

## 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平行执行。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 $4.5\times$ 超过单代理基线<sup>1</sup>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。

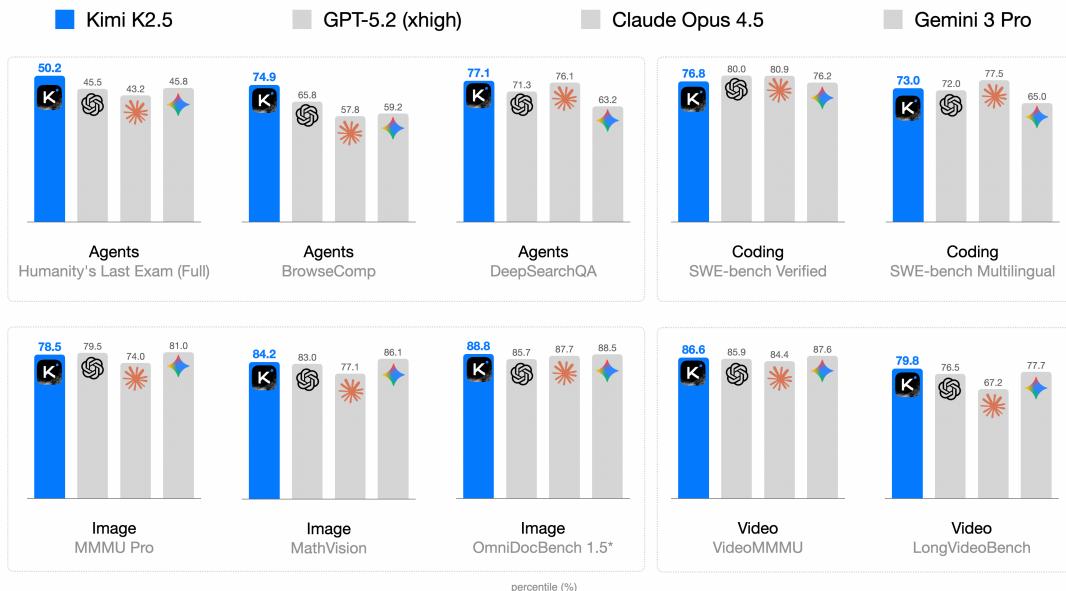


图1：Kimi K2.5主要结果。

## 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。

<sup>1</sup><https://huggingface.co/moonshotai/Kimi-K> 2.5

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。

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%	<b>25.8</b>	<b>43.8</b>	<b>65.7</b>	<b>45.5</b>	58.5	<b>24.8</b>
Mid	50%	20%:80%	25.0	40.7	64.1	43.9	<b>58.6</b>	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。

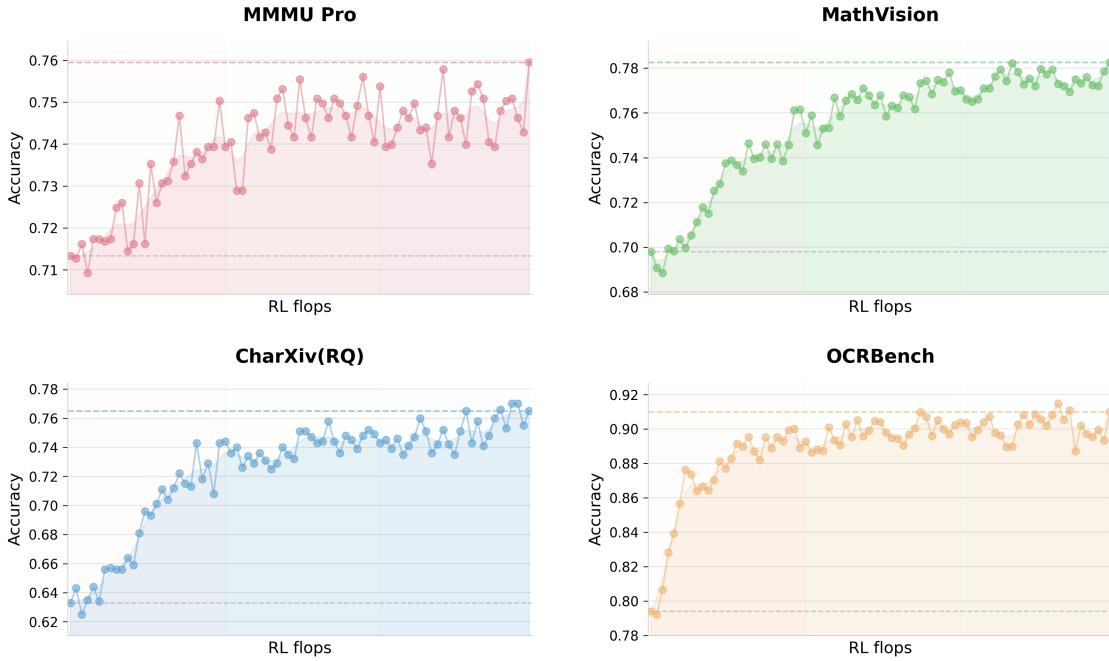


图2：从无视觉SFT开始，在视觉基准上进行视觉强化学习训练的曲线。通过扩展视觉RLFLOPs，性能持续提升，证明无视觉激活与长时间运行的强化学习足以获取稳健的视觉能力。

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，顺序系统往往遇到显著的瓶颈[5, 6, 7]。

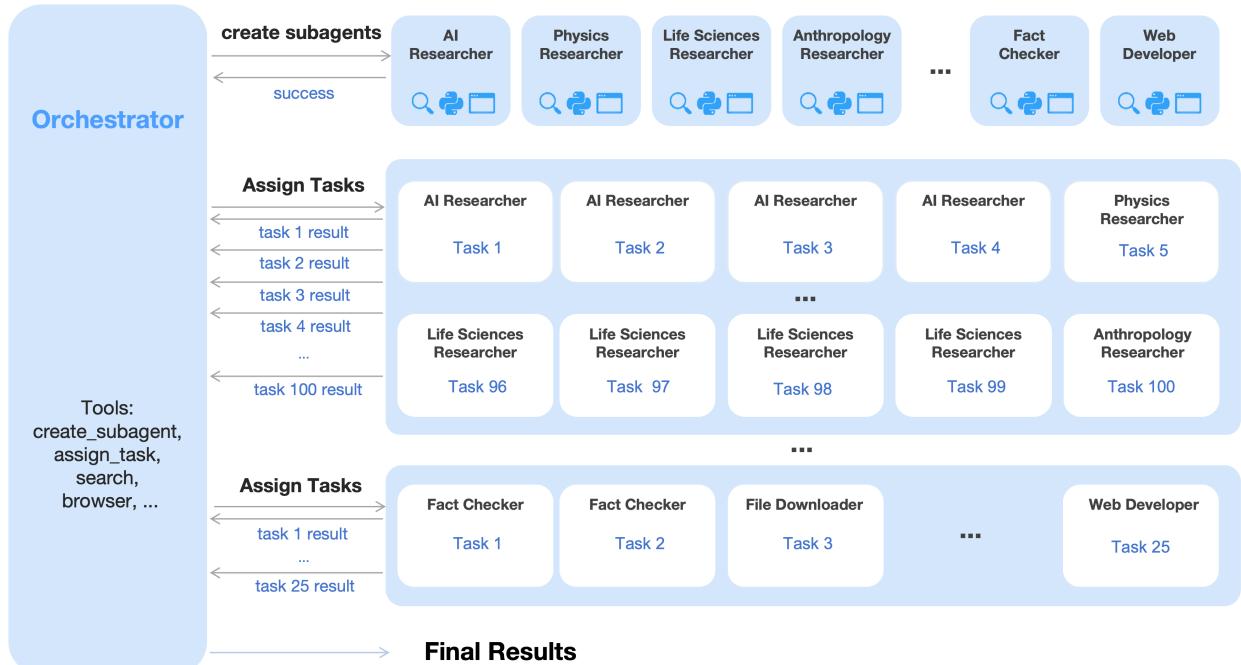


图3：一个代理群拥有可训练的编排器，该编排器动态创建专门化的冻结子代理，并将复杂任务分解为可并行化的子任务，以实现高效的分布式执行。

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  $r_{\text{perf}}$  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  $r_{\text{parallel}}$  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

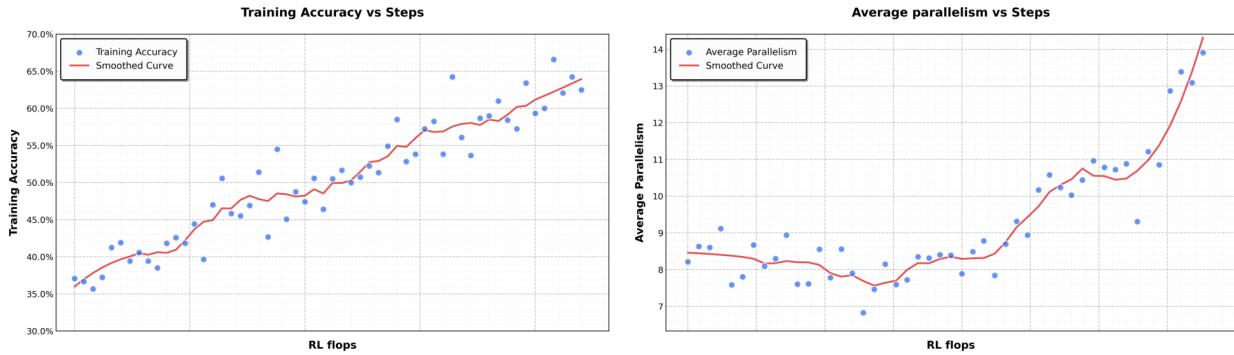


图4：在我们的并行代理强化学习环境中，随着训练的进行，训练准确率平滑增长。同时，训练过程中的并行度也逐渐提高。

空间。奖励  $r_{\text{finish}}$  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， $r_{\text{finish}}$  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  $\lambda_1$  和  $\lambda_2$  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  $t = 1, \dots, T$  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]，

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。

**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  $\pi_{\text{old}}$  生成  $K$  个响应  $\{y_1, \dots, y_K\}$ 。我们针对以下目标优化模型  $\pi_{\theta}$ ：

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

其中  $\alpha, \beta, \tau > 0$  是超参数， $y_0^j$ ： $i$  是第  $j$  个响应的前缀，直到第  $i$  个标记， $N = \sum_{i=1}^K |\mathbf{y}_i|$  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。

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

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

其中  $\lambda$  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  $\lambda$  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。

$$\text{budget}(x) = \text{Percentile}(\{|y_j| \mid r(x, y_i) = 1, i = 1, \dots, K\}, \rho). \quad (2)$$

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

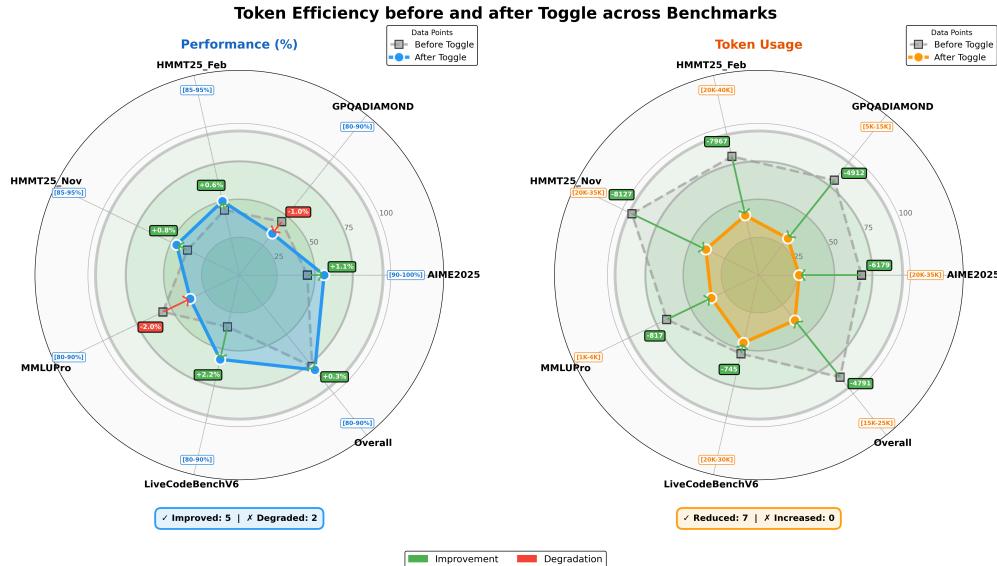


图 5：比较 Kimi K2 T 的性能和 token 使用情况 采用 token 高效强化学习进行思考。

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].

- **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], OJ Bench (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 <sup>2</sup>; (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 GPT-5.2 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。

<sup>2</sup><https://github.com/MoonshotAI/WorldVQA>

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

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	Gemini-3.0	DeepSeek-V3.2
	思考	Pro	思考	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 VideoMMU 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%。

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	<b>78.4</b>	60.6	37.0	65.8	77.9
WideSearch	<b>79.0</b>	72.7	76.2	-	-
内部群集基准	<b>58.3</b>	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。

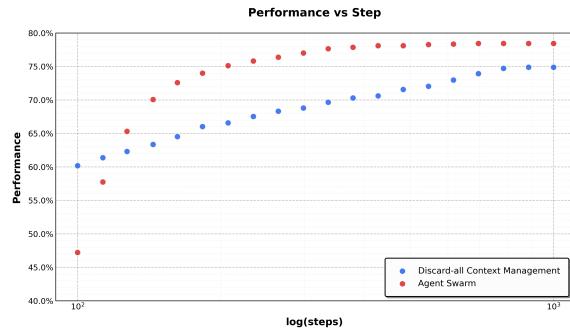


图7: 比较Kimi K2.5在Agent Swarm和Discard-all上下文管理中的性能表现（在BrowseComp中）。

(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

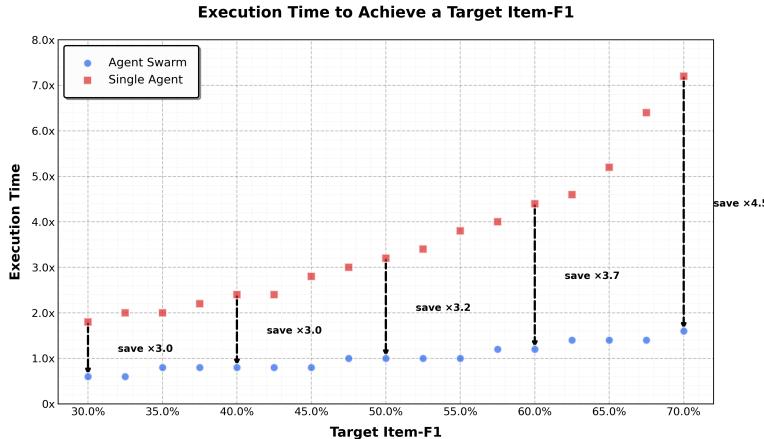


图8：在WideSearch测试中，随着目标Item-F1从30%增加到70%，Agent Swarm相比单代理基线实现了3×–4.5×的更快执行时间。

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。

## 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](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] 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] 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/3f5ee243547dee91fdbd053c1c4a845aa-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fdbd053c1c4a845aa-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](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. “OJ Bench: 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] 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).

## 贡献

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

作者名单按姓氏字母顺序排列。

†香港大学

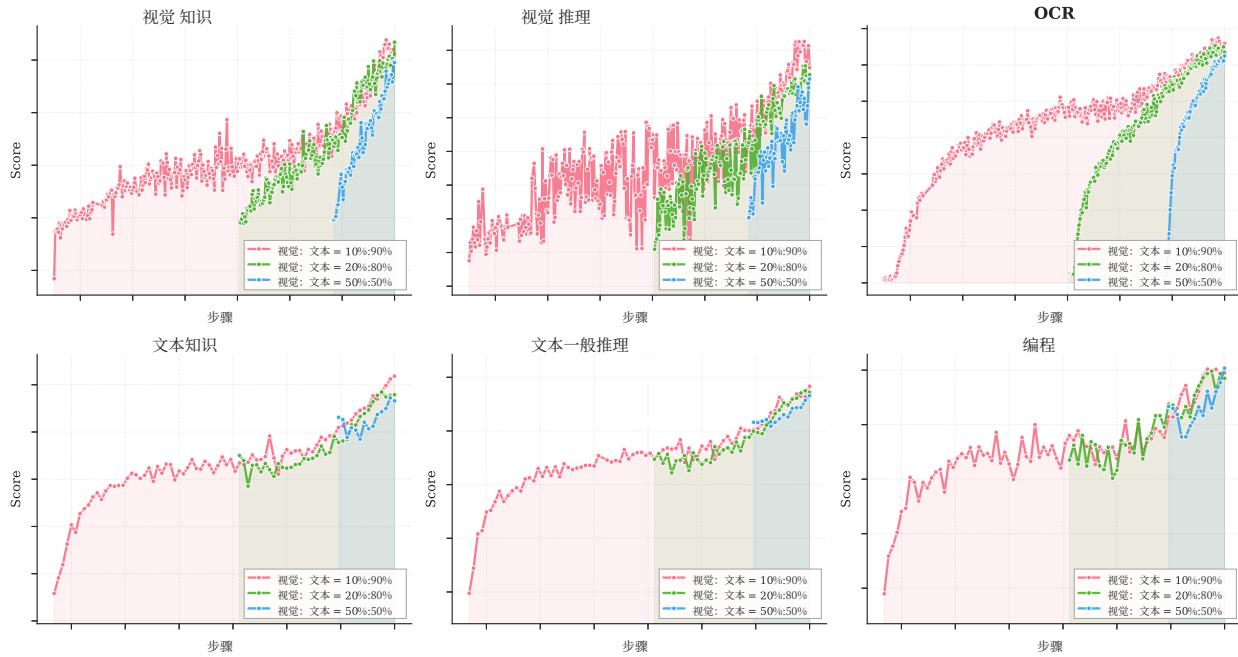


图 9：比较在不同视觉-文本标记预算下（10:90、20:80、50:50）的视觉到文本比例学习曲线。在视觉和语言任务中，早期融合且视觉比例较低时，往往能获得更好的结果。

## 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 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  $8 \times 400$  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

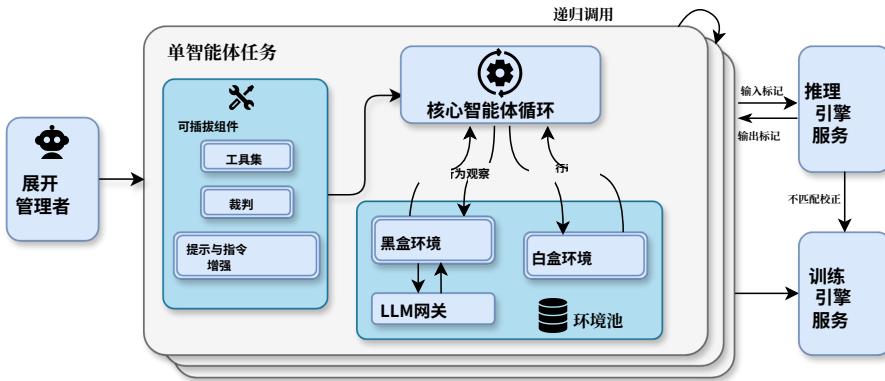


图 10：我们自主代理式强化学习框架的概述。

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 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 we 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 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  
    当前时间戳：  
    E 是动态设置为 th

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。

**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": "代理的系统提示" } } }}
```

```
"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。

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。

**USER**

作为一名视频网站管理员，请分析所有这些文件（在 ./video/wukong/ 中），并创建一个酷炫的 HTML 页面：

1. 确定视频事件的时间线
2. 确定玩家等级升级的时间线
3. 展示你的 (K2.5 代理) 强大的长视频理解和网页设计能力。

输出结果到 ./video/wukong-result-video-web-master/。



**32Parts, 24Hours, 40G**

**思考**

<思考> 用户希望我分析视频文件在 ./video/wukong/ 并创建一个酷炫的 HTML 页面，该页面：

1. 阐明视频事件的时间顺序
2. 识别玩家等级升级的时间线
3. 展示了我擅长视频的理解和网页设计能力

让我先探索一下目录，了解我们正在处理哪些文件。</Think>

**摘要**

我会分析指定目录中的视频文件并创建HTML页面。让我先探索一下我们有什么。

**TOOL**

```
ls -la ./video/wukong/
```

**思考**

<思考> 我们拥有32个视频文件，总计约40GB！这些是《黑神话：悟空》——一款中国动作角色扮演游戏。让我检查文件大小：

- 大部分文件大小为700MB - 2GB
- 总计40GB+

根据AGENTS.md，对于大视频（≥25MB），我应该并行使用子代理。每个子代理应独立处理帧提取和分析。

让我制定一个策略：

1. 每个视频都相当大（全部 > 25MB），所以我将为每个使用子代理
2. 每个子代理将分析一个视频，用于：
  - 事件时间线
  - 升级时间线
  - 关键时刻/场景
3. 然后我会汇总结果并创建一个HTML页面

考虑到规模（32个大视频），我应该并行启动多个代理。让我创建子代理来分析每个视频。</思考>

**摘要**

这是一个庞大的项目！《黑神话：悟空》的32个游戏玩法视频（总共~40GB）。我将部署一个并行分析策略，使用子代理为每个视频，然后创建一个令人惊叹的HTML展示分析，然后创建一个令人惊叹的HTML展示

**TOOL**

<创建子代理 x 32>

**思考**

<思考> 良好，我可以看到所有分析文件都已创建。现在让我从重要的部分提取一些关键视频片段。我将挑选关键的BOSS战和升级时刻，为HTML页面制作简短的视频片段。让我：

1. 使用ffmpeg提取一些关键视频片段
2. 创建综合性的HTML页面
3. 让我先阅读几个分析文件，以识别最佳的提取时刻。</思考>

**摘要**

让我阅读一些分析文件，以确定视频片段的最佳时刻，然后并行提取它们

**TOOL**

<读取文件 x 32>

**思考**

<思考> 我现在对内容有了很好的理解。让我为HTML提取一些关键视频片段

页面。我将提取：

1. 序幕 - 与二郎神的史诗战斗 (0:05:40-0:06:10)
2. 第一章Boss战 - 黑风王 (0:27:00-0:27:30)
6. 终章 - 筋斗云飞 (0:12:30-0:13:00)

让我生成子代理来并行提取这些片段。</思考>

**摘要**

现在让我提取用于HTML展示的关键视频片段。我将生成子代理来从最史诗的时刻。

**TOOL**

<创建子代理 x 9>

**思考**

<思考> 我现在已有足够的数据。让我再获取一些分析文件，然后将其全部编译成令人惊叹的HTML展示。</思考>

**摘要**

让我再收集几个关键分析文件，然后构建完整的HTML页面。

**TOOL**

<构建前端>

**思考**

<思考> 所有任务已完成。让我向用户提供一个关于已完成工作的全面总结。</思考>

**摘要**

```
## 分析完成！黑神话：悟空之旅展示
### 如何查看
在网页浏览器中打开 ./video/wukong-result-video-web-master/index.html，体验完整的交互式展示，其中包含嵌入式视频片段、动画图表以及史诗《黑神话：悟空》的完整时间线：悟空之旅！
```

**子代理 32**

主代理  
分析黑神话：悟空游戏玩法视频： ./video/wukong/01-25895506462\_Prologue.mp4  
这是视频 #01 - 序幕 (302MB, ~5分钟)  
...  
将你的分析保存到： ./video/wukong-result-video-web-master/analysis\_01.json

**子代理 3**

从属代理 2

从属代理 1

**子代理 1**

主代理  
分析黑神话：悟空游戏玩法视频： ./video/wukong/01-25895506462\_Prologue.mp4  
这是视频 #01 - 序幕 (302MB, ~5分钟)  
...  
将你的分析保存到： ./video/wukong-result-video-web-master/analysis\_01.json

**思考**

<思考> 用户希望我分析《黑神话：悟空》游戏玩法视频  
...  
让我先探索文件并提取帧。</思考>

**摘要**

我会分析这个《黑神话：悟空》游戏玩法视频。让我先检查视频并提取关键帧进行分析。

**TOOL**

<分析视频> & <使用FFMPEG提取帧> & <读取媒体文件x N>



**摘要**

## 分析完成

我已经成功分析了《黑神话：悟空》Prologue视频并创建了结构化分析文件。

\*\*已创建文件：\*\*  
 \*\*--分析 JSON:\*\* \*\*./video/wukong/01-25895506462\_Prologue.json\*\*  
 \*\*--提取的帧:\*\* \*\*82 帧在 ./video/wukong-result-video-web-master/frames\_01/

**子代理 1**

主代理  
从以下内容中提取 10 秒视频片段：  
来源： ./video/wukong/01-25895506462\_序幕.mp4  
时间： 00:05:40 至 00:05:50  
输出： ./video/wukong-result-video-web-master/clips/01\_序幕战斗.mp4  
使用ffmpeg以高质量提取此片段。这展示了与二郎神的史诗教程战斗开始。  
命令格式：  
ffmpeg -ss 00:05:40 -t 10 -i ./video/wukong/01-25895506462\_序幕.mp4 -c:v libx264 -crf 23 -preset fast -an ./video/wukong-result-video-web-master/clips/01\_序幕战斗.mp4  
如果需要，先创建输出目录。

**思考**

<思考> 用户希望我使用 ffmpeg。让我先创建输出目录（如果不存在），然后运行 ffmpeg 提取片段的命令。</思考>

**TOOL**

<使用FFMPEG提取视频片段>



**思考**

视频片段已成功提取。让我为主代理总结结果。

**摘要**

\*\*视频片段提取成功！\*\*  
 ./video/wukong-result-video-web-master/clips/01\_序幕\_战斗.mp4  
 \*\*剪辑详情：\*\*  
 ...  
 展示二郎神史诗教程战的10秒片段已被提取  
 高质量设置 (CRF 23, 快速预设) 并保存到输出目录。

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**  
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。  
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，似乎有一个小蚂蚁，这很可能是起点。  
另一个小蚂蚁是终点。这个迷宫本身是一个二值化结构。  
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：

**截图**  
`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（铃铛）和结束（蚂蚁）的位置。  
2. 使用广度优先搜索算法来寻找路径。  
3. 可视化  
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：

**截图**  
`python:绘制点和广度优先搜索>`

**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：

**截图**  
`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  
- 路径：从左侧进入↑向右走→横穿地下  
迷宫↑向上蜿蜒↑到达右侧出口。  
- 优先级：优先级保证这是最短路径且无回溯。  
- 特别注意：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；红色代表步行路线！

**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. 导入模块  
2. 引入饼图  
3. 计算每部分的百分比  
4. 将颜色和百分比相加  
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提供空间，然后对每种颜色进行遍历并计算像素数量。

**摘要**  
我会帮你分析这个饼图中绿色和蓝色区域的百分比。

**截图**  
`python:DifferentImage>`

**思考**  
这是一个“找不同”游戏。图片展示了来自《疯狂动物城》的闪电熊熊场景。  
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**  
正在进行像素级比较，以准确找出所有差异：

**截图**  
`python:Differencem>`

**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。

**截图**  
`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8个主要差异，并输出更精确的坐标。

**截图**  
`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）。  
背景植物叶序存在细微差异。  
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. DMV/EMU/MVM 标志；  
2. 日落照明变化；  
3. 树懒表情：沉睡与张嘴；  
4. 尼克的眼部状态：眯眼与张眼；  
5. 朱迪面部调整；  
6. 撕裂纸张与飞机；  
7. 显示器显示内容；  
8. 马克杯文字变化：“你想它何时？”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 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。