估模型，展示了其在代理式深度研究、信息合成和多步工具编排方面的卓越能力。

[2]https://github.com/MoonshotAI/WorldVQA

Table 4: Performance comparison of Kimi K2.5 against open-source and proprietary models. **Bold** denotes the global SOTA; Data points marked with * are taken from our internal evaluations. † refers to their scores of text-only subset.

Benchmark	Kimi K2.5	Proprietary			Open Source	
		Claude Opus 4.5	GPT-5.2 (xhigh)	Gemini 3 Pro	DeepSeek-V3.2	Qwen3-VL-235B-A22B
Reasoning & General						
HLE-Full	30.1	30.8	34.5	**37.5**	25.1†	-
HLE-Full w/ tools	**50.2**	43.2	45.5	45.8	40.8†	-
AIME 2025	96.1	92.8	**100**	95.0	93.1	-
HMMT 2025 (Feb)	95.4	92.9*	**99.4**	97.3*	92.5	-
IMO-AnswerBench	81.8	78.5*	**86.3**	83.1*	78.3	-
GPQA-Diamond	87.6	87.0	**92.4**	91.9	82.4	-
MMLU-Pro	87.1	89.3*	86.7*	**90.1**	85.0	-
SimpleQA Verified	36.9	44.1	38.9	**72.1**	27.5	-
AdvancedIF	75.6	63.1	**81.1**	74.7	58.8	-
LongBench v2	61.0	64.4*	54.5*	**68.2***	59.8*	-
Coding						
SWE-Bench Verified	76.8	**80.9**	80.0	76.2	73.1	-
SWE-Bench Pro (public)	50.7	55.4*	**55.6**	-	-	-
SWE-Bench Multilingual	73.0	**77.5**	72.0	65.0	70.2	-
Terminal Bench 2.0	50.8	**59.3**	54.0	54.2	46.4	-
PaperBench (CodeDev)	63.5	**72.9***	63.7*	-	47.1	-
CyberGym	41.3	**50.6**	-	39.9*	17.3*	-
SciCode	48.7	49.5	52.1	**56.1**	38.9	-
OJBench (cpp)	57.4	54.6*	-	**68.5***	54.7*	-
LiveCodeBench (v6)	85.0	82.2*	-	**87.4***	83.3	-
Agentic						
BrowseComp	**60.6**	37.0	65.8	37.8	51.4	-
BrowseComp (w/ ctx manage)	**74.9**	57.8		59.2	67.6	-
BrowseComp (Agent Swarm)	**78.4**	-	-	-	-	-
WideSearch	72.7	**76.2***	-	57.0	32.5*	-
WideSearch (Agent Swarm)	**79.0**	-	-	-	-	-
DeepSearchQA	**77.1**	76.1*	71.3*	63.2*	60.9*	-
FinSearchCompT2&T3	**67.8**	66.2*	-	49.9	59.1*	-
Seal-0	**57.4**	47.7*	45.0	45.5*	49.5*	-
GDPVal-AA	41.0	45.0	**48.0**	35.0	34.0	-
Image						
MMMU-Pro	78.5	74.0	79.5*	**81.0**	-	69.3
MMMU (val)	84.3	80.7	86.7*	**87.5***	-	80.6
CharXiv (RQ)	77.5	67.2*	**82.1**	81.4	-	66.1
MathVision	84.2	77.1*	83.0	**86.1***	-	74.6
MathVista (mini)	**90.1**	80.2*	82.8*	89.8*	-	85.8
SimpleVQA	**71.2**	69.7*	55.8*	69.7*	-	56.8*
WorldVQA	46.3	36.8	28.0	**47.4**	-	23.5
ZeroBench	**9**	3*	**9***	8*	-	4*
ZeroBench w/ tools	11	9*	7*	**12***	-	3*
BabyVision	36.5	14.2	34.4	**49.7**	-	22.2
BLINK	**78.9**	68.8*	-	78.7*	-	68.9
MMVP	87.0	80.0*	83.0*	**90.0***	-	84.3
OmniDocBench 1.5	**88.8**	87.7*	85.7	88.5	-	82.0*
OCRBench	**92.3**	86.5*	80.7*	90.3*	-	87.5
InfoVQA (test)	**92.6**	76.9*	84*	57.2*	-	89.5
Video						
VideoMMMU	86.6	84.4*	85.9	**87.6**	-	80.0
MMVU	80.4	77.3*	**80.8***	77.5*	-	71.1
MotionBench	**70.4**	60.3*	64.8*	70.3	-	-
Video-MME	87.4	77.6*	86.0*	**88.4***	-	79.0
LongVideoBench	**79.8**	67.2*	76.5*	77.7*	-	65.6*
LVBench	**75.9**	57.3	-	73.5*	-	63.6
Computer Use						
OSWorld-Verified	63.3	**66.3**	8.6*	20.7*	-	38.1
WebArena	58.9	**63.4***	-	-	-	26.4*

表4：Kimi K2.5与开源和专有模型的性能对比。 **加粗**表示全局SOTA；带*的数据点来自我们的内部评估。† 指其仅文本子集的得分。

基准测试	Kimi K2.5	专有			开源	
		Claude Opus 4.5	GPT-5.2 (xhigh)	Gemini 3 Pro	DeepSeek-V3.2	Qwen3-VL-235B-A22B
推理 & 一般						
HLE-Full	30.1	30.8	34.5	**37.5**	25.1†	-
HLE-Full w/ 工具	**50.2**	43.2	45.5	45.8	40.8†	-
AIME 2025	96.1	92.8	**100**	95.0	93.1	-
HMMT 2025 (Feb)	95.4	92.9*	**99.4**	97.3*	92.5	-
IMO-AnswerBench	81.8	78.5*	**86.3**	83.1*	78.3	-
GPQA-Diamond	87.6	87.0	**92.4**	91.9	82.4	-
MMLU-Pro	87.1	89.3*	86.7*	**90.1**	85.0	-
SimpleQA Verified	36.9	44.1	38.9	**72.1**	27.5	-
AdvancedIF	75.6	63.1	**81.1**	74.7	58.8	-
LongBench v2	61.0	64.4*	54.5*	**68.2***	59.8*	-
编程						
SWE-Bench Verified	76.8	**80.9**	80.0	76.2	73.1	-
SWE-Bench Pro (public)	50.7	55.4*	**55.6**	-	-	-
SWE-Bench 多语言	73.0	**77.5**	72.0	65.0	70.2	-
Terminal Bench 2.0	50.8	**59.3**	54.0	54.2	46.4	-
PaperBench (代码开发)	63.5	**72.9***	63.7*	-	47.1	-
CyberGym	41.3	**50.6**	-	39.9*	17.3*	-
SciCode	48.7	49.5	52.1	**56.1**	38.9	-
OJBench (cpp)	57.4	54.6*	-	**68.5***	54.7*	-
LiveCodeBench (v6)	85.0	82.2*	-	**87.4***	83.3	-
自主代理式						
BrowseComp	**60.6**	37.0	65.8	37.8	51.4	-
BrowseComp (带上下文管理)	**74.9**	57.8		59.2	67.6	-
BrowseComp (Agent Swarm)	**78.4**	-	-	-	-	-
WideSearch	72.7	**76.2***	-	57.0	32.5*	-
WideSearch (Agent Swarm)	**79.0**	-	-	-	-	-
DeepSearchQA	**77.1**	76.1*	71.3*	63.2*	60.9*	-
FinSearchCompT2&T3	**67.8**	66.2*	-	49.9	59.1*	-
Seal-0	**57.4**	47.7*	45.0	45.5*	49.5*	-
GDPval-AA	41.0	45.0	**48.0**	35.0	34.0	-
图片						
MMMU-Pro	78.5	74.0	79.5*	**81.0**	-	69.3
MMMU (val)	84.3	80.7	86.7*	**87.5***	-	80.6
CharXiv (RQ)	77.5	67.2*	**82.1**	81.4	-	66.1
MathVision	84.2	77.1*	83.0	**86.1***	-	74.6
MathVista (mini)	**90.1**	80.2*	82.8*	89.8*	-	85.8
SimpleVQA	**71.2**	69.7*	55.8*	69.7*	-	56.8*
WorldVQA	46.3	36.8	28.0	**47.4**	-	23.5
ZeroBench	**9**	3*	**9***	8*	-	4*
ZeroBench w/ 工具	11	9*	7*	**12***	-	3*
BabyVision	36.5	14.2	34.4	**49.7**	-	22.2
BLINK	**78.9**	68.8*	-	78.7*	-	68.9
MMVP	87.0	80.0*	83.0*	**90.0***	-	84.3
OmniDocBench 1.5	**88.8**	87.7*	85.7	88.5	-	82.0*
OCRBench	**92.3**	86.5*	80.7*	90.3*	-	87.5
InfoVQA (测试)	**92.6**	76.9*	84*	57.2*	-	89.5
视频						
VideoMMMU	86.6	84.4*	85.9	**87.6**	-	80.0
MMVU	80.4	77.3*	**80.8***	77.5*	-	71.1
MotionBench	**70.4**	60.3*	64.8*	70.3	-	-
Video-MME	87.4	77.6*	86.0*	**88.4***	-	79.0
长视频基准测试	**79.8**	67.2*	76.5*	77.7*	-	65.6*
LVBench	**75.9**	57.3	-	73.5*	-	63.6
计算机使用						
OSWorld-Verified	63.3	**66.3**	8.6*	20.7*	-	38.1
WebArena	58.9	**63.4***	-	-	-	26.4*

Table 5: Performance and token efficiency of some reasoning models. Average output token counts (in thousands) are shown in parentheses.

Benchmark	Kimi K2.5	Kimi K2 Thinking	Gemini-3.0 Pro	DeepSeek-V3.2 Thinking
AIME 2025	96.1 (25k)	94.5 (30k)	95.0 (15k)	93.1 (16k)
HMMT Feb 2025	95.4 (27k)	89.4 (35k)	97.3 (16k)	92.5 (19k)
HMMT Nov 2025	91.1 (24k)	89.2 (32k)	94.5 (15k)	90.2 (18k)
IMO-AnswerBench	81.8 (36k)	78.6 (37k)	83.1 (18k)	78.3 (27k)
LiveCodeBench	85.0 (18k)	82.6 (25k)	87.4 (13k)	83.3 (16k)
GPQA Diamond	87.6 (14k)	84.5 (13k)	91.9 (8k)	82.4 (7k)
HLE-Text	31.5 (24k)	23.9 (29k)	38.4 (13k)	25.1 (21k)

Vision Reasoning, Knowledge and Perception Kimi K2.5 demonstrates strong visual reasoning and world knowledge capabilities. It scores 78.5% on MMMU-Pro, spanning multi-disciplinary multimodal tasks. For world knowledge question answering, K2.5 achieves 71.2% on SimpleVQA and 46.3% on WorldVQA. For visual reasoning, it achieves 84.2% on MathVision, 90.1% on MathVista (mini), and 36.5% on BabyVision. For OCR and document understanding, K2.5 delivers outstanding results with 77.5% on CharXiv (RQ), 92.3% on OCRBench, 88.8% on OmniDocBench 1.5, and 92.6% on InfoVQA (test). On the challenging ZeroBench, Kimi K2.5 achieves 9% and 11% with tool augmentation, substantially ahead of competing models. On basic visual perception benchmarks BLINK (78.9%) and MMVP (87.0%), we also observe competitive performance of Kimi K2.5, demonstrating its robust real-world visual perceptions.

Video Understanding Kimi K2.5 achieves state-of-the-art performance across diverse video understanding tasks. It attains 86.6% on VideoMMMU and 80.4% on MMVU, rivaling frontier leaderships. With the context-compression and dense temporal understanding abilities of MoonViT-3D, Kimi K2.5 also establishes new global SOTA records in long-video comprehension with 75.9% on LVBench and 79.8% on LongVideoBench by feeding over 2,000 frames, while demonstrating robust dense-motion understanding at 70.4% on the highly-dimensional MotionBench.

Computer-Use Capability Kimi K2.5 demonstrates state-of-the-art computer-use capability on real-world tasks. On the computer-use benchmark OSWorld-Verified [72, 73], it achieves a 63.3% success rate relying solely on GUI actions without external tools. This substantially outperforms open-source models such as Qwen3-VL-235B-A22B (38.1%) and OpenAI's computer-use agent framework Operator (o3-based) (42.9%), while remaining competitive with the current leading CUA model, Claude Opus 4.5 (66.3%). On WebArena [81], an established benchmark for GUI-based web browsing, Kimi K2.5 achieves a 58.9% success rate, surpassing OpenAI's Operator (58.1%) and approaching the performance of Claude Opus 4.5 (63.4%).

5.2 Agent Swarm Results

Benchmarks To rigorously evaluate the effectiveness of the agent swarm framework, we select three representative benchmarks that collectively cover deep reasoning, large-scale retrieval, and real-world complexity:

- **BrowseComp**: A challenging deep-research benchmark that requires multi-step reasoning and complex information synthesis.
- **WideSearch**: A benchmark designed to evaluate the ability to perform broad, multi-step information seeking and reasoning across diverse sources.
- **In-house Swarm Bench**: An internally developed Swarm benchmark, designed to evaluate the agent swarm performance under real-world, high-complexity conditions. It covers four domains: *WildSearch* (unconstrained, real-world information retrieval over the open web), *Batch Download* (large-scale acquisition of diverse resources), *WideRead* (large-scale document comprehension involving more than 100 input documents), and *Long-Form Writing* (coherent generation of extensive content exceeding 100k words). This benchmark incorporates extreme-scale scenarios that stress-test the orchestration, scalability, and coordination capabilities of agent-based systems.

Performance Table 6 presents the performance of Kimi K2.5 Agent Swarm against single-agent configurations and proprietary baselines. The results demonstrate substantial performance improvements from multi-agent orchestration. On BrowseComp, Agent Swarm achieves 78.4%, representing a 17.8% absolute gain over the single-agent K2.5

表5：一些推理模型的性能和token效率。平均输出token数（以千为单位）显示在括号中。

基准测试	Kimi K2.5	Kimi K2 思考	Gemini-3.0 Pro	DeepSeek-V3.2 思考
AIME 2025	96.1 (25k)	94.5 (30k)	95.0 (15k)	93.1 (16k)
HMMT 2025年2月	95.4 (27k)	89.4 (35k)	97.3 (16k)	92.5 (19k)
HMMT 2025年11月	91.1 (24k)	89.2 (32k)	94.5 (15k)	90.2 (18k)
IMO-AnswerBench	81.8 (36k)	78.6 (37k)	83.1 (18k)	78.3 (27k)
LiveCodeBench	85.0 (18k)	82.6 (25k)	87.4 (13k)	83.3 (16k)
GPQA Diamond	87.6 (14k)	84.5 (13k)	91.9 (8k)	82.4 (7k)
HLE-文本	31.5 (24k)	23.9 (29k)	38.4 (13k)	25.1 (21k)

视觉推理、知识感知 Kimi K2.5展现出强大的视觉推理和世界知识能力。在MMMU-Pro上得分78.5%，涵盖跨学科多模态任务。在知识问答方面，K2.5在SimpleVQA上达到71.2%，在WorldVQA上达到46.3%。在视觉推理方面，它在MathVision上得分84.2%，在MathVista（mini）上得分90.1%，在BabyVision上得分36.5%。在OCR和文档理解方面，K2.5表现优异，在CharXiv（RQ）上得分77.5%，在OCRBench上得分92.3%，在OmniDocBench 1.5上得分88.8%，在InfoVQA（测试）上得分92.6%。在具有挑战性的ZeroBench上，Kimi K2.5在工具增强下分别达到9%和11%，显著领先于竞品模型。在基础视觉感知基准BLINK（78.9%）和MMVP（87.0%）上，我们也观察到Kimi K2.5具有竞争力的表现，证明其强大的现实世界视觉感知能力。

视频理解Kimi K2.5 在多种视频理解任务上实现了最先进的性能。它在 VideoMMMU 上达到 86.6%，在 MMVU 上达到 80.4%，与前沿领先水平相当。凭借 MoonViT-3D 的上下文压缩和密集时序理解能力，Kimi K2.5 在长视频理解方面也建立了新的全球 SOTA 记录，通过处理超过 2,000 帧，在 LVBench 上达到 75.9%，在 LongVideoBench 上达到 79.8%，同时在高度维度的 MotionBench 上展现出 70.4% 的稳健密集运动理解能力。

计算机使用能力 Kimi K2.5 在实际任务上展示了最先进的计算机使用能力。在计算机使用基准测试 OSWorld-Verified [72, 73], 它仅依靠 GUI 操作而无需外部工具即可实现 63.3% 的成功率。这显著优于开源模型 Qwen3-VL-235B-A22B (38.1%) 和 OpenAI' 的计算机使用代理框架 Operator (o3-based) (42.9%)，同时与当前领先的 CUA 模型 Claude Opus 4.5 (66.3%) 保持竞争力。在 WebArena [81], 一个用于基于 GUI 的网络浏览的成熟基准测试中，Kimi K2.5 实现了 58.9% 的成功率，超越了 OpenAI 的 Operator (58.1%)，并接近 Claude Opus 4.5 (63.4%) 的性能。

5.2 Agent Swarm 结果

基准测试 为了严格评估代理群集框架的有效性，我们选择了三个具有代表性的基准测试，它们共同覆盖了深度推理、大规模检索和现实世界复杂性：

- **BrowseComp**：一个具有挑战性的深度研究基准，需要多步推理和复杂的信息综合。
- **WideSearch**：一个旨在评估执行广泛、多步信息查找和推理能力的基准，这些推理跨越不同的来源。
- **内部群集基准**：一个内部开发的群集基准，旨在评估代理群集在现实世界、高复杂性条件下的性能。它涵盖四个领域：*WildSearch*（无约束的、现实世界的信息检索，跨越开放网络）、批量下载（大规模获取多样化资源）、*WideRead*（大规模文档理解，涉及超过100个输入文档），以及 长篇写作（生成超过100k字的有条理的扩展内容）。此基准包含了极端规模的场景，这些场景对基于代理的系统的编排、可扩展性和协调能力进行了压力测试。

性能表格6 展示了Kimi K2.5智能体群与单智能体配置和专有基线相比的性能。结果表明，多智能体编排带来了显著的性能提升。在BrowseComp上，智能体群达到了78.4%，比单智能体K2.5绝对提升了17.8%。

Table 6: Performance comparison of Kimi K2.5 Agent Swarm against single-agent and proprietary baselines on agentic search benchmarks. **Bold** denotes the best result per benchmark.

Benchmark	K2.5 Agent Swarm	Kimi K2.5	Claude Opus 4.5	GPT-5.2	GPT-5.2 Pro
BrowseComp	**78.4**	60.6	37.0	65.8	77.9
WideSearch	**79.0**	72.7	76.2	-	-
In-house Swarm Bench	**58.3**	41.6	45.8	-	-

Figure 6: The word cloud visualizes heterogeneous K2.5-based sub-agents dynamically instantiated by the Orchestrator across tests.

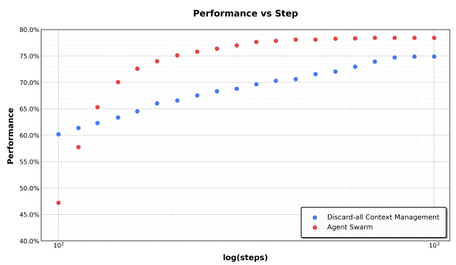


Figure 7: Comparison of Kimi K2.5 performance under Agent Swarm and Discard-all context management in BrowseComp.

(60.6%) and surpassing even GPT-5.2 Pro (77.9%). Similarly, WideSearch sees a 6.3% improvement (72.7% → 79.0%) on Item-F1, enabling K2.5 Agent Swarm to outperform Claude Opus 4.5 (76.2%) and establish a new state-of-the-art. The gains are most pronounced on In-house Swarm bench (16.7%), where tasks are explicitly designed to reward parallel decomposition. These consistent improvements across benchmarks validate that Agent Swarm effectively converts computational parallelism into qualitative capability gains, particularly for problems requiring broad exploration, multi-source verification, or simultaneous handling of independent sub-tasks.

Execution Time Savings via Parallelism Beyond improved task performance, Agent Swarm achieves substantial wall-clock time reductions through parallel subagent execution. On the WideSearch benchmark, it reduces the execution time required to reach target performance by $3\times \sim 4.5\times$ compared to a single-agent baseline. As shown in Figure 8, this efficiency gain scales with task complexity: as the target Item-F1 increases from 30% to 70%, the single agent's execution time grows from approximately $1.8\times$ to over $7.0\times$ the baseline, whereas Agent Swarm maintains near-constant low latency in the range of $0.6\times \sim 1.6\times$. These results indicate that Agent Swarm effectively transforms sequential tool invocations into parallel operations, preventing the linear growth in completion time typically observed as task difficulty increases.

Dynamic Subagent Creation and Scheduling Within an agent swarm, subagents are dynamically instantiated rather than pre-defined. Through PARL, the orchestrator learns adaptive policies to create and schedule self-hosted subagents in response to evolving task structures and problem states. Unlike static decomposition approaches, this learned policy enables the Orchestrator to reason about the requisite number, timing, and specialization of subagents based on query. Consequently, a heterogeneous agent group emerges organically from this adaptive allocation strategy (Figure 6).

Agent Swarm as Proactive Context Management Beyond better performance and runtime acceleration, an agent swarm is a kind of proactive and intelligent context management enabled by multi-agent architecture [5]. This approach differs from test-time context truncation strategies such as Hide-Tool-Result [2], Summary [71], or Discard-all [14], which react to context overflow by compressing or discarding accumulated histories. While effective at reducing token usage, these methods are inherently reactive and often sacrifice structural information or intermediate reasoning.

In contrast, Agent Swarm enables proactive context control through explicit orchestration. Long-horizon tasks are decomposed into parallel, semantically isolated subtasks, each executed by a specialized subagent with a bounded local context. Crucially, these subagents maintain independent working memories and perform local reasoning without directly mutating or contaminating the global context of the central orchestrator. Only task-relevant outputs—rather than full interaction traces—are selectively routed back to the orchestrator. This design induces context sharding

表6：Kimi K2.5智能体群与单智能体和专有基线在代理式搜索基准测试上的性能比较。**加粗**表示每个基准测试的最佳结果。

基准测试	K2.5智能体群	Kimi K2.5	Claude Opus 4.5	GPT-5.2	GPT-5.2 Pro
BrowseComp	**78.4**	60.6	37.0	65.8	77.9
WideSearch	**79.0**	72.7	76.2	-	-
内部群集基准	**58.3**	41.6	45.8	-	-

图6：词云可视化展示了编排器在测试中动态实例化的异构K2.5子代理。

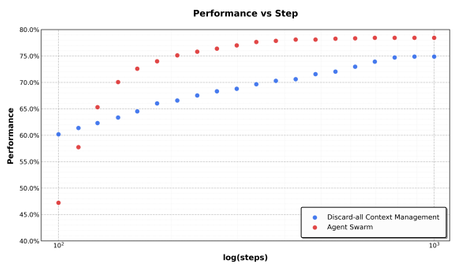


图7：比较Kimi K2.5在Agent Swarm和Discard-all上下文管理中的性能表现（在BrowseComp中）。

(60.6%)，甚至超过了GPT-5.2 Pro (77.9%)。同样，在WideSearch上，Item-F1指标提升了6.3% (72.7% → 79.0%)，使K2.5智能体群能够超越Claude Opus 4.5 (76.2%) 并建立新的顶尖水平。这些提升在内部群集基准 (16.7%) 上最为明显，该基准的任务明确设计为奖励并行分解。这些跨基准的持续改进验证了智能体群能够有效将计算并行性转化为定性能力提升，尤其适用于需要广泛探索、多源验证或同时处理独立子任务的问题。

通过并行性实现的执行时间节省除了提升任务性能，Agent Swarm通过并行子代理执行显著减少了实际运行时间。在WideSearch基准测试中，它将实现目标性能所需的执行时间减少了 $3\times \sim 4$。$5\times$ 与单代理基线相比。如图8所示，这种效率提升随着任务复杂度的增加而扩展：当目标Item-F1从30%增加到70%时，单代理的执行时间从大约$1.8\times$ 增长到超过$7.0\times$ ，而基线则显著增加，而Agent Swarm在 $0.6\times \sim 1.6\times$范围内维持近乎恒定的低延迟。这些结果表明，Agent Swarm有效地将顺序工具调用转换为并行操作，防止了随着任务难度增加而通常观察到的完成时间的线性增长。

动态子代理创建与调度 在代理群中，子代理是动态实例化的，而不是预定义的。通过 PARL，编排器学习自适应策略，以根据不断变化的任务结构和问题状态来创建和调度自托管子代理。与静态分解方法不同，这种学习到的策略使编排器能够根据查询推理子代理的必要数量、时间和专业化。因此，这种自适应分配策略有机地形成了一个异构代理组（图6）。

代理群作为主动式上下文管理 除了更好的性能和运行时加速外，代理群是一种由多代理架构启用的主动式智能上下文管理。这种方法不同于测试时上下文截断策略，如隐藏工具结果 [2]，摘要 [71]，或丢弃全部[14]，，这些策略通过压缩或丢弃累积的历史记录来应对上下文溢出。虽然这些方法在减少标记使用方面很有效，但它们本质上具有反应性，并且经常牺牲结构信息或中间推理。

相比之下，Agent Swarm 通过显式编排实现主动式上下文控制。长期任务被分解为并行、语义隔离的子任务，每个子任务由具备有限本地上下文的专用子代理执行。关键在于，这些子代理保持独立的工 作记忆，并在不直接修改或污染中央编排器全局上下文的情况下进行本地推理。只有与任务相关的输出——而非完整交互轨迹——会被选择性地路由回编排器。这种设计促使上下文分片

Figure 8: Agent Swarm achieves 3×–4.5× faster execution time compared to single-agent baselines as target Item-F1 increases from 30% to 70% in WideSearch testing.

rather than context truncation, allowing the system to scale effective context length along an additional architectural dimension while preserving modularity, information locality, and reasoning integrity.

As shown in Figure 7, this proactive strategy outperforms Discard-all in both efficiency and accuracy on BrowseComp. By preserving task-level coherence at the orchestrator level while keeping subagent contexts tightly bounded, Agent Swarm enables parallel execution with selective context persistence, retaining only high-level coordination signals or essential intermediate results. Consequently, Agent Swarm operates as an active, structured context manager, achieving higher accuracy with substantially fewer critical steps than uniform context truncation.

6 Conclusions

Kimi K2.5 shows that scalable and general agentic intelligence can be achieved through joint optimization of text and vision together with parallel agent execution. By unifying language and vision across pre-training and reinforcement learning, the model achieves strong cross-modal alignment and visual–text reasoning. Agent Swarm enables concurrent execution of heterogeneous sub-tasks, reducing inference latency while improving performance on complex agentic workloads. Grounded in vision–text intelligence and agent swarms, Kimi K2.5 demonstrates strong performance on benchmarks and real-world tasks. By open-sourcing the post-trained checkpoints, we aim to support the open-source community in building scalable and general-purpose agentic systems and to accelerate progress toward General Agentic Intelligence.

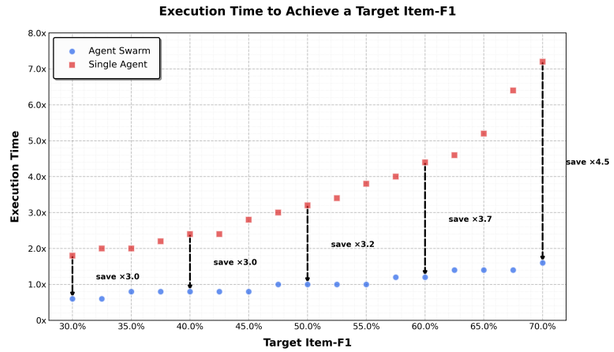


图8：在WideSearch测试中，随着目标Item-F1从30%增加到70%，Agent Swarm相比单代理基线实现了 3×–4.5× 的更快执行时间。

而非上下文截断，使系统能够在增加额外架构维度的情况下有效扩展上下文长度，同时保持模块化、信息局部性和推理完整性。

如图7所示，这种主动策略在BrowseComp上同时优于Discard-all的效率和准确性。通过在编排器级别保持任务级连贯性，同时将子代理上下文紧密限制，Agent Swarm实现了并行执行与选择性上下文持久化，仅保留高级协调信号或关键中间结果。因此，Agent Swarm作为一个主动、结构化的上下文管理器运行，在显著减少关键步骤的同时实现了更高的准确性。

6 结论

Kimi K2.5 展示了通过文本与视觉的联合优化以及并行智能体执行，可以实现可扩展的通用智能体智能。通过在预训练和强化学习过程中统一语言和视觉，该模型实现了强大的跨模态对齐和视觉-文本推理。智能体群能够并发执行异构子任务，在降低推理延迟的同时，提升了复杂智能体工作负载的性能。基于视觉-文本智能和智能体群，Kimi K2.5 在基准测试和实际任务中表现出色。通过开源预训练后的检查点，我们旨在支持开源社区构建可扩展的通用智能体系统，并加速通用智能体智能的进展。

References

[1] Moonshot AI. *Introducing Kimi K2 Thinking*. 2025. URL: https://moonshotai.github.io/Kimi-K2/thinking.html.

[2] Moonshot AI. *Kimi-Researcher End-to-End RL Training for Emerging Agentic Capabilities*. 2025. URL: https://moonshotai.github.io/Kimi-Researcher/.

[3] Amazon Web Services. *Amazon Simple Storage Service (Amazon S3)*. Web. Available at: https://aws.amazon.com/s3/. 2023. URL: https://aws.amazon.com/s3/ (visited on 12/15/2023).

[4] Mathematical Association of America. *2025 American Invitational Mathematics Examination I*. Held on February 6, 2025. 2025. URL: https://artofproblemsolving.com/wiki/index.php/2025_AIME_I.

[5] Anthropic. *Building multi-agent systems: when and how to use them*. 2026. URL: https://claude.com/blog/building-multi-agent-systems-when-and-how-to-use-them.

[6] Anthropic. *Claude Opus 4.5 System Card*. 2025. URL: https://www-cdn.anthropic.com/bf10f64990cfda0ba858290be7b8cc6317685f47.pdf.

[7] Anthropic. *How we built our multi-agent research system*. 2025. URL: https://www.anthropic.com/engineering/multi-agent-research-system.

[8] Shuai Bai et al. *Qwen3-VL Technical Report*. 2025. arXiv: 2511.21631 [cs.CV]. URL: https://arxiv.org/abs/2511.21631.

[9] Yushi Bai et al. *LongBench v2: Towards Deeper Understanding and Reasoning on Realistic Long-context Multitasks*. 2025. arXiv: 2412.15204 [cs.CL]. URL: https://arxiv.org/abs/2412.15204.

[10] Greg Brockman et al. *OpenAI Gym*. 2016. arXiv: 1606.01540 [cs.LG]. URL: https://arxiv.org/abs/1606.01540.

[11] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL]. URL: https://arxiv.org/abs/2005.14165.

[12] Liang Chen et al. *BabyVision: Visual Reasoning Beyond Language*. 2026. arXiv: 2601.06521 [cs.CV]. URL: https://arxiv.org/abs/2601.06521.

[13] Xianfu Cheng et al. *SimpleVQA: Multimodal Factuality Evaluation for Multimodal Large Language Models*. 2025. arXiv: 2502.13059 [cs.CL]. URL: https://arxiv.org/abs/2502.13059.

[14] DeepSeek-AI et al. *DeepSeek-V3.2: Pushing the Frontier of Open Large Language Models*. 2025. arXiv: 2512.02556 [cs.CL]. URL: https://arxiv.org/abs/2512.02556.

[15] Mostafa Dehghani et al. *Patch n' Pack: NaViT, a Vision Transformer for any Aspect Ratio and Resolution*. 2023. arXiv: 2307.06304 [cs.CV]. URL: https://arxiv.org/abs/2307.06304.

[16] Xiang Deng et al. "SWE-Bench Pro: Can AI Agents Solve Long-Horizon Software Engineering Tasks?" In: *arXiv preprint arXiv:2509.16941* (2025).

[17] Chaoyou Fu et al. *Video-MME: The First-Ever Comprehensive Evaluation Benchmark of Multi-modal LLMs in Video Analysis*. 2025. arXiv: 2405.21075 [cs.CV]. URL: https://arxiv.org/abs/2405.21075.

[18] Xingyu Fu et al. *BLINK: Multimodal Large Language Models Can See but Not Perceive*. 2024. arXiv: 2404.12390 [cs.CV]. URL: https://arxiv.org/abs/2404.12390.

[19] Samir Yitzhak Gadre et al. "Datacomp: In search of the next generation of multimodal datasets". In: *Advances in Neural Information Processing Systems* 36 (2024).

[20] Google. *Gemini 3 Pro*. 2025. URL: https://deepmind.google/models/gemini/pro/.

[21] Dong Guo et al. *Seed1.5-VL Technical Report*. 2025. arXiv: 2505.07062 [cs.CV]. URL: https://arxiv.org/abs/2505.07062.

[22] Lukas Haas et al. *SimpleQA Verified: A Reliable Factuality Benchmark to Measure Parametric Knowledge*. 2025. arXiv: 2509.07968 [cs.CL]. URL: https://arxiv.org/abs/2509.07968.

[23] Yun He et al. *AdvancedIF: Rubric-Based Benchmarking and Reinforcement Learning for Advancing LLM Instruction Following*. 2025. arXiv: 2511.10507 [cs.CL]. URL: https://arxiv.org/abs/2511.10507.

[24] Wenyi Hong et al. *MotionBench: Benchmarking and Improving Fine-grained Video Motion Understanding for Vision Language Models*. 2025. arXiv: 2501.02955 [cs.CV]. URL: https://arxiv.org/abs/2501.02955.

[25] Kairui Hu et al. *Video-MMMU: Evaluating Knowledge Acquisition from Multi-Discipline Professional Videos*. 2025. arXiv: 2501.13826 [cs.CV]. URL: https://arxiv.org/abs/2501.13826.

参考文献

[1] *Moonshot AI.* 介绍 *Kimi K2* 思维。*2025*。*URL: https://moonshotai.github.io/Kimi-K2/thinking.html.*[2] *Moonshot AI. Kimi-Researcher* 端到端强化学习训练，用于新兴智能体能力。*2025*。*URL: https://moonshotai.github.io/Kimi-Researcher/.*[3] 亚马逊网络服务。亚马逊简单存储服务（亚马逊 *S3*）。网络。可在 *https://aws.amazon.com/s3/* 获取。*2023*。*URL: https://aws.amazon.com/s3/* (于 *2023* 年 *12* 月 *15* 日访问)。[4] 美国数学协会。*2025* 年美国邀请数学考试 *I*。于 *2025* 年 *2* 月 *6* 日举行。*2025*。*URL: https://artofproblemsolving.com/wiki/index.php/2025_AIME_I.*[5] *Anthropic*。构建多代理系统：何时以及如何使用它们。*2026*。*URL: https://claude.com/ blog/building-multi-agent-systems-when-and-how-to-use-them.*[6] *Anthropic*。*Claude Opus 4.5* 系统卡。*2025*。*URL: https://www-cdn.anthropic.com/bf10f64990cfda0ba858290be7b8cc6317685f47.pdf.*[7] *Anthropic*。我们如何构建我们的多代理研究系统。*2025*。*URL: https://www.anthropic.com/ engineering/multi-agent-research-system.*[8] *Shuai Bai* 等人。*Qwen3-VL* 技术报告。*2025*。*arXiv: 2511.21631 [cs.CV]*。*URL: https://arxiv.org/abs/2511.21631.*[9] *Yushi Bai* 等人。*LongBench v2*：迈向更深入理解现实长文本多任务推理。*2025*。*arXiv: 2412.15204 [cs.CL]*。*URL: https://arxiv.org/abs/2412.15204.*[10] *Greg Brockman* 等人。*OpenAI Gym*。*2016*。*arXiv: 1606.01540 [cs.LG]*。*URL: https://arxiv.org/abs/1606.01540.*[11] *Tom B. Brown* 等人。语言模型是少样本学习者。*2020*。*arXiv: 2005.14165 [cs.CL]*。*URL: https://arxiv.org/abs/2005.14165.*[12] *Liang Chen* 等人。*BabyVision*：超越语言的视觉推理。*2026*。*arXiv: 2601.06521 [cs.CV]*。*URL: https://arxiv.org/abs/2601.06521.*[13] *Xianfu Cheng* 等人。*SimpleVQA*：用于多模态大型语言模型的多模态事实性评估。*2025*。*arXiv: 2502.13059 [cs.CL]*。*URL: https://arxiv.org/abs/2502.13059.*[14] *DeepSeek-AI* 等人。*DeepSeek-V3.2*：推动开放大型语言模型的边界。*2025*。*arXiv: 2512.02556 [cs.CL]*。*URL: https://arxiv.org/abs/2512.02556.*[15] *Mostafa Dehghani* 等人。*Patch n' Pack*：*NaViT*，一个适用于任何宽高比和分辨率的视觉 *Transformer*。*2023*。*arXiv: 2307.06304 [cs.CV]*。*URL: https://arxiv.org/abs/2307.06304.*[16] *Xiang Deng* 等人。*"SWE-Bench Pro: AI* 代理能否解决长时程软件工程任务？*"*载于 *arXiv* 预印本 *arXiv:2509.16941 (2025)*。[17] *Chaoyou Fu* 等人。*Video-MME*：视频分析中多模态大型语言模型首个综合评估基准。*2025*。*arXiv: 2405.21075 [cs.CV]*。*URL: https://arxiv.org/abs/2405.21075.*[18] *Xingyu Fu* 等人。*BLINK*：多模态大型语言模型能看但不能感知。*2024*。*arXiv: 2404.12390 [cs.CV]*。*URL: https://arxiv.org/abs/2404.12390.*[19] *Samir Yitzhak Gadre* 等人。*"Datacomp*：寻找下一代多模态数据集*"*。载于神经信息处理系统进展 *36 (2024)*。[20] 谷歌。*Gemini 3 Pro*。*2025*。*URL: https://deepmind.google/models/gemini/pro/.*[21] *Dong Guo* 等人。*Seed1.5-VL* 技术报告。*2025*。*arXiv: 2505.07062 [cs.CV]*。*URL: https://arxiv.org/abs/2505.07062.*[22] *Lukas Haas* 等人。*SimpleQA Verified*：一种可靠的用于衡量参数化知识的事实性基准。*2025*。*arXiv: 2509.07968 [cs.CL]*。*URL: https://arxiv.org/abs/2509.07968.*[23] *Yun He* 等人。*AdvancedIF*：基于评分的基准测试和强化学习，用于推进大型语言模型指令遵循。*2025*。*arXiv: 2511.10507 [cs.CL]*。*URL: https://arxiv.org/abs/2511.10507.*[24] *Wenyi Hong* 等人。*MotionBench*：对视觉语言模型的细粒度视频运动理解进行基准测试和改进。*2025*。*arXiv: 2501.02955 [cs.CV]*。*URL: https://arxiv.org/abs/2501.02955.*[25] *Kairui Hu* 等人。*Video-MMMU*：评估从多学科专业视频获取知识。*2025*。*arXiv: 2501.13826 [cs.CV]*。*URL: https://arxiv.org/abs/2501.13826.*

[26] Liang Hu et al. *FinSearchComp: Towards a Realistic, Expert-Level Evaluation of Financial Search and Reasoning*. 2025. arXiv: 2509.13160 [cs.CL]. URL: https://arxiv.org/abs/2509.13160.

[27] Yanping Huang et al. *GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism*. 2019. arXiv: 1811.06965 [cs.CV]. URL: https://arxiv.org/abs/1811.06965.

[28] Naman Jain et al. "Livecodebench: Holistic and contamination free evaluation of large language models for code". In: *arXiv preprint arXiv:2403.07974* (2024).

[29] Carlos E Jimenez et al. "Swe-bench: Can language models resolve real-world github issues?" In: *arXiv preprint arXiv:2310.06770* (2023).

[30] Keller Jordan et al. *Muon: An optimizer for hidden layers in neural networks*. 2024. URL: https://kellerjordan.github.io/posts/muon/.

[31] Kimi Team. "Kimi k1. 5: Scaling reinforcement learning with llms". In: *arXiv preprint arXiv:2501.12599* (2025).

[32] Hugo Laurençon et al. "Obelics: An open web-scale filtered dataset of interleaved image-text documents". In: *Advances in Neural Information Processing Systems* 36 (2024).

[33] Dmitry Lepikhin et al. "Gshard: Scaling giant models with conditional computation and automatic sharding". In: *arXiv preprint arXiv:2006.16668* (2020).

[34] Jingyuan Liu et al. "Muon is Scalable for LLM Training". In: *arXiv preprint arXiv:2502.16982* (2025).

[35] Yuliang Liu et al. "OCRBench: on the hidden mystery of OCR in large multimodal models". In: *Science China Information Sciences* 67.12 (Dec. 2024). ISSN: 1869-1919. DOI: 10.1007/s11432-024-4235-6. URL: http://dx.doi.org/10.1007/s11432-024-4235-6.

[36] Pan Lu et al. *MathVista: Evaluating Mathematical Reasoning of Foundation Models in Visual Contexts*. 2024. arXiv: 2310.02255 [cs.CV]. URL: https://arxiv.org/abs/2310.02255.

[37] Thang Luong et al. "Towards Robust Mathematical Reasoning". In: *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. Ed. by Christos Christodoulopoulos et al. Suzhou, China: Association for Computational Linguistics, Nov. 2025, pp. 35418–35442. ISBN: 979-8-89176-332-6. DOI: 10.18653/v1/2025.emnlp-main.1794. URL: https://aclanthology.org/2025.emnlp-main.1794/.

[38] Minesh Mathew et al. *InfographicVQA*. 2021. arXiv: 2104.12756 [cs.CV]. URL: https://arxiv.org/abs/2104.12756.

[39] Mike A Merrill et al. "Terminal-Bench: Benchmarking Agents on Hard, Realistic Tasks in Command Line Interfaces". In: *arXiv preprint arXiv:2601.11868* (2026).

[40] Deepak Narayanan et al. *Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM*. 2021. arXiv: 2104.04473 [cs.CL]. URL: https://arxiv.org/abs/2104.04473.

[41] OpenAI. *Introducing GPT 5.2*. 2025. URL: https://openai.com/index/introducing-gpt-5-2/.

[42] Linke Ouyang et al. *OmniDocBench: Benchmarking Diverse PDF Document Parsing with Comprehensive Annotations*. 2025. arXiv: 2412.07626 [cs.CV]. URL: https://arxiv.org/abs/2412.07626.

[43] Tejal Patwardhan et al. *GDPval: Evaluating AI Model Performance on Real-World Economically Valuable Tasks*. 2025. arXiv: 2510.04374 [cs.LG]. URL: https://arxiv.org/abs/2510.04374.

[44] Bowen Peng et al. "Yarn: Efficient context window extension of large language models". In: *arXiv preprint arXiv:2309.00071* (2023).

[45] Thinh Pham et al. *SealQA: Raising the Bar for Reasoning in Search-Augmented Language Models*. Seal-0 is the main subset of this benchmark. 2025. arXiv: 2506.01062 [cs.CL]. URL: https://arxiv.org/abs/2506.01062.

[46] Long Phan et al. *Humanity's Last Exam*. 2025. arXiv: 2501.14249 [cs.LG]. URL: https://arxiv.org/abs/2501.14249.

[47] David Rein et al. "Gpqa: A graduate-level google-proof q&a benchmark". In: *First Conference on Language Modeling*. 2024.

[48] Jonathan Roberts et al. *ZeroBench: An Impossible Visual Benchmark for Contemporary Large Multimodal Models*. 2025. arXiv: 2502.09696 [cs.CV]. URL: https://arxiv.org/abs/2502.09696.

[49] Christoph Schuhmann et al. "Laion-5b: An open large-scale dataset for training next generation image-text models". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 25278–25294.

[50] John Schulman et al. "Proximal Policy Optimization Algorithms". In: *arXiv preprint arXiv:1707.06347* (2017). URL: https://arxiv.org/abs/1707.06347.

[26] 梁华等. FinSearchComp: 旨在实现真实、专家级的金融搜索与推理评估. 2025. arXiv: 2509.13160 [cs.CL]. URL: https://arxiv.org/abs/2509.13160.[27] 黄艳萍等. GPipe: 利用流水线并行高效训练巨型神经网络. 2019. arXiv: 1811.06965 [cs.CV]. URL: https://arxiv.org/abs/1811.06965.[28] 贾南曼等. "Livecodebench: 对大型语言模型进行代码的全面且无污染评估". 载于arXiv预印本 arXiv:2403.07974 (2024).[29] 卡洛斯·E·希门内斯等. "Swe-bench: 语言模型能否解决真实的GitHub问题？" 载于arXiv预印本 arXiv:2310.06770 (2023).[30] 乔丹·凯勒等. Muon: 神经网络中隐藏层的优化器. 2024. URL: https://kellerjordan.github.io/posts/muon/.[31] Kimi 团队. "Kimi k1. 5: 利用大语言模型扩展强化学习". 载于 arXiv 预印本 arXiv:2501.12599 (2025).[32] 洛伦索·雨果等. "Obelics: 一个开放的网络规模过滤数据集，包含图像-文本交织文档". 载于神经信息处理系统进展36 (2024).[33] 列奥尼德·列皮京等. "Gshard: 利用条件计算和自动分片扩展巨型模型". 载于arXiv预印本 arXiv:2006.16668 (2020).[34] 刘景源等. "Muon对LLM训练是可扩展的". 载于arXiv预印本 arXiv:2502.16982 (2025).[35] 刘雨良等. "OCRBench: 关于大型多模态模型中OCR的隐藏之谜". 载于科学中国信息科学67.12 (2024年12月). ISSN: 1869-1919. DOI: 10.1007/s11432-024-4235-6. URL: http://dx.doi.org/10.1007/s11432-024-4235-6.[36] 陆磐等. MathVista: 在视觉环境中评估基础模型数学推理能力. 2024. arXiv: 2310.02255 [cs.CV]. URL: https://arxiv.org/abs/2310.02255.[37] 陆祥龙等. "迈向鲁棒的数学推理". 载于2025年实验自然语言处理会议. 由克里斯托斯·克里斯托多普洛普洛斯等编辑. 苏州，中国：计算语言学协会，2025 年 11 月，第 35418–35442 页 . ISBN: 979-8-89176-332-6. DOI: 10.18653 / v1 / 2025 . emnlp - main . 1794. URL: https : / / aclanthology . org / 2025 . emnlp - main.1794/.[38] 马修斯·米内什等. InfographicVQA. 2021. arXiv: 2104.12756 [cs.CV]. URL: https://arxiv. org/abs/2104.12756.[39] 迈克·A·梅里尔等. "Terminal-Bench: 在命令行界面中基准测试代理在困难、现实任务上的表现". 载于arXiv预印本 arXiv:2601.11868 (2026).[40] Deepak·纳拉扬等. 使用Megatron-LM在GPU集群上进行高效的大规模语言模型训练. 2021. arXiv: 2104.04473 [cs.CL]. URL: https://arxiv.org/abs/2104.04473.[41]OpenAI. 发布GPT 5.2. 2025. URL: https://openai.com/index/introducing- gpt-5- 2/.[42] 欧阳林克等. OmniDocBench: 对具有全面注释的多样化PDF文档解析进行基准测试. 2025. arXiv: 2412.07626 [cs.CV]. URL: https://arxiv.org/abs/2412.07626.[43] 帕杰·帕特瓦德汉等. GDPval: 在真实世界具有经济价值的任务上评估AI模型性能. 2025. arXiv: 2510.04374 [cs.LG]. URL: https://arxiv.org/abs/2510.04374.[44] 彭 Bowen等. "Yarn: 大语言模型的高效上下文窗口扩展". 载于arXiv预印本 arXiv:2309.00071 (2023).[45] 范 Thin等. SealQA: 提升搜索增强语言模型中的推理水平. Seal-0是此基准测试的主要子集. 2025. arXiv: 2506.01062 [cs.CL]. URL: https://arxiv.org/ abs/2506.01062.[46] 范龙等. 人类最后考试. 2025. arXiv: 2501.14249 [cs.LG]. URL: https://arxiv. org/abs/2501.14249.[47] 大卫·莱因等. "Gpqa: 一个研究生级别的谷歌证明问答基准". 载于第一次语言建模会议. 2024.[48] 乔纳森·罗伯茨等. ZeroBench: 对当代大型多模态模型的不可能视觉基准. 2025. arXiv: 2502.09696 [cs.CV]. URL: https://arxiv.org/abs/2502.09696.[49] 克里斯托夫·舒曼曼等. "Laion-5b: 一个用于训练下一代图像-文本模型的开放大规模数据集". 载于神经信息处理系统进展35 (2022), 第25278–25294页.[50] 约翰·舒尔曼等. "近端策略优化算法". 载于arXiv预印本 arXiv:1707.06347 (2017). URL: https://arxiv.org/abs/1707.06347.

[51] Tianhui Song et al. *Towards Pixel-Level VLM Perception via Simple Points Prediction*. 2026. arXiv: 2601.19228 [cs.CV]. URL: https://arxiv.org/abs/2601.19228.

[52] Giulio Starace et al. "PaperBench: Evaluating AI's Ability to Replicate AI Research". In: *arXiv preprint arXiv:2504.01848* (2025).

[53] Kimi Team et al. "Kimi k2: Open agentic intelligence". In: *arXiv preprint arXiv:2507.20534* (2025).

[54] Kimi Team et al. "Kimi-vl technical report". In: *arXiv preprint arXiv:2504.07491* (2025).

[55] Meituan LongCat Team et al. "Longcat-flash-omni technical report". In: *arXiv preprint arXiv:2511.00279* (2025).

[56] Minyang Tian et al. "Scicode: A research coding benchmark curated by scientists". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 30624–30650.

[57] Shengbang Tong et al. *Eyes Wide Shut? Exploring the Visual Shortcomings of Multimodal LLMs*. 2024. arXiv: 2401.06209 [cs.CV]. URL: https://arxiv.org/abs/2401.06209.

[58] Harvard-MIT Mathematics Tournament. *Harvard-MIT Mathematics Tournament, February 2025*. Held on February 15, 2025. 2025. URL: https://www.hmmt.org/www/archive/282.

[59] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[60] Nikhita Vedula et al. *DeepSearchQA: Bridging the Comprehensiveness Gap for Deep Research Agents*. 2025. URL: https://storage.googleapis.com/deepmind-media/DeepSearchQA/DeepSearchQA_benchmark_paper.pdf.

[61] Ke Wang et al. *Measuring Multimodal Mathematical Reasoning with MATH-Vision Dataset*. 2024. arXiv: 2402.14804 [cs.CV]. URL: https://arxiv.org/abs/2402.14804.

[62] Weihan Wang et al. *LVBench: An Extreme Long Video Understanding Benchmark*. 2025. arXiv: 2406.08035 [cs.CV]. URL: https://arxiv.org/abs/2406.08035.

[63] Xinyuan Wang et al. *OpenCUA: Open Foundations for Computer-Use Agents*. 2025. arXiv: 2508.09123 [cs.AI]. URL: https://arxiv.org/abs/2508.09123.

[64] Yubo Wang et al. *MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark*. 2024. arXiv: 2406.01574 [cs.CL]. URL: https://arxiv.org/abs/2406.01574.

[65] Zhexu Wang et al. "OJBench: A Competition Level Code Benchmark For Large Language Models". In: *arXiv preprint arXiv:2506.16395* (2025).

[66] Zhun Wang et al. "CyberGym: Evaluating AI Agents' Cybersecurity Capabilities with Real-World Vulnerabilities at Scale". In: *arXiv preprint arXiv:2506.02548* (2025).

[67] Zirui Wang et al. *CharXiv: Charting Gaps in Realistic Chart Understanding in Multimodal LLMs*. 2024. arXiv: 2406.18521 [cs.CL]. URL: https://arxiv.org/abs/2406.18521.

[68] Jason Wei et al. *BrowseComp: A Simple Yet Challenging Benchmark for Browsing Agents*. 2025. arXiv: 2504.12516 [cs.CL]. URL: https://arxiv.org/abs/2504.12516.

[69] Ryan Wong et al. *WideSearch: Benchmarking Agentic Broad Info-Seeking*. 2025. arXiv: 2508.07999 [cs.CL]. URL: https://arxiv.org/abs/2508.07999.

[70] Haoning Wu et al. *LongVideoBench: A Benchmark for Long-context Interleaved Video-Language Understanding*. 2024. arXiv: 2407.15754 [cs.CV]. URL: https://arxiv.org/abs/2407.15754.

[71] Xixi Wu et al. *ReSum: Unlocking Long-Horizon Search Intelligence via Context Summarization*. 2025. arXiv: 2509.13313 [cs.CL]. URL: https://arxiv.org/abs/2509.13313.

[72] Tianbao Xie et al. "Introducing OSWorld-Verified". In: *xlang.ai* (July 2025). URL: https://xlang.ai/blog/osworld-verified.

[73] Tianbao Xie et al. *OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments*. 2024. arXiv: 2404.07972 [cs.AI].

[74] Feng Yao et al. *Your Efficient RL Framework Secretly Brings You Off-Policy RL Training*. Aug. 2025. URL: https://fengyao.notion.site/off-policy-rl.

[75] Jiahui Yu et al. *CoCa: Contrastive Captioners are Image-Text Foundation Models*. 2022. arXiv: 2205.01917 [cs.CV]. URL: https://arxiv.org/abs/2205.01917.

[76] Xiang Yue et al. *MMMU-Pro: A More Robust Multi-discipline Multimodal Understanding Benchmark*. 2025. arXiv: 2409.02813 [cs.CL]. URL: https://arxiv.org/abs/2409.02813.

[77] Xiang Yue et al. "MMMU: A Massive Multi-discipline Multimodal Understanding and Reasoning Benchmark for Expert AGI". In: *Proceedings of CVPR*. 2024.

[51] 宋天辉等. 迈向像素级VLM感知通过简单的点预测. 2026. arXiv: 2601.19228 [cs.CV]. URL: https://arxiv.org/abs/2601.19228.[52] 吉尤利奥·斯塔拉切等. "PaperBench: 评估AI复制AI研究的能力". 载于 arXiv 预印本 arXiv:2504.01848 (2025).[53] Kimi 团队等 . "Kimi k2: 开放式自主智能 ". 载于arXiv预印本 arXiv:2507.20534 (2025).[54] Kimi团队等. "Kimi-vl技术报告". 载于arXiv预印本 arXiv:2504.07491 (2025).[55] 美团龙猫团队等. "Longcat-flash-omni技术报告". 载于arXiv预印本 arXiv:2511.00279 (2025).[56] 田明阳等. "Scicode: 由科学家策划的研究编码基准". 载于神经信息处理系统进展37 (2024), 第30624–30650页.[57] 董胜邦等. 眼睛闭着? 探索多模态LLM的视觉缺陷. 2024. arXiv: 2401.06209 [cs.CV]. URL: https://arxiv.org/abs/2401.06209.[58] 哈佛-麻省理工学院数学锦标赛. 哈佛-麻省理工学院数学锦标赛, 2025年2月. 于2025年2月15日举行. 2025. URL: https://www.hmmt.org/www/archive/282.[59] 阿希什·瓦西尼等. "注意力就是全部你需要". 载于神经信息处理系统. 由I. 盖翁等编辑. 第30卷. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.[60] 尼基塔·维达拉等. DeepSearchQA: 桥接深度研究代理的全面性差距. 2025. URL: https://storage.googleapis.com/deepmind-media/DeepSearchQA/DeepSearchQA_benchmark_paper.pdf.[61] 王 Ke等. 使用MATH-Vision数据集测量多模态数学推理. 2024. arXiv: 2402.14804 [cs.CV]. URL: https://arxiv.org/abs/2402.14804.[62] 王伟汉等. LVBench: 一个极端长视频理解基准. 2025. arXiv: 2406.08035[cs.CV]. URL: https://arxiv.org/abs/2406.08035.[63]王晓媛等. OpenCUA: 计算机使用代理的开放基础. 2025. arXiv: 2508. 09123[cs.AI]. URL: https://arxiv.org/abs/2508.09123.[64] 王宇博等. MMLU-Pro: 一个更鲁棒和更具挑战性的多任务语言理解基准. 2024. arXiv: 2406.01574 [cs.CL]. URL: https://arxiv.org/abs/2406.01574.[65] 王哲旭等. "OJBench: 为大型语言模型提供一个竞赛级别的代码基准". 载于arXiv预印本 arXiv:2506.16395 (2025).[66] 王准等. "CyberGym: 使用真实世界漏洞大规模评估AI代理的网络能力". 载于arXiv预印本 arXiv:2506.02548 (2025).[67] 王子睿等. CharXiv: 绘制多模态LLM中真实图表理解的差距. 2024. arXiv: 2406.18521 [cs.CL]. URL: https://arxiv.org/abs/2406.18521.[68] 韦杰森等. BrowseComp: 一个简单但具有挑战性的浏览代理基准. 2025. arXiv: 2504.12516 [cs.CL]. URL: https://arxiv.org/abs/2504.12516.[69] 王瑞安等. WideSearch: 基准测试代理的广泛信息寻求. 2025. arXiv: 2508 . 07999[cs.CL]. URL: https://arxiv.org/abs/2508.07999.[70] 吴浩宁等. LongVideoBench: 一个长上下文交织视频-语言理解基准. 2024. arXiv: 2407.15754 [cs.CV]. URL: https://arxiv.org/abs/2407.15754.[71] 吴熙熙等. ReSum: 通过上下文摘要解锁长时程搜索智能. 2025. arXiv: 2509.13313 [cs.CL]. URL: https://arxiv.org/abs/2509.13313.[72] 谢天宝等. "介绍OSWorld-Verified". 载于xlang.ai (2025年7月). URL: https://xlang.ai/ blog/osworld-verified.[73] 谢天宝等 . OSWorld: 在真实计算机环境中对多模态代理进行开放式任务基准测试. 2024. arXiv: 2404.07972 [cs.AI].[74] 姚峰等. 你的高效RL框架悄悄为你带来离线策略RL训练. 2025年8月. URL: https://fengyao.notion.site/off-policy-rl.[75] 余佳辉等. CoCa: 对比性描述者都是图像-文本基础模型. 2022. arXiv: 2205.01917[cs.CV]. URL: https://arxiv.org/abs/2205.01917.[76] 岳翔等. MMMU-Pro: 一个更鲁棒的多学科多模态理解基准. 2025. arXiv: 2409.02813 [cs.CL]. URL: https://arxiv.org/abs/2409.02813.[77] 岳翔等. "MMMU: 一个大规模多学科多模态理解和推理基准，用于专家AGI". 载于CVPR会议论文集. 2024.

[78] Xiaohua Zhai et al. *Sigmoid Loss for Language Image Pre-Training*. 2023. arXiv: 2303.15343 [cs.CV]. URL: https://arxiv.org/abs/2303.15343.

[79] Xin Zhao et al. *Small Leak Can Sink a Great Ship–Boost RL Training on MoE with IcePop!* Sept. 2025. URL: https://ringtech.notion.site/icepop.

[80] Yilun Zhao et al. *MMVU: Measuring Expert-Level Multi-Discipline Video Understanding*. 2025. arXiv: 2501.12380 [cs.CV]. URL: https://arxiv.org/abs/2501.12380.

[81] Shuyan Zhou et al. "WebArena: A Realistic Web Environment for Building Autonomous Agents". In: *arXiv preprint arXiv:2307.13854* (2023). URL: https://webarena.dev.

[82] Wanrong Zhu et al. "Multimodal c4: An open, billion-scale corpus of images interleaved with text". In: *Advances in Neural Information Processing Systems* 36 (2024).

[78] 夏华翟等. Sigmoid Loss用于语言图像预训练. 2023. arXiv: 2303.15343 [cs.CV]. URL: https://arxiv.org/abs/2303.15343.[79] 赵欣等. 小洞能沉大船—使用IcePop增强MoE的RL训练！2025年9月. URL: https://ringtech.notion.site/icepop.[80] 赵一龙等. MMVU: 测量专家级多学科视频理解. 2025. arXiv: 2501.12380 [cs.CV]. URL: https://arxiv.org/abs/2501.12380.[81] 周舒岩等. "WebArena: 用于构建自主代理的现实网络环境". 载于arXiv预印本arXiv:2307.13854 (2023). URL: https://webarena.dev.[82] 朱万荣等. "多模态c4: 一个开放的、十亿规模的图像与文本交错语料库". 载于神经信息处理系统进展36 (2024).

A Contributions

Tongtong Bai
Yifan Bai
Yiping Bao
S.H. Cai
Yuan Cao
Y. Charles
H.S. Che
Cheng Chen
Guanduo Chen
Huarong Chen
Jia Chen
Jiahao Chen
Jianlong Chen
Jun Chen
Kefan Chen
Liang Chen
Ruijue Chen
Xinhao Chen
Yanru Chen
Yanxu Chen
Yicun Chen
Yimin Chen
Yingjiang Chen
Yuankun Chen
Yujie Chen
Yutian Chen
Zhirong Chen
Ziwei Chen
Dazhi Cheng
Minghan Chu
Jialei Cui
Jiaqi Deng
Muxi Diao
Hao Ding
Mengnan Dong
Yuxin Dong
Yuhao Dong
Ang'ang Du
Chenzhuang Du
Dikang Du
Lingxiao Du
Yulun Du
Yu Fan
Shengjun Fang
Qiulin Feng
Yichen Feng
Garimugai Fu
Kelin Fu
Hongcheng Gao
Tong Gao
Yuyao Ge
Shangyi Geng
Chengyang Gong
Xiaochen Gong
Zhuoma Gongque
Qizheng Gu
Xinran Gu
Yicheng Gu
Longyu Guan
Yuanying Guo
Xiaoru Hao
Weiran He
Wenyang He
Yunjia He
Chao Hong
Hao Hu
Jiaxi Hu
Yangyang Hu
Zhenxing Hu
Ke Huang
Ruiyuan Huang
Weixiao Huang
Zhiqi Huang
Tao Jiang
Zhejun Jiang
Xinyi Jin
Yu Jing
Guokun Lai
Aidi Li
C. Li
Cheng Li
Fang Li
Guanghe Li
Guanyu Li
Haitao Li
Haoyang Li
Jia Li
Jingwei Li
Junxiong Li
Lincan Li
Mo Li
Weihong Li
Wentao Li
Xinhang Li
Xinhao Li
Yang Li
Yanhao Li
Yiwei Li
Yuxiao Li
Zhaowei Li
Zheming Li
Jiawei Lin
Xiaohan Lin
Zhishan Lin
Zichao Lin
Cheng Liu
Chenyu Liu
Hongzhang Liu
Liang Liu
Shaowei Liu
Shudong Liu
Shuran Liu
Tianwei Liu
Tianyu Liu
Weizhou Liu
Xiangyan Liu
Yangyang Liu
Yanming Liu
Yibo Liu
Yuanxin Liu
Yue Liu
Zhengying Liu
Zhongnuo Liu
Enzhe Lu
Haoyu Lu
Zhiyuan Lu
Junyu Luo
Tongxu Luo
Yashuo Luo
Long Ma
Yingwei Ma
Shaoguang Mao
Yuan Mei
Xin Men
Fanqing Meng
Zhiyong Meng
Yibo Miao
Minqing Ni
Kun Ouyang
Siyuan Pan
Bo Pang
Yuchao Qian
Ruoyu Qin
Zeyu Qin
Jiezhong Qiu
Bowen Qu
Zeyu Shang
Youbo Shao
Tianxiao Shen
Zhennan Shen
Juanfeng Shi
Lidong Shi
Shengyuan Shi
Feifan Song
Pengwei Song
Tianhui Song
Xiaoxi Song
Hongjin Su
Jianlin Su
Zhaochen Su
Lin Sui
Jinsong Sun
Junyao Sun
Tongyu Sun
Flood Sung
Yunpeng Tai
Chuning Tang
Heyi Tang
Xiaojuan Tang
Zhengyang Tang
Jiawen Tao
Shiyuan Teng
Chaoran Tian
Pengfei Tian
Ao Wang
Bowen Wang
Chensi Wang
Chuang Wang
Congcong Wang
Dingkun Wang
Dinglu Wang
Dongliang Wang
Feng Wang
Hailong Wang
Haiming Wang
Hengzhi Wang
Huaqing Wang
Hui Wang
Jiahao Wang
Jinhong Wang
Jiuzheng Wang
Kaixin Wang
Linian Wang
Qibin Wang
Shengjie Wang
Shuyi Wang
Si Wang
Wei Wang
Xiaochen Wang
Xinyuan Wang
Yao Wang
Yejie Wang
Yipu Wang
Yiqin Wang
Yucheng Wang
Yuzhi Wang
Zhaoji Wang
Zhaowei Wang
Zhengtao Wang
Zhexu Wang
Zihan Wang
Zizhe Wang
Chu Wei
Ming Wei
Chuan Wen
Zichen Wen
Haoning Wu
Junyan Wu
Rucong Wu
Wenhao Wu
Yuefeng Wu
Yuhao Wu
Yuxin Wu
Zijian Wu
Chenjun Xiao
Jin Xie
Xiaotong Xie
Yuchong Xie
Yifei Xin
Bowei Xing
Boyu Xu
Jianfan Xu
Jing Xu
Jinjing Xu
L.H. Xu
Lin Xu
Suting Xu
Weixin Xu
Xinbo Xu
Xinran Xu
Yangchuan Xu
Yichang Xu
Yuemeng Xu
Zelai Xu
Ziyao Xu
Junjie Yan
Yuzi Yan
Guangyao Yang
Hao Yang
Kai Yang
Ningyuan Yang
Ruihan Yang
Xiaofei Yang
Xinlong Yang
Ying Yang
Yi (弋) Yang
Yi (翌) Yang
Zhen Yang
Zhilin Yang
Zonghan Yang
Haotian Yao
Dan Ye
Wenjie Ye
Zhuorui Ye
Bohong Yin
Chengzhen Yu
Longhui Yu
Tao Yu†
Tianxiang Yu
Enming Yuan
Mengjie Yuan
Xiaokun Yuan
Yang Yue
Weihao Zeng
Dunyuan Zha
Haobing Zhan
Dehao Zhang
Hao Zhang
Jin Zhang
Puqi Zhang
Qiao Zhang
Rui Zhang
Xiaobin Zhang
Y. Zhang
Yadong Zhang
Yangkun Zhang
Yichi Zhang
Yizhi Zhang
Yongting Zhang
Yu Zhang
Yushun Zhang
Yutao Zhang
Yutong Zhang
Zheng Zhang
Chenguang Zhao
Feifan Zhao
Jinxiang Zhao
Shuai Zhao
Xiangyu Zhao
Yikai Zhao
Zijia Zhao
Huabin Zheng
Ruihan Zheng
Shaojie Zheng
Tengyang Zheng
Junfeng Zhong
Longguang Zhong
Weiming Zhong
M. Zhou
Runjie Zhou
Xinyu Zhou
Zaida Zhou
Jinguo Zhu
Liya Zhu
Xinhao Zhu
Yuxuan Zhu
Zhen Zhu
Jingze Zhuang
Weiyu Zhuang
Ying Zou
Xinxing Zu
Kimi K2
Kimi K2.5

The listing of authors is in alphabetical order based on their last names.
†Hongkong University

贡献

白同桐
白一帆
包一平
蔡S.H.
曹元
查尔斯Y.
谢H.S.
陈程
陈冠多
陈华荣
贾晨
嘉豪·陈
建龙·陈
俊·陈
科帆·陈
亮·陈
瑞杰·陈
新豪·陈
艳如·陈
彦旭·陈
奕存·陈
一民·陈
英江·陈
陈元坤
陈宇杰
陈宇天
陈志荣
陈子伟
程大智
楚明瀚
崔佳蕾
邓嘉琪
貂木溪
丁浩
董梦楠
董宇欣
董宇豪
杜昂昂
杜陈庄
杜狄康
杜凌晓
杜宇伦
范宇
方胜军
冯秋林
冯奕辰
付嘉瑞
付科林
高洪成
高通
葛宇瑶
耿上艺
Gong成阳
Gong晓辰
Gongque朱玛
Gu齐正
Gu欣然
Gu一成
Guan龙宇
Guo袁莹
Hao晓如
He魏然
何文阳
何云嘉
洪超
胡浩
胡嘉熙
胡阳阳
胡振星
黄科
黄瑞元
黄伟晓
黄志奇
蒋涛
蒋哲俊
金新怡
景宇
来国昆
李爱迪
李C
李成
李方
李广和
李冠宇
李海涛
李浩阳
李嘉
李景伟
李军雄
李林灿
李墨
李伟红
李文涛
李新航
李新浩
李杨
李彦豪
李艺伟
李宇晓
李兆伟
李哲明
林佳伟
林晓寒
林志山
林子超
刘程
刘晨宇
刘洪章
刘亮
刘少伟
刘舒东
刘舒然
刘天伟
刘天宇
刘伟周
刘向岩
刘阳阳
刘艳明
刘一博
刘元欣
刘越
刘正英
刘中诺
陆恩哲
陆浩宇
陆志远
罗军宇
罗通旭
罗亚寿
马龙
马英伟
毛少光
梅元
门新
孟繁庆
孟志勇
苗一博
倪明清
欧阳坤
潘思源
Bo Pang
Yuchao Qian
Ruoyu Qin
Zeyu Qin
Jiezhong Qiu
Bowen Qu
Zeyu Shang
邵友波
沈天晓
沈振南
石娟凤
石立冬
石胜元
宋非凡
宋鹏伟
宋天辉
宋晓希
苏洪金
苏建林
苏赵晨
隋林
孙金松
孙俊瑶
孙通宇
沈洪水
台云鹏
唐春
唐Heyi
唐晓娟
唐正阳
陶嘉文
滕诗源
田超然
田鹏飞
王澳
王 Bowen
王宸
王创
王聪聪
王丁坤
王丁录
王东亮
王峰
王海龙
王海明
王恒志
王华清
王辉
王嘉豪
王金红
王九正
王凯欣
王林岩
王启斌
王胜杰
王舒毅
王思
王伟
王晓辰
王晓媛
王瑶
王晔杰
王奕普
王怡琴
王宇成
王宇智
王赵基
王兆伟
王正涛
王哲旭
王志涵
王子哲
朱伟
魏明
温传
温智辰
吴浩宁
吴俊岩
吴若丛
吴文浩
吴越峰
吴宇豪
吴雨欣
吴子健
肖晨俊
谢进
谢晓彤
谢宇崇
辛怡飞
刑博文
徐伯宇
徐建藩
徐静
徐金晶
徐L.H.
徐琳
徐素婷
徐微信
徐新波
徐新然
徐杨川
徐宜昌
徐月萌
徐泽来
徐子晔
闫俊杰
闫宇齐
杨广耀
杨浩
杨凯
杨宁远
杨瑞涵
杨晓飞
杨新龙
杨颖
杨弋
杨翌
杨振
杨志林
杨宗汉
姚浩天
叶丹
叶文杰
叶卓睿
尹博文
余成臻
余龙辉
余涛†
余天翔
袁恩明
袁梦洁
袁晓坤
岳杨
曾伟豪
翟 Dunyuan
战昊冰
张德华
张浩
张金
张普奇
张俏
张瑞
张晓斌
张毅
张亚东
张阳坤
张奕驰
张奕之
张永庭
张宇
张宇顺
张宇涛
张宇通
张政
赵晨光
赵飞帆
赵金祥
赵帅
赵祥宇
赵奕凯
赵子嘉
郑华斌
郑瑞涵
郑少杰
郑腾阳
钟俊峰
钟龙光
钟伟明
周M
周润吉
周新宇
周载大
朱金国
朱丽雅
朱新浩
朱宇轩
朱振
庄景泽
庄伟宇
邹颖
Zu新行
Kimi K2
Kimi K2.5

作者名单按姓氏字母顺序排列。
† 香港大学

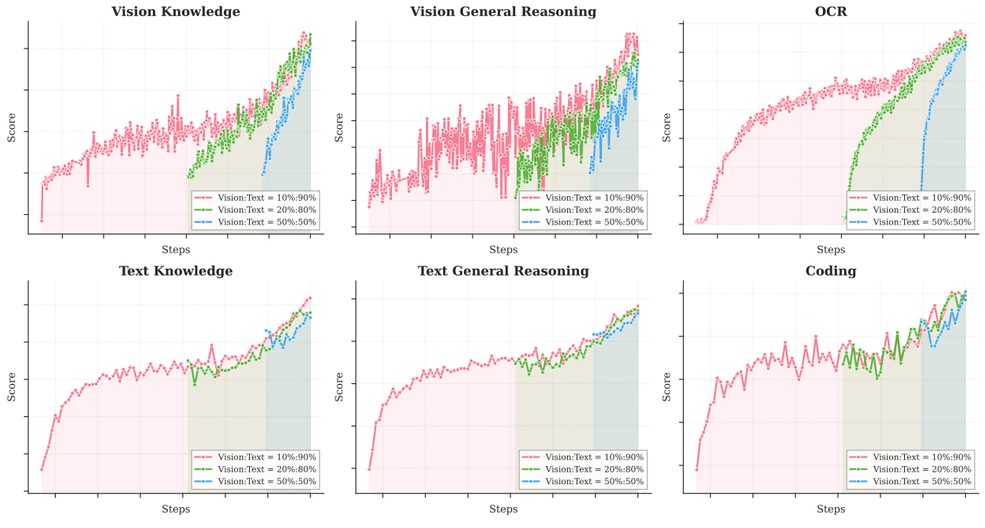


Figure 9: Learning curves comparing vision-to-text ratios (10:90, 20:80, 50:50) under fixed vision-text token budget across vision and language tasks. Early fusion with lower vision ratios tend to yield better results.

B Pre-training

B.1 Joint-Training

We further provide the full training curves for all configurations in Figure 9. Notably, we observe a "dip-and-recover" pattern in text performance during mid-fusion and late-fusion stages: when vision data is first introduced, text capability initially degrades before gradually recovering. We attribute this to the modality domain shift—the sudden introduction of vision tokens disrupts the established linguistic representation space, forcing the model to temporarily sacrifice text-specific competence for cross-modal alignment.

In contrast, early fusion maintains a healthier and more stable text performance curve throughout training. By co-optimizing vision and language from the outset, the model naturally evolves unified multimodal representations without the shock of late-stage domain migration. This suggests that early exposure not only prevents the representation collapse observed in late fusion but also facilitates smoother gradient landscapes for both modalities. Collectively, these findings reinforce our proposal of native multimodal pre-training: moderate vision ratios combined with early fusion yield superior convergence properties and more robust bi-modal competence under fixed token budgets.

B.2 Text data

The Kimi K2.5 pre-training text corpus comprises curated, high-quality data spanning four primary domains: Web Text, Code, Mathematics, and Knowledge. Most data processing pipelines follow the methodologies outlined in Kimi K2 [53]. For each domain, we performed rigorous correctness and quality validation and designed targeted data experiments to ensure the curated dataset achieved both high diversity and effectiveness.

Enhanced Code Intelligence We upweighted code-centric data, significantly expanding (1) repository-level code supporting cross-file reasoning and architectural understanding, (2) issues, code reviews and commit histories from the internet capturing real-world development patterns, and (3) code-related documents retrieved from PDF and webtext corpora. These efforts strengthen repository-level comprehension for complex coding tasks, improve performance on agentic coding subtasks such as patch generation and unit test writing, and enhance code-related knowledge capabilities.

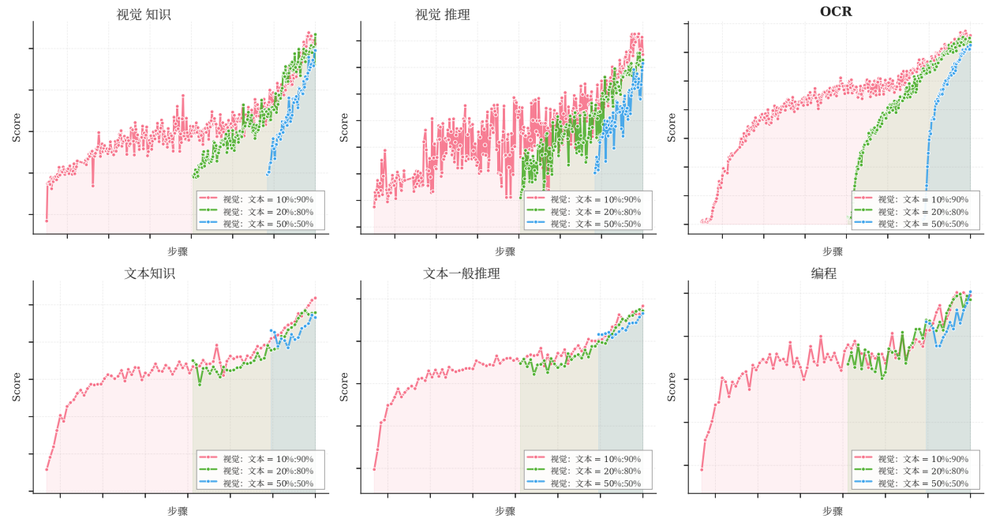


图 9：比较在不同视觉-文本标记预算下（10:90、20:80、50:50）的视觉到文本比例学习曲线。在视觉和语言任务中，早期融合且视觉比例较低时，往往能获得更好的结果。

B 预训练

B.1 联合训练

我们进一步提供了所有配置在图9中的完整训练曲线。值得注意的是，我们在中期融合和后期融合阶段观察到文本性能出现“先下降后回升”的模式：当首次引入视觉数据时，文本能力最初会下降，然后逐渐恢复。我们将此归因于模态域迁移—视觉标记的突然引入扰乱了已建立的 linguistic 表示空间，迫使模型暂时牺牲文本特定能力以进行跨模态对齐。

相比之下，早期融合在整个训练过程中保持了更健康和稳定的文本性能曲线。从一开始就协同优化视觉和语言，模型自然地演化出统一的模态表示，而不会受到后期域迁移的冲击。这表明早期接触不仅防止了后期融合中观察到的表示崩溃，还促进了两种模态更平滑的梯度图。综合来看，这些发现加强了我们对原生模态预训练的提议：中等视觉比例结合早期融合，在固定标记预算下能获得更好的收敛特性和更鲁棒的模态能力。

B.2 文本数据

Kimi K2.5 预训练文本语料库包含精选的高质量数据，涵盖四个主要领域：网络文本、代码、数学和知识。大多数数据处理流程遵循 Kimi K2 [53] 中概述的方法论。对于每个领域，我们进行了严格正确性和质量验证，并设计了针对性的数据实验，以确保精选数据集同时实现了高度多样性和有效性。

增强代码智能我们加重了以代码为中心的数据，显著扩展了（1）仓库级别的代码支持跨文件推理和架构理解，（2）来自互联网的问题、代码审查和提交历史，捕捉现实世界的开发模式，（3）从PDF和网络文本语料库中检索到的与代码相关的文档。这些工作加强了仓库级别对复杂编码任务的认知，提高了在自主代理式编程子任务（如补丁生成和单元测试编写）上的性能，并增强了与代码相关的知识能力。

B.3 Vision data

Our multimodal pre-training corpus includes seven categories: caption, interleaving, OCR, knowledge, perception, video, and agent data. Caption data [49, 19] provides fundamental modality alignment, with strict limits on synthetic captions to mitigate hallucination. Image-text interleaving data from books, web pages, and tutorials [82, 32] enables multi-image comprehension and longer context learning. OCR data spans multilingual text, dense layouts, and multi-page documents. Knowledge data incorporates academic materials processed via layout parsers to develop visual reasoning capabilities.

Furthermore, we curate a specialized multimodal problem-solving corpus to bolster reasoning within Science, Technology, Engineering, and Mathematics domains. This data is aggregated through targeted retrieval and web crawling; for informational content lacking explicit query formats, we employ in-context learning [11] to automatically reformulate raw materials into structured academic problems spanning K-12 to university levels. To bridge the modality gap between visual layouts and code data, we incorporate extensive image-code paired data. This includes a diverse array of code formats—such as HTML, React, and SVG, among others—paired with their corresponding rendered screenshots, enabling the model to align abstract structural logic with concrete visual geometry.

For agentic and temporal understanding, we collect GUI screenshots and action trajectories across desktop, mobile, and web environments, including human-annotated demonstrations. Video data from diverse sources enables both hour-long video comprehension and fine-grained spatio-temporal perception. Additionally, we incorporate grounding data to enhance fine-grained visual localization, including perception annotations (bounding boxes), point-based references. We also introduce a new contour-level segmentation task [51] for pixel-level perception learning. All data undergoes rigorous filtering, deduplication, and quality control to ensure high diversity and effectiveness.

C Infra

Kimi K2.5 is trained on NVIDIA H800 GPU clusters with 8×400 Gbps RoCE interconnects across nodes. We employ a flexible parallelism strategy combining 16-way Pipeline Parallelism (PP) with virtual stages [27, 40], 16-way Expert Parallelism (EP) [33], and ZeRO-1 Data Parallelism, enabling training on any number of nodes that is a multiple of 32. EP all-to-all communication is overlapped with computation under interleaved 1F1B scheduling. To fit activations within GPU memory constraints, we apply selective recomputation for `LayerNorm`, `SwiGLU`, and `MLA` up-projections, compress insensitive activations to FP8-E4M3, and offload remaining activations to CPU with overlapped streaming.

C.1 Data Storage and Loading

We employ S3 [3] compatible object storage solutions from cloud providers to house our VLM datasets. To bridge the gap between data preparation and model training, we retain visual data in its native format and have engineered a highly efficient and adaptable data loading infrastructure. This infrastructure offers several critical advantages:

- **Flexibility:** Facilitates dynamic data shuffling, blending, tokenization, loss masking, and sequence packing throughout the training process, enabling adjustable data ratios as requirements evolve;
- **Augmentation:** Allows for stochastic augmentation of both visual and textual modalities, while maintaining the integrity of 2D spatial coordinates and orientation metadata during geometric transformations;
- **Determinism:** Guarantees fully deterministic training through meticulous management of random seeds and worker states, ensuring that any training interruption can be resumed seamlessly — the data sequence after resumption remains identical to an uninterrupted run;
- **Scalability:** Achieves superior data loading throughput via tiered caching mechanisms, robustly scaling to large distributed clusters while regulating request frequency to object storage within acceptable bounds.

Furthermore, to uphold uniform dataset quality standards, we have built a unified platform overseeing data registration, visualization, statistical analysis, cross-cloud synchronization, and lifecycle governance.

D Unified Agentic Reinforcement Learning Environment

Environment To support unified Agentic RL, our RL framework features a standardized Gym-like [10] interface to streamline the implementation of diverse environments. Such design empowers users to implement and customize

B.3 视觉数据

我们的多模态预训练语料库包含七个类别：文本、交错、OCR、知识、感知、视频和代理数据。文本数据 [49, 19] 提供基本的模态对齐，对合成文本有严格限制以减轻幻觉。来自书籍、网页和教程的图像-文本交错数据 [82, 32] 支持多图像理解和更长上下文学习。OCR数据涵盖多语言文本、密集布局和多页文档。知识数据包含通过布局解析器处理的学术材料，以发展视觉推理能力。

此外，我们精心构建了一个专门的多模态问题解决语料库，以增强科学、技术、工程和数学领域的推理能力。这些数据通过定向检索和网络爬虫收集；对于缺乏明确查询格式的信息内容，我们采用上下文学习 [11] 自动将原始材料重构为结构化的学术问题，涵盖K-12至大学级别。为了弥合视觉布局与代码数据之间的模态差距，我们整合了大量的图像-代码配对数据。这包括多种代码格式——如HTML、React和SVG等——及其对应的渲染截图，使模型能够将抽象的结构逻辑与具体的视觉几何对齐。

为了实现自主代理式和时序理解，我们收集了跨桌面、移动和网页环境的GUI截图和操作轨迹，包括人工标注的演示。来自不同来源的视频数据支持小时级长视频理解和细粒度的时空感知。此外，我们引入了用于像素级感知学习的轮廓级分割任务 [51] 。所有数据都经过严格的过滤、去重和质量控制，以确保高度的多样性和有效性。

C 基础设施

Kimi K2.5 在 NVIDIA H800 GPU 集群上训练，节点间使用 8×400 Gbps RoCE 互连。我们采用灵活的并行策略，结合 16 路流水线并行（PP）、虚拟阶段 [27, 40], 16 路专家并行（EP） [33], 和 ZeRO-1 数据并行，可在任何 32 的倍数节点上训练。EP 全对全通信与计算在交错 1F1B 调度下重叠。为适应 GPU 内存限制，我们对 LayerNorm、SwiGLU 和 MLA 上投影应用选择性重计算，将不敏感的激活压缩为 FP8-E4M3，并将剩余激活通过重叠流式传输卸载到 CPU。

C.1 数据存储和加载

我们采用来自云服务提供商的 S3 [3] 兼容对象存储解决方案来存储我们的 VLM 数据集。为弥合数据准备与模型训练之间的差距，我们保留视觉数据的原生格式，并设计了一套高效且可适应的数据加载基础设施。该基础设施提供了以下关键优势：

- **灵活性：** 支持训练过程中的动态数据洗牌、融合、分词、损失掩码和序列打包，可根据需求调整数据比例；
- **增强：** 允许对视觉和文本模态进行随机增强，同时在几何变换过程中保持二维空间坐标和方向元数据的完整性；
- **确定性：** 通过严谨管理随机种子和工作状态，确保完全确定性的训练，确保任何训练中断都可以无缝恢复——恢复后的数据序列与不间断运行时完全一致；
- **可扩展性：** 通过分层缓存机制实现卓越的数据加载吞吐量，在大型分布式集群中稳健扩展，同时将请求频率调节在对象存储的可接受范围内。

此外，为了维护统一的数据集质量标准，我们已构建了一个统一平台，负责数据注册、可视化、统计分析、跨云同步和生命周期管理。

D 统一代理式强化学习环境

环境为了支持统一的自主代理式强化学习，我们的强化学习框架具备标准化的Gym-like [10] 接口，以简化多样化环境的实现。这种设计赋予用户实现和定制的能力

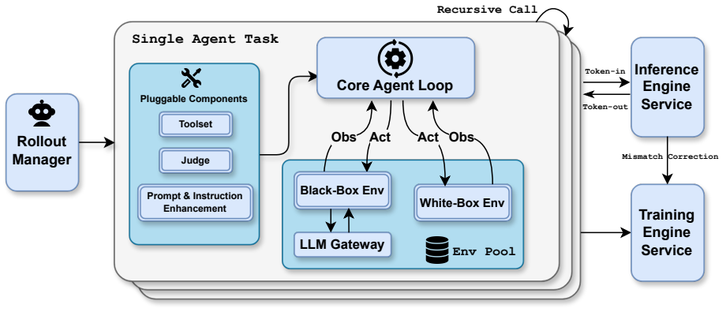


Figure 10: Overview of our agentic RL framework.

environments with minimal overhead. Our design prioritizes compositional modularity by integrating a suite of pluggable components, such as a *Toolset* module for supporting various tools with sandboxes, a *Judge* module for multi-faceted reward signals, and specialized modules for prompt diversification and instruction-following enhancement. These components can be dynamically composed with core agent loops, offering high flexibility and enhancing model generalization.

At the execution level, our RL framework treats every agent task as an independent asynchronous coroutine. Each task can recursively trigger sub-task rollouts, simplifying the implementation of complex multi-agent paradigms such as *Parallel-Agent RL* and *Agent-as-Judge*. As shown in the figure 10, a dedicated `Rollout Manager` orchestrates up to 100,000 concurrent agent tasks during the RL process, providing fine-grained control to enable features like partial rollout [31]. Upon activation, each task acquires an environment instance from a managed pool, equipped with a sandbox and specialized tools.

Inference Engine Co-design Our framework strictly follows a *Token-in-Token-out* paradigm. We also record log probabilities for all inference engine outputs to perform train-inference mismatch correction, ensuring stable RL training. A co-design of inference engine for RL requirements has allowed us to support these features by custom inference APIs for RL.

Besides a comprehensive suite of built-in white-box environments, there are also black-box environments that can only run under standard LLM API protocol, missing the opportunity to use advanced features offered by our custom API protocol. To facilitate model optimization under black-box environments, we developed *LLM Gateway*, which is a proxy service that keeps detailed records of rollout requests and responses under our custom protocol.

Monitoring and debugging It is a challenging task to optimize performance of a highly-parallel asynchronous execution system, while ensuring correctness. We develop a series of tools for performance monitoring, profiling, data visualization and data verification. We found these to be instrumental in debugging and ensuring both the efficiency and correctness of our Agentic RL.

E Evaluation Settings

This section provides comprehensive configuration details and testing protocols for all benchmarks reported in Table 4.

E.1 General Evaluation Protocol

Unless explicitly stated otherwise, all experiments for Kimi-K2.5 adhere to the following hyperparameter configuration:

- **Temperature:** 1.0
- **Top-p:** 0.95
- **Context Length:** 256k tokens

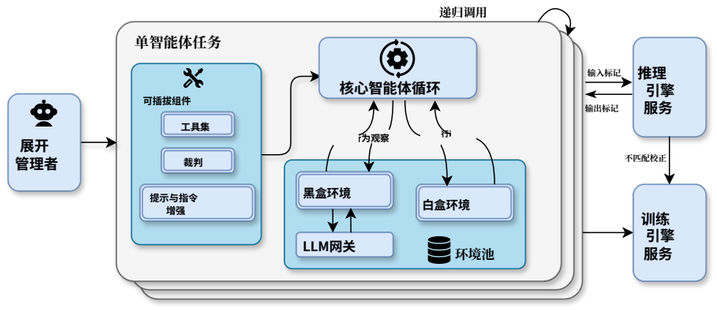


图 10：我们自主代理式强化学习框架的概述。

环境的能力，且开销极小。我们的设计优先考虑组合式模块化，通过集成一系列可插拔组件，例如用于支持带沙箱的各类工具的工具集模块、用于多维度奖励信号的裁判模块，以及用于提示多样化和指令跟随增强的专用模块。这些组件可以与核心智能体循环动态组合，提供高灵活性并增强模型泛化能力。

在执行层面，我们的强化学习框架将每个代理任务视为一个独立的异步协程。每个任务可以递归地触发子任务展开，简化了复杂多代理范式（如 并行代理式强化学习 和 代理式裁判）的实现。如图 10所示，一个专用的 展开管理器 在强化学习过程中可协调高达 100,000 个并发代理任务，提供细粒度控制以支持部分展开 [31]。任务激活时，会从管理池中获取一个环境实例，该实例配备沙箱和专用工具。

推理引擎协同设计 我们的框架严格遵循 输入标记-输出标记 范式。我们还记录所有推理引擎输出的日志概率，以执行训练-推理不匹配校正，确保稳定的强化学习训练。针对强化学习需求的推理引擎协同设计，使我们能够通过自定义的强化学习推理 API 支持这些功能。

除了全面的内置白盒环境外，还有黑盒环境，这些环境只能运行在标准LLM API协议下，无法使用我们自定义API协议提供的先进功能。为了在黑盒环境下进行模型优化，我们开发了<样式 id='1'>LLM网关</样式>，这是一个代理服务，用于记录我们自定义协议下的展开请求和响应的详细记录。

<样式 id='1'>监控和调试</样式>优化高度并行异步执行系统的性能，同时确保正确性是一项具有挑战性的任务。我们开发了一系列用于性能监控、分析、数据可视化和数据验证的工具。我们发现这些工具在调试和确保我们自主代理式强化学习（Agentic RL）的效率和正确性方面发挥了重要作用。

E 评估设置

<样式 id='1'>本节提供了表</样式><样式 id='3'>4</样式><样式 id='5'>中报告的所有基准测试的全面配置细节和测试协议。</样式>

E.1 一般评估协议

除非另有说明，所有Kimi-K2.5的实验均遵循以下超参数配置：

- **温度：** 1.0
- **Top-p:**0.95
- **上下文长度：** 256k 标记

E.2 Baselines

For baseline models, we report results under their respective high-performance reasoning configurations:

- **Claude Opus 4.5:** Extended thinking mode
- **GPT-5.2:** Maximum reasoning effort (`xhigh`)
- **Gemini 3 Pro:** High thinking level
- **DeepSeek-V3.2:** Thinking mode enabled (for text-only benchmarks)
- **Qwen3-VL-235B-A22B:** Thinking mode (for vision benchmarks only)

For vision and multimodal benchmarks, GPT-5.2-xhigh exhibited an approximate 10% failure rate (i.e., no output generated despite three retry attempts) during vision evaluations. These failures were treated as incorrect predictions, meaning that the reported scores may be conservative lower bounds of the model's true capability.

In addition, because we were unable to consistently access a stable GPT-5.2 API, we skipped some benchmarks with high evaluation costs, such as WideSearch.

E.3 Text Benchmarks

Reasoning Benchmarks. For high-complexity reasoning benchmarks, including HLE-Full, AIME 2025, HMMT 2025, GPQA-Diamond, and IMO-AnswerBench, we enforce a maximum completion budget of 96k tokens to ensure sufficient reasoning depth. To reduce variance arising from stochastic reasoning paths, results on AIME 2025 and HMMT 2025 (Feb) are averaged over 64 independent runs (Avg@64), while GPQA-Diamond is averaged over 8 runs (Avg@8).

LongBench v2. For a fair comparison, we standardize all input contexts to approximately 128k tokens using the same truncation strategy as in [9]. We observe that GPT5.2-xhigh frequently produces free-form question–answer style responses rather than the required multiple-choice format. Therefore, we report results using GPT5.2-high, which consistently adheres to the expected output format.

E.4 Image and Video Benchmarks

All image and video understanding evaluations utilize the following configuration:

- **Maximum Tokens:** 64k
- **Sampling:** Averaged over 3 independent runs (Avg@3)

ZeroBench (w/ tools). Multi-step reasoning evaluations use constrained step-wise generation:

- **Max Tokens per Step:** 24k
- **Maximum Steps:** 30

MMMU-Pro. We adhere strictly to the official evaluation protocol: input order is preserved for all modalities, with images prepended to text sequences as specified in the benchmark guidelines.

Sampling Strategies for Video Benchmarks. For short video benchmarks (VideoMMMU, MMVU & Motion-Bench), we sample 128 uniform input frames with a maximum spatial resolution at 896; 2048 uniform frames are sampled for long video benchmarks (Video-MME, LongVideoBench & LVBench) with 448 spatial resolution.

Specialized Metrics.

- **OmniDocBench 1.5:** Scores are computed as $(1 - \text{normalized Levenshtein distance}) \times 100$, where higher values indicate superior OCR and document understanding accuracy.
- **WorldVQA:** Access available at https://github.com/MoonshotAI/WorldVQA. This benchmark evaluates atomic, vision-centric world knowledge requiring fine-grained visual recognition and geographic understanding.

E.2 基线

For baseli 对于这些模型，我们在它们各自的高性能推理配置下报告结果 :

- **ClaudeOpus 4.5:** 扩展思考模式
- **GPT-5.2:** 最大推理力度（xhigh）
- **Gemini3 Pro:** 思考水平高
- **DeepSeek-V3.2:** 已启用思考模式（适用于纯文本基准测试）
- **Qwen3-VL-235B-A22B:** 思考模式（仅适用于视觉基准测试）

对于视觉和多模态基准测试，GPT-5.2-xhigh在视觉评估期间表现出约10%的失败率（即，尽管尝试了三次重试仍未生成输出）。这些失败被视为错误预测，这意味着报告的分数可能是模型真实能力的保守下限。

此外，还因为 w e 我们无法持续访问稳定的 GPT-5.2 API，因此跳过了部分基准测试
高评估成本， 例如 WideSearch。

E.3 文本基准测试

推理基准测试。 对于高复杂度推理基准测试，包括 HLE-Full、AIME 2025、HMMT 2025、GPQA-Diamond 和 IMO-AnswerBench，我们强制执行最大完成预算为 96k 标记，以确保足够的推理深度。为了减少随机推理路径产生的方差，AIME 2025 和 HMMT 2025（2月）的结果在 64 次独立运行中取平均（Avg@64），而 GPQA-Diamond 在 8 次运行中取平均（Avg@8）。

LongBench v2. 为了公平比较，我们使用与 [9]相同的截断策略，将所有输入上下文标准化为约 128k 标记。我们观察到 GPT5.2-xhigh 经常产生自由形式的问答风格响应，而不是所需的单选题格式。因此，我们使用始终遵循预期输出格式的 GPT5.2-high 报告结果。

E.4 图像和视频基准测试

所有图像和视频理解评估都使用以下配置：

- **最大标记数：** 64k
- **采样：** 在3次独立运行中取平均值（Avg@3）

ZeroBench（带工具）。 多步推理评估使用约束的逐步生成：

- **每步最大标记数：** 24k
- **最大步数：** 30

MMMU-Pro. 我们严格遵循官方评估协议：所有模态的输入顺序均保持不变，图像按照基准测试指南的要求预置于文本序列之前。

视频基准测试的采样策略。 对于短视频基准测试（VideoMMMU、MMVU & Motion-Bench），我们采样128帧均匀输入帧，最大空间分辨率为896；对于长视频基准测试（Video-MME、长视频基准测试& LVBench），我们采样2048帧均匀帧，空间分辨率为448。

专用指标。

- **OmniDocBench 1.5:** 分数计算为 $(1 - \text{归一化Levenshtein距离}) \times 100$，其中较高值表示OCR和文档理解精度更高。
- **WorldVQA:** 可在 https://github.com/MoonshotAI/WorldVQA访问。此基准测试评估原子、以视觉为中心的世界知识，需要细粒度的视觉识别和地理理解。

E.5 Coding and Software Engineering

Terminal Bench 2.0. All scores are obtained using the default Terminus-2 agent framework with the provided JSON parser. Notably, we evaluate under **non-thinking mode** because our current context management implementation for thinking mode is technically incompatible with Terminus-2's conversation state handling.

SWE-Bench Series. We employ an internally developed evaluation framework featuring a minimal tool set: `bash`, `create_file`, `insert`, `view`, `str_replace`, and `submit`. System prompts are specifically tailored for repository-level code manipulation. Peak performance is achieved under **non-thinking mode** across all SWE-Bench variants (Verified, Multilingual, and Pro).

CyberGym. Claude Opus 4.5 results for this benchmark are reported under non-thinking settings as specified in their technical documentation. We report scores in the difficulty level 1 (the primary setting).

PaperBench. We report the scores under the CodeDev setting.

Sampling. All coding task results are averaged over 5 independent runs (Avg@5) to ensure stability across environment initialization and non-deterministic test case ordering.

E.6 Agentic Evaluation

Tool Setting. Kimi-K2.5 is equipped with web search tool, code interpreter (Python execution environment), and web browsing tools for all agentic evaluations, including HLE with tools and agentic search benchmarks (BrowseC-omp, WideSearch, DeepSearchQA, FinSearchComp T2&T3 and Seal-0).

Context Management Strategies. To handle the extended trajectory lengths inherent in complex agentic tasks, we implement domain-specific context management protocols. Unless otherwise specified below, **no context management** is applied to agentic evaluations; tasks exceeding the model's supported context window are directly counted as failures rather than truncated.

- **Humanity's Last Exam (HLE).** For the HLE tool-augmented setting, we employ a *Hide-Tool-Result Context Management* strategy: when the context length exceeds predefined thresholds, only the most recent round of tool messages (observations and return values) is retained, while the reasoning chain and thinking processes from all previous steps are preserved in full.
- **BrowseComp.** For BrowseComp evaluations, our evaluation contains both with and without context management settings. Under the context management setting, we adopt the same *discard-all* strategy proposed by DeepSeek, where all history is truncated once token thresholds are exceeded.

System Prompt. All agentic search and HLE evaluations utilize the following unified system prompt, where `DATE` is dynamically set to the current timestamp:

```
You are Kimi, today's date: DATE.
Your task is to help the user with their questions by using various tools,
thinking deeply, and ultimately answering the user's questions.

Please follow the following principles strictly during the deep research:
1. Always focus on the user's original question during the research process,
   avoiding deviating from the topic.
2. When facing uncertain information, use search tools to confirm.
3. When searching, filter high-trust sources (such as authoritative websites,
   academic databases, and professional media) and maintain a critical mindset
   towards low-trust sources.
4. When performing numerical calculations, prioritize using programming tools
   to ensure accuracy.
5. Please use the format [^index^] to cite any information you use.
6. This is a **Very Difficult** problem——do not underestimate it. You must use
   tools to help your reasoning and then solve the problem.
7. Before you finally give your answer, please recall what the question is
   asking for.
```


E.5 编程与软件工程

TerminalBench 2.0。 所有分数都是使用提供的 JSON 解析器，通过默认的 Terminus-2 代理框架获得的。值得注意的是，我们评估时处于 **非思考模式**，因为当前用于思考模式的上下文管理实现与 Terminus-2 的对话状态处理在技术上不兼容。

SWE-Bench 系列。 我们采用一个内部开发的评估框架，包含极简的工具集：bash、create_file、insert、view、str_replace 和 submit。系统提示针对仓库级代码操作进行了专门定制。在所有 SWE-Bench 变体（Verified、Multilingual 和 Pro）中，**非思考模式** 下均达到峰值性能。

CyberGym。 Claude Opus 4.5 在此基准测试中的结果，根据其技术文档中指定的非思考设置进行报告。我们报告难度级别 1（主要设置）的分数。

PaperBench。 我们在 CodeDev 设置下报告分数。

采样。 所有编程任务的结果在5次独立运行中取平均值（Avg@5）以确保跨环境初始化和非确定性测试用例排序的稳定性。

E.6 自主代理式评估

工具设置。 Kimi-K2.5配备了网络搜索工具、代码解释器（Python执行环境）和网络浏览工具，用于所有自主代理式评估，包括带工具的人类最后考试（HLE）和自主代理式搜索基准测试（BrowseC-omp、WideSearch、DeepSearchQA、FinSearchComp T2&T3和Seal-0）。

上下文管理策略。 为了处理复杂自主代理式任务中固有的扩展轨迹长度，我们实现了特定领域的上下文管理协议。除非以下另有说明，**不应用上下文管理**；超出模型支持上下文窗口的任务将直接计为失败，而不是被截断。

- **人类最后考试（HLE）**。 对于HLE工具增强设置，我们采用了一种隐藏工具结果上下文管理 策略：当上下文长度超过预定义阈值时，仅保留最近一轮的工具消息（观察值和返回值），而所有先前步骤的推理链和思考过程将完整保留。
- **BrowseComp.** 对于 BrowseComp 评估，我们的评估包含有和没有上下文管理设置的两种情况。在上下文管理设置下，我们采用了 DeepSeek 提出的相同 *Discard-all* 策略，一旦超过 token 阈值，所有历史记录将被截断。

系统提示。 所有 代理式搜索和HLE评估使用以下统一的系统提示，其中DAT E 是动态设置为 th 当前时间戳:

```
你是Kimi，今天的日期：日期。你的任务是使用各种工具，深入思考，并最终回答用户的问题。

在进行深度研究时，请严格遵循以下原则：1. 研究过程中始终聚焦用户原始问题，避免偏离主题。2. 遇到
不确定信息时，使用搜索工具进行确认。3. 搜索时筛选高信任来源（如权威网站、学术数据库和专业媒体），
对低信任来源保持批判性思维。4. 进行数值计算时，优先使用编程工具以确保准确性。5. 请使用格式 [^
索引^]来引用您使用过的任何信息。6. 这是一个**非常困难**的问题——不要低估它。您必须使用工具来
辅助您的推理，然后才能解决问题。7. 在最终给出答案前，请回忆问题要求的是什么。
```

Sampling Protocol. To account for the inherent stochasticity in search engine result rankings and dynamic web content availability, results for Seal-0 and WideSearch are averaged over 4 independent runs (Avg@4). All other agentic benchmarks are evaluated under single-run protocols unless explicitly stated otherwise.

E.7 Computer-Use Evaluation

Hyperparameter Settings. We set `max_steps_per_episode` = 100 for all experiments, with `temperature` = 0 for OSWorld-Verified and `temperature` = 0.1 for WebArena. Due to resource constraints, all models are evaluated in a one-shot setting. Adhering to the OpenCUA configuration [63], the agent context includes the last 3 history images, the complete thought history, and the task instruction. For WebArena, we manually corrected errors in the evaluation scripts and employed `GPT-4o` as the judge model for the `fuzzy_match` function. To ensure fair comparison, Claude Opus 4.5 is evaluated solely with computer-use tools (excluding browser tools), a departure from the System Card configuration [6].

System Prompt We utilize a unified system prompt for all computer use tasks:

```
You are a GUI agent. You are given an instruction, a screenshot of the screen and your
previous interactions with the computer. You need to perform a series of actions to
complete the task. The password of the computer is {password}.

For each step, provide your response in this format:
{thought}
## Action:
{action}
## Code:
{code}

In the code section, the code should be either pyautogui code or one of the following
functions wrapped in the code block:
- {"name": "computer.wait", "description": "Make the computer wait for 20 seconds
for installation, running code, etc.", "parameters": {"type": "object", "properties":
{}, "required": []}}
- {"name": "computer.terminate", "description": "Terminate the current task and report
its completion status", "parameters": {"type": "object", "properties": {"status":
{"type": "string", "enum": ["success", "failure"], "description": "The status of the
task"}, "answer": {"type": "string", "description": "The answer of the task"}},
"required": ["status"]}}
```

E.8 Agent Swarm Configuration

Tool Setting. In addition to the core toolset described in Appendix E.6 (web search, code interpreter, and web browsing), the orchestrator is equipped with two specialized tools for sub-agent creation and scheduling:

- `create_subagent`: Instantiates a specialized sub-agent with a custom system prompt and identifier for reuse across tasks.
- `assign_task`: Dispatches assignments to created sub-agents.

The tool schemas are provided below:

```
{
 "name": "create_subagent",
 "description": "Create a custom subagent with specific system prompt
   and name for reuse.",
 "parameters": {
   "type": "object",
   "properties": {
     "name": {
       "type": "string",
       "description": "Unique name for this agent configuration"
     },
     "system_prompt": {
       "type": "string",
```


采样协议。为考虑搜索引擎结果排名的固有随机性和动态网页内容可用性的变化，Seal-0和WideSearch的结果在4次独立运行中取平均值（Avg@4）。所有其他自主代理式基准测试均在单次运行协议下进行评估，除非另有说明。

E.7 计算机使用评估

超参数设置。我们设置 `max_步数_每_个` = 100 实验 = 100，其中温度 = 0 用于 OSWorld-验证，温度 = 0.1 用于 WebArena。由于资源限制，所有模型均在单次设置下进行评估。遵循 OpenCUA 配置 [63]，代理上下文包括最后 3 个历史图像、完整的思维历史和任务指令。对于 WebArena，我们手动更正了评估脚本中的错误，并采用`GPT-4o` 作为模糊_匹配功能的裁判模型。为确保公平比较，Claude Opus 4.5 仅使用计算机使用工具（不包括浏览器工具）进行评估，这与系统卡配置 [6]不同。

系统提示 我们为所有计算机使用任务使用统一的系统提示：

你是 GUI 代理。你将获得一个指令、屏幕截图以及你与计算机的先前交互。你需要执行一系列操作来完成任务。计算机的密码是 {password}。

对于每个步骤，请以以下格式提供您的响应：{thought} ## 操作：
{action} ## 代码：{code}

在代码部分，代码应为pyautogui代码或以下函数之一，并用代码块包裹：

- {"name": "computer.wait", "description": "使计算机等待20秒进行安装、运行代码等", "parameters": {"type": "object", "properties": {}, "required": []}}
- {"name": "computer.terminate", "description":"终止当前任务并报告其完成状态", "parameters": {"type": "object", "properties": {"status": {"type": "string", "enum": ["success", "failure"], "description": "任务的状态"}, "answer": {"type": "string", "description": "任务的答案"}}, "required": ["status"]}}

E.8 Agent Swarm Configuration

工具设置。除了附录 E.6 (网络搜索、代码解释器和网络浏览)中描述的核心工具集之外，编排器配备了两个用于子代理创建和调度的专用工具：

- 创建_子代理：使用自定义系统提示和标识符实例化一个专门的子代理，以便跨任务重用。
- 分配_任务：将任务分配给已创建的子代理。

工具模式如下提供：

{ "name": "create_subagent", "description":"创建一个具有特定系统提示和名称的自定义子代理，以便重复使用。", "parameters": { "type": "object", "properties": { "name": { "type": "string", "description":"此代理配置的唯一名称" }, "system_prompt": { "type": "string",

```
      "description": "System prompt defining the agent's role,
        capabilities, and boundaries"
    }
  },
  "required": ["name", "system_prompt"]
 }
}
{
 "name": "assign_task",
 "description": "Launch a new agent.\nUsage notes:\n
   1. You can launch multiple agents concurrently whenever possible,
      to maximize performance;\n
   2. When the agent is done, it will return a single message back to you.",
 "parameters": {
   "type": "object",
   "properties": {
     "agent": {
       "type": "string",
       "description": "Specify which created agent to use."
     },
     "prompt": {
       "type": "string",
       "description": "The task for the agent to perform"
     }
   },
   "required": ["agent", "prompt"]
 }
}
```

Step Limits. When operating in Agent Swarm mode, we set computational budgets for the orchestrator and sub-agents. Step limits apply to the aggregate count of tool invocations and environment interactions.

- **BrowseComp:** The orchestrator is constrained to a maximum of 15 steps. Each spawned sub-agent operates under a limit of 100 steps (i.e., up to 100 tool calls per sub-agent).
- **WideSearch:** Both the orchestrator and each sub-agent are allocated a maximum budget of 100 steps.
- **In-house Bench:** The orchestrator is constrained to a maximum of 100 steps. Each spawned sub-agent operates under a limit of 50 steps .

System Prompt.

```
You are Kimi, a professional and meticulous expert in information collection and organization.
You fully understand user needs, skillfully use various tools, and complete tasks with the
highest efficiency.
# Task Description
After receiving users' questions, you need to fully understand their needs and think
about and plan how to complete the tasks efficiently and quickly.
# Available Tools
To help you complete tasks better and faster, I have provided you with the following tools:
1. Search tool: You can use the search engine to retrieve information, supporting multiple
queries in parallel.
2. Browser tools: You can visit web links (web pages, PDFs, etc.), get page content, and
perform interactions such as clicking, inputting, finding, and scrolling.
3. Sub Agent tools:
   - `create_subagent`: Create a new sub-agent with a unique name and clear, specific
   system prompt.
   - `assign_task`: Delegate tasks to created sub-agents. Sub-agents can also use search
   and browser tools.
4. Other tools: Including code execution (IPython, Shell).
```

E.9 GDPVal

We cite the GDPVal-AA evaluation by Artificial Analysis, and the scores reported in Table 4 reflect the official leaderboard metrics as of January 28, 2026.


```
"description":"定义代理的角色、功能和边界"}}, "required": ["name", "system_prompt"]}}
{"name": "assign_task", "description":"启动一个新的代理。\n使用说明:\n 1. 尽可能多地同时启动
多个代理，以最大化性能；\n 2. 代理完成后，将返回一条消息给您。", "parameters": {"type": "
object", "properties": {"agent": {"type": "string", "description":"指定要使用的已创建代理。"}, "
prompt": {"type": "string", "description":"代理要执行的任务"}}, "required": ["agent", "
prompt"]}}
```

步骤限制。 在 Agent Swarm 模式下，我们为编排器和子代理设置计算预算。步骤限制适用于工具调用和环境交互的总计数。

- **BrowseComp:** 编排器被限制在最多15个步骤。每个生成的子代理在100个步骤的限制下运行（即每个子代理最多100个工具调用）。
- **WideSearch:** 编排器和每个子代理都被分配了最多100个步骤的预算。
- **内部基准：** 编排器最多限制为100个步骤。每个生成的子代理都在50个步骤的限制下运行。

系统提示。

```
你是Kimi，一位在信息收集和组织方面专业且细致的专家。                                                    n.
你完全理解用户需求，熟练使用各种工具，并以最高效率完成任务。# 任务描述在接收到用户的问题后，你需要充分理解他
们的需求，并思考如何高效快速地完成任务。# 可用工具为了帮助你更好地、更快地完成任务，我为你提供了以下工具：1.
搜索工具：你可以使用搜索引擎检索信息，支持并行多查询。2. 浏览器工具：你可以访问网页链接（网页、PDF等），获取
页面内容，并进行点击、输入、查找、滚动等交互。3. 子代理工具：- `create_subagent`：创建一个具有唯一名称和清
晰、具体系统提示的新子代理。- `assign_task`：将任务分配给创建的子代理。子代理也可以使用搜索和浏览器工具。
4. 其他工具：包括代码执行（IPython、Shell）。
```

E.9 GDPval

我们引用了由人工智能分析进行的 GDPVal-AA 评估，表 4 中报告的分数反映了截至 2026 年 1 月 28 日的官方排行榜指标。

Figure 11: Qualitative example of Kimi K2.5 analyzing a complete playthrough of *Black Myth: Wukong* (24 hours of continuous gameplay across 32 videos at 1080p) using parallel visual agents. See generated webpage and source videos (all rights reserved by source authors).

图11：Kimi K2.5分析完整通关过程的定性示例黑神话：悟空(24小时连续游戏，覆盖32个1080p视频)使用并行视觉代理。参见生成网页和源视频(所有权利均归源作者所有)。

USER

Please help the ant walk through the maze. Do not cross walls.

THINK

The user uploaded a maze image and asked me to help find a path from the start to the end without crossing walls, using the ipython tool for assistance.
First, I need to examine the maze image to determine the start and end positions:
Looking at the image, there is a small bell on the left, which is likely the start.
There is an ant or insect character on the right, which is likely the end.
The maze itself is an underground tunnel structure.
Let me use ipython to analyze this maze image and find the path.
Steps:
1. Analyze the maze structure (binarization, identify walls and paths).
2. Identify the start and end points.
3. Use a pathfinding algorithm (like BFS or A*) to find the shortest path.
4. Visualize the path.^p
Let me start writing the code.

SUMMARY

The little bell on the left is the starting point, and the little ant on the right is the finish line. Now, let me convert the maze into a computable format and find the path:

TOOL

<python: Binarization algorithm>

THINK

The binarization process worked well. Now I need to:
1. Manually mark the positions of the start (bell) and the end (ant).
2. Implement the BFS algorithm to find the path.
Based on the image:
1. The starting point (bell) is on the left, roughly around (50, 250).
2. The endpoint (ant) is on the right, roughly around (980, 450).

SUMMARY

Now, let me determine the coordinates for the start and end points, and use the BFS (Breadth-First Search) algorithm to find the shortest path:

TOOL

<python: Draw point and BFS>

The path contains 3,288 points.

THINK

Great! The path has been found, consisting of 3,288 points. Now I need to visualize it for the user. Let me create a colored path map and overlay the red path on the original image.

SUMMARY

The entire route consists of 3,288 steps. Now, let me plot the path onto the maze:

TOOL

<python: Draw path>

THINK

Perfect! The path has been successfully found and visualized.

SUMMARY

Final Path Characteristics:
📏 Total Length: 3,288 pixels
🚶 Route: Enters from the left → heads down → traverses right across the underground maze → winds upward → reaches the exit on the right.
🧭 Algorithm: BFS guarantees this is the shortest path with no backtracking.
🚧 Constraints: Strictly followed the corridors with no wall-clipping.
The maze path is clearly marked in the image above; the red line represents the walking route!

USER

What is the total percentage of the green and blue areas?

THINK

The user is asking for the total percentage of the green and blue areas in the image. The image shows a pie chart with three parts: Section 1 (Red), Section 2 (Blue), and Section 3 (Green).
Let me write code to:
1. Load the image
2. Identify the colored sectors
3. Calculate the percentage of each color
4. Sum green and blue
Actually, since this is a simple pie chart with distinct colors, I'll convert to HSV color space for better color segmentation, then mask each color and calculate the pixel counts.

SUMMARY

I'll help you analyze the percentage of the green and blue areas in this pie chart.

TOOL

<python: Sample color algorithm>
# Convert to HSV for better color segmentation
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

Define color ranges for each section...

Total colored pixels: 61609
Red/Pink pixels: 26994
Blue pixels: 13752
Green pixels: 18032
Sum of all colors: 58778

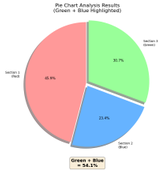
THINK

Good, I can see the masks are working but there's some discrepancy (58778 vs 61609 total pixels, probably due to black borders or anti-aliasing). Let me calculate the percentages based on the sum of the three detected colors, which should give us the relative proportions.

TOOL

<python: Visualization Results>
# Calculate percentages based on sum of detected colored pixels
total_detected = red_pixels + blue_pixels + green_pixels

red_pct = (red_pixels / total_detected) * 100
blue_pct = (blue_pixels / total_detected) * 100
green_pct = (green_pixels / total_detected) * 100
...



THINK

Perfect! I've successfully analyzed the pie chart using image processing techniques. The results show:
Section 1 (Red/Pink): 45.9%
Section 2 (Blue): 23.4%
Section 3 (Green): 30.7%
Total of Green + Blue = 54.1%
Given the standard pie chart proportions, this looks like roughly:
Red: ~46% (close to 50% or half)
Blue: ~23% (close to 25% or a quarter)
Green: ~31% (close to 30%)
So Green + Blue = ~54%, which is slightly more than half the pie chart.
The analysis is complete and I've provided both the detailed breakdown and the final answer.

SUMMARY

Based on the image analysis using Python, here are the results:
Individual Percentages:
Section 1 (Red/Pink): 45.9%
Section 2 (Blue): 23.4%
Section 3 (Green): 30.7%
Answer to your question: The total percentage of the green and blue areas is:
54.1%
(Green 30.7% + Blue 23.4% = 54.1%)
This means the green and blue sections combined make up slightly more than half of the entire pie chart.

USER

Find the differences in th picture.

THINK

This is a "Spot the Difference" game. The image shows the Flash the Sloth scene from Zootopia.
Task: Analyze the differences between the top and bottom images.
I need to load the image for pixel-level difference analysis.

SUMMARY

Now conducting a pixel-wise comparison to find all differences accurately:

TOOL

<python: Difference map>

THINK

Now I can see the obvious difference regions. Let me convert the contours into bounding boxes and normalize the coordinates to the original image dimensions.

SUMMARY

Extracting bounding boxes from the detected differences and transforming them to normalized coordinates.

TOOL

<python: Filter with threshold>

THINK

The automated detection identified 27 regions, but some may overlap or contain multiple differences. I will manually refine the primary points of interest based on visual observation and provide more precise coordinates.

SUMMARY

27 change regions were auto-detected, though many overlap. Let me manually refine these into the 8 major differences and output more accurate coordinates.

TOOL

<python: Filter main difference>

THINK

I have now identified all the differences and labeled their coordinates. Let me double-check for any other obvious discrepancies.
Upon further comparison, I also noticed:
The sloth's hair/fur is different (smoother in the top image, more frizzy in the bottom).
There are subtle differences in the leaves of the background plants.
The paper next to the stamp on the desk.
However, the 10 main differences should cover the most prominent changes. Let me summarize the final answer.

SUMMARY

After a thorough comparative analysis, a total of 10 major differences have been identified.
1. DMV vs EMV sign;
2. Calendar image variation;
3. Sloth expression: Closed vs Open mouth;
4. Nick's eye state: Squinting vs Open;
5. Judy's facial adjustments;
6. Crumpled paper vs Paper airplane;
7. Monitor display content;
8. Mug text change: 'YOU WANT IT WHEN?' vs 'I'LL GET TO IT';
9. Watch face/color variation;
10. File folder position/shape

Figure 12: Qualitative examples of Kimi K2.5 solving visual reasoning tasks via tool use.

USER

请帮助蚂蚁穿过迷宫。不要穿越墙壁。

思考

用户上传了一张迷宫图片，并要求我帮助找到一条从起点到终点且不穿越墙壁的路径。并使用 Python 工具协助。
使用 Python 工具协助。
首先，我需要检查迷宫图片以确定起点和终点位置：
观察图片，左边有一个小铃铛，这很可能是起点。
右侧有一个蚂蚁或昆虫角色，这可能是终点。
这个迷宫本身是一个地下隧道结构。
让我用Python来分析这个迷宫图像并找到路径。
步骤：
1. 分析迷宫结构（二值化，识别墙壁和路径）。
2. 确定起点和终点。
3.使用路径规划算法（如广度优先搜索或A*算法）来寻找最短路径。
4.可视化路径。^p
让我开始编写代码。

摘要

左侧的小铃铛是起点，右侧的小蚂蚁是终点。
线。现在，让我将迷宫转换为可计算的格式并寻找路径：

</>

<python:二值化算法>

思考

二值化过程运行良好。现在我需要：
1.手动标记起始（铃）和结束（蚂蚁）的位置。
2.实现广度优先搜索算法来寻找路径。
根据图像：
1. 起点（铃铛）在左侧，大致位于(50,250)。
2. 终点（蚂蚁）在右侧，大致位于(980,450)。

摘要

现在，我来确定起点和终点的坐标。并使用广度优先搜索(BFS)算法寻找最短路径：
广度优先搜索(BFS)算法来寻找最短路径：

</>

<python:绘制点和广度优先搜索>

路径包含3,288个点。

思考

太好了！路径已找到。包含3,288个点。现在我需要将其可视化
以供用户使用。让我创建一个彩色路径图，并将红色路径叠加到原始图像上
。

摘要

整个路线包含3,288个步骤。现在，让我在迷宫中绘制路径：

</>

<python:绘制路径>

思考

太棒了！路径已成功找到并可视化。

摘要

最终路径特征：
📏总长度：3,288像素
🚶路线：从左侧进入 → 向下走 → 横穿地下
迷宫 → 向上蜿蜒 → 到达右侧的出口。
🧭算法：广度优先搜索保证这是最短路径且无回溯。
🚧约束：严格遵循走廊且无墙切割。
迷宫路径在上方图像中清晰标注，红线代表
步行路线！

USER

绿色和蓝色区域的百分比是多少?

思考

用户正在询问图像中绿色和蓝色区域的总百分比。
图像显示一个饼图，分为三个部分：第一部分（红色）、第二部分（蓝色）和第三部分（绿色）。
让我们编写代码：
1.加载图像
2.识别彩色区域
3.计算每种颜色的百分比
4.将绿色和蓝色相加
实际上，由于这是一个具有不同颜色的简单饼图，我将转换为HSV颜色为更好的颜色分割视觉空间，然后对每种颜色进行遮罩并计算像素数量。

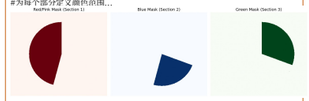
摘要

我会帮你分析这个饼图中绿色和蓝色区域的百分比。

</>

<python:示例颜色算法>
#转换为HSV以进行更好的颜色分割
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

#为每个部分定义颜色范围...



总着色像素：61609
红/粉红像素：26994
蓝色像素:13752
绿色像素:18032
所有颜色总和:58778

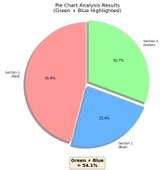
思考

好的，我能看到遮罩正在工作，但存在一些差异（58778 vs 61609
总像素，可能由于黑色边框或抗锯齿）。让我来计算一下
基于三种检测到的颜色的总和计算百分比，这应该能给我们
相对比例。

</>

<python:可视化结果>
#根据检测到的彩色像素总和计算百分比
total_detected =红色像素 +蓝色像素 +绿色像素

red_pct =(红色像素/total_detected)*100
蓝色占比 =(蓝色像素/总检测)*100
绿色占比 =(绿色像素/总检测)*100
总



思考

完美！我已成功使用图像处理技术分析了饼图。结果如下：
显示：
Section1(红/粉):45.9%
Section2(蓝):23.4%
Section3(绿):30.7%
Green 和Blue 的 54.1%
根据标准饼图比例，这看起来大约是：
红：~46%（接近50%或一半）
蓝色：~23%（接近25%或四分之一）
绿色：~31%（接近30%）
So Green 和蓝色 的~54%，这略高于饼图的一半。
分析已完成，我已提供详细的分解和最终
答案。

摘要

基于使用Python进行的图像分析，以下是结果：
个体百分比：
Section1(红/粉):45.9%
Section2(蓝):23.4%
Section3(绿):30.7%
回答你的问题：绿色和蓝色区域的总百分比是：
54.1%
(绿30.7% 和蓝23.4% 的 54.1%)
这意味着绿色和蓝色部分加起来占到了略微超过一半。
整个饼图。

USER

在图片中找出不同之处。

思考

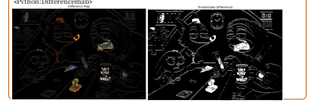
这是一个“找不同”游戏。图片展示了来自《疯狂动物城》的闪电鲶熊场景。
疯狂动物城。
任务：分析上下两张图片之间的差异。
我需要加载图片进行像素级差异分析。

摘要

现在进行像素级比较，以准确找出所有差异：

</>

<Python:Differencemap>



思考

现在我能看到明显的差异区域。让我将轮廓转换为
边界框，并将坐标规范化到原始图像尺寸。

摘要

从检测到的差异中提取边界框，并将其转换为
标准化坐标。

</>

<python:使用阈值过滤>

思考

自动化检测识别出27个区域，但其中一些可能存在重叠或包含多个差异。我将根据视觉观察手动优化主要关注点，并提供更精确的坐标。

摘要

27 个区域已被自动检测，尽管许多区域存在重叠。让我手动进行优化
这些区域可以归纳为6个主要差异，并输出更精确的坐标。

</>

<Python：过滤主要差异>

思考

我已经识别了所有差异并标注了它们的坐标。让我再检查一遍是否有其他明显的差异。
任何其他明显的差异。
进一步比较后，我还注意到：
树懒的毛发/毛皮不同（顶部图像更光滑，底部更蓬松）。
背景植物叶片存在细微差异。
桌子上的花瓣旁边。
然而，10个主要差异应该涵盖了最显著的变化。让我
总结最终答案。

摘要

经过全面对比分析，已发现10个主要差异。
已被确定。
1.DMVvsEMV标志;
2.日历图像变化;
3.树懒表情：闭嘴与张嘴;
4.尼克的眼部状态：眯眼与张眼;
5.朱迪面部调整;
6.揉皱纸张与纸飞机;
7.显示器显示内容;
8.马克杯文字变化：‘你想要它何时？’与‘我会去拿它’;
9.手表表盘颜色变化;
10.文件文件夹位置/形状

图12：Kimi K2.5通过工具使用解决视觉推理任务的定性示例。

F Visualization

Figure 11 demonstrates our Agent Swarm tackling a challenging long-form video understanding task: analyzing a complete playthrough of *Black Myth: Wukong* (24 hours of continuous gameplay across 32 videos, totaling 40GB). The system employs a hierarchical multi-agent architecture where a Main Agent orchestrates parallel Sub Agents to process individual video segments independently. Each sub agent performs frame extraction, temporal event analysis, and key moment identification (e.g., boss fights, level-ups). The Main Agent subsequently aggregates these distributed analyses to synthesize a comprehensive HTML showcase featuring chronological timelines, embedded video clips, and interactive visualizations. This example demonstrates the system's ability to handle massive-scale multimodal content through parallelization while maintaining coherent long-context understanding.

Figure 12 presents qualitative examples of Kimi K2.5 solving diverse visual reasoning tasks via tool-augmented reasoning. The model demonstrates: **(1)** Maze Solving—processing binary image segmentation and implementing pathfinding algorithms (BFS) to navigate complex mazes; **(2)** Pie Chart Analysis—performing pixel-level color segmentation and geometric calculations to determine precise area proportions; and **(3)** Spot-the-Difference—employing computer vision techniques to detect pixel-level discrepancies between image pairs. These examples highlight the model's capability to decompose complex visual problems into executable code, iteratively refine strategies based on intermediate results, and synthesize precise answers through quantitative visual analysis.

F 可视化

图 11 展示了我们的Agent Swarm如何应对一项具有挑战性的长视频理解任务：分析一个完整的《黑神话：悟空》游戏通关过程（跨越32个视频，连续游戏24小时，总计40GB）。该系统采用分层多智能体架构，主智能体协调并行工作的子智能体独立处理单个视频片段。每个子智能体执行帧提取、时序事件分析和关键时刻识别（例如，Boss战、升级）。随后，主智能体聚合这些分布式分析，合成一个包含时间线、嵌入视频片段和交互式可视化的HTML展示。该案例展示了系统通过并行化处理海量多模态内容的能力，同时保持对长时序上下文的理解。

图 12 展示了Kimi K2.5通过工具增强型推理解决多样化视觉推理任务的定性示例。该模型展示了： **(1)** 迷宫求解—处理二值图像分割并实现路径规划算法（广度优先搜索）以导航复杂迷宫； **(2)** 饼图分析—执行像素级颜色分割和几何计算以确定精确的面积比例；以及 **(3)** 找不同—运用计算机视觉技术检测图像对之间的像素级差异。这些示例突出了模型分解复杂视觉问题为可执行代码、基于中间结果迭代优化策略以及通过定量视觉分析综合精确答案的能力。