步骤3.5 Flash：开启前沿级智能，拥有11B主动参数

StepFun团队

GitHub HuggingFace 模型博客

摘要

我们介绍了 **步骤3.5 Flash**，一种稀疏专家混合（MoE）模型，它弥合了前沿级智能体智能与计算效率之间的差距。我们专注于构建智能体时最关键的因素：敏锐的推理和快速可靠的执行。反映这些优先事项，步骤3.5 Flash 配备了一个 **196B参数基础**用于高保真建模，以及**11B激活参数**用于高效推理，通过交错 **3:1滑动窗口/全注意力**和 **多标记预测**(MTP-3) 优化，以最小化多轮智能体交互的延迟和成本。为了实现前沿级智能，我们设计了一个可扩展的强化学习框架，该框架集成了可验证信号和偏好反馈，同时在大规模离线策略训练期间保持稳定性，以在数学、代码和工具使用方面推动持续的自我改进。步骤3.5 Flash 在智能体、编码和数学任务中表现出色，在 IMO-AnswerBench 上达到 85.4%，在 LiveCodeBench-v6 (2024.08–2025.05) 上达到 86.4%，在 τ^2-Bench 上达到 88.2%，在 BrowseComp (带上下文管理) 上达到 69.0%，在 Terminal-Bench 2.0 上达到 51.0%——其性能与 GPT-5.2 xHigh 和 Gemini 3.0 Pro 等前沿模型相当。通过重新定义效率前沿，步骤3.5 Flash 为在现实世界的工业环境中部署复杂智能体提供了一个高密度基础。

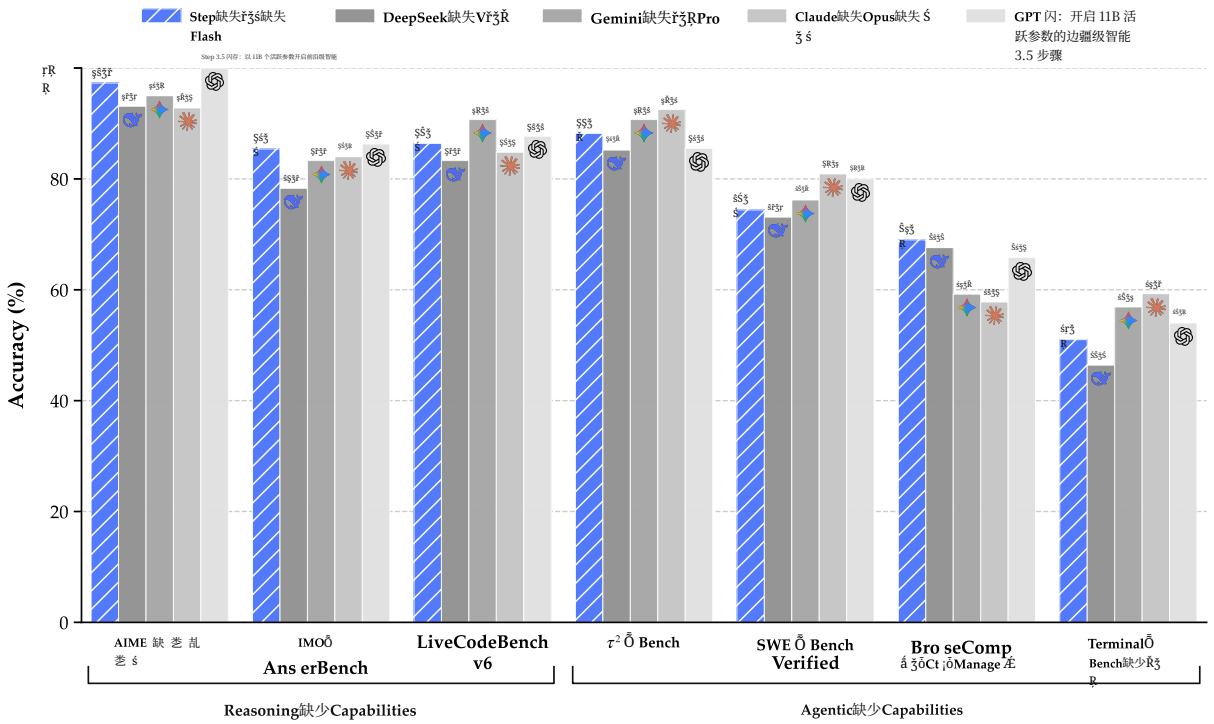


图1：步骤3.5 Flash 仅用11B个激活参数（196B MoE）即可实现前沿级智能，与领先的封闭式和开源模型相当。

目录

1 简介 n 4

2 架构 5

2.1 设计理念.5 2.2 稀疏MoE主干网络与混合注意力.6 2.3 架构消融与结果.8

3 基础设施9

3.1 计算集群.9 3.2 训练框架.9 3.3 高吞吐量轻量级监控.10

4 预训练和中间训练 10

4.1 训练稳定性.11

4.1.1 质子对数值敏感性.11 4.1.2 专家崩溃超越路由崩溃.12 4.1.3 MoE层中的局部激活爆炸.12

4.2 训练课程.13

4.2.1 数据混合.13 4.2.2 计划.14 4.2.3 超参数.15

5 训练后 g 15

5.1 专家模型构建与自蒸馏.15 5.2 可扩展的强化学习.16

5.2.1 MIS-过滤策略优化（MIS-PO）.16 5.2.2 奖励系统.18 5.2.3 超参数.19

5.3 数据合成与策展.19 5.3.1 通用与推理.19 5.3.2 通用工具学习.20 5.3.3 代码代理.20

5.3.4 搜索和研究代理.20

5.4 AgentInfrastructure.21

6 评估 **21**

6.1 预训练评估.21 6.2 训练后评估.22

7 讨论与局限性 23

A 架构细节 **25**

A.1 逐头门控注意力.25 A.2 注意力增强的速度基准测试.26 A.3 元标记.27 A.4 预训练消融细节.27

B 局部激活爆炸的详细分析 **28**

C步骤预训练数据基础 **30**

C.1 知识数据构建.30 C.2 代码数据.31 C.3 数学 & STEM数据.32 C.4 数据基础设施.33 C.5 数据消融设置.33

D 训练后详情 33

D.1 SFT 详情.33 D.2 RL 详情及消融实验.34 D.3 工具集成推理与并行推理.37

E 详细评估协议和提示 **38**

E.1 预训练模型评估详情.38 E.2 后训练模型评估详情.43 E.3 内部评估 - 基准测试和方法论.50

1. 简介

虽然开源大语言模型（大语言模型） [1–6]在可验证任务 [10–12], 上迅速缩小了与闭源前沿系统的性能差距 [7–9] , 但随着智能体系统的普及，新的挑战也随之出现。特别是，开源模型在复杂推理方面仍落后于闭源前沿。此外，关键效率瓶颈阻碍了它们在长上下文智能体任务 [13–21], 中的应用，更不用说在边缘或资源受限环境中部署了。

在设计步骤3.5 Flash的架构时，我们关注两个核心方面：效率和容量。我们采用了一种稀疏专家混合（MoE） [22–26] 架构，总参数量为196B，每个token仅激活11B，并结合了3:1的滑动窗口注意力（SWA） [27] 与全注意力以及多token预测（MTP-3） [3, 28–30] 来减少长上下文延迟。为了在混合注意力下提高容量且开销最小，我们将滑动窗口注意力（SWA）层中的查询头数量从64增加到96，并使用逐头门控注意力 [31]。这种设计支持大规模在线部署，在OpenRouter [1]上线第一周期间，在Hopper GPU上可维持 ~170 tokens/s的吞吐量。

在预训练方面，我们将稳定性视为首要要求，通过轻量级异步指标服务器和微批处理级别的持续日志记录，构建了一个全面的可观察性和诊断栈。该基础设施能够系统性地识别和缓解大规模MoE故障模式（例如，与Muon相关的精度敏感性、专家坍缩 [32], 和激活爆炸 [5, 33]）。结合改进的Muon优化器 [34] 提供的更准确和稳定的更新，我们实现了在17.2T高质量多样化token上的稳定训练，且仅出现一次瞬态损失尖峰。通过这种稳定训练机制，步骤3.5 Flash Base在数学、编程和知识基准测试上与更大规模的模型（如DeepSeek-V3.2-Exp Base [1] 和Kimi-K2-Base [5], ）相比实现了竞争性能。值得注意的是，在SimpleQA [35], 上，它得分31.6%，尽管参数量仅为其三分之一，但仍超越了DeepSeek-V3.2-Exp Base。

为迈向前沿级智能，当前的任务后训练系统面临两大紧密耦合的挑战：针对特定领域的专家迭代效率低下以实现自我蒸馏 [1–4], 以及强化学习（RL）在MoE模型上的长时程推理扩展性有限。训练单个通才直接覆盖多样化领域往往牺牲特定领域的专业知识，而维护独立的专家模型则会导致碎片化，并带来持续多模型迭代的不可持续成本。同时，随着模型扩展到更深的推理轨迹，离线策略回放中的微小标记级差异会累积成高方差梯度。这种效应在MoE模型中尤为严重，因为专家级路由会引发更大的分布偏移，并使前沿性能区域内的优化过程不稳定 [1, 36–38]。

为应对这些挑战，我们提出一种基于共享SFT基础的大规模RL统一后训练配方。该框架在特定领域专业化和全局合成之间交替切换，在保持单一高性能通用专家的同时，实现高效的专家迭代。一个专门的中期训练阶段将上下文窗口扩展到128k，并通过合成数据强化核心的代理和推理能力，为下游后训练提供强大的初始化。为在这个统一的框架内支持稳定且可扩展的RL，我们引入了Metropolis独立性采样-过滤策略优化（MIS-PO） [39, 40], , 用离散分布过滤替代了标记和轨迹级别的连续重要性加权。通过将优化限制在稳定信任区域内的样本中，MIS-PO大幅降低了梯度方差，同时保留了有效的学习信号，使RL能够可靠地扩展到长时程推理和代理行为。

[1]https://openrouter.ai

Step 3.5 Flash在推理和代理基准测试中，尽管只有11B个激活参数，但仍取得了与前沿领先模型和系统的竞争性能。它在标准推理任务推理下的结果表现强劲，包括在IMO-AnswerBench [41] 上达到85.4%，在LiveCodeBench-v6（2024.08–2025.05） [12], 上达到86.4%，同时在88.2%的 τ^2-Bench [15], 69上展示了强大的长时程、工具增强能力，在带有上下文管理的BrowseComp [17], 上达到0%，在Terminal-Bench 2.0 [16]上达到51.0%。通过PaCoRe [42]深度思考推理，Step 3.5 Flash进一步提升了需要长时间推理和多轮合成的推理密集型基准测试的性能。综合来看，这些结果表明Step 3.5 Flash在推理和代理设置中，显著缩小了先进开放模型与前沿专有系统之间的差距。

2. 架构

2.1. 设计理念

步骤3.5 Flash的架构反映了模型-系统协同设计的范式转变。除了智能和成本的传统目标外，自主代理时代提升了一个关键的约束条件：推理延迟。在交互式代理工作流中 [43, 44], 最小化延迟直接转化为任务完成所需的总时间减少，反之，则允许在固定时间预算内通过测试时扩展 [42, 45-47]提升智能水平。

代理工作负载通常表现出独特的特征：广泛的上下文预填充，随后是长时间的、多轮的交互式解码。因此，我们针对步骤3.5 Flash的三条耦合轴协同设计了低总时间延迟：注意力(以加速长上下文处理，并具有良好的MTP兼容性)，稀疏*MoE* (以防止分布式部署中的滞后者，从而提高吞吐量)，以及多*token*预测 (MTP；以通过推测解码促进快速生成)。

注意力。 为加速预填充，我们采用了一种混合注意力机制 [33, 48, 49] 来缓解长上下文处理的二次复杂度。在解码阶段，我们优先考虑与推测解码 [50], 的架构兼容性，因为验证效率是带宽受限硬件上的主要杠杆。这些考虑促使了两个注意力设计决策：

- 滑动窗口注意力（***SWA***）。我们选择 SWA [27] 而非线性注意力 [10, 51] 以最大化解码效率。尽管两者都具有线性复杂度，但线性注意力的状态更新机制使高效草稿树生成和并行树验证（用于推测解码 [52–54]）变得复杂。相比之下，SWA 保留了标准注意力语义，并通过 KV 掩码的方式本质上易于并行验证。此外，在没有充分实证证据表明线性注意力在代理任务中能实现更优的长上下文建模的情况下，我们发现窗口大小为 W=512 的 SWA 在内核效率和捕获局部依赖性之间取得了有利平衡。

- 硬件对齐分组查询注意力（***GQA-8***）。针对在标准 8-GPU 服务器节点上的部署，我们配置模型使用八个 KV 头（GQA-8） [55]。这使 KV- 缓存分片与 8 路张量并行对齐，并改善了内存访问模式。关键在于，虽然 GQA-8 使注意力更受内存带宽限制，但它也创造了可吸收推测草稿和验证开销的计算冗余，从而能够在不产生成比例延迟惩罚的情况下实现激进的多元组推测。

稀疏 MoE。 在前馈侧，我们采用细粒度 MoE [22–26]来降低平均 FFN 计算量，同时保持容量。专家并行（EP) [25] 被用于实现可扩展

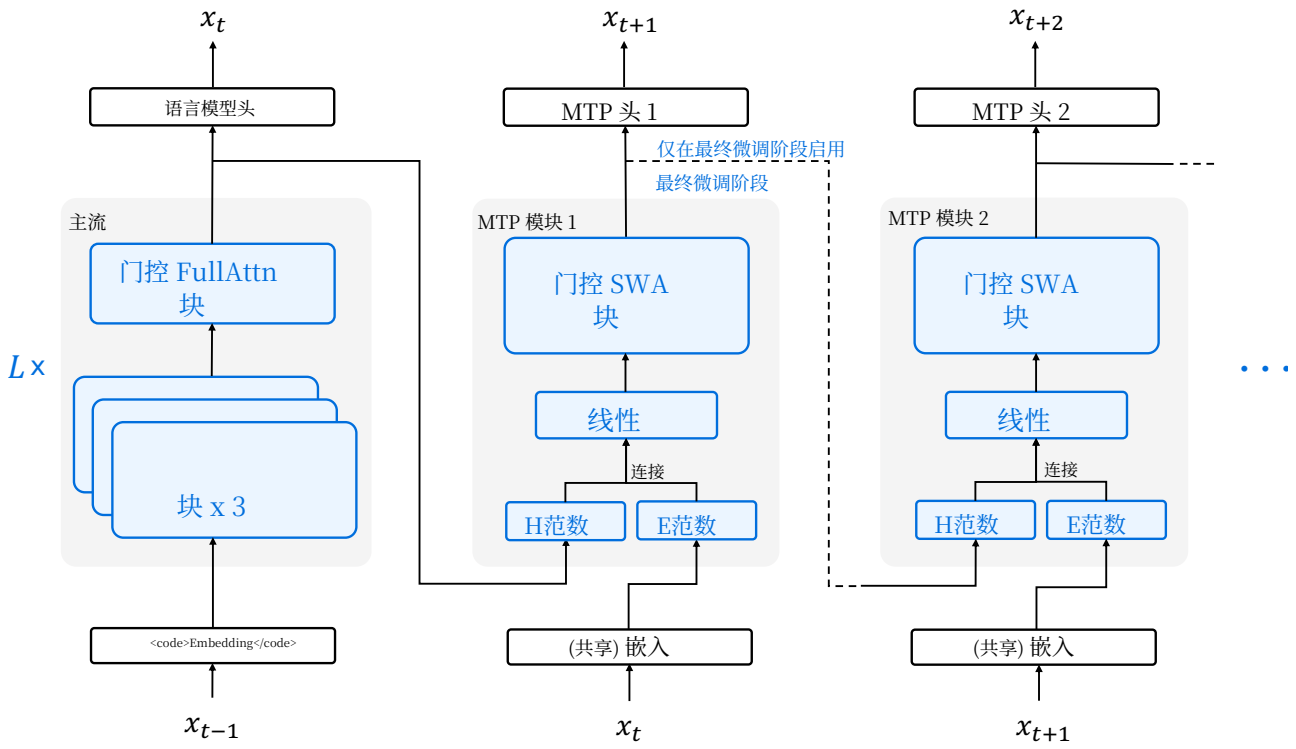


图2：Step 3.5 Flash 的示意图。该模型使用逐头门控注意力 [31]，由一个领先的密集注意力层后接 $L = 11$ 混合块组成，每个混合块交错嵌入3个滑动窗口注意力（SWA）层和一个密集注意力层（为视觉清晰起见，图中省略了第一层）。我们全程应用零中心 RMSNorm [57]。前三个块使用密集 FFN；后续块采用稀疏MoE FFN。MTP 模块使用SWA和密集FFN。为限制开销，仅在主流训练期间训练MTP模块1；MTP模块2-3从其克隆并联合在轻量级最终阶段进行微调。

部署。然而，在EP下，端到端延迟可能被路由不平衡引起的滞留者 所主导：token分配偏差将工作负载集中在少数专家及其托管GPU上，在同步点处限制吞吐量。因此，我们引入了一种*EP*-组平衡*MoE*路由策略。

多标记预测（MTP）。为进一步降低自回归延迟，我们引入多标记预测（MTP） [29, 56] 作为推测解码 [50]的辅助手段。为保持推测的轻量化，我们通过利用SWA和密集前馈网络 [3]来简化MTP头部。

我们进一步将模型参数量控制在200B以下，使其能够在高端工作站128GB的内存预算内实现高性能推理。

2.2. 稀疏MoE主干与混合注意力

如图2所示，Step 3.5 Flash采用一个45层稀疏MoE Transformer主干（3层密集层和42层MoE层），搭配一个专门混合注意力层布局。每层MoE包含288个路由专家加上一个共享专家，每个token由一个top-k 路由器激活 k=8 个专家。这种配置在保持广泛知识容量（196B总参数）的同时，将每个token的激活限制在11B，确保推理延迟足够低，以支持高度

响应的agent交互。表6总结了Step 3.5 Flash的关键架构超参数。

混合注意力层布局。 为了平衡长上下文效率与鲁棒的长距离连接性，步骤3.5 Flash 利用受 [33,49,58], 启发的、以3:1比例（SWA : 全注意力）交织的注意力布局，记为 $S3F1$。该配置重复一个由四层组成的模式，包括三个SWA层 (W=512) 后接一个全GQA-8层。然而，在我们的初步实验中，一种简单的交织策略在各种基准测试（表10）中始终不如密集注意力基线表现好。为了在不增加实际开销的情况下弥合这一性能差距，我们采用了两个互补的增强措施：(i) 增加SWA查询头数量，以及 (ii) 采用逐头门控注意力 [31]。

SWA中的增强查询头。 使用更高的查询头数量（从64增加到96）有效地缓解了在从均匀全注意力架构过渡到 $S3F1$ 布局时通常观察到的性能下降（表10）。我们认为这几乎是一种“免费午餐”。因为在长文本场景中，简单SWA的开销非常小，即使我们的解决方案扩展显著。

逐头门控。 朴素SWA的一个局限性在于，当输入窗口 [31,59–61] 中没有有用信息时，它无法有效地吸收未使用的注意力权重。先前工作 [3,33] 将可学习、与数据无关的汇流标记引入窗口中来解决这个问题。相反，我们选择了一种不同的方法，通过集成参数高效的逐头门控机制 [31,62,63],，这可以看作是集成与数据相关的汇流标记。有关实现细节和进一步讨论，请参阅附录A.1。逐头门控对理论FLOPs和实践延迟的影响也微乎其微。我们在附录A.2中报告了更多关于门控和增加SWA头数的性能分析及基准测试。

MoE 专家并行负载均衡。 我们使用无损负载均衡 [29,64] 来鼓励 全局 token 在专家之间平衡。然而，这种方法并不能保证在微批次级别上跨 EP 排的负载均衡，可能会导致 stragglers 和吞吐量降低。因此，我们引入了一个 EP 级别的平衡损失，它明确地促进均匀的排级利用率 [26]。

EP分区专家E为 G 个互斥组$\{\mathrm{E}_g\}_{g=1}^{G}$ 按等级划分。对于标记 t，用 S_t 表示前K位专家（掩码$s_{t,e} = \mathbf{1}[e \in S_t]$），用$p_{t,\cdot}$表示路由概率。那么，EP负载均衡损失$\mathrm{L}_{EP}$为：

$$p_e = \frac{1}{T}\sum_{t=1}^{T} p_{t,e}, \quad f_e = \frac{1}{TK}\sum_{t=1}^{T} s_{t,e}, \quad p_g = \sum_{e\in\mathcal{E}_g} p_e, \quad f_g = \sum_{e\in\mathcal{E}_g} f_e, \quad \mathcal{L}_{\mathrm{EP}} = G\sum_{g=1}^{G} f_g\, p_g. \tag{1}$$

多标记预测（MTP）。为了加速长上下文智能体工作负载上的推测解码，我们附加了三个轻量级多标记预测（MTP）头。每个MTP头由一个SWA和一个密集FFN组成，仅增加了0.81B参数（~0.41%）。我们通过它们相对于标准LM头的额外预测偏移来索引这些头：对于 附加预测偏移 $h \in \{1, 2, 3\}$，MTP-h 根据骨干隐藏状态在位置 t 预测标记 x_{t+1+h}。为了控制训练开销，我们在大多数训练阶段仅激活和优化MTP-1。一旦骨干训练完成，我们从MTP-1初始化MTP-2和MTP-3，并在一个轻量级后训练阶段联合训练所有MTP头。受Fast-MTP [65], 启发，我们在MTP头的预测偏移上采用位置相关损失重加权，以防止过度优化远距离标记预测。

布局	SWA 头	相对 FLOPs	预训练 Avg.	下游性能						
		解码 / 预填充		推理	Math	Code	Sci	通用	LongCtx	Avg.
FFFF	32	~2.68 / 2.90	54.1	40.8	40.9	**19.6**	42.7	26.5	28.8	33.2
*S*1*F*1	32	~1.58 / 1.65	54.6	**42.1**	**42.3**	19.3	**44.5**	**26.8**	**29.6**	**34.1**
*S*3*F*1	32	**1.00 / 1.00**	53.6	40.2	40.4	18.9	42.4	25.4	27.5	32.5
*S*3*F*1+Head	48	~1.01 / 1.02	**55.7**	40.6	40.3	18.3	44.0	26.0	28.2	32.9

表1：在30B-A3B上的下游结果。 *F* 表示全注意力， *S* 表示SWA. *S*3*F*1 表示混合布局中三个 *S* 层后跟一个 *F* 层。相对FLOPs相对于 *S*3*F*1配置进行归一化，并在64K/256K上下文中平均（表8）。预训练平均结果跨通用、数学和代码基准聚合（表16）。

2.3. 架构消融与结果

我们对步骤3.5 Flash中的关键设计选择进行了广泛的实验验证，重点关注 (i) 注意力布局，包括SWA和头规模，以及 (ii) 逐头门控注意力与汇流标记。为确保我们的效率优化不会降低模型性能，我们采用了两种互补的消融协议：一种评估涵盖预训练、32k长上下文扩展和64k上下文长度监督微调 (SFT) 的完整端到端管道，另一种将分析扩展到100B参数，以研究这些设计选择在规模下的表现。所有表格的详细架构和评估设置在附录A.4中提供。这些大规模实验的关键发现总结如下。

SWA 相对于长上下文。 我们通过完整管道（1.4T-token预训练后进行SFT）训练一个30B-A3B模型，以评估混合注意力对推理和长上下文性能的端到端影响。我们消融了四种注意力布局：全部全注意力 (*FFFF*)、交替SWA/全 (*S*1*F*1)、3:1 SWA到全布局 (*S*3*F*1)，以及一个增加了SWA查询头的 *S*3*F*1 变体 (*S*3*F*1+Head)。为隔离注意力结构效应，我们将SWA窗口大小固定为 W=512 ，并禁用MTP（见附录，表9和表10）。

表1展示了不同布局下的成本-质量权衡关系。 *S*3*F*1 实现了最低的归一化注意力侧FLOPs（分别针对prefill和decode归一化为1.00），而 *FFFF*则 ~2。68×/2.90× 与 *S*3*F*1一样昂贵；然而， *S*3*F*1 表现出持续的质量退化（例如，LongCtx从28.8降至27.5）。

增加SWA查询头的数量可以很大程度上弥补这种损失。值得注意的是， *S*3*F*1+Head在预训练阶段就已经超越了 *FFFF* （55.7 vs. 54.1），并且在后训练阶段仍然保持竞争力：LongCtx从27.5提升到28.2，Sci从42.4提升到44.0，几乎消除了与 *FFFF*基线的差距，且注意力成本的增加可以忽略不计。剩余的缺点是有限的且局部的（例如，Code上出现适度下降至18.3），而整体质量趋势有利于 *S*3*F*1+Head。

有趣的是，交替 *S*1*F*1 布局提供了最佳的总体SFT质量，并获得了最强的LongCtx分数（29.6），但需要显著更高的注意力侧预填充/解码FLOPs(~1.58/1.65)，相对于 *S*3*F*1+Head的成本增加了约60%。因此，我们采用 *S*3*F*1+Head 作为长上下文智能型工作负载的默认配置，优先考虑其远低的预填充/解码成本，同时兼顾强大且稳定的长上下文性能。

逐头门控注意力 vs. 汇流标记我们在一个100B-A10B MoE上进行了规模化的、受控的预训练实验，以研究在真实规模条件下注意力侧面的机制。

方法	BBH	MMLU	GPQA	MBPP	C-EVAL	CMMLU	平均
汇流标记	70.6	65.1	27.2	61.2	76.2	74.6	62.5
逐头门控	**73.7**	**67.0**	**28.1**	**62.6**	**77.9**	**77.1**	**64.4**

表 2：在 $S3F1$ 布局下对 100B-A10B 模型进行仅预训练评估。逐头门控始终优于固定汇流标记，在所有基准测试（包括总体平均）中表现更佳。

具体而言，我们比较汇流标记和逐头门控，同时保持注意力布局固定为相同的 $S3F1$ 配置，窗口大小为 W=512。如表2所示，逐头门控始终提高质量，将平均性能从62.46提高到64.43 (+1.97)。因此，我们在后续研究中采用逐头门控注意力作为默认机制。

3. 基础设施

3.1. 计算集群

步骤3.5 Flash 在一个包含 4,096 个 NVIDIA H800 GPU 的大规模集群上进行训练。每个节点包含 8 个通过 NVLink 和 NVSwitch 互连的 GPU，用于高带宽的节点内通信。对于节点间连接，该集群依赖于 8×200 Gbps RoCE 链路，以在规模上保持高效的同步和数据交换。

3.2. 训练框架

步骤3.5 Flash 的训练由我们内部 *Steptron* 框架提供支持，这是一个基于 PyTorch [66] 和 Megatron-LM [67] 构建的超轻量高性能系统。Steptron 统一了完整模型开发流程，支持大规模预训练、后训练和强化学习（RL）工作负载，并集成在单一工程栈中。

步骤3.5 Flash 采用混合并行化策略，包括 8 路流水线并行（PP） [68]（含虚拟流水线阶段 VPP），以及 8 路专家并行（EP） [25], 和 ZeRO-1 数据并行（DP） [69]。为了促进步骤3.5 Flash 的高效训练，我们采用了以下工程技术。

解耦并行。 遵循 Megatron-Core [70], ，我们实现了一种解耦并行化方案，允许注意力模块和 MoE 模块使用不同的并行化策略。我们为它们分配独立的并行组，并在每个模块对应的数据并行组内进行梯度缩减和缩放。

通信优化。 并发 DP 通信流用于解耦注意力机制和 MoE 可能会饱和 RoCE 链路，由于拥塞导致 DP 开销显著增加。为解决此问题，我们提出了两种互补的通信优化方案，联合将迭代时间缩短高达 5%。首先，*感知网络的通信调度* 将 DP 流量划分为节点内 NVLink 和节点间 RoCE 阶段，并对其进行流水线处理以充分利用两种网络。其次，*感知通信的排名放置* 使用作业级通信配置将排名放置在交换机上，减少跳数并将大量流量从交换机热点区域引导开。

Muon ZeRO-1 重分片。 Muon [34]需要完整的（未分片）每个参数的梯度用于牛顿-舒尔茨正交化，这与 ZeRO-1 [69]reduce-scatter 冲突，后者将参数的梯度分片到 DP 排名上。Megatron-LM 中的当前实现通过在 Muon 更新之前天真地全归约 FP32 梯度以重建完整梯度来解决此不匹配，但几乎将通信量翻倍。我们改为将整个参数分配给 DP 排名，并将梯度缓冲区重新打包为排名主序缓冲区，以便单个 reduce-scatter 将每个参数的完整梯度交付给其所有者。由于填充到最宽排名的开销随数据并行规模增长，我们仅将其应用于专家参数，并使用 DP 全归约处理非专家参数。这种混合策略与天真全归约基线相比，端到端迭代时间减少了约 5%，且额外内存不到 4 GB。

GPU内核优化。 我们还应用内核级优化来提高训练效率。在注意力机制中，我们将QK归一化与RoPE融合。在MoE中，我们融合多个小算子以减少内核启动开销和内存流量，并实现了一个与SonicMoE [71]类似的分组GEMM融合的MoE gather/scatter。

细粒度选择性检查点。 我们的训练框架支持按层、按子模块级别的激活重新计算开关（例如，注意力、FFN、归一化、SiLU 和 MoE 排列），能够仅对最内存密集的组件进行选择性重新计算，以最小的开销降低峰值内存。

3.3. 高吞吐量轻量级监控

我们收集了一套全面的指标（例如，每个微批次的专家分布和梯度范数），用于对训练进行细粒度监控。然而，遥测数据的规模是巨大的：一个 4,096 个 GPU 的工作负载每次迭代会产生近 600 万条消息。在主循环中执行同步全局缩减会引入几秒钟的重大开销，实际上将迭代时间翻倍，这显然对于高性能训练是不可接受的。为了缓解这个问题，我们开发了一个轻量级指标服务器，将遥测数据处理与训练路径解耦。每个 rank 利用 *StepRPC*（一个内部异步通信框架）异步地将本地指标卸载到远程服务器。这种方法将遥测开销降低到每次迭代约 100 毫秒。

指标服务器缓存传入的指标，并且仅在与所有参与排名接收到 迭代结束 信号后才触发缩减和数据库持久化，从而消除了主循环中的同步。为了以低延迟摄取和处理数百万条消息，服务器被实现为一个高并发的多进程系统，包含两个解耦的模块：(i) 专为高吞吐量摄取优化的消息接收器，以及 (ii) 负责聚合和持久化的缩减处理器。通过利用这些模块内部和跨模块的多核并行性，服务器能够与遥测流保持同步，并确保指标管理始终不会落后于训练。

4. 预训练和中训练

概述。 本节总结了我们的预训练和中间训练过程，重点强调大规模稀疏MoE训练的实用稳定性约束。我们首先描述训练稳定性诊断和缓解措施（第4.1节），然后详细说明预训练和中间训练所使用的课程，包括数据混合、时间表和关键超参数（第4.2节）。

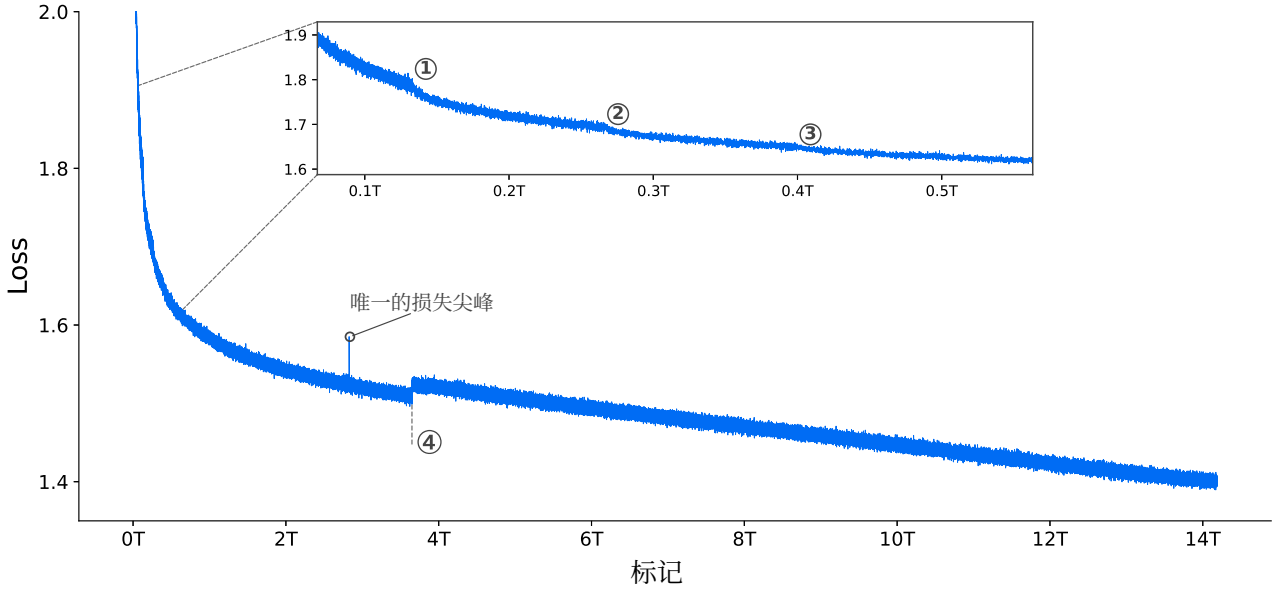


图3：步骤3.5 Flash的每步训练损失，绘制时<样式 id='1'>未进行平滑或子采样</样式>。我们在整个训练过程中仅观察到<样式 id='3'>一个</样式>孤立的损失尖峰。为清晰起见，省略了初始训练步骤。标记<样式 id='5'>1</样式>–<样式 id='7'>3</样式>分别表示批大小增加到8,192、12,288和16,384。标记<样式 id='9'>4</样式>表示元标记上的损失掩码激活（有关详细信息，请参见附录A.3）。

4.1. 训练稳定性

训练稳定性是大规模稀疏MoE预训练的**头等** 要求。为了使稳定性具有可操作性，我们构建了一个基于轻量级异步指标服务器和微批次级持续日志的综合可观察性和诊断堆栈（在第3.3节中描述）。该基础设施提供了对优化器级和专家级信号的细粒度可见性，能够系统地缓解大规模MoE训练中反复出现的故障模式。

在实践中，我们发现指标堆栈有助于及早发现并精确定位三种主要的不稳定性：(i) 暂时性损失尖峰和偶尔出现的随机数值爆炸，这是由于Muon在降低精度下的数值敏感极因子迭代引起的；(ii) 即使路由调度统计仍然看似健康时也可能发生的专家端崩溃（“死专家”）；以及(iii) 仅限于一小部分专家的局部激活爆炸。

通过这些诊断指导的缓解措施，预训练损失在整个运行过程中保持平滑，仅表现出一个损失尖峰。图3显示了学习率冷却前的完整曲线。

*4.1.1. Muon*数值敏感性

Muon通过牛顿-舒尔茨（NS）迭代 [72]近似一个半正交更新方向。在早期实验中，我们发现使用收敛速度更快的正交化近似时，损失减少效果温和且一致。因此我们采用Polar Express [73] 迭代，并运行固定T=6 步数来平衡优化质量和吞吐量。

然而，尽管我们使用了推荐的安全缩放 [73]，但偶尔仍会观察到剧烈且无法恢复的损失尖峰。这些尖峰是非确定性的（通常通过从附近的检查点恢复来避免），这表明存在数值病理问题。模拟表明，在某些更新统计下，`bfloat16` Polar Express 由于累积误差，很少会产生极端中间异常值。因此，我们将 **仅** Polar Express 迭代（状态和中间值）转换为 `float16`

同时保持训练的其他部分使用混合精度。在此更改后，尖峰不再出现。

4.1.2. 专家坍塌超越路由坍塌

步骤-3，我们的先前工作 [32] 报告称，MoE训练可能会出现“专家死亡”现象，通常被描述为专家在长时间内接收到的token调度微乎其微，因此获得的梯度信号很少。在我们的先前调查中，我们发现专家崩溃也可能表现为一种专家侧 病理，即使路由器调度保持稳定，即 专家激活消失以及专家参数范数停滞或衰减。

我们观察到有两个因素特别具有影响力：(i) 路由专家聚合需要显式缩放。当引入共享专家时，引入一个显式缩放因子来校准共享专家和路由专家的相对贡献非常重要。虽然较小的模型可能会隐式学习这种平衡，但较大的模型在自我校准方面不太可靠。不匹配可能会抑制路由专家的有效贡献，即使路由频率看起来是健康的。(ii) 微批平衡在细粒度稀疏情况下可能过于严格。对于稀疏的细粒度MoE设计，微批级别的负载平衡约束（如Switch式路由 [22]中常见地实现的那样）可能会变得过于严格。正如 [74], 中分析的那样，微批LBL可能会诱导过多的跨专家竞争，并阻碍有效专业化。

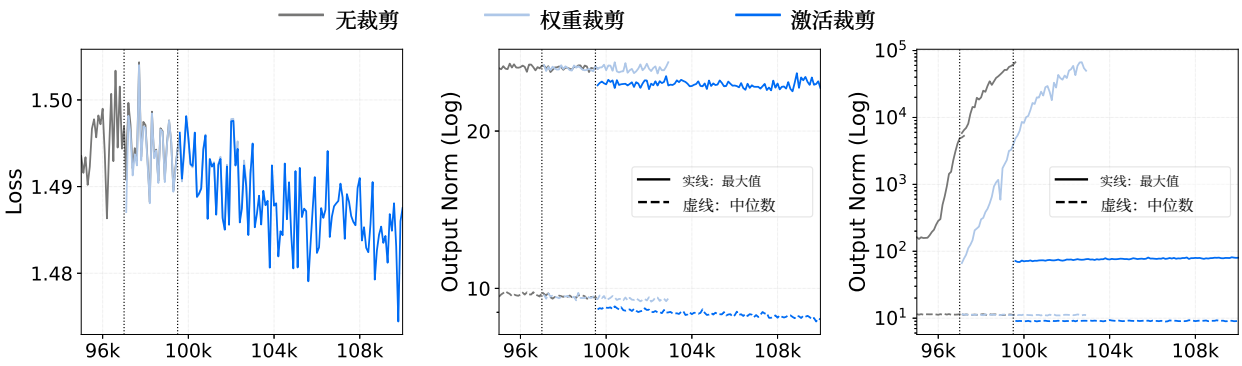
因此，我们更倾向于更广泛范围的平衡（例如，全局批次统计） [74, 75] 或基于观察到的负载的无损失偏差调整 [29, 64]。在实践中，路由调度统计通常是稳定的，并不是专家崩溃的敏感指标。我们建议监控专家端的信号，包括每个专家的激活范数（例如，MoE FFN中间的RMS/均值范数）和参数范数（例如，专家投影矩阵的Frobenius范数）。当一部分专家的激活/更新漂移到接近零，而中位数保持稳定（例如，最小值到中位数的比率下降）时，它提供了专家“死亡”的早期预警。

*4.1.3. MoE*层的局部激活爆炸

随着专家专业化在主要训练阶段成熟，我们在更深层的MoE层观察到一种局部稳定性病理。具体来说，一小部分专家（通常是每层一个或两个）的激活**范数**快速增长，而同一层的多数专家仍然表现良好。这种差异导致重尾激活分布：中位数专家激活范数保持稳定，但最大激活范数爆炸，显著增加了数值溢出和下游不稳定的风险。

图4说明了这种失效模式。**值得注意的是，这种内部不稳定性完全被训练损失所掩盖**，尽管在(a)面板中显示了底层范数的爆炸性增长，但训练损失的变化可以忽略不计。我们通过监控每个专家FFN输出范数的分散程度来追踪这种现象。从(b)和(c)面板中观察到，虽然中间层（例如，第38层等）保持稳定的分布，但最终层（即，第45层即）在最大值（实线）和中位数（虚线）之间显示出快速扩大的差距。这表明激活能正在深层网络中少数“流氓”专家那里高度集中。为了缓解这一问题，我们评估了两种不同的干预措施：

- 专家投影的权重裁剪：我们对MoE FFN专家投影矩阵的范数进行约束。对于每个专家投影矩阵 W，如果其最大激活范数$\max_x \|Wx\|$超过阈值 τ，我们通过 $W \leftarrow W \cdot \frac{\tau}{\max_x \|Wx\|}$对其进行缩放。这与注意力中的MuonClip类似 [5],，但我们是在检查点上进行离线裁剪，而不是实时进行。
- 专家内的激活裁剪：我们直接将元素级裁剪应用于MoE FFN中间激活，在输出投影之前，正如 [33] 中所述。



(a) 损失 vs. 步数　(b) 专家输出（第38层）　(c) 专家输出（第45层）

图4：专家激活稳定性分析及缓解策略。在(b)–(c)面板中，实线表示最大专家输出范数，而虚线表示中位数。(1)深度相关的不稳定性：虽然训练损失在所有方法中看起来相同（a面板），并且中间层保持稳定（例如，b面板中的第38层），但最终层（即，c面板中的第45层）在<No clipping>基线中遭受灾难性的范数爆炸。(2)缓解：<Weight clipping>仅仅延迟了这种爆炸。相比之下，<Activation clipping>有效地限制最大范数，确保所有层的稳定性。

尽管在图4 (a)中，不同缓解策略的训练损失看起来无法区分，但最大值-中位数比率可靠地揭示了潜在的不稳定性。如面板(b)和(c)所示，激活裁剪确保了内部范数的稳定轨迹，而仅权重裁剪无法防止异常专家的再次出现。因此，我们建立每个专家激活范数的最大值-中位数比率作为监测训练稳定性的可靠且必要的指标。

激活膨胀由多个因素驱动。我们观察到高频二元组可以触发专家专门化。在使用预规范 [76,77]，时，单个专家可以无限制地放大其输出并主导最终输出范数，导致近乎确定性预测行为。这种风险在 SwiGLU [78]，中被加剧，其中门控和上投影分支之间强对齐会产生具有极端幅度的稀疏激活。Muon 通过放大持续的低秩更新进一步加速了这种崩溃。详细分析在附录 B 中提供。

4.2. 训练课程

训练从广泛的开放领域覆盖开始，逐步转向更具自主性和长上下文的专门化。我们首先在4k上下文中对广泛的开放领域混合数据进行预训练，以建立通用能力，然后将混合数据退火向更高质量的知识和更多软件开发数据（代码、拉取请求、问题和提交）发展，同时将上下文窗口扩展到32k。接下来，一个专门的中训练阶段将上下文窗口从32k扩展到128k，以加强长时程推理，并为下游的后续训练和自主性工作负载改进初始化。总体而言，我们在预训练阶段训练了大约17.6T个token，在中训练阶段训练了750B个token。

4.2.1. 数据混合

我们的语料库结合了通用开放域数据与以智能体为导向的数据。我们总结了以下关键来源，更多细节可在附录C中参考。

通用知识数据。 为支持广泛的世界知识，我们构建了 **StepCrawl** (附录C.1.1)，这是一个超越标准 Common Crawl [79], 的内部爬取和管理基础设施，用于从网页（HTML）和书籍/文档类来源（ ePub/PDF）大规模采集数万亿高质量标记。所有内容都经过多阶段质量过滤、站点/类别标记、去重和净化处理。

代码数据。 强大的代码能力是智能体模型的基础。我们的代码语料库使用改进的 OpenCoder [80] 管道进行策展和精炼。我们从零容忍政策放宽过滤标准，允许每篇文档存在 0–6 个启发式违规（附录 C.2.1 ），在平衡质量和多样性的同时，在退火和中期训练期间对代码相关数据进行上采样，以增强与智能体相关的编程。

PR/Issue/提交数据。 为了更好地匹配真实的软件工程工作流程，我们从具有 10+ 星级的GitHub仓库中整理了一个全面的PR/Issue/提交数据集（附录C.2.2）。这包括 (1) 经验证的Base数据（与基准测试 [14, 81]去重）；(2) 使用Agentless风格的模板 [82] 从PR线程和提交中派生的PR-对话数据，用于文件本地化和代码修复；以及 (3) 用于中训练和后训练的衍生软件工程语料库。

工具使用和推理数据。 为提高工具使用的鲁棒性和多步推理能力，我们添加了涵盖数学/代码/科学/通用知识以及针对搜索代理、SWE代理和工具执行的特定领域样本的合成和半合成数据。在中期训练期间，我们进一步引入长上下文样本（自然长文档和长格式合成任务），以加强在扩展上下文中的规划和推理。

4.2.2. 安排

预训练安排。 预训练包括两个阶段:

1. **预训练阶段1：开放域预训练** (14.6T tokens, 4k 上下文)。进行广泛的开放域训练，以最大化覆盖范围和基础能力。
2. **预训练阶段2：退火 +长上下文初始化** (3T tokens, 4k 至 32k 上下文)。我们将数据混合退火至代码和 PR/Issue/Commit 中心化来源，同时增加高质量知识和密集推理样本的份额。此阶段从 2T tokens 和 4k 上下文开始，然后在相同的退火混合下过渡到 1T tokens 和 32k 上下文，以初始化长上下文训练。

中训练安排。 中训练也包括两个阶段:

1. **中训练阶段1：32k 专业化**(386B tokens, 32k 上下文)。我们从预训练中重放 81B tokens(21%), 以减轻分布偏移并稳定专业化，同时强调软件工程师和工具使用中心化的混合。

2. **中期训练阶段2：长上下文专门化**(364B个token，128k上下文)。我们保留10.5B个重放token，并进一步专门化长上下文能力，结合合成长时程推理和自然长文档（从预训练数据中选取长度为 > 32 k的文档），以及代码代理、搜索代理和工具使用的领域特定数据。

4.2.3. 超参数

预训练超参数。我们整个预训练过程中使用Muon优化器 [34] ，将权重衰减设置为0.1，梯度裁剪设置为1.0。学习率从0线性预热到2.5×10^{-4}，在最初的2,000步内，然后余弦衰减到 5×10^{-5}，在预训练阶段1。在预训练阶段2，我们对4k部分（2T个token）从 5×10^{-5} 余弦衰减到 2×10^{-5}，并将学习率固定在 2×10^{-5}，对32k部分（1T个token）保持不变。全局批大小在最初的400B个token内逐渐从4096增加到16384，在剩余训练中保持16384，在32k部分的退火中设置为2k。MTP损失权重在预训练阶段1设置为0.3，在预训练阶段2设置为0.1，遵循 [29]。对于无损负载均衡，前14.6T个token的偏差更新率为0.001，在退火期间衰减到0.0，并在整个预训练过程中应用系数为0.001的EP组平衡损失。对于RoPE [83], ，我们在4k训练期间对全注意力机制和滑动窗口注意力（SWA）都使用 $\theta = 10,000$，并将 $\theta\text{Full} = 1{,}000{,}000$仅用于全注意力机制，并保持 $\theta\text{SWA} = 10{,}000$，用于32k部分的退火。

训练中期的超参数。 我们在训练中期继续使用 Muon [34]。我们冻结了 MoE 路由器权重，禁用了 EP组平衡损失，并将 MTP 损失权重固定为 0.1，适用于训练中期的两个阶段。学习率在迭代的前 3% 内从 0 热身到 2×10^{-5}，在训练中期阶段 1 保持不变，并在训练中期阶段 2 衰减到 7.3×10^{-6}。对于 RoPE 选择性缩放，我们在 32k（训练中期阶段 1）时设置为 $\theta\text{Full} = 1{,}000{,}000$，并在 128k（训练中期阶段 2）时增加到 $\theta\text{Full} = 5{,}000{,}000$，同时在整个训练中期保持 $\theta\text{SWA} = 10{,}000$ [84]。

5. 训练后

在本节中，我们介绍了一种针对大规模强化学习 (RL) 的统一训练后配方，该配方以统一的监督微调 (SFT) 模型开始。该框架通过结合可验证的奖励信号与人类偏好反馈，实现一致的自我改进，同时在为专家混合 (MoE) 模型进行大规模离线策略训练时保持稳定性。该过程遵循与先前工作 [2, 85] 类似的两阶段方法。首先，我们通过在数学、代码、STEM、工具使用、长上下文理解、人类偏好和代理推理等特定领域构建 **专家模型** 来增强统一的 SFT 基线，从而构建这些专家。然后，使用 **自我蒸馏** 和 **可扩展 RL** 将这些专业专家蒸馏成一个通才模型，确保最终模型在多样化任务中与专业基线保持竞争力。通过系统地交替进行针对性专业化和广泛综合，我们实现了稳健的泛化，同时不牺牲专家级性能。

5.1. 专家模型构建与自蒸馏

我们采用两阶段SFT流程来为后续RL构建一个稳健的基础。第一阶段执行大规模多领域SFT，涵盖数学、代码、STEM、逻辑性、通用问答、代码代理、工具使用、搜索代理和长上下文理解。应用难度感知过滤和策略性平衡来培养广泛的智能体行为。第二阶段通过注入分布外（OOD）信号 [46, 86], 来显式最大化推理密度，这些信号包含 ~30K个专家级化学轨迹和合成算术任务。这种针对不同推理模式的定向接触仅需三个epoch即可解锁模型中的潜在能力，使模型具备初始化后续领域特定RL阶段所需的复杂结构能力。

遵循特定领域的强化学习，我们将分散的专家能力整合到一个统一的学生模型中，该模型从训练中途的检查点初始化。在此阶段，专家模型生成高

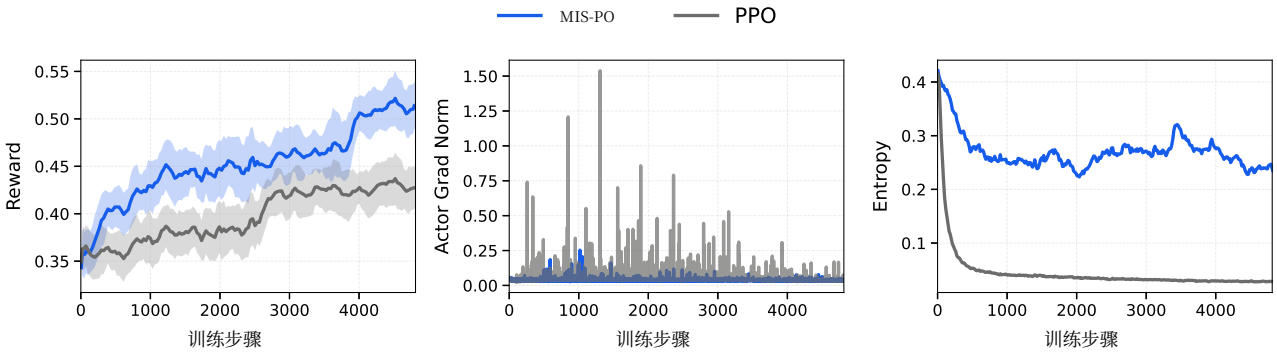


图 5：**在我们的内部模型上 MIS-PO 与 PPO 的可扩展性比较。 (1) 效率**：MIS-PO 展示了更优的样本效率，以加速收敛趋势实现更高的奖励平台。 **(2) 稳定性**：MIS-PO 通过抑制梯度噪声和消除策略梯度范数中的大幅尖峰，显著稳定了训练动态。 **(3) 探索持续性**：MIS-PO 表现出更慢的熵衰减，从而实现更好的探索-利用平衡。

通过这种方式，专家模型能够生成更高质量的轨迹，从而提高整体性能。这种方法特别适用于需要高度稳定性和效率的场景，例如在自然语言处理任务中。通过拒绝采样，可以有效地避免一些常见的错误模式，从而提高模型的鲁棒性。总之，这种方法为强化学习提供了一种更有效、更稳定的优化路径。

超参数。 使用Muon优化器 [34] ，采用3%的预热和从1.0×10^{-5}到5.0×10^{-6}的余弦衰减。我们冻结MoE路由器权重，并禁用EP组平衡损失，类似于中期训练。SFT训练使用MTP损失权重为0.1，全局批大小为32，全局序列长度为128k。关于旋转位置嵌入（RoPE） [83],，我们保持 $\theta_{SWA} = 10$，000并调整 $\theta_{Full} = 5$，000，000以适应128k的上下文长度 [84]。

5.2. 可扩展强化学习

在大语言模型的强化学习中，我们优化策略 π_θ 以在轨迹 $\tau = (s_0, a_0, \ldots, s_T)$ 上最大化终端奖励，其中 a_t 表示在状态 s_t 生成的标记。然而，对于推理任务，该过程由于高梯度方差而面临严重的不稳定性，而超长时序和模型规模（图 5 (2)）进一步放大了这种不稳定性。这种方差主要源于高吞吐量推理引擎和训练框架之间的 **基础设施差异**，以及迭代更新中固有的 **离策略错位**。在这种情况下，重要性采样本质上是不稳定的，因为微小的标记级概率变化会累积成干扰收敛的噪声梯度。

5.2.1. MIS-Filtered Policy Optimization(MIS-PO)

为解决这些稳定性挑战，我们提出了MIS-PO方法，该方法受Metropolis独立采样（MIS） [39,40]启发。我们将推理策略视为提议分布，将训练策略视为目标分布，限制更新仅针对保持在目标分布附近足够近的样本。与重要性采样不同，后者通过有界比率缩放梯度且常受高方差影响，MIS-PO应用二进制掩码过滤离分布样本，并将保留的轨迹视为有效在策略上，从而显著降低梯度方差并实现稳定优化。

形式上，我们定义一个二元指示函数 $\mathbb{I}(x) = \mathbb{1}[\rho\min \le x \le \rho\max]$，并在两个不同的粒度上应用它。在<样式id='6'>token级别</样式>，该函数过滤概率比率$x_t = \pi_{\theta_{\text{old}}}(a_t|s_t)/\pi_{\theta_{\text{vllm}}}(a_t|s_t)$，以抑制训练策略与推理策略之间的局部不匹配 [37]。在<样式id='13'>轨迹级别</样式>，我们对几何平均比率 $\bar{\rho}(\tau) = (\prod_t x_t)^{\frac{1}{T}}$ 应用相同的指示器，有效丢弃已显著偏离目标分布的整个轨迹。重新表述的演员损失用这些双层离散掩码替换了连续的重要性权重：

$$\mathcal{L}_{actor} = -\mathbb{E}_{\tau\sim\pi_{\theta_{\text{vllm}}}}\left[\mathbb{I}(x_t)\cdot\mathbb{I}(\bar{\rho}(\tau))\cdot\log\pi_\theta(a_t|s_t)\cdot\hat{A}_t\right]. \tag{2}$$

通过将有效样本视为在策略样本，该目标在信任区域约束下显著降低了长时程推理任务的梯度方差。图5展示了约5000个训练步数的消融研究，其中MIS-PO在演员梯度范数上的噪声显著低于PPO，表明可扩展性得到提升。更多消融实验展示在附录D.2.3中。

为进一步稳定训练动态，我们采用多种技术：**截断感知值引导** [87] 来纠正由上下文长度截断引入的雄心勃勃的奖励偏差，以及**路由置信度**监控来预测MoE架构特有的不稳定性。

截断感知值引导。将零奖励分配给上下文截断轨迹会将截断与任务失败混淆。这种歧义通过无法区分不完整和错误的结果来惩罚长链推理。为解决这个问题，我们用最终状态的引导值估计来替换零奖励，将截断视为一个时程中断而非终端失败。轨迹 τ_i 的修改后奖励定义为：

$$\hat{R}_i = \begin{cases} V_\phi(s_T) & \text{if the response is truncated,} \\ R_i & \text{otherwise.} \end{cases} \tag{3}$$

从经验上看，这种截断感知值引导能在高达20%的截断率下稳定训练，防止因轨迹不完整通常引发的奖励退化 [88,89]。消融研究表明，这种技术对竞争级基准测试特别有益，因为在这些测试中，长时程推理使得截断效应最为普遍。

路由置信度作为稳定性代理。 近期研究 [36, 38] 将RL稳定性与MoE路由一致性联系起来。在此基础上，我们提出将**路由置信度**(Σ_k)作为稳定性代理，即激活专家的平均概率质量。低 Σ_k 意味着高路由不确定性，这会放大训练-推理不匹配。通过初步实验，我们识别出一种独特的稳定性相变：路由置信度低的模型非常脆弱，需要极端的稳定化（例如，Router Replay [1,36,38], 严格的在线策略更新 [90]）。相比之下，路由置信度高的模型保持鲁棒性，无需复杂的干预即可进行离线策略训练。

强化学习训练动态。 为了全面展示我们的方法，我们在图6中展示了具有可验证奖励（RLVR）的强化学习训练动态以及步骤3.5 Flash在下游评估中的改进。训练奖励的稳步上升表明了一个稳定有效的学习过程。此外，步骤3.5 Flash在多种评估基准上实现了持续的性能提升。具体来说，我们在IMO-AnswerBench上观察到了 +3.2%的显著改进，在CF-Div2-Stepfun-cpp（我们的自定义编码基准，如附录E.2.1所述）上观察到了 +6.1%的显著改进，在ARC-AGI-1 [92], 上观察到了 +10.6%的显著改进，在HLE_{text} [93]上观察到了 +3.4%的显著改进。

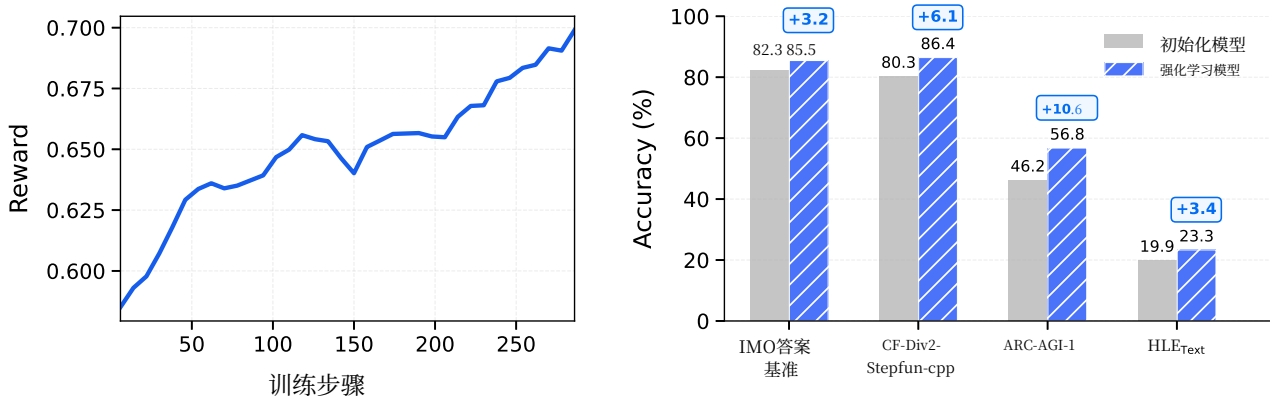


图6：**步骤3.5 Flash的强化学习训练动态及跨域改进**。强化学习驱动持续奖励增长（左图）并在多个基准测试中提供稳定准确率提升（右图）。

5.2.2. 奖励系统

我们将强化学习框架解耦为具有可验证奖励（RLVR [94]）的强化学习和具有不可验证奖励（例如，RLHF [95]）的强化学习，每种强化学习都由一个针对其监督特征的独特奖励支持。

可验证奖励。对于RLVR，每个提示都与一个特定任务的验证器配对，验证器会输出奖励。基于规则的检查器用于逻辑、指令遵循和代码，而基于模型的验证器则用于STEM任务。在我们的内部模型上进行的450步RL训练消融研究中，使用基于模型的验证器处理STEM任务的表现优于直接使用Vanilla数学验证，平均提升2.0%；更多细节请参见附录D.2.2。

不可验证奖励。我们使用成对生成式奖励模型（GenRM [96]）处理不可验证任务，该模型将响应与固定参考进行比较进行基准测试。GenRM是一个推理模型，输出一个置信分数，表示响应获胜的可能性。该分数随后被转换为Bradley–Terry胜率 [97]，作为奖励信号。长度控制被建模为GenRM中的置信分数惩罚，并传递到胜率奖励中，从而在RL训练过程中有效抑制过度长度增长。我们进一步通过为包含伪造引用、过度自信主张或语言不一致的响应分配零奖励来确保鲁棒性。

代理奖励。搜索任务使用基于实体匹配分数的大语言模型进行评估。对于报告生成，基于评分标准的大语言模型裁判 评估研究查询、评分标准规范和候选报告，产生三元判断（满意、部分满意、不满意）[98]。由于中间类别经常与专家偏好不一致，我们将输出映射为非对称二元奖励，从而产生更清晰的学习信号，并更快地收敛到与专家偏好一致的行为。

GenRM 训练和 MetaRM。我们通过使用 RM 特定的提示对 SFT 模型进行微调来初始化 GenRM。对于 RL 训练，我们使用与标量奖励模型公式类似的 logsigmoid 损失的精选成对偏好数据。为了提高 GenRM 的鲁棒性，我们通过集成 MetaRM（一个额外的验证器）来惩罚表现出虚假推理的响应（即，从有缺陷的逻辑中得出正确偏好），当检测到此类模式时，该验证器会降低训练奖励

领域	样本数	标记数 i	
Math	68055	0.98B	11.19%
Code	86421	1.23B	21.10%
STEM	120399	0.55B	6.31%
逻辑性	93323	0.81B	13.87%
通用	314495	0.80B	9.16%
代码代理	37240	0.90B	17.70%
工具使用	114507	0.76B	8.72%
搜索代理	20256	0.50B	8.75%
长上下文	15565	0.70B	4.00%
总计	870687	7.23B	100.00%

表3：第一阶段SFT的数据统计

时。在我们的内部模型上进行的涵盖 200 个 RL 训练步的消融研究中，MetaRM 增强的 GenRM 在每个基准测试中都比 vanilla GenRM 高 0.5% - 3%。

5.2.3. 超参数

对于推理过程，我们将采样温度和top-p 均设置为1.0，最大序列长度为128k个token。每个生成过程中，对于推理任务，我们采样256个唯一提示，每个提示生成16个响应；对于人类偏好任务，采样512个唯一提示，每个提示生成8个响应；对于工具使用任务，采样128个唯一提示，每个提示生成8个响应。推理完成后，完成的样本被划分为小批量，并在单个epoch内用于训练，其中演员使用4个小批量，评价者使用12个小批量。优化过程使用Muon优化器，权重衰减为0.1。演员使用学习率 2×10^{-6} 和20个预热步骤进行训练，而评价者使用学习率5×10^{-6} 和50个预热步骤。遵循ORZ [90], , 我们将 γ 和 λ 均设置为1。我们进一步采用无偏KL损失 [85]，在最终阶段系数为0.001。对于公式(2)，token级别和轨迹级别的掩码边界分别设置为 $[0.5, 2]$ 和 [0。996、1.001], 。

5.3. 数据合成与策展

我们通过聚合开源数据、合成生成和用户轨迹构建了一个多样化且难度平衡的提示池。应用了统一的合成与策展流程，结合严格的全球过滤和特定领域的细化，以最大化推理密度。通过基于规则的启发式方法和基于模型的保真度检查相结合，确保了数据质量。生成的数据集包含87.1万个样本（723亿个token），详细统计数据汇总在表3中。

5.3.1. 通用与推理

我们的训练语料库聚合了来自不同开源社区提示、专家回复和合成数据，包括 **数学** [90,99–110], **编程** [111–113], 以及 **科学和开放式问答** [114–117]。为最大化推理密度，我们采用统一流程，结合严格的全球过滤和特定领域的细化，通过基于规则的启发式方法和基于模型的保真度检查来确保质量。具体来说，在数学领域，我们通过专家指导的拒绝采样和合成大数算术来确保数值稳定性。对于编程，我们通过选择严谨的算法挑战来优先考虑离线可执行性，同时严格清除与RAG相关的幻觉。特别是，我们缓解模型错误声称可以访问外部搜索引擎或假装检索在线解决方案的倾向。此外，我们将科学数据限制为具有明确、可确定解的不模糊问题。

外部搜索引擎或假装检索在线解决方案。此外，我们将科学数据限制为具有明确、可确定解的不模糊问题。
为实现在实际场景中的泛化能力，我们扩展开源检查器[2] 并使用多种现实世界约束增强样本。同时，我们从开源、合成和用户轨迹中收集通用提示，形成一个多样化且难度均衡的提示池。该过程产生了一个包含数百万样本、达到百亿级token规模的高保真数据集。

5.3.2. 通用工具学习

我们提出了一种执行驱动数据生成框架，用于在智能体中学习可靠的工具使用行为，解决了现有合成管道的关键局限性，如数据不一致性、缺乏可验证性和模型幻觉。我们的方法不依赖于随机探索 [118, 119] 或基于模型的模拟 [5, 120], ，而是将工具使用行为分解为原子意图，并使用有限状态机（FSM）对它们进行建模，明确地将抽象的工具调用逻辑与参数化执行约束分开。数据通过带有拒绝采样的采样-执行-验证循环生成，其中所有候选轨迹都在真实环境中执行，并由确定性反馈进行验证，确保保真度并消除幻觉行为。通过组合式结合原子意图，该框架支持复杂、可控的工具使用场景的可扩展生成。使用这种范式，我们构建了超过 100K 的高质量轨迹，总计数十亿个 token，为基于工具的计划、推理和执行提供了精确的监督。

5.3.3. 代码代理

代码代理可以通过验证环境构建和解决方案生成之间的闭环干预进行自我提升，其中可执行的反馈持续优化双方的能力。我们将环境构建视为与错误修复和功能实现同等重要的第一类能力，在可验证的奖励信号下将其综合。为此，我们开发了一个从SWE-factory [121] 框架演化而来的专用代理管道，其中包含一个跨任务内存池，该内存池将历史构建成功作为少样本演示进行检索，并包含一个循环检测机制以防止冗余探索。该管道实现了40%的环境构建成功率，通过从构建轨迹（包括shell命令和错误恢复）的密集监督中形成正向反馈循环，促进模型自我进化。为了进一步提高信号质量，我们通过抽象和屏蔽不贡献于最终解决方案的瞬态故障和冗余执行模式来规范化环境构建轨迹。引导环境作为动态测试平台，利用执行反馈和单元测试生成高质量的合成数据和奖励信号，以实现持续对齐。经验上，我们观察到双向迁移：构建专业知识加速编码性能，而在此类环境中编码进一步提高了构建精度，如DockSmith [122]所示。利用该进化管道，我们整理了50K个验证环境，涵盖超过15K个GitHub仓库和20多种编程语言。这个多样化的集合捕捉了广泛的现实世界场景，为训练通用代码代理提供了坚实的基础。

5.3.4. 搜索和研究代理

为了促进高级信息检索，我们的管道集成了基于图和多文档合成来执行多跳推理。通过在知识图谱（例如，Wikidata5m [123]）上执行拓扑扩展，并模拟跨网站浏览轨迹，我们生成了反映现实世界研究复杂性的数据。关键的是，为了保证外部检索的必要性，我们

[2] https://github.com/allenai/open-instruct/tree/main/open_instruct/IFEvalG

验证生成的查询系统性地排除可通过此强推理模型无需工具交互解决的实例。生成的轨迹通过一个结构化报告生成管道 [125] 进行精炼，该管道强制执行严格的指令合规性和结构完整性。具体而言，我们强制遵守预设的研究计划，丢弃任何偏离结构的轨迹。随后，有效的输出通过基于模型的判别器和启发式规则进行迭代清理，以解决诸如非正式写作、时间幻觉和混合语言伪影等细粒度问题。这种端到端方法在 RESEARCHRUBRICS [21] 基准测试上实现了行业领先性能。

5.4. AGENT 基础设施

使用工具使用模板设计进行推理。 为了将推理和 AGENT 能力有效集成到一个单一的基础模型中，确定思考过程和工具使用的适当模板至关重要。关于推理模板，我们评估了三种管理策略。每次都丢弃推理历史的做法 [124], ，同时激励独立生成，会导致长时程任务（例如，超过 100 轮的编码会话）的任务失败。相反，保留完整的推理历史会导致过高的上下文消耗，这会迅速饱和模型的容量并阻止后续的工具调用。为了解决这个问题，我们采用了一种选择性保留策略：仅保留由最新用户指令触发的工具使用轨迹的推理轨迹。这种设计在推理连贯性和上下文效率之间实现了最佳权衡，与最近的前沿模型 [85,126] 的实践一致。关于工具使用模板，我们比较了流行的 JSON 和 XML 格式。JSON 的严格语法，包括转义序列和分隔符，经常在小型、训练不足的模型中导致解析错误。相比之下，XML 格式允许扁平字符串输出，语法开销显著更低。因此，我们选择 XML 格式，以确保在复杂的现实世界 AGENT 编码场景中的鲁棒性。

可扩展代码代理基础设施。 我们的集成架构专注于可扩展的会话管理和跨框架泛化，以促进高吞吐量的代理式编码。其核心是一个专有的会话路由器，该路由器通过 Kubernetes 管理容器生命周期，并通过 Tmux 确保交互一致性。该架构支持数千个并发环境，并具有无缝的状态持久化，无需手动进行特定脚手架的 Docker 配置。为确保跨多样化的代理式工作流程实现高泛化能力，我们训练模型以适应广泛的各种交互框架，从学术标准（例如，Open-Hands [127], SWE-agent [128], 和 Terminus-2 [16]）到企业级协议（例如，Kilocode [129],Roocode [130], 和 ClaudeCode [131]）。通过在训练过程中向模型暴露这些多样的交互范式，我们有效地防止其过度拟合特定的管道模式，确保无论底层执行环境如何，模型都能保持稳健。

6. 评估

6.1. 预训练评估

评估设置。 我们在一系列基准测试上评估了 Step 3.5 Flash，涵盖了各种能力：(1) 通用语言理解和推理，包括 BBH [132],MMLU [133],MMLU-Redux [134], MMLU-Pro [135], HellaSwag [136], WinoGrande [137], GPQA [138], SuperG- PQA [139], 和 SimpleQA [140]。(2) 数学推理，包括 GSM8K [141] 和 MATH [142]。(3) 编程，包括 HumanEval [143], MBPP [144], HumanEval+, MBPP+ [145] 和MultiPL-E [146]。(4) 中文理解，包括 C-Eval [147], CMMLU [148], 和 C-SimpleQA [149]。

基准测试	# 样本	步骤3.5 Flash Base	MiMo-V2 闪存 Base	GLM-4.5 Base	DeepSeek V3.1 Base	DeepSeek V3.2 实验基础	Kimi-K2 Base
# 激活参数	-	**11B**	15B	32B	37B	37B	32B
# 总参数	-	**196B**	309B	355B	671B	671B	1043B
通用							
BBH	3-shot	**88.2**	88.5	86.2	88.2†	88.7†	**88.7**
MMLU	5-shot	**85.8**	86.7	86.1	87.4†	**87.8**†	87.8
MMLU-Redux	5-shot	**89.2**	**90.6**	-	90.0†	90.4†	90.2
MMLU-Pro	5-shot	**62.3**	**73.2**	-	58.8†	62.1†	69.2
HellaSwag	10-shot	**90.2**	88.5	87.1	89.2†	89.4†	**94.6**
WinoGrande	5-shot	**79.1**	83.8	-	**85.9**†	85.6†	85.3
GPQA	5-shot	**41.7**	43.5*	33.5*	43.1*	37.3*	43.1*
SuperGPQA	5-shot	**41.0**	41.1	-	42.3†	43.6†	**44.7**
SimpleQA	5-shot	**31.6**	20.6	30.0	26.3†	27.0†	**35.3**
数学							
GSM8K	8-shot	**88.2**	**92.3**	87.6	91.4†	91.1†	92.1
MATH	4-shot	**66.8**	**71.0**	62.6	62.6†	62.5†	70.2
CODE							
HumanEval	3-shot	**81.1**	77.4*	79.8*	72.5*	67.7*	**84.8***
MBPP	3-shot	**79.4**	81.0*	81.6*	74.6*	75.6*	**89.0***
HumanEval+	0-shot	**72.0**	70.7	-	64.6†	67.7†	-
MBPP+	0-shot	**70.6**	71.4	-	**72.2**†	69.8†	-
MultiPL-E HumanEval	0-shot	**67.7**	59.5	-	45.9†	45.7†	60.5
MultiPL-E MBPP	0-shot	**58.0**	56.7	-	52.5†	50.6†	**58.8**
中文							
C-EVAL	5-shot	**89.6**	87.9	86.9	90.0†	91.0†	**92.5**
CMMLU	5-shot	**88.9**	87.4	-	88.8†	88.9†	**90.9**
C-SimpleQA	5-shot	**63.2**	61.5	70.1	70.9†	68.0†	**77.6**

表4：预训练评估结果。*表示原始分数不可用；我们报告了在步骤3.5 Flash相同测试条件下评估的结果，以进行公平比较。†表示来自MiMo-V2-Flash报告的DeepSeek分数 [30]。

评估结果。 表4总结了 Step 3.5 Flash 在通用推理、数学、代码和中文基准测试上的预训练评估。尽管仅激活了 11B 参数（总共 196B），Step 3.5 Flash 仍然与远大于的稀疏基线（激活 15–37B；总共 309–1043B ）具有广泛的竞争力，展示了强大的准确率 - 效率权衡。在核心通用基准测试上，Step 3.5 Flash 在 BBH 上达到 88.2（在最佳值附近 0.5），在 MMLU 上达到 85.8。值得注意的是，Step 3.5 Flash 在 SimpleQA 上达到 31.6，优于 DeepSeek-V3.2-Exp Base (27.0)，而使用的总参数仅为 196B，相比之下是 671B（即 ~3.4× 总参数），突显了更强的每参数预算能力密度。Step 3.5 Flash 进一步展示了强大的编程能力，包括 HumanEval 上的 81.1、MultiPL-E HumanEval 上的 67.7 和 MultiPL-E MBPP 上的 58.0。总体而言，这些结果表明 Step 3.5 Flash 每个激活的计算提供了强大的性能，为下游推理和代理后训练奠定了坚实的基础。

6.2. 训练后评估

我们在代表性的基准测试上评估了步骤3.5 Flash，包括推理相关的HLE（文本子集） [150], MMLU-Pro [135], GPQA-Diamond [138], AIME2025 [10], HMMT [11], IMO-AnswerBench [91]；编码相关的 LiveCodeBench-v6(2024.08-2025.05) [12], CF-Div2-Step(Ap-

pendix E.2.1):我们的自定义CodeForces[3] Div2基准测试，SWE-Bench验证版 [13]和SWE-Bench多语言版 [14]；代理系列 τ^2-Bench [15], Terminal-Bench 2.0 [16], GAIA [19], BrowseComp [17], xbench-DeepSearch [20], BrowseComp-zh [18], 和RESEARCHRUBRICS [21]；通用相关的 ArenaHard v2 [151],IFBench [152]和MultiChallenge [153]；以及长上下文相关的LongBench v2 [154], MRCR [155] [4], FRAMES [156] 和RepoQA [157]。

我们进一步通过采用<样式 id='1'>并行协调推理（PaCoRe）</样式>范式 [42]，研究了步骤3.5 Flash在推理、通用和长上下文基准测试上的测试时扩展特性。利用步骤3.5 Flash的极致推理效率，该方法通过启动并行推理轨迹，并通过多轮协调将它们的见解合成更高保真度的解决方案，将推理能力与上下文限制解耦。具体来说，我们采用多轮PaCoRe轨迹配置作为 $\vec{K} = [4, 4, 4, 4]$，在所有基准测试中都取得了显著提升。

我们保持最大序列长度为256k，使用默认解码配置，解码温度和top-p均为1.0。我们还对原始128k位置嵌入应用YaRN [158]，缩放因子为2.0，并仅限制在完整注意力层上。我们基于每个问题多次独立生成的平均性能，报告所有方法的pass@1准确率：AIME 2025、HMMT 2025 二月和HMMT 2025 十一月为64；IMO-AnswerBench、LiveCodeBench、GPQA-Diamond和MultiChallenge为8；HLE为1；其他所有基准测试为4次运行。更多细节请参见附录E.2。

评估结果。 表5展示了步骤3.5 Flash与一系列强大的基线模型在推理、代码代理、通用代理、长上下文理解和通用能力基准测试方面的全面对比。尽管仅激活了11B参数（总共196B），步骤3.5 Flash在广泛任务中表现出色，特别是在推理密集型基准测试上，如AIME 2025、HMMT 2025 二月、HMMT 2025 十一月、IMO-AnswerBench和LiveCodeBench-v6。它始终优于参数数量更大的开源模型，并达到了与前沿模型（如GPT-5.2 xHigh和Gemini 3.0 Pro）相当的性能。值得注意的是，步骤3.5 Flash在代理评估中取得了优异结果，包括SWE-Bench Verified、Terminal-Bench 2.0、BrowseComp（带上下文管理器）、GAIA和 τ^2-Bench，突显了其强大的工具使用和长时程决策能力。总体而言，这些结果表明步骤3.5 Flash以显著更低的活跃参数数量，提供了强大的代理和推理性能。

7. 讨论与局限性

Token效率。 Step 3.5 Flash实现了前沿级智能，但目前达到可比质量需要比Gemini 3.0 Pro更长的生成轨迹。下一步我们将修剪和压缩思考以获得更好的效率，同时保持相同的竞争性能。

高效通用掌握。 我们旨在将通用多功能性与深度领域专业知识相结合。为实现高效，我们正在推进在线策略蒸馏的变体，使模型能够以更高的样本效率内化专家行为。

[3]https://codeforces.com/[4]
https://huggingface.co/datasets/openai/mrcr

基准测试	Step 3.5 Flash Vanilla	Step 3.5 Flash PaCoRe	MiniMax M2.1	MiMo V2 Flash	GLM 4.7	DeepSeek V3.2	Kimi K2.5	Gemini 3.0 Pro	Claude Opus 4.5	GPT-5.2 xHigh
# 激活参数	11B		10B	15B	32B	37B	32B	-	-	-
# 总参数	196B		230B	309B	355B	671B	1T	-	-	-
REASONING										
AIME 2025	97.3	99.9	83.0	95.1*	95.7	93.1	96.1	95.0	92.8	**100.0**
HMMT 2025 二月	98.4	100.0	71.0*	95.4*	97.1	92.5	95.4	97.5 †	92.9 †	**99.4**
HMMT 2025 十一月	94.0	97.8	74.3*	91.0*	93.5	90.2	91.1	94.5 †	91.7*	**97.1***
IMO-AnswerBench	85.4	88.8	60.4*	80.9*	82.0	78.3	81.8	83.3 †	84.0 †	**86.3†**
LiveCodeBench-v6	86.4	88.9	75.4*	81.6*	84.9	83.3	85.0	**90.7†**	84.8 †	87.7 †
CF-Div2-Stepfun-cpp	86.1	93.3	59.0*	46.9*	74.1*	81.6*	73.6*	83.5*	72.2*	-
MMLU-Pro	84.4	84.8	88.0	84.9	84.3	85.0	87.1	**90.1†**	89.5 †	87.4 †
GPQA-Diamond	83.5	85.0	83.0	84.1*	85.7	82.4	87.6	91.9	87.0	**92.4**
HLE_{text}	23.1	27.9	22.2	22.1	24.8	25.1	31.5	**37.7†**	30.8 †	35.5 †
代码代理										
SWE Verified	74.4	-	74.0	73.4	73.8	73.1	76.8	76.2	**80.9**	80.0
SWE Multilingual	67.4	-	72.5	71.7	66.7	70.2	73.0	65.0 †	**77.5†**	72.0 †
Terminal-Bench 2.0	51.0	-	47.9	38.5	41.0	46.4	50.8	56.9 †	**59.3†**	54.0 †
通用AGENT										
BrowseComp	51.6	-	47.4	45.4	52.0	51.4	**60.6**	37.8 †	37.0 †	-
带上下文管理的浏览组件	69.0	-	62.0	58.3	67.5	67.6	**74.9**	59.2 †	57.8 †	65.8
BrowseComp-ZH	66.9	-	47.8*	51.2*	66.6	65.0	62.3*	66.8*	62.4*	**76.1***
GAIA	84.5	-	64.3*	78.2*	61.9*	75.1*	75.9*	76.6*	76.1*	83.5*
xbench-DeepSearch-2505	83.7	-	68.7*	69.3*	72.0*	78.0*	76.7*	78.3*	77.0*	83.0*
xbench-DeepSearch-2510	56.3	-	43.0*	44.0*	52.3*	55.7*	40.0 †	57.7*	59.3*	**67.0***
研究评估标准	65.3	-	60.2*	54.3*	62.0*	55.8*	59.5*	50.1*	61.6*	57.8*
τ^2-Bench	88.2	-	86.6*	84.1*	87.4	85.2*	85.4*	90.7	**92.5**	85.5*
通用										
Arena-Hard-v2.0	74.0	93.1	63.1*	68.2*	73.1*	66.0*	**85.8***	81.7 †	76.7 †	80.6 †
MultiChallenge	55.7	60.8	50.5*	44.3*	67.8*	57.1*	**73.6***	71.8*	65.8*	71.9*
IFBench	67.4	56.8	70.0	64.0 †	68.0 †	61.0 †	72.8*	70.4 †	58.0 †	**75.4†**
LONG CONTEXT										
LongBench v2	57.5	62.0	53.9*	60.6 †	59.1*	58.4 †	61.0	**70.0***	67.8*	62.4*
MRCR-8needle	28.8	26.3	20.0 †	19.9 †	25.4 †	27.2 †	36.5*	73.0 †	54.0*	**88.2***
FRAMES-Oracle	76.5	77.2	76.5*	78.0*	75.1*	80.1*	77.4*	79.7*	85.8*	**87.3***
RepoQA	88.5	88.7	88.2*	91.2*	89.5*	91.9*	89.8*	91.5*	**95.7***	93.8*

表5：步骤3.5 Flash与封闭式/开放式模型的比较。*表示原始分数不可用或劣于我们的重现结果；因此我们报告了在步骤3.5 Flash相同测试条件下评估的结果以进行公平比较。†表示分数引自非官方来源，包括技术报告或独立评估平台。我们在HLE上的评估专注于纯文本子集。带上下文管理的浏览组件表示启用上下文管理时对BrowseComp的评估。

用于开放世界智能体任务的强化学习。 虽然步骤3.5 Flash 在学术智能体基准测试上表现出竞争性能，但智能体AI的下一个前沿需要将强化学习应用于专业工作、高级工程和科学研究中发现的复杂、专家级任务。解决这些挑战是部署能够实现真正自主的智能体的先决条件。

操作范围和限制。 步骤3.5 Flash 专为编码和工作相关任务而设计，但在分布偏移（distributionshifts）期间可能会出现稳定性降低的情况。这通常发生在高度专业化的领域或长时程、多轮对话中，此时模型可能会表现出重复性推理、混合语言输出或时间与身份意识不一致的情况。

附录

A. 架构细节

表6总结了步骤3.5 Flash的关键架构超参数。

超参数	值
BACKBONE	
词汇量 (V)	128,896
模型宽度 ($d$model)	4096
Transformer 块	45 (3 密集 + 42 MoE)
MoE FFN	
每个 MoE 块的专家数	288 + 1 共享
路由	top-k = 8
密集FFN隐藏大小	11,264
MoE专家隐藏大小	1,280
注意	
混合块结构	3 个 SWA 块 + 1 全注意力块
SWA 窗口大小	512
KV 头 (GQA)	8
查询头 (全 / SWA)	64 / 96
门类型	输出头向
头维度	128
RoPE θ	10,000
RoPE 维度（完整 / SWA）	64 / 128
多标记预测	
MTP 块	3 (密集 SWA)
参数统计	
总参数（骨干网络）	196B
激活参数 / token (主干)	11B
总参数 (含MTP3)	198B
激活参数 / token (含MTP3)	13B

表6：步骤3.5 Flash的关键架构超参数。“激活参数”按token报告，不包括嵌入/输出矩阵。

A.1. 逐头门控注意力

每个注意力头被分配一个轻量级、输入相关的标量门控，允许模型在混合布局中动态调节信息流，且计算开销可忽略不计。

形式上，对于一个维度为 d的（单个）头，令 $\boldsymbol{q}_i, \boldsymbol{k}_j, \boldsymbol{v}_j \in \mathbb{R}^d$ 分别表示位置 i 的查询向量以及位置 j的键向量和值向量，缩放点积分数 s、相应的注意力权重 α以及输出 $\boldsymbol{y}$ 按照如下方式计算：

$$s_{i,j} = \langle \boldsymbol{q}_i, \boldsymbol{k}_j \rangle / \sqrt{d}, \qquad Z_i = \sum_{j'} \exp(s_{i,j'}), \qquad \alpha_{i,j} = \exp(s_{i,j})/Z_i, \qquad \boldsymbol{y}_i = \sum_j \alpha_{i,j} \boldsymbol{v}_j. \tag{4}$$

给定位置 i的输入表示 $\boldsymbol{x}_i$，我们计算逐头门控 g_i 来调制头输出：

$$g_i = \sigma(\boldsymbol{w}_{gate}^{\top}\boldsymbol{x}_i), \qquad o_i^{\text{gate}} = g_i\,\boldsymbol{y}_i, \tag{5}$$

其中 $\sigma(\cdot)$ 是 sigmoid 函数，而 $\boldsymbol{w}_{gate}$ 是一个可学习的向量。

逐头门控可以看作是在注意力机制中引入一个 输入相关的汇流标记 [33]。将 $\sigma(g) = \frac{1}{1+\exp(-g)}$ 代入公式5，我们得到

$$\boldsymbol{o}_i^{\text{gate}} = \sum_j \frac{\exp(s_{i,j})}{Z_i + e^{-g_i} Z_i}\boldsymbol{v}_j, \tag{6}$$

其中 $\exp(-g_i)Z_i$ 在 softmax 归一化中充当一个 输入相关的汇流质量 。如第2.3节所示，这种自适应公式始终优于固定的（与输入无关）汇流标记。

A.2. 注意力增强的速度基准测试

我们使用 MTP-3 进行模拟，以评估在理想工作负载下两种增强的延迟开销。表7展示了理论FLOPs和延迟的相对增量。增加SWA的查询头数量会略微提高FLOPs，但对延迟影响较小。这是由于查询头与 kv 的比例为12，即使考虑MTP-3，SWA仍处于IO受限区域。对于逐头门控，由于其轻量级特性，FLOPs和延迟都没有明显差异。

主干网络	SWA <Heads>	设置	解码（FLOPs / 延迟）		预填充（FLOPs / 延迟）	
			64k	256k	64k	256k
步骤3.5 Flash ($S3F1$ 布局)	64	无门	1.00 / 1.00	1.00 / 1.00	1.00 / 1.00	1.00 / 1.00
	96	无门	1.02 / 1.01	1.01 / 1.00	1.08 / 1.06	1.04 / 1.03
	64	从头部	1.00 / 1.00	1.00 / 1.00	1.00 / 1.02	1.00 / 1.01
	96	横向	1.02 / 1.02	1.01 / 1.00	1.08 / 1.08	1.04 / 1.05

表 7：不同 SWA 头数和门控策略下的相对增量。指标以 FLOPs / 延迟呈现。基线配置（第一行）已归一化为 1.0。

主干网络	布局	SWA 头部	解码		预填充	
			64K	256K	64K	256K
步骤3.5 Flash	$S3F1$	64	1.00	1.00	1.00	1.00
	$S3F1$+Head	96	1.02	1.01	1.08	1.04
	$S1F1$	64	1.18	1.47	1.38	1.71
	$FFFF$	64	1.51	2.33	2.07	3.00
内部30B-A3B	$S3F1$	32	1.00	1.00	1.00	1.00
	$S3F1$+Head	48	1.02	1.01	1.05	1.02
	$S1F1$	32	1.42	1.74	1.50	1.80
	$FFFF$	32	2.21	3.16	2.47	3.34

表8：不同骨干网络和注意力模式下的相对FLOPs成本。头部数量指的是SWA头部。对于每个骨干网络，具有最小FLOPs（$S3F1$ 减少头部）的配置是基线（1.0）。

A.3. 元标记

近期文献 [159–161] 已从理论和实证两方面证明，在预训练序列前添加结构化元数据可以提高数据效率并加速收敛：通过暴露高级属性（例如，模态、语言、领域），元数据提供了全局线索，减少了关于即将出现内容的 uncertainty，从而使下一个词的预测更容易。

受此范式驱动，我们将每个训练样本与一个人类可读格式的元数据字符串 $\mathbf{M}$ 相关联，该字符串包括内容类型（例如，代码、书籍、论文、网页）、语言（例如，英文、中文）、领域和来源。然后，我们将 $\mathbf{M}$ 添加到原始标记序列 $\mathbf{x}$ 的开头，形成一个单一的训练序列 $\mathbf{s} = [\mathbf{M};\mathbf{x}]$。在预训练期间，模型被训练以最大化 $\mathbf{s}$ 的似然性：

$$\mathcal{L}_{\text{full}}(\theta) = -\sum_{t=1}^{|\mathbf{s}|} \log P_\theta(s_t \mid \mathbf{s}_{<t}). \tag{7}$$

经过大约 3.8T 个标记的初始阶段后，我们保留 $\mathbf{M}$ 在上下文中，但将其位置从损失中遮蔽，同时继续预测有效载荷标记：

$$\mathcal{L}_{\text{mask}}(\theta) = -\sum_{t=|\mathbf{M}|+1}^{|\mathbf{s}|} \log P_\theta(s_t \mid \mathbf{s}_{<t}) = -\sum_{t=1}^{|\mathbf{x}|} \log P_\theta(x_t \mid \mathbf{M}, \mathbf{x}_{<t}). \tag{8}$$

我们假设到这个阶段，模型已经学会有效地将元数据用作条件信号。遮蔽元数据损失因此将优化压力完全分配给有效载荷标记，同时仍然受益于对数据特征的显式条件化。

A.4. 预训练消融细节

我们进行受控的预训练消融实验，以隔离（i）不同混合注意力布局和（ii）汇流标记与逐头门控注意力的效果。

混合注意力布局。 我们采用<code>30B-A3B MoE</code>架构来评估在固定token预算下不同混合注意力布局的下游影响。训练遵循严格的、多阶段的流程：一个30B-token的预热阶段，接着是1T-token的主预训练阶段，一个300B-token的冷却阶段，以及一个额外的100B-token的长上下文专门化阶段——总共约1.4T-token。然后在一个0.1× 下采样的数据集上进行监督微调（SFT）。完整的训练细节在表9中提供。

门控与汇流标记（缩放设置）。 我们预训练一个<code>100B-A10B MoE</code>模型，用于比较汇流标记和逐头门控在大规模环境下的表现。

架构消融的预训练结果展示在表2和表10中。我们采用第6.1节中详细描述的评估协议。具体来说，GPQA [138] 使用5-shot提示进行评估，而HumanEval [162]和MBPP [163]则使用3-shot提示进行评估。

表1中的后训练结果如下聚合：

- **推理：** MMLU-Pro [135], GPQA-Diamond [138],LiveCodeBench v6 [12], 和LiveBench [164]的平均值。
- **数学：** AIME 2024 [165], AIME 2025 [166], HMMT 2025 二月 [167], 和CNMO 2024[5]的平均值。
- **代码：** CF-Div2-Stepfun 和 LiveCodeBench v6 [12] 的平均值。

[5]https://www.cms.org.cn/Home/comp/comp/cid/12.html

超参数	100B-A10B	30B-A3B
总token数	250B	1.4T
优化器	Muon [34]	
峰值学习率	1.31×10^{-4}	1.1×10^{-3}
批大小预热	-	30B tokens
层	43	48
维度	4096	2048
主导密集层	1	1
路由专家	96	128
活跃专家	4	8
共享专家	1	1
负载均衡方法	无损失 [64]	
注意力模块	GQA8	
序列长度	4096	
词汇量大小	129280	
批大小	8192	16384
权重衰减	0.1	
部分RoPE	禁用	启用
MTP	启用	禁用

表 9：100B-A10B 和 30B-A3B 架构消融套件的训练配置。

• **科学：** 由 GPQA-Diamond [138] 表示。
• **通用：** IFEval [168], IFBench [152], WildBench [169], Arena-Hard [151], 和MultiChallenge [153]的平均值。
• **长上下文：** 六个基准级平均值的平均值：(i) 在RULER [170], 上针对上下文长度8k-128k的平均分数 (ii) 在Long-Bench v2 [154],的短集和中等集中的平均分数 (iii) 在HELMET [171], 上针对上下文长度8k-128k的平均分数 (iv) GSM-无限 [172], (v) 在FRAMES [156],上的总分数和 (vi) 在RepoQA [157]上的总分数。

表1和表10显示，vanilla *S3F1* 布局在通用预训练基准上的表现不如全注意力基线，并且持续降低SFT质量 (例如，BBH: –4.3；SFT平均： –0.7)。增加SWA查询头的数量可以显著缩小这一差距（例如，MMLU-Pro： +3.7；SFT推理： +0.4），在SFT代码 (–0.6) 上只有轻微的回归，同时在多个指标上达到或超过全注意力基线。表2进一步证明，头门控注意力在汇流标记指标上平均提升了62.5到64.4 (+1.9)。

B. 局部激活膨胀的详细分析

为调查局部激活爆炸的根因，我们分析所有层中触发最大专家激活的标记，并识别出两种不同的显著激活模式：(1) 特定词汇项，如特殊标记和标点符号，通常会引发较大但不剧烈的激活，尤其是在较浅的层中。该模式不被我们视为一种故障模式，因为激活没有快速增长，它可能作为语义的内部机制

布局	SWA 头	预训练评估											
		BBH	MMLU	MMLU-Redux	MMLU-Pro	SimpleQA	GSM8K	MATH	HumanEval	MBPP	C-EVAL	CMMLU	平均
FFFF	32	**66.0**	64.5	69.7	35.7	7.2	70.0	39.2	**48.8**	53.4	69.7	70.5	54.1
S1F1	32	64.1	64.7	69.8	**37.7**	7.5	70.1	43.9	47.0	56.2	69.8	69.8	54.6
S3F1	32	61.7	64.2	69.4	33.7	**8.0**	67.4	41.5	47.6	56.0	69.5	70.9	53.6
S3F1+Head	48	65.3	**65.9**	**71.0**	37.4	7.5	**72.2**	**44.5**	**48.8**	**58.6**	**70.2**	**71.0**	**55.7**

表10：混合注意力布局消融实验的预训练评估结果(W=512)在30B-A3B上.F 表示全注意力， S 表示SWA； $S3F1$ 表示混合布局中有三个 S 和一个 F .$S3F1$+Head将SWA头的数量从32增加到48。

建模 [60,173]。另一个模式是 (2) 一些高频的二元组在第一个词上触发极大的激活，这代表了我们所调查的失效模式。该模式由多个因素触发：二元组的出现频率足够高，并且 MoE FFN 足够精细，允许一个专家专门处理该二元组而不受负载均衡机制的限制。这种专门化充当了一个捷径：一旦专家被激活，输出就变得确定性，其他网络不再影响预测。虽然找到捷径是减少损失的一个合理方法，但在具有预规范架构 [76,77], 的 MoE 模型中，有一种直接且病态的解决方案来实现这种确定性预测，如下所述。模型的最终表示是所有层的输出的总和，然后进行 RMSNorm。这可以表示为专家的输出和注意力层的输出的组合：

$$\boldsymbol{h}_{\text{final}} = \text{RMSNorm}(\underbrace{\text{expert}_{\text{outlier}}}_{\boldsymbol{h}_{\text{outlier}}} + \underbrace{\sum_{l=1}^{L} \text{attn}_l + \sum_{l,e \text{ is not a outlier}} \text{expert}_{l,e}}_{\boldsymbol{h}_{\text{others}}}), \tag{9}$$

其中 attn、MoE 和 expert 分别代表各自模块的输出隐藏状态，而 L 和 E 分别表示层数和专家数。直接的方法是无限扩大 $\text{expert}_{\text{outlier}}$，然后

$$\text{RMSNorm}(\boldsymbol{h}_{\text{final}}) = \lim_{c\to\infty} \text{RMSNorm}(c \cdot \hat{\boldsymbol{h}}_{\text{outlier}} + \boldsymbol{h}_{\text{others}}) = \text{RMSNorm}(\boldsymbol{h}_{\text{outlier}}), \tag{10}$$

其中我们将 $\boldsymbol{h}$ 异常值与幅度 c 解耦，并用单位向量 $\hat{\boldsymbol{h}}$ 异常值表示其方向。

SwiGLU 是步骤3.5 Flash 中的专家架构，提供了一种生成大输出的方法，即使权重衰减有效抑制了权重范数。SwiGLU 定义如下：

$$\text{SwiGLU}(\boldsymbol{x}) = \boldsymbol{W}_{\text{down}} \left(\text{SiLU}(\boldsymbol{W}_{\text{gate}}\boldsymbol{x}) \cdot \boldsymbol{W}_{\text{up}}\boldsymbol{x}\right). \tag{11}$$

我们分析了 $\boldsymbol{W}$ 门控 $\boldsymbol{x}$ 和 $\boldsymbol{W}$ 上 $\boldsymbol{x}$ 的激活范数，发现异常专家和正常专家之间没有显著差异。然而，逐元素乘积产生了异常输出，这些输出具有

$$\|\text{SiLU}(\boldsymbol{W}_{\text{gate}}\boldsymbol{x})\| \cdot \|\boldsymbol{W}_{\text{up}}\boldsymbol{x}\| \approx \|\text{SiLU}(\boldsymbol{W}_{\text{gate}}\boldsymbol{x}) \cdot \boldsymbol{W}_{\text{up}}\boldsymbol{x}\|, \tag{12}$$

在异常专家中。这只有在 SiLU($\boldsymbol{W}$门控$\boldsymbol{x}$)和 $\boldsymbol{W}$上$\boldsymbol{x}$ 高度一致并且专注于非常有限数量的维度时才能实现。因此，由于输入的稀疏性，仅利用了来自 $\boldsymbol{W}$上的有限数量的行。这一观察结果使我们倾向于选择激活裁剪而不是权重裁剪，因为激活的数值特性直接导致膨胀和稀疏性，而激活裁剪可以迅速解决这些问题。此外，激活裁剪的负面影响可以忽略不计，因为行为良好的激活很少超过阈值。

在使用Muon优化器时，门控线性单元（如SwiGLU）容易受到logit爆炸的影响。这种脆弱性源于与注意力爆炸相似的机制，正如在 [5]中报告的那样。对于一个专门针对某些特定双字的异常值专家，路由到它的隐藏状态预计与其路由嵌入紧密对齐。我们通过将路由嵌入输入到异常值专家中，并直接根据该专家的输出来预测输出，来验证这一点。预测的分布与真实数据的分布以及整个网络的性能一致。结合过度单一的训练目标（预测双字中的第二个词元），我们认为异常值专家的参数（ $\boldsymbol{W}$gate、 $\boldsymbol{W}$ up和 $\boldsymbol{W}$down）的梯度不仅异常低秩（记为 r），而且始终指向一个强调分析中第一个因素所分析的幅度的方向，而没有旋转。令参数矩阵 $\boldsymbol{W} \in \mathbb{R}^{N\times N}$ 的更新矩阵为

$$\Delta \boldsymbol{W} = \sum_i \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^\top = \underbrace{\sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^\top}_{\text{low rank signal}} + \underbrace{\sum_{j=r+1}^{N} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^\top}_{\text{noise}} \tag{13}$$

累积优化步数中的更新将迅速增加低秩信号的奇异值，导致权重参数爆炸。在GLU结构中， $\|\text{SiLU}(\boldsymbol{W}\text{gate}\boldsymbol{x}) \cdot \boldsymbol{W}\text{up}\boldsymbol{x}\|$在我们的强对齐情况下平方了谱范数，使进展更加剧烈。此外，Muon完全消除了梯度幅值的影响。在爆炸过程中，RMSNorm降低了大输入的梯度。在使用Adam优化器时，其 ϵ 在学习率适应过程中充当阈值，以过滤掉小梯度，这可能会阻碍进展。相比之下，Muon始终如一且有效地正交化梯度，导致更激进的更新。

C. 步骤预训练数据基础

C.1. 知识数据构建

C.1.1. StepCrawl

超越标准网络规模数据集（例如 CommonCrawl），我们开发了 **StepCrawl**，这是一个内部爬取和整理系统，旨在大规模获取 高质量和 多样化的 token。StepCrawl 既是高信号网页的主要数据来源，也是文档式内容（尤其是 PDF）的主要数据来源，这些内容通常包含长文本、高信息密度的材料。

StepCrawl 的一个关键组件是一个由 WebOrganizer 风格模型 [174] 驱动的站点和 URL 选择层。我们借鉴了 WebOrganizer 中引入的功能，并对一个为我们的管道量身定制的版本进行了进一步微调。在爬取过程中，该模型会分析每个抓取到的网页，形成一个轻量级的 LM 在回路中的反馈循环，该循环 (i) 过滤 SEO 驱动的和其他低效用页面，以及 (ii) 通过平衡站点类别（例如，防止工具和电商站点被不成比例地爬取）来指导爬取预算分配，以保持语料库多样性和减少主题偏差。在实践中，StepCrawl 在这种质量和多样性感知的调度策略下，每天处理大约 ~1B 页面。

所有爬取活动均严格遵循 `robots.txt` 和站点特定的访问策略。收集到的内容随后会通过多阶段过滤流程（质量评分、去重和净化），确保仅保留高效用且符合政策的数据用于训练。

C.1.2. 质量优化与分层

质量分层。 受 Nemotron-CC [175]-风格质量分桶启发，我们将内部网络数据划分为质量层级，并优先从较高层级进行采样。我们使用六种轻量级评分器/分类器组合对每份文档进行标记，并在评分器间整合层级分配。在最终方案中，我们保留高/中高/中层级，而丢弃中低/低层级，这显著提升了Token效率。对于书籍和论文语料库，我们应用相同的分层策略，但在退火阶段仅保留高/中高层级以最大化多样性。除了共享的六评分器组合外，我们还集成了针对STEM和知识密集型内容的领域特定过滤器，并对过代表现的领域进行下采样，以确保均衡代表性。

基于嵌入的聚类重平衡。 我们利用基于嵌入的语料库平衡作为进一步减少冗余和缓解分布倾斜的原则性方法。具体来说，我们嵌入大规模中文/英文网络数据，运行 k-means 聚类（100K+ 个聚类），并对质量不均的聚类进行下采样。在消融实验中，冷却阶段中的聚类级重平衡提高了广泛的基准测试。

知识密集型挖掘与增强。 我们使用一个轻量级两阶段流水线构建一个专门的知识子集，该流水线基于上述共享嵌入表示。首先，使用一个经过筛选的高价值实体、概念和关系清单，在嵌入空间中从完整语料库中检索知识密集型文档和段落；这些候选文档由知识密度模型和简单的覆盖率启发式算法进行排序。其次，对于检索到的部分内容，我们应用目标转换，如受控改写和问答合成，以提高可学习性。生成的样本被重新混合回训练混合物中，以增加有效知识信号密度。我们观察到该流水线在消融实验中始终如一地带来了增益，而对其益处的详细因果分析则留待未来工作。

C.2. 代码数据

C.2.1. 纯代码

我们使用 OpenCoder 过滤规则的修改版来优化内部编程数据集 [80]，，引入校准的放宽机制以平衡数据质量和多样性。在我们的流程中，应用 OpenCoder 过滤器会为每个文档生成一组“命中”，每个命中代表一个启发式规则的违反（表明潜在的噪声）。我们根据这些命中数量对语料库进行分类：`hit0` 对于干净文档（零违规），`hit1` 对于一个违规，等等。

我们的内部消融实验揭示了一个明显的质量-多样性权衡：严格过滤（例如 `hit0` 仅）过度修剪语料库，而不过滤则引入过多的噪声。我们发现 `hit0-6` 配置（接受最多有 6 个违规的文档）产生了最佳的基准测试性能，与原始严格约束相比，保留了更广泛的高信号代码。

C.2.2. PR/Issue/Commit 数据

为提升软件工程能力，我们从GitHub上超过10星级的仓库构建了一个综合数据集，包含拉取请求、问题和提交。我们对仓库流行度和内容质量进行严格筛选，并使用大语言模型生成缺失的问题描述，形成了一个500万样本基础。由此，我们衍生出四个训练子集：

(1) 基础PR/Issue/Commit数据：我们通过GHArchive和GitHub API爬取数据，包括完整的提交历史。我们提取变更，并对照 `git diff`真实情况验证一小部分样本，然后筛选至 20+ 主流语言（例如，Python、Java、C++）。我们严格与SWE-Bench验证版 [13]和SWE-Bench多语言版 [14] 进行去重，以防止数据泄露。

(2) 拼接的 PR 对话数据 (90B tokens): 我们通过应用两种 Agentless 启发的模板 [82] 生成了 90B tokens 的代码编辑训练数据：(1)文件定位：给定问题描述和仓库结构，识别目标文件路径；(2) 代码修复：给定问题描述和文件内容，通过 SEARCH/REPLACE 块生成精确的修改。

我们将这 90B 代码编辑数据整合到两个具有特定掩码策略的训练阶段中。在预训练的退火阶段，仅掩码模板骨架；在中训练阶段，数据转换为带用户提示的聊天对话。内部消融实验表明，在冷却阶段和中训练阶段，与 SWE-Bench 验证版和 SWE-Bench 多语言版相比，始终获得一致的提升。

(3) 重写的面向推理数据 (12B tokens): 从我们的基础数据集的 Python 子集中，我们通过 LLM 变更类型标注导出 bug 修复样本。我们应用两种简洁的重写策略：(1) 推理重建：LLM 重建 PR 作者的问题解决过程（问题分析、根本原因识别、解决方案设计和代码实现），注入到 PR-Dialogue 格式中。通过基于规则的 LLM 验证过滤幻觉/不一致的轨迹。(2) 主动阅读笔记本：PR/issue/commit 数据转换为结构化学习大纲（动机、根本原因、设计决策、见解），然后合成连贯的技术笔记。这些重写数据集 (~12B tokens) 在中训练阶段被整合，在 SWE-Bench 验证版上进一步提升了性能。

(4) 基于环境的种子数据。 我们使用附录 E.2.2 中描述的环境构建管道，从原始 PR、问题和提交记录中整理出可执行环境。候选样本经过严格过滤以确保测试补丁的包含，并通过严格的基于规则的准则进行验证以保证环境可复现性。此外，选中的问题会进行针对性重写以提升数据质量和覆盖范围。生成的数据集包含数十万种子样本，包括问题描述、代码变更和测试函数，作为增强 AGENT 编码能力的基础基石，显著提升了下游 AGENT 任务的性能。

C.3. 数学 & STEM 数据

为了增强推理能力并从知识中提取智能，我们精心策划了一个大规模的数学和STEM数据集。在先前工作中使用的标准Common Crawl数据之外 [176,177], ，我们利用自家的StepCrawl系统来采集大规模的额外数学相关数据。具体来说，我们实现了一个受MegaMath [177], 启发的过滤管道，利用内部分类器集合和FineMath [178]。这使我们能够在Common Crawl之外保留数百亿个数学相关Token。我们进一步收集了一个包含练习、测验和教学内容的1亿样本多样化教育数据集。这个收藏品弥合了学术理论与专业应用之间的差距，涵盖了从K-12数学/物理/化学和人文学科到成人职业考试（CPA、法律）的领域。早期实验证实，这种问题解决数据对于在预训练期间优化Token效率至关重要。

C.4. 数据基础设施

我们的数据构建和策展管道运行在一个高通量、内部数据基础设施系统上，该系统专为大规模去重、挖掘和模型推理过滤而设计。我们使用混合 CPU/GPU 集群，并使用 `Spark` 和 `Ray` 等分布式框架来执行大容量处理（例如基于 minhash 的去重）和模型驱动的策展工作负载（例如嵌入生成和分类器/LM 推理），并由一个存储层支持，该层跨越对象存储 (OSS)、 `HDFS` 和 `JuiceFS` ，用于高效读取/写入原始语料库和中间工件。

C.5. 数据消融设置

为了严格评估数据质量和数据整理策略的影响，我们使用经过Muon优化器优化的30B-A3B MoE架构进行了一系列消融实验，与主线设置一致（表9）。遵循严格的Token效率协议，我们为所有实验设置了固定的训练预算。模型在6.1节中列出的综合基准测试上进行了评估，同时还包括一系列精心设计的保留压缩（困惑度）测试集。我们观察到，压缩指标通常能提供更直接的衡量知识容量的方法，为主流基准测试提供互补信号。

对30B-A3B MoE模型的内部实验表明，其性能和稳定性优于更小的代理模型。虽然更小的模型计算成本更低，但它们往往无法捕捉复杂推理的细微之处，并且缺乏记忆长尾模式的能力，导致对数据重复产生人为的偏差。从经验上看，30B-A3B规模提供了更强的稳定性，并能更好地反映大规模趋势的保真度。

D. 训练后详情

本节描述了将基础模型优化为高性能智能系统的训练后过程，涵盖使用严格的数据处理和质量控制进行监督微调（SFT），随后通过大规模强化学习进一步提升推理能力、工具使用能力和泛化能力。

D.1. SFT 详情

D.1.1. SFT 数据处理流程

在所有领域，我们应用统一的数据处理流程，强调答案的可验证性、推理质量和执行真实性。为确保整体数据完整性，聚合数据集会经过严格的两个阶段过滤流程：

1. **基于规则的过滤：** 我们移除表现出退化模式的质量低劣数据，例如无限重复、有害内容和可识别个人身份信息。2. **基于模型的过滤：** 我们利用专用模型检测并过滤掉语言不一致的数据。通过识别并移除包含不自然语言混合的样本，我们显著提升了数据集的语言纯净度和整体质量。3. **去污染：** 我们进行全面的基准测试去污染，以防止测试集泄露。这涉及精确匹配（使用数字掩码以捕获数值修改）和 N-gram 匹配。

该过程生成一个最终精炼的数据集，包含871k个样本，总计7.23B个token。SFT数据的详细分布情况展示在表3中。

D.2. 强化学习细节与消融实验

本节详细介绍了大规模RL后训练，涵盖数据管理、异步搜索代理训练以及密集模型和MoE模型的消融实验。

D.2.1. 数据管理

我们通过聚合开源集合和竞赛存档中的问题来管理强化学习训练数据集，涵盖竞赛编程、STEM 和合成数据，用于通用强化学习虚拟现实训练。为防止数据污染，我们严格排除 2024–2026 年期间举办的竞赛中的问题。该数据集进一步通过以下方式增强：(i) 涉及 11–13 位整数的合成算术问题；(ii) 一个生成器-验证器管道，用于为编程任务合成额外的测试用例；(iii) 用于通用推理任务的合成环境，例如谜题和指令遵循。

我们应用两阶段过滤流程。首先，确定性基于规则的剪枝会移除包含图像、外部链接或开放式要求且没有唯一最终答案的提示。其次，基于准确性的过滤器会排除琐碎或退化的问题。在训练期间，每个批次都是根据预定义的采样概率从不同领域中进行采样的。

D.2.2. 奖励系统

可验证奖励。对于 STEM 任务，我们采用 gpt-oss-120b [33] 作为验证模型，使用以下结构化提示（最初为中文）严格评估最终答案的正确性。对于编程任务，我们利用沙盒来验证代码执行，并根据测试用例给予软性奖励。

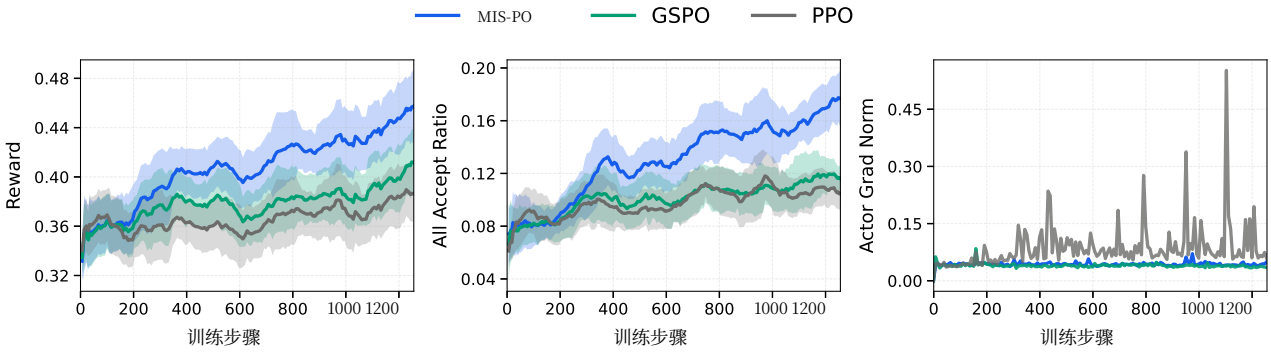
```
You are a strict grader. Below you are given the problem the student's answer and the ,,
reference answer. Please determine whether the student's answer is correct according to
the rules below:1. Overall check: If the student's response is incomplete, lacks a clear final answer,
or contains repeated content multiple times → mark as incorrect.
2. Final-answer match: Extract the student's explicit final answer and compare it with


the reference answer:

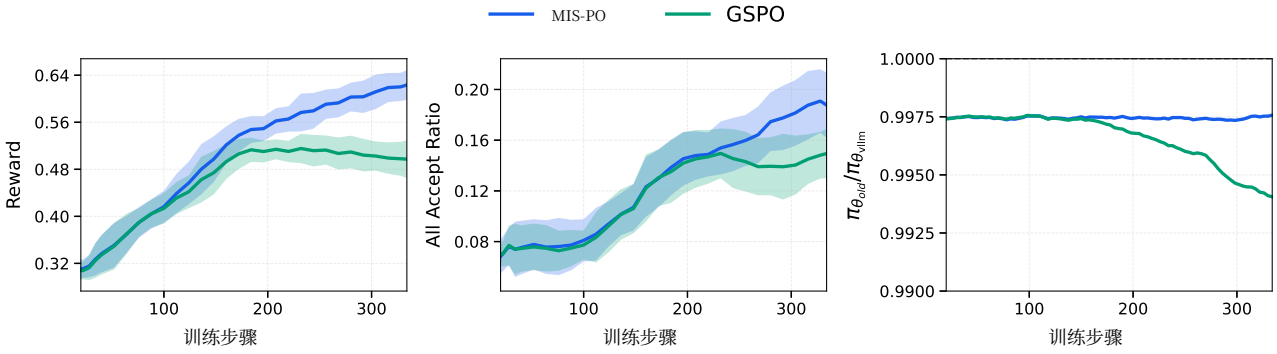
   • If they are exactly equivalent semantically or mathematically → proceed to process
     check.
   • If numerical computation is involved and the discrepancy is solely due to rounding
     → proceed to process check.
   • Otherwise → mark as incorrect.
3. Process check: Carefully verify each reasoning step:
   • If there are errors, contradictions, obvious irrelevance to the problem, or
   the student merely copies the prompt without a substantive solution → mark as incorrect.
   • If the solution process is correct, clear, and consistent → mark as correct.
4. Format requirements: If the problem requires a specific format (e.g., units,

step-by-step answers, or explicit equations) and the student does not satisfy it → mark
as incorrect.5. Multiple sub-questions: If the problem contains multiple sub-questions,
 the student must answer all of them correctly to be marked correct.
6. Other cases: If the above rules do not cover the situation, make an overall judgment
from the perspective of whether the student truly knows how to solve the problem.
Output requirement:Your final output must be strictly one of the following:

   • <correct> True </correct>
   • <correct> False </correct>
Now begin:
<question>
{question}
</question>
<student_answer>
{student_answer}
</student_answer>
<reference_answer>
{reference_answer}
</reference_answer>
```



(a) **在密集模型上的比较。** GSPO 虽然也能有效降低动作梯度范数的方差，但其效率不如 MIS-PO。在相同的迭代预算下，MIS-PO 实现了更高的奖励和所有接受率。



(b) **MoE模型对比。 (1) 效率：** MIS-PO展现出更优的样本效率，在加速收敛的同时获得更高的奖励，而GSPO在迭代200左右趋于平稳。 **(2) 稳定性：** GSPO在训练过程中表现出不断增加的训练-推理差异，该差异通过密度比$\pi_{\theta_{\text{old}}}/\pi_{\theta_{\text{vllm}}}$量化（其中 $\pi_{\theta_{\text{vllm}}}$ 是推理后端中的滚动策略，$\pi_{\theta_{\text{old}}}$ 是训练后端中的预更新策略快照）。相反，MIS-PO始终将此差异维持在稳定范围内。

图7：**MIS-PO与GSPO的性能对比。**顶部图形（a）展示了在密集模型上的结果，底部图形（b）展示了在MoE模型上的结果。MIS-PO在不同架构下始终在效率和稳定性方面优于GSPO。

*D.2.3. RL*消融细节

MIS-PO 与 GSPO。 为严格验证我们方法的有效性，我们将 MIS-PO 与 GSPO [36] 在密集架构和 MoE 架构上进行基准测试。我们选择 GSPO 作为主要基线，因为它代表了一种竞争性策略，用于减少重要性采样中固有的梯度方差。在我们的实现中，我们通过将其广义重要性采样机制集成到演员损失中，将原始 GSPO 估计器扩展到演员-评论家设置。具体来说，我们用轨迹级比率的几何平均值替换了标准的 token 级重要性采样比率。得到的演员损失公式如下 ($\gamma = \lambda = 1$)：

$$r_\tau(\theta) = \left(\prod_{t=0}^{T-1} \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \right)^{\frac{1}{T}} \tag{14}$$

$$\hat{A}_t = \hat{R} - V_\phi(s_t) \tag{15}$$

$$\mathcal{L}_{\text{actor}}^{\text{GSPO}} = -\mathbb{E}_{\tau \sim \pi_{\theta_{\text{vllm}}}} \left[\mathbb{I}(x_t) \cdot \mathbb{I}(\bar{\rho}(\tau)) \cdot \min(r_\tau(\theta)\hat{A}_t, \text{clip}(r_\tau(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right] \tag{16}$$

为确保公平比较，我们应用 MIS-PO 中使用的相同 token 级和样本级掩码策略，以排除具有显著训练-推理不匹配的数据。关于剪裁比率 ϵ，我们在 $\{1, 2, 3, 4\} \times 10^{-4}$ 上进行网格搜索。我们采用 $\epsilon = 10^{-4}$ 进行所有实验，因为它在 200 步 RL 训练后实现了最佳基准性能。此外，我们观察到该设置产生的剪裁分数约为 15%，与原始 GSPO [36] 一致。

图7展示了比较结果。经验上，MIS-PO比GSPO表现出更高的样本效率和可扩展性。关键的是，MIS-PO有效地将训练-推理不匹配约束在一个稳定范围内。这种稳定性对于MoE模型的规模化RL训练尤其关键，其中基线GSPO无法保持一致的收敛。

MoE的扩展训练动态。 为了进一步验证我们方法的可扩展性，我们使用一个具有挑战性的数据集，在MoE模型上对MIS-PO进行扩展训练运行。如图8所示，模型在奖励、演员梯度范数和熵水平方面均保持持续上升的稳定趋势。这些结果从经验上证实了MIS-PO对于大规模MoE离线策略RL训练的可靠性。

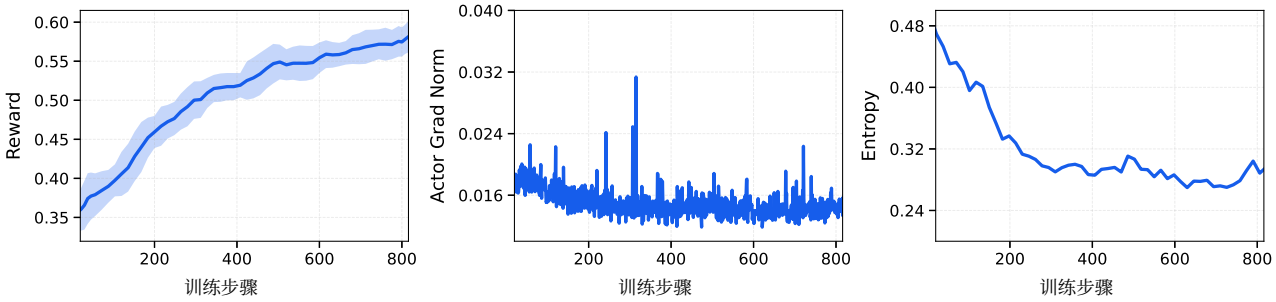


图8：**MIS-PO在MoE模型上的扩展训练动态。** 指标包括奖励（左侧）、演员梯度范数（中间）和熵（右侧）。值得注意的是，中间面板显示了未进行平滑或降采样的原始梯度范数，以突出优化的稳定性。

D.2.4. 搜索代理

关于训练架构，早期的客户端-服务器单步离策略框架受到长尾延迟的严重瓶颈：大约5%的样本占用了约80%的生成成本。然而，我们的观察表明该策略对陈旧性具有很强的鲁棒性，即使延迟达到约20步也能保持稳定性能。因此，我们采用FullyAsync范式，将生成和更新解耦为完全异步的过程。此外，为了在多轮交互中最小化推理开销，我们实现了粘性调度，将同一会话始终分配到同一节点以最大化KV缓存重用。总体而言，该配置实现了约 10× 的效率提升，同时保持了训练稳定性。

在整个训练过程中，FullyAsync范式展现了强大的稳定性，这体现在奖励的持续增长以及截断重要性采样（TIS）截断率始终保持在可控范围内，从而表明异步性引起的策略漂移有限。值得注意的是，与我们观察到的“从零开始”的强化学习在训练预算方面的有限可扩展性不同，在训练中期注入与任务相关的知识和工具使用先验，显著提升了性能并使RL过程中能力的涌现更加稳定。

模型	BrowseComp I		GAIA	xbench DeepSearch-2505	xbench DeepSearch-251(	
AGENT Δ平均@3（指标：通过率%）						
步骤3.5 Flash*	1.5 ▲50.1	25.0 ▲41.9	17.0 ▲67.5	26.0 ▲57.7	11.3 ▲42.7	**52.0**
Kimi K2-Thinking*	3.6 ▲37.9	23.8 ▲38.5	18.8 ▲36.6	28.7 ▲39.3	14.3 ▲27.0	35.9
Kimi K2.5*	7.4 ▲53.2	40.3 ▲22.0	26.7 ▲49.2	36.0 ▲40.3	19.7 ▲36.6	40.2
DeepSeek V3.2	8.1 ▲43.3	41.2 ▲23.8	23.4 ▲51.7	35.7 ▲41.3	18.7 ▲30.6	38.1
GLM-4.7	3.4 ▲48.6	30.2 ▲36.4	19.6 ▲26.5	29.7 ▲34.6	19.3 ▲23.4	33.9
MiniMax M2.1	1.3 ▲46.1	10.1 ▲37.7	15.4 ▲30.9	18.7 ▲46.6	6.0 ▲36.3	39.5
MiMo-V2 闪存	0.9 ▲44.5	12.9 ▲38.3	12.9 ▲42.3	19.7 ▲49.6	6.3 ▲13.7	37.7
Gemini 3.0 Pro	25.2 ▲12.6	-	32.1 ▲44.5	45.0 ▲32.0	-	29.7
Claude Sonnet 4.5	1.4 ▲22.7	21.2 ▲19.6	16.2 ▲54.7	24.7 ▲42.6	7.3 ▲37.7	35.5

表11：**工具使用对AGENT性能的影响**。每个单元格显示**基线分数**(仅内部知识)随后是启用搜索工具所实现的 ▲**性能提升**。最终分数是这两个值的总和。**平均提升**突出了模型利用外部信息改进结果的能力。带*标记的模型表示在256K设置下测量的工具结果；其他模型的设置未指明。

讨论。 为了从参数记忆中严格评估代理能力，我们关注工具使用提升，其定义为：

$$\Delta_{\text{tool}} = \text{Score}_{\text{with tools}} - \text{Score}_{\text{no tools}}$$

该指标将模型的固有知识与动态利用外部工具的能力解耦。如表11所示，**步骤3.5 Flash**展示了最强大的外部信息利用能力，实现了最高的平均增益（52.0），并在GAIA和xbench-DeepSearch等复杂基准测试中表现显著领先。

这种区分至关重要，因为在BrowseComp等基准测试上获得的高绝对分数有时可能源于强大的内部知识而非有效的搜索策略。高性能模型中较小的Δ工具可能模糊地表明模型要么效率很高（模型已经“知道”答案），要么未能有效利用工具来提升结果。相反，较大的 Δ工具明确表明模型在通过检索弥补知识差距方面的熟练度。因此，我们主张未来的优化不应仅仅追求更高的绝对分数（“基准磨炼”），而应旨在最大化在长上下文、证据关键场景中的 Δ工具。这确保了智能体真正掌握了过程信息检索与推理，而不是过度拟合静态知识或基准测试的伪影。

D.3. 工具集成推理与并行推理

在本节中，我们介绍了步骤3.5 Flash中用于测试时扩展的两个主要方法：工具集成推理和并行推理。

工具集成推理对于复杂的推理任务，我们将模型与Python解释器集成，以促进工具辅助推理。在此框架中，模型在沙盒内运行，迭代地思考并执行代码，用于计算、模拟和可视化。在我们的实验中，我们在AIME 2025、HMMT 2025、IMO-AnswerBench、GPQA、HLE_{text}和ARC-AGI-1上评估，限制为100轮。如表12所示，工具集成推理显著

提升在具有挑战性的数学、STEM 和谜题基准测试中的性能，突显了步骤3.5闪存的高级代理推理能力。

基准测试	步骤3.5闪存	步骤3.5闪存+Python
AIME 2025	97.3	**99.8** (+2.5)
HMMT 2025 二月	98.4	**98.7** (+0.3)
HMMT 2025 十一月	94.0	**98.0** (+4.0)
IMO-AnswerBench	85.4	**86.7** (+1.3)
GPQA-Diamond	83.5	**84.4** (+0.9)
HLE_{text}	23.1	**26.5** (+3.4)
ARC-AGI-1	54.8	**56.5** (+1.7)

表12：步骤3.5闪存与步骤3.5闪存+Python的比较。

工具集成并行推理我们初步探索了将PaCoRe扩展到多轮交互环境。按设计，PaCoRe保留了标准LLM消息接口。这种兼容性允许其无缝集成到使用多轮工具交互的现有代理框架中。为了将PaCoRe适配到此环境，我们实现了一个状态感知输入序列化协议，如表14所示。

我们在GPQA和HLE_{text} 基准测试上使用配备Python解释器的步骤3.5闪存评估了这种方法。如表13所示，将并行推理扩展到这些代理循环相对于标准推理基线显著提高了性能。这些发现表明PaCoRe有效地泛化到了需要交互反馈的环境，突出了代理测试时扩展的一个有前景的途径。

带Python的基准测试	步骤3.5闪存	步骤3.5闪存 +PaCoRe
GPQA-Diamond	84.4	**85.7** (+1.3)
HLE_{text}	26.5	**28.2** (+1.7)

表13：步骤3.5闪存+Python与使用PaCoRe测试时缩放的同款模型的对比

E. 详细评估协议和提示

本节提供了我们评估套件的实现细节。我们概述了具体的提示模板、少样本配置以及在不同基准测试中使用的判别模型。对于复杂指标，例如用于长上下文或推理任务的指标，我们还详细说明了其底层计算逻辑和评分标准，以确保可复现性。在提供的模板中，`{question}` 表示文本问题描述的占位符，而其他占位符（例如 `{test}`、`{context}`）表示特定任务的信息。

E.1. 预训练模型的评估细节

E.1.1. 通用语言理解和推理基准测试

BBH。 我们使用BBH [132], 的官方CoT-prompts [6]，仅将“Q:”和“A:”替换为“问题：”和“解决方案：”，如下所示：

[6]https://github.com/suzgunmirac/BIG-Bench-Hard/tree/main/Cot-prompts

Panel A：标准用户查询(最后角色： `user`)

您将获得一个问题和一份参考响应列表。您的任务是分析这些参考并给出自己的响应。

原始问题：
{{ original_content }}

参考响应： 注意：某些参考可能包含 <工具_调用> 标签，表明参考意图进行工具调用。这些工具调用尚未被执行——它们仅作为分析参考显示。{% for response in ref_响应 %} 参考 {{ loop.index }}： {{ response }}{% endfor %}

现在，根据上面的原始问题和参考响应，请提供您自己的综合解决方案。

Panel B：工具观察(最后角色： `tool`)

你将获得一个工具 响应以及一份分析该响应的参考响应列表。你的任务是分析这些参考，并提供你自己的响应。

原始工具响应：
{{ original_content }}

参考响应：
[与Panel A中相同的关于未执行工具调用的前缀]
{% for response in ref_responses %}
参考 {{ loop.index }}：
{{ response }}
{% endfor %}

现在，根据上述原始工具响应和参考响应，请提供您自己的综合分析和下一步行动。

表14：**工具集成PaCoRe的输入序列化模板**。我们引入了不同的模板来处理初始用户查询（A面板）和后续工具观察（B面板）。请注意，`tool_calls` 在参考分支中均序列化为文本以供分析。

```
Problem:
{question}

Solution:
```

MMLU。 我们使用 MMLU 的官方评估指标 [133] 并采用 5-shot。我们使用以下任务特定的系统提示：

```
The following are multiple choice questions (with answers) about {category}.
```

相应的问题提示结构如下：

```
{question}
Answer:
```

MMLU-Redux。 我们使用 MMLU-Redux 的官方评估指标 [134] 并采用 5-shot。并使用以下问题提示：

```
Answer the question and place the option (A/B/C/D...)  inside \boxed{}.
{question}
```

MMLU-Pro。 我们遵循MMLU-Pro的官方评估指标 [135] 进行5-shot评估。所有评估都使用以下系统提示：

```
The following are multiple choice questions (with answers) about {category}.
Think step by step and then output the answer in the format of "The answer is
(X)" at the end.
```

问题提示　　问题提示的结构如下，并在最后一个句号后故意留有一个空格：

```
Question:  {question}
Answer:  Let's think step by step.
```

值得注意的是，我们观察到原始MMLU-Pro数据集（12,102个问题中的470个）在真实选项前存在不一致的空格。我们明确移除这些空格，以减轻潜在的格式偏差并确保评估一致性。

HellaSwag. 我们使用HellaSwag的官方评估指标 [136] ，并采用10-shot。我们使用以下问题提示：

```
Question:  {question}
A. {option_0}
B. {option_1}
C. {option_2}
D. {option_3}
Answer:
```

WinoGrande. 我们使用WinoGrande的官方评估指标 [137]，并采用5-shot。问题提示的结构是为了清晰地呈现二进制选择：

```
Question:  {question}
Options:
A. {option_0}
B. {option_1}

Answer:
```

GPQA. 我们使用GPQA的官方评估指标 [138] ，并采用5-shot。问题提示的结构是为了清晰地呈现选择：

```
Question:  {question}
Options:
A. {option_0}
B. {option_1}
C. {option_2}
D. {option_3}

Answer:  Let's think step by step.
```

SuperGPQA。 我们使用SuperGPQA的官方评估指标 [139] ，并结合5-shot进行评估。问题提示遵循思维链（CoT）结构，其中每个少样本示例都包含逐步推导最终答案的过程：

```
Question:
{question}

Answer:  Let's think step by step.
```

SimpleQA. 我们使用 SimpleQA 的官方评估指标 [140] 并采用 5-shot。由于 SimpleQA 要求开放式简短答案，我们采用基于大语言模型的评估方法，具体使用 gpt-oss-120b [33] 作为评判模型。问题提示被格式化为一个简洁的查询：

```
Question:  {question} Answer:
```

E.1.2. 数学推理基准测试

GSM8K. 我们使用 GSM8K 的官方评估指标 [141] 并采用 8-shot。问题提示被设计为通过以下模板来引导思维链推理：

```
Q: {question}
A: Let's think step by step.
```

MATH. 我们使用 MATH 的官方评估指标 [142] 并采用 4-shot。问题提示被构建为包含明确问题和解决方案分隔符的结构：

```
Problem:
{question}


Solution:
```

E.1.3. 编程基准测试

HumanEval. 我们使用 HumanEval 的官方评估指标 [143] 并采用 3-shot。问题提示被构建为包含三个真实示例的结构，以提供代码生成的上下文指导：

```
# Below are the ground-truth solutions:

def add_two_numbers(a, b):""" Given two numbers a and b,
 return the sum of a and b. """# get the sum of a and b
sum of a and b = a + b_  _ _   _return sum of a and b
_ _ _   _def reverse_list(some_list: list) -> list:
""" Given a list, return a reversed copy of the list. """
```

```
    new_list = []# iterate over the listfor item in some list:_
# insert item into new listnew_list.insert(0, item)return new list_
def fast_reverse_list(some_list: list) -> list:""" Given a list,
 return a reversed copy of the list. Be fast! """
# use faster built-in reversesome_list.reverse()return some list_
{question}
```

MBPP。 我们遵循MBPP [144] 的官方评估指标，采用3-shot方式。

HumanEval+。 我们遵循HumanEval+ [145] 的官方评估指标，采用3-shot方式。 .

MBPP+。 我们使用MBPP+ [145] 的官方评估指标，采用零样本方式。我们采用结构化指令提示，明确任务要求，并包含一个示例测试用例以进行对齐：

```
    You are an expert Python programmer,
 and here is your task:{question}
Your code should pass the test:{test}
Here is the corresponding code:``` python
```

MultiPL-E。 我们使用MultiPL-E [146] 的官方评估指标，采用零样本方式。我们遵循官方测试用例来评判生成的代码。

E.1.4. 中文理解基准测试

C-Eval. 我们使用 C-Eval [147] 的官方评估指标，并添加了一个 5-shot 设置。我们采用了以下系统提示：

```
你是一个中文 文人工智能助手，以下是中国关于{category}考试的单项选择题，请选出其中的正确答案。
```

对应的问题提示结构如下：

```
{question}
答案:
```

CMMLU。 我们使用CMMLU的官方评估指标 [148]，并添加了5-shot设置。我们采用以下系统提示：

```
你是一个中文人工智能助手,以下是中国关于{category}考试的单项选择题,请选出其中的正确答案。
```

相应的 问题提示 结构如下:

```
{question}
答案:
```

C-SimpleQA. 我们使用中文 SimpleQA [149] 的官方 评估指标 和 基于大语言模型的 判断协议。我们添加了 5-shot 设置，并使用 gpt-oss-120b [33] 作为判断模型。我们采用以下 问题提示:

```
问题:{question} 答案:
```

E.2. 后训练模型的评估细节

在本节中，我们详细介绍了用于评估后训练模型在多种代理任务上的评估协议。我们的评估涵盖了代码中心和通用代理设置，包括软件工程、终端交互、深度搜索、研究工作流和现实世界工具使用。我们在严格控制的环境和推理预算下报告标准化指标，以确保跨基准测试的公平、稳定比较。

E.2.1. 推理基准测试

CF-Div2-Stepfun. 我们使用一个自定义的 CodeForces Div. 2 基准测试来评估我们模型的竞争编程能力。该基准测试包含 53 个问题，这些问题来自 2024 年 9 月至 2025 年 2 月期间官方举办的 CodeForces Division 2 比赛。我们开发了一个离线评估框架，该框架使用本地评分机制作为实时在线提交的替代方案。我们尝试构建与原始测试用例相似的测试用例。具体来说，我们首先生成足够多的中小规模测试用例以评估正确性覆盖率，然后添加随机数据以进行大规模测试。最后，通过分析常见错误模式和实际用户的“黑客”提交，我们进行了对抗性边缘用例的构建。一些边缘用例也通过压力测试技术自动生成，该技术不断生成无数测试用例，直到能够区分失败的提交和正确的提交。为了验证此基准测试的可靠性，我们运行了从原始比赛中选出的正确和具有代表性的失败提交。我们的评估器正确地将 100% 的接受提交识别为“通过”，同时 92.45% 的失败提交被准确标记。

模型	平均@8准确率 C++	Python	Java	Codeforces C++ 通过@8评级
Step 3.5 Flash	**86.1%**	**81.5%**	77.1%	**2489**
DeepSeek V3.2	81.6%	66.5%	80.7%	2319
GLM-4.7	74.1%	63.0%	70.5%	2156
Kimi K2-Thinking	67.9%	60.4%	58.5%	1976
Minimax-M2.1	59.0%	46.4%	58.0%	1869
MiMo-V2 闪存	46.9%	43.6%	39.6%	1658
Gemini 3.0 Pro	83.5%	74.1%	**81.6%**	2397
Claude Opus 4.5	72.2%	68.4%	68.9%	2100

表15：CF-Div2-Stepfun中变量模型的完整评估结果。

我们为每个问题采样8个响应，并报告平均准确率。此过程使用的用户提示是：

```
    You are a coding expert.  Given a competition-level coding problem, you need to
write a {LANGUAGE} program to solve it.  You may start by outlining your thought
process.  In the end, please provide the complete code in a code block enclosed
with ``` ```.{question}
```

C++、Python、Java的编译和执行命令如下：

```
g++ -std=c++20 -fno-asm -fsanitize=bounds -fno-sanitize-recover=bounds -static
-O2 -DONLINE_JUDGE -o code.exe code.cpp
./code.exe
```

```
python3 code.py
```

```
javac -J-Xmx544m {JAVA_CLASS_NAME}.java
java -XX:+UseSerialGC -Xmx544m -Xss64m -DONLINE_JUDGE {JAVA_CLASS_NAME}
```

为与竞赛编程规范保持一致，并避免JIT"预热"期间的不一致开销，我们使用标准Python解释器并设置双倍时间限制，而不是PyPy[7]。我们将相同的双倍时间限制应用于所有Java提交。

虽然表15报告了原始准确率，但我们认识到问题难度差异很大。因此，评分分数提供了更稳健的指标。尽管像CodeELO [179]这样的框架可以计算竞争性评分，但目前顶尖模型在2级竞赛中表现如此出色，以至于它们的评分可能导致统计异常值。此外，我们采用了一种简化的评分计算方法，通过假设所有解决方案在竞赛开始时就提交来忽略时间惩罚。虽然这种方法偏离了实证竞争场景，并且可能导致与人类参与者无法直接比较的评分，但它为一致的跨模型比较提供了一个标准基准。

[7]https://pypy.org/

LiveCodeBench-v6。 我们使用LiveCodeBench [12]的官方评估方法。我们采用以下系统提示:

```
You are an expert Python programmer.  You will be given a question (problem
specification) and will generate a correct Python program that matches the
specification and passes all tests.
```

相应的 问题提示 结构如下:

```
### Question:
{question}
### Format:  You will use the following starter code to write the solution to
the problem and enclose your code within delimiters.
```` python
{starter_code}
````
Answer: (use the provided format with backticks)
```

AIME 2025. 我们使用 AIME 2025 [166] 的官方评估方法，并采用 repeat@64。我们使用以下问题提示:

```
Answer the question and place the answer inside \boxed {} with MathTeX format.
{question}
```

HMMT 2025 二月/十一月。 我们使用 HMMT 2025 [11] 的官方评估方法，并采用 repeat@64。我们使用以下问题提示:

```
Answer the question and place the answer inside \boxed {} with MathTeX format.
{question}
```

IMO-AnswerBench. 我们使用 IMO-AnswerBench [91] 的官方评估方法，并采用 repeat@64。我们使用以下问题提示:

```
Answer the question and place the answer inside \boxed {} with MathTeX format.
{question}
```

MMLU-Pro. 我们使用 MMLU-Pro [135] 的官方评估方法。数据集的处理与我们的预训练 MMLU-Pro 评估方法保持一致（详情请参见附录 E.1.1）。

```
Answer the question and place the option (A/B/C/D...)  inside \boxed{}.

{question}
```

GPQA-Diamond。 我们使用 GPQA-Diamond 的官方评估方法 [138]。我们采用以下问题提示：

```
Answer the question and place the option (A/B/C/D...)  inside \boxed{}.

{question}
```

HLE_{text}. 我们使用 HLE 的官方评估指标和基于大语言模型的判断协议。我们使用 gpt-oss-120b [33] 作为判断模型。

E.2.2. 代码代理基准测试

SWE-Bench. SWE-Bench 验证版 [13]是原始 SWE-bench 数据集的高质量子集，包含 500 个经过人类专家开发者严格验证的软件工程任务，以确保可靠和准确的评估。SWE-Bench 多语言版将原始基准扩展到 9 种编程语言的 300 个真实世界软件工程任务。

我们使用内部代理基础设施测试了 Step 3.5 Flash 的软件工程代理能力，该基础设施基于所述会话路由器架构构建，并在 SWE-Bench 验证版和 SWE-Bench 多语言版上进行。对于每个评估实例，我们通过 Kubernetes 提供一个容器化会话。然后执行针对 SWE-Bench 的环境初始化，这包括移除未来的提交以防止数据泄露，以及配置网络代理和关键系统设置。关于代理骨架，我们采用了 OpenHands [127] CodeAct 代理框架，该框架在研究界被广泛使用。我们启用了默认的四套工具：execute_bash、str_replace_editor、finish 和 think。最大交互回合数设置为 350。

由于编译型语言资源消耗较大，我们在多语言环境下分配了12GB内存，而验证实例则限制在4GB以内。在评估中，工具执行超时时间设置为1200秒，模型推理参数为：温度=1，top-p=0.95。根据以上设置，Step 3.5 Flash在SWE-Bench验证版上达到74.4%，在SWE-Bench多语言版基准测试上达到67.4%，平均分数为4次运行的平均值。我们还对Step 3.5 Flash在其他流行代理框架上进行了交叉评估：SWE-Agent [128]使用原始代理管道设置在SWE-Bench验证版上达到74.2%的准确率，标准Claude代码 [8]环境在每次实例扩展4小时时间限制且单工具执行无时间限制的情况下得分72.0%。

终端基准测试2.0。 我们在任务无关的远程容器中测试终端基准测试 [16] 。我们将容器内存限制在16GB。我们部署了一个内部Artifactory仓库，并更新了所有Docker容器的默认包源。在会话创建和测试阶段的依赖安装过程中，如果发生错误，系统将多次重试。为简化通信，我们修改了Terminus 2框架，使其自动中断超时命令，并阻止同一轮中的后续命令执行，向代理返回超时警告。相应地，我们修改了原始系统提示中的命令时长控制部分：

```
Keystroke duration sets the command hard timeout.  The system automatically
interrupts timed-out commands and prevents subsequent commands in the same
```

[8]https://github.com/anthropics/claude-code

```
round from executing.  You can simply continue with your next round - no special
action is required.
```

在推理过程中，我们将模型的单轮输出限制为64k，并将最大上下文窗口限制为256k，适用于所有交互。思考过程将保留在多轮历史记录中。如果模型输出超过256k上下文窗口限制，我们将执行剪枝上下文管理：保留问题陈述和最后50%的历史记录，然后重试。我们使用top-p=0.95和temperature=1的推理参数。交互协议主要使用XML格式化的结构化响应进行。AGENT被限制为200轮交互，一旦达到此限制，将直接进入测试阶段。交互和测试的总时间限制为6小时。

为确保一致性，我们验证并优化每个任务的检查器，以问题陈述为准，这使整体准确率提高了约1.5%。每个任务都在8次试验中执行。值得注意的是，88.6%的成功轨迹在30次交互内完成。最终pass@8为67/89，avg@8为50.98%。我们的AGENT在89个任务中的23个任务中，在所有8次试验中都实现了100%的成功率。在成功轨迹中，9.41%的运行触发了历史剪枝以管理上下文限制。

设置	最大输出	最大轮次	超时上下		Avg@8
基线	64k	200	6h	✓	**50.98%**
限制16k	16k	200	6h	✓	48.03%
限制16k（无剪枝）	16k	200	6h	×	45.22%
轮次100	64k	100	6h	✓	50.42%
超时 2 小时	64k	200	2h	✓	49.72%

表 16：Terminal-Bench 2.0 推理约束的消融研究。

消融研究表明，限制 16k 会导致最大的性能下降，因为模型在处理复杂任务时的长推理过程常常在输出终端命令之前就耗尽了 token 限制。在 16k 限制下禁用上下文管理时，性能进一步下降至 45.22%。同时，轮次 100 的影响最小，因为大多数任务会提前完成。超时 2 小时的下降反映了某些涉及模型训练、重度编译或复杂环境配置的任务需要更多时间才能完成。

E.2.3. 通用 *AGENT* 基准测试

深度搜索。 我们在多个基准测试（例如，**BrowseComp [17], BrowseComp-ZH [18], GAIA [19], xbench-DeepSearch [20]**）上评估了我们智能体的深度搜索能力。表5中报告的结果基于avg@3指标；GPT-5.2 xHigh使用avg@1。该智能体配备了包括以下核心工具集：

- **搜索**：并行执行多个搜索查询。
- **访问**：分析网页内容，基于大语言模型回答特定问题。
- **谷歌_学术**：搜索学术论文和技术文献。
- **Python_解释器**：运行Python代码进行计算和数据分析。
- **文件**：从直接URL下载并保存文件。

在推理过程中，我们采用256k令牌的上下文窗口，生成长度无限制。推理使用top-p = 0.95、温度 = 1.0和存在惩罚 = 1.1，允许高达400步的执行预算。

agent和大语言模型裁判的详细系统提示，与GitHub仓库中与 [125]关联的配置一致。

带上下文管理的浏览组件。表5中报告的69.0分是带上下文管理的浏览组件（带上下文管理的浏览组件）在完整浏览组件数据集上使用丢弃全部方法评估的结果。这种方法与DeepSeek V3.2 [1], 相同，当上下文长度超过预定义阈值时触发，此时代理丢弃其整个上下文并重新初始化操作循环。在最大迭代约束为1000步的情况下，该策略为浏览组件设置72k个token的上下文长度阈值，为浏览组件-ZH设置41k个token的上下文长度阈值。

我们还对来自浏览组件的200个实例子集评估了各种上下文管理策略，包括摘要、保留首尾K、丢弃全部和多代理协调。如表17所示，我们的模型在这些不同的范式中展现出强大的适应性。在单代理策略中，丢弃全部的准确率达到了66.0%。我们认为丢弃全部是一种测试时pass@k 策略，迫使模型从头重新推理，直到找到自我验证路径。性能遵循清晰的层级关系：多代理通过利用主代理分解任务并派遣专业代理进行并行推理排名第一，其次是丢弃全部、保留首尾K 和摘要——这与实际步数的增加密切相关。这种关联反映了推理成本（步数）和准确率之间的直接权衡，表明密集的上下文管理能够有效地将增加的计算量转化为更优的性能。

方法	准确率 (%)	实际步数
步骤3.5 Flash	49.5	86
+ 摘要	57.0	131
+ 保留首尾K	58.0	244
+ 全部丢弃	66.0	302
+ 多智能体	68.5	721

表17：上下文管理方法评估结果。

R研究问题集R标准。为评估深度研究能力，我们采用研究问题集 [21]基准测试。该数据集包含101个领域多样化的研究任务，每个任务配有20-43条专家编写的细粒度评分标准，用于评估事实准确性、推理合理性和清晰度。我们将性能与两大代表性系统系列进行基准测试：商业智能体系统和ReAct智能体。

对于商业智能体，我们通过其官方网页界面（数据捕获于2025年12月2-15日）在默认配置下收集报告。如表18所示，领先的商业系统（Gemini DeepResearch）获得63.69的总分。

对于ReAct智能体，详细的性能对比见表5。我们的模型获得**65.3**的分数，超越了复杂的专有商业基线。值得注意的是，在标准化的ReAct框架内评估Gemini 3.0 Pro时，我们观察到50.1的分数。我们将这一性能差距归因于在处理开放式研究问题时搜索深度不足；模型倾向于依赖内部参数化知识，而非执行广泛的外部检索。因此，生成的报告缺乏全面性，未能充分覆盖用户的隐性标准。

我们为 ReAct 智能体标准化了执行环境，最多支持 30 轮推理，且每轮输出限制为 16k 个 token。对于推理参数，其他基于 API 的模型使用其默认设置，而我们的模型则配置了温度为 1 和 top-p{v1}.95。所有输出随后将由大语言模型裁判根据每个标准进行三档评分。为支持端到端的研究工作流程，我们的 ReAct 框架提供了以下工具集的访问权限：

智能体系统	分数
Gemini DeepResearch	63.69
OpenAI DeepResearch	60.67
Kimi Researcher	53.67
MiniMax Agent Pro	51.85
Qwen DeepResearch	49.24

表18： 商业代理系统在R E S E A R C HR U B R I C S基准测试中的性能 k.

所有输出随后将由大语言模型裁判根据每个标准进行三档评分。为支持端到端的研究工作流程，我们的 ReAct 框架提供了以下工具集的访问权限：

- **批量_网络_冲浪者**：用于并发网络搜索和多页面浏览。
- **文件**：用于稳健的文件操作，包括读取、写入和迭代编辑。
- **文件_解析器**：用于将文件转换为Markdown格式。
- **shell**: 用于交互式命令执行和环境交互。
- **待办**: 用于动态任务状态管理和跟踪。
- **Tmux**: 用于模拟具有持久会话和滚动历史的复用终端环境。

τ^2-基准测试. τ^2-基准测试 [15] 是一个评估通用工具使用能力的代理式基准测试，涵盖三个客户服务领域：航空、零售和电信。我们使用原始代码库中的官方设置评估 Step 3.5 Flash。具体来说，我们使用默认的大语言模型代理框架，并将温度设置为1.0，top-p设置为0.95，最大序列长度设置为256K。用户模型设置为GPT-4.1，温度为0.0，以确保评估过程中交互的稳定性。对于航空领域，由于它存在不正确的真实答案，我们使用来自Claude Opus 4.5的固定版本以确保评估的可靠性 [9]。对于零售和电信领域，我们也遵循Claude Opus 4.5，在用户提示中包含通用提示补充，以避免用户因错误结束交互而导致失败模式 [10]。我们报告8次运行的平均分数，以确保评估结果的稳定性。

E.2.4. 通用基准测试

Arena-Hard-v2.0. 我们使用 Arena-Hard-v2.0 [151] 的官方评估指标，并使用 GPT- 4.1 [180] 作为评判模型。

MultiChallenge. 我们使用 MultiChallenge [153] 的官方评估指标，并使用 o3-mini [181] 作为评判模型。这遵循了 GPT-5 [182] 发布的发现，即 GPT-4o [180] 经常对复杂响应评分错误，导致结果被低估。

IFBench。 我们使用 IFBench 的官方评估方法 [152]。

[9] https://github.com/sierra-research/tau2-bench/pulls/chrisgorgo [10] https://github.com/anthropics/model-cards/tree/main/claude-opus-4-5-20251101/tau2

E.2.5. 长上下文基准测试

LongBench v2。 我们使用 LongBench v2 [154] 的官方评估方法。

MRCR-8needle. 对于 MRCR-8needle [155] 基准测试，我们报告了曲线下面积（AUC）指标，遵循 ContextArena [11]建立的协议。具体来说，我们使用 **AUC@128k**指标，该指标提供了一个单一的 holistic 分数，总结了对长达 131,072 个 token 的上下文长度的性能表现。

AUC 是通过绘制每个上下文 bin（范围从 8k 到 128k）的平均检索准确率与其最大上下文长度，来计算的。我们在线性尺度上应用梯形规则来测量所得曲线下的面积，然后通过总上下文宽度（128k 减去初始 bin 大小）进行归一化，得到一个 0% 到 100% 之间的百分比分数。这个指标有效地惩罚了随着上下文序列长度增加而性能下降的情况。

FRAMES-Oracle. 我们使用 FRAMES [156]的官方评估指标。由于我们的重点是长上下文能力，我们特别报告了 **Oracle 提示**子集的结果。在这种设置下，模型会获得问题以及所有在人工标注过程中使用的 ground-truth 维基百科文章。这种配置作为模型性能的上限，模拟了一个完美的检索系统，将所有相关上下文都提供给模型。

RepoQA. 我们使用REPOQA的官方评估方法 [157]。

E.3. 内部评估 - 基准测试与方法论

E.3.1. 数据分析基准测试

为了可靠地评估步骤3.5 Flash在Claude代码环境中执行实际数据分析任务的能力，我们开发了一个内部数据分析基准测试，用于评估在现实商业约束下端到端的分析问题解决能力。该基准测试通过系统地提炼高级从业者的隐性专业知识，构建了一个基于评分标准的评估套件。这种方法捕捉了现实世界分析的模糊性和上下文细微差别，同时通过标准化的评分标准和可验证的真实结果工件确保一致的评估。

该基准测试采用专家驱动、评分标准为基础的协议构建，以确保领域真实性和评分可靠性。来自中国主要互联网公司的十位高级数据分析领导者，每位都具有超过15年的经验，通过结构化访谈贡献了现实世界的商业案例，这些案例引出了核心分析模式和决策逻辑。这个过程产生了具有代表性的任务以及专家认可解决方案策略。

面试材料被标准化为机器可消费的任务，每个任务包含一个问题描述、一个CSV数据集、一个参考分析和一个加权清单式评分标准。生成的基准测试包含50个项目，涵盖多样的分析意图，每个任务平均包含26.9个评分标准项。通过迭代专家评审确保质量，对任务定义、数据、参考解决方案和评估标准进行对齐，以提高有效性和可重复性。

我们进一步实现了一个统一的端到端评估框架，涵盖任务执行、自动评分和报告合成。该框架支持基于代码、面向研究和基于文本的分析，可在单一管道内完成，能够在异构环境中以低集成开销实现可扩展和可复现的评估。

[11]https://contextarena.ai/

分析，能够在异构环境中以低集成开销实现可扩展和可复现的评估。

评估方法。 每个任务由基于模型的评估器进行评估，该评估器将生成的输出与专家定义的评分标准进行比对，并在3次相同的运行中取平均值以减少随机波动，确保可靠且可比的跨模型评估。

模型	Avg@3(%)
Claude Opus 4.5	45.0
步骤3.5 Flash	39.6
GPT-5.2	39.3
Gemini 3.0 Pro	33.6
DeepSeek V3.2	27.9

表19：数据分析基准测试评估结果

评估结果。 表19展示了在数据分析基准测试上的结果。Claude Opus 4.5总体排名第一，而步骤3.5 Flash获得第二名的强劲表现（39.58%），并且非常接近GPT-5.2（39.31%）。其竞争性能可能部分与在Claude Code环境中相对良好的适应性有关。此外，步骤3.5 Flash展示了有利的速度-能力权衡，在保持稳固的分析质量的同时提供更快的响应。这些结果将步骤3.5 Flash定位为真实世界数据分析任务中高效且具有竞争力的选项。

E.3.2. 咨询与建议基准测试

为了在真实世界的咨询场景中严格评估步骤3.5 Flash，我们精选了一个包含500个多样化查询的基准，这些查询来自Reddit、Stack Exchange和各种社区论坛等真实社交平台。这些查询代表了日常生活、学术学习、娱乐和专业工作环境中的真实用户意图。

这里，我们实现了一个“基于锚点的”评分框架来评估候选模型。在此过程中，我们首先利用领先模型，包括 GPT-5.2、Claude Opus 4.5 和 DeepSeek V3.2 来为每个查询生成独立响应。这些高级输出随后由人类专家综合和提炼，以创建一个参考响应作为基准。该参考作为高质量的“锚点”，具有 88/100 的标准化性能值。

我们随后在四个关键维度上衡量模型的性能，采用严格的评分标准，包括实用性、逻辑性、指令遵循和语气。实用性评估模型是否提供可直接使用的解决方案，以专家级深度、可执行步骤和可行建议有意义地解决任务。逻辑性评估事实准确性和结构合理性，检查是否存在幻觉、不正确的引用、无效的结论或因果关系和时序不一致，以及整体连贯性和论证流程。指令遵循衡量对显式约束（例如、格式、长度和陈述要求）和嵌入在用户查询中的隐式上下文期望的遵守程度。语气评估沟通质量，包括语言和语域的适当性、解释复杂推理的清晰度，以及校准的表达方式，避免过度自信，同时在适当的时候清晰地标示不确定性。

我们采用混合LLM作为裁判的系统。认识到不同的前沿模型具有不同的

评估优势，我们分配了具体的评分职责如下：逻辑性、指令遵循和实用性：这三个维度由GPT-5.2评估，利用其在事实验证、约束检查和客观问题解决方面的行业领先能力。语气：这个维度由Claude Opus 4.5评估，利用其在语言风格、情感校准和“类人”共鸣方面的卓越能力。裁判的可靠性通过与人专家的对齐研究得到验证，AI和人类分配的分数之间具有高皮尔逊相关性。最终分数使用四个维度（各25%）的等权重计算得出，确保了一个平衡的评估，它共同反映了技术正确性和沟通质量。

模型	平均	实用性	逻辑性	Tone	指令遵循
GPT-5.2	77.8%	77.2%	81.9%	73.0%	79.6%
Kimi K2.5	72.2%	77.1%	62.1%	72.7%	77.3%
Gemini 3.0 Pro	70.6%	73.9%	61.7%	72.3%	74.4%
步骤3.5 Flash	70.5%	73.3%	62.1%	72.4%	74.2%
DeepSeek V3.2	70.3%	72.5%	64.4%	71.2%	72.9%
GLM-4.7	70.3%	73.5%	61.5%	72.5%	73.6%
Claude Opus 4.5	68.5%	69.7%	66.5%	65.9%	72.1%
MiMo-V2 闪存	67.9%	71.5%	58.0%	70.6%	71.4%
MiniMax M2.1	67.1%	70.7%	60.1%	67.2%	70.4%

表20：在咨询与建议基准测试上的评估结果

评估结果。 表20显示，Step 3.5 Flash在咨询与推荐基准测试中平均得分为70.5%，位列第四。Step 3.5 Flash在所有维度上与Gemini 3.0 Pro表现相当，实现了可比的Pro级分数（70.5% vs. 70.6%），同时提供了显著更低的推理成本和延迟。与许多为了速度而牺牲推理质量的快速模型不同，Step 3.5 Flash在逻辑维度上超越了更大模型，减少了幻觉和逻辑错误，使其非常适合需要高度事实准确性的自动化咨询工作流。

E.3.3. 步骤3.5 Flash + Step-GUI

为验证Step 3.5 Flash在真实世界智能体场景中的有效性，我们在**AndroidDailyHard** [183], 上进行了评估。这是一个为中文移动应用环境设计的具有挑战性的基准测试。该基准测试包含涵盖电子商务交易、多媒体交互和日常移动操作的组合任务，为评估GUI智能体在复杂、多步骤工作流中的能力提供了自然测试平台，这些工作流代表了生产环境中的实际部署。

我们通过实证研究两种架构实现方式：(1) **Step-GUI** [183], 轻量级设备端代理（仅边缘），该代理使用本地计算资源自主执行任务，以及 (2) **Step3.5 Flash + Step-GUI**，这是一个边缘-云协同框架，其中 Step 3.5 Flash 作为云端推理协调器，合成高级任务计划，通过 GUI-MCP 协议将它们分解为可执行的原语，并将低级控制委托给设备端的 Step-GUI 代理。这种分层架构利用了云端推理的规模优势和边缘效率的互补性：Step 3.5 Flash 的 11B 活跃参数支持复杂的多步规划和上下文理解，而 Step-GUI 则确保低延迟动作执行和隐私保护本地控制。

定量结果。 边缘-云协同范式在 AndroidDaily Hard 上实现了 **57.0%** 的成功率，显著优于仅边缘的基线（**40.0%**）。这一结果表明，结合强大的云端推理和高效的边缘执行是一种有效的策略，用于应对多轮代理交互中部署约束的挑战。

通用泛化。 关键地，这种协作模式超越了移动生态系统，扩展到包括台式计算机和汽车信息娱乐系统在内的异构平台。通过将认知编排（云）与具身执行（边缘）解耦，该框架建立了一种可扩展的范式，用于在资源受限的工业环境中部署复杂的agent——这与Step 3.5 Flash的设计目标直接一致，即重新定义生产级agentic系统的效率前沿。结果表明，有效的现实世界agent不仅需要先进的推理能力，还需要能够在基础设施层级之间协调计算分布的架构。

参考文献

[1] DeepSeek-AI. Deepseek-v3.2-exp: 通过深度探索稀疏注意力提升长上下文效率，2025.[2] 高安增，吕欣，郑钦凯，侯振宇，陈斌，谢成星，王存祥，尹达，曾浩，张嘉杰，等. Glm-4.5：具有代理、推理和编码（ARC）能力的基座模型. *arXiv* 预印本 *arXiv:2508.06471*, 2025.[3] LLM-Core 小米. Mimo-v2-flash 技术报告，2026.[4] 美团龙猫团队，北李，雷冰叶，王波，容波林，王超，张超，高陈，张陈，孙成，等. 龙猫-flash 技术报告. *arXiv* 预印本 *arXiv:2509.01322*, 2025.[5] Kimi 团队，白一帆，鲍一平，陈观多，陈嘉豪，陈宁馨，陈瑞杰，陈延儒，陈元坤，陈宇天，等. Kimi k2：开放代理智能. *arXiv* 预印本 *arXiv:2507.20534*, 2025.[6] MiniMax 团队. Minimax-m2.1，2025.

[7] OpenAI。GPT-5.2，2025。

[8] Google DeepMind. Gemini 3 pro模型卡，2025.[9] Anthropic. 系统卡：Claude opus 4.5，2025.[10] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, 和 François Fleuret. Transformers are rnns: 快速自回归Transformer与线性注意力机制. 在 机器学习国际会议论文集, ICML’20. JMLR.org, 2020.[11] HMMT. Hmmt 2025 二月，2025.[12] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, 和 Ion Stoica. Livecodebench: 对代码大型语言模型进行全面且无污染的评估. *arXiv*预印本 *arXiv:2403.07974*, 2024.[13] OpenAI. 推出SWE-bench验证版 我们发布了一个经过人工验证的SWE-bench子集，2024.[14] John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, 和 Diyi Yang. Swe-smith: 为软件工程AGENT扩展数据规模，2025.

[15] Sierra Research. tau2-bench. https://github.com/sierra-research/tau2-bench, 2025.[16] Mike A Merrill, Alexander G Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, I van Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E Kelly Buchanan, et al. Terminal-bench: 基准测试在命令行界面中针对困难、现实任务的代理。 *arXivpreprint arXiv:2601.11868*, 2026.[17] Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. Browsecomp: 一个简单但有挑战性的基准测试，用于浏览代理，2025.[18] Peilin Zhou, Bruce Leon, Xiang Ying, Can Zhang, Yifan Shao, Qichen Ye, Dading Chong, Zhiling Jin, Chenxuan Xie, Meng Cao, Yuxin Gu, Sixin Hong, Jing Ren, Jian Chen, Chao Liu, and Yining Hua. Browsecomp-zh: 基准测试中文大语言模型的网络浏览能力，2025.[19] Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: 一个通用人工智能助手基准测试，2023.[20] Kaiyuan Chen, Yixin Ren, Yang Liu, Xiaobo Hu, Haotong Tian, Tianbao Xie, Fangfu Liu, Haoye Zhang, Hongzhang Liu, Yuan Gong, et al. xbench: 使用与职业对齐的真实世界评估跟踪代理的生产力扩展。 *arXiv preprint arXiv:2506.13651*, 2025.[21] Manasi Sharma, Chen Bo Calvin Zhang, et al. Researchrubrics: 一个用于评估深度研究代理的提示和基准测试。 *arXiv preprint arXiv:2511.07685*, 2025.[22] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: 使用简单高效的稀疏性扩展到万亿参数模型。 *arXiv preprint arXiv:2101.03961*, 2021.[23] Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: 设计稳定和可迁移的稀疏专家模型，2022.[24] Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten P Bosma, Zongwei Zhou, Tao Wang, Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. GLaM: 使用专家混合高效扩展语言模型。在 Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, 和 Sivan Sabato 编辑的 机器学习国际会议论文集，第162卷的 机器学习研究论文集，第5547–5569页。 PMLR, 17–23 Jul 2022.[25] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: 使用条件计算和自动分片扩展巨型模型，2020.[26] Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. Deepseekmoe: 朝向混合专家语言模型中的终极专家专业化，2024.[27] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 使用稀疏转换器生成长序列。 *CoRR*, abs/1904.10509, 2019.[28] Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. 通过多标记预测改进和加快大型语言模型。 *arXivpreprint arXiv:2404.19737*, 2024.[29] DeepSeek-AI. Deepseek-v3 技术报告。 *arXiv preprint arXiv:2412.19437*, 2024.[30] LLM Xiaomi, Bingquan Xia, Bowen Shen, Dawei Zhu, Di Zhang, Gang Wang, Hailin Zhang, Huaqiu Liu, Jiebao Xiao, Jinhao Dong, et al. Mimo: 解锁语言模型的推理潜力——从预训练到后训练。 *arXiv preprint arXiv:2505.07608*, 2025.

[31] 邱子涵, 王泽坤, 郑博, 黄泽宇, 温凯悦, 杨松林, 严睿, 余乐, 黄飞, 黄寿之, 刘代恒, 周景润, 和 Junyang Lin. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free, 2025.[32] StepFun 团队. Step3: Cost-effective multimodal i ntelligence.[33] OpenAI. Gpt-oss-120b & gpt-oss-20b model card, 2025.[34] Keller Jordan, Jin Yuchen, Boza Vlado, You Jiacheng, Cesista Franz, Newhouse Laker, 和 Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024.[35] Wei Jason, Nguyen Karina, Chung Hyung Won, Jiao Yunxin Joy, Papay Spencer, Glaese Amelia, Schulman John, 和 Fedus William. Measuring short-form factuality in large language models.*arXiv preprint arXiv:2411.04368*, 2024.[36] Zheng Chujie, Liu Shixuan, Li Mingze, Chen Xiong-Hui, Yu Bowen, Gao Chang, Dang Kai, Liu Yuqiong, 严睿, Yang An, 等. Group sequence policy optimization. *arXivpreprintarXiv:2507.18071*, 2025.[37] Yao Feng, Liu Liyuan, Zhang Dinghuai, Dong Chengyu, Shang Jingbo, 和 Gao Jianfeng. Your efficient rl framework secretly brings you off-policy rl training, August 2025.[38] Ma Wenhan, Zhang Hailin, Zhao Liang, Song Yifan, Wang Yudong, Sui Zhifang, 和 Luo Fuli. Stabilizing moe reinforcement learning by aligning training and inference routers. *arXivpreprintarXiv:2510.11370*, 2025.[39] Metropolis Nicholas, Rosenbluth Arianna W, Rosenbluth Marshall N, Teller Augusta H, 和 Teller Edward. Equation of state calculations by fast computing machines. *Thejournal of chemicalphysics*, 21(6):1087–1092, 1953.[40] Hastings W Keith. Monte carlo sampling methods using markov chains and their applications. 1970.[41] Luong Thang, Hwang Dawsen, Nguyen Hoang H., Ghiasi Golnaz, Chervonyi Yuri, Seo Insuk, Kim Junsu, Bingham Garrett, Lee Jonathan, Mishra Swaroop, Zhai Alex, Huiyi Hu Clara, Michalewski Henryk, Kim Jimin, Ahn Jeonghyun, Bae Junhwi, Song Xingyou, Trinh Trieu H., Le Quoc V., 和 Jung Junehyuk. Towards robust mathematical reasoning, 2025.[42] Hu Jingcheng, Zhang Yinmin, Shang Shijie, Yang Xiaobo, Peng Yue, Huang Zhewei, Zhou Hebin, Wu Xin, Cheng Jie, Wan Fanqi, Kong Xiangwen, Yao Chengyuan, Yan Kaiwen, Huang Ailin, Zhou Hongyu, Han Qi, Ge Zheng, Jiang Daxin, Zhang Xiangyu, 和 Shum Heung-Yeung. Pacore: Learning to scale test-time compute with parallel coordinated reasoning, 2026.[43] Building effective agents. `https://www.anthropic.com/engineering/building-effective-agents`.[44] Unrolling the codex agent loop. `https://openai.com/index/unrolling-the-codex-agent-loop/`.[45] Snell Charlie Victor, Lee Jaehoon, Xu Kelvin, 和 Kumar Aviral. Scaling LLM test-time com-pute optimally can be more effective than scaling parameters for reasoning. In *The ThirteenthInternational Conference on Learning Representations*, 2025.[46] Muennighoff Niklas, Yang Zitong, Shi Weijia, Li Xiang Lisa, Fei-Fei Li Li, Hajishirzi Hannaneh, Zettlemoyer Luke, Liang Percy, Candès Emmanuel, 和 Hashimoto Tatsunori B. s1: Simple test-time scaling. In *Proceedings ofthe 2025 Conference on Empirical Methods in Natural LanguageProcessing*, pages 20286–20332, 2025.[47] Yang Xinyu, An Yuwei, Liu Hongyi, Chen Tianqi, 和 Chen Beidi. Multiverse: Your language models secretly decide how to parallelize and merge generation. In *The Thirty-ninthAnnualConference on Neural Information Processing Systems*, 2025.

[48] Iz Beltagy, Matthew E. Peters, 和 Cohan Arman. Longformer: The long-document transformer.*arXiv:2004.05150*, 2020.[49] Gemma 团队, Kamath Aishwarya, Ferret Johan, 等. Gemma 3 technical report, 2025.[50] Leviathan Yaniv, Kalman Matan, 和 Matias Yossi. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conferenceon Machine Learning*, ICML'23. JMLR.org, 2023.[51] Schlag Imanol, Irie Kazuki, 和 Schmidhuber Jürgen. Linear transformers are secretly fast weight programmers. In *International conference onmachinelearning*, pages 9355–9366. PMLR, 2021.[52] Wang Jikai, Su Yi, Li Juntao, Xia Qingrong, Ye Zi, Duan Xinyu, Wang Zhefeng, 和 Zhang Min. Opt-tree: Speculative decoding with adaptive draft tree structure. *Transactions of the Associationfor Computational Linguistics*, 13:188–199, 2025.[53] Xiong Yunfan, Zhang Ruoyu, Li Yanzeng, 和 Zou Lei. Dyspec: Faster speculative decoding with dynamic token tree structure. *World WideWeb*, 28(3):36, 2025.[54] You Haoran, Fu Yichao, Wang Zheng, Yazdanbakhsh Amir, 和 Lin Yingyan (Celine). When linear attention meets autoregressive decoding: towards more effective and efficient linearized large language models. In *Proceedings ofthe 41st International Conference onMachine Learning*, ICML'24. JMLR.org, 2024.[55] Ainslie Joshua, Lee-Thorp James, de Jong Michiel, Zemlyanskiy Yury, Lebron Federico, 和 Sanghai Sumit. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In Bouamor Houda, Pino Juan, 和 Bali Kalika, editors, *Proceedingsof the 2023Conference onEmpirical Methodsin NaturalLanguage Processing*, pages 4895–4901, Singapore, December 2023. Association for Computational Linguistics.[56] Li Yuhui, Wei Fangyun, Zhang Chao, 和 Zhang Hongyang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *International Conference on Machine Learning*, 2024.[57] Gemma 团队, Mesnard Thomas, Hardin Cassidy, Dadashi Robert, Bhupatiraju Surya, Pathak Shreya, Sifre Laurent, Rivière Morgane, Kale Mihir Sanjay, Love Juliette, 等. Gemma: Open models based on gemini research and technology, 2024.[58] Team Cohere, :, Aakanksha, Ahmadian Arash, Ahmed Marwan, 等. Command a: An enterprise-ready large language model, 2025.[59] Xiao Guangxuan, Tian Yuandong, Chen Beidi, Han Song, 和 Lewis Mike. Efficient streaming language models with attention sinks. In *The TwelfthInternational Conference onLearningRepresentations*, 2024.[60] Sun Mingjie, Chen Xinlei, Kolter J Zico, 和 Liu Zhuang. Massive activations in large language models. In *First Conference onLanguage Modeling*, 2024.[61] Gu Xiangming, Pang Tianyu, Du Chao, Liu Qian, Zhang Fengzhuo, Du Cunxiao, Wang Ye, 和 Lin Min. When attention sink emerges in language models: An empirical view. In *TheThirteenthInternational Conference on Learning Representations*, 2025.[62] Jumper John, Evans Richard, Pritzel Alexander, Green Tim, Figurnov Michael, Ronneberger Olaf, Tunyasuvunakool Kathryn, Bates Russ, Žídek Augustin, Potapenko Anna, Bridgland Alex, Meyer Clemens, Kohl Simon A. A., Ballard Andrew J., Cowie Andrew, Romera- Paredes Bernardino, Nikolov Stanislav, Jain Rishub, Adler Jonas, Back Trevor, Petersen Stig, Reiman David, Clancy Ellen, Zielinski Michal, Steinegger Martin, Pacholska Michalina, Berghammer Tamas, Bodenstein Sebastian, Silver David, Vinyals Oriol, Senior Andrew W., Kavukcuoglu Koray, Kohli Pushmeet, 和 Hassabis Demis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, August 2021.

[63] 林智轩、Evgenii Nikishin、何旭和Aaron Courville。遗忘Transformer：带有遗忘门的Softmax注意力机制。在第 *13* 届学习表示国际会议，2025年.[64] 王 Lean、高华祖、赵成刚、孙旭和戴大梅。无辅助损失的专家混合负载均衡策略。*arXiv*预印本 *arXiv:2408.15664*，2024年.[65] 蔡雨轩、梁晓专、王兴华、马金、梁海津、罗金文、左新宇、段立升、尹宇阳和陈西。Fastmtp：通过增强多标记预测加速LLM推理，2025年.[66] Ansel Jason、杨 Edward、何 Horace、Gimelshein Natalia、Jain Animesh、Voznesensky Michael、包斌、Bell Peter、Berard David、Burovski Evgeni、Chauhan Geeta、Chourdia Anjali、Constable Will、Desmaison Alban、DeVito Zachary、Ellison Elias、Feng Will、Gong Jiong、Gschwind Michael、Hirsh Brian、Huang Sherlock、Kalambarkar Kshiteej、Kirsch Laurent、Lazos Michael、Lezcano Mario、梁岩波、梁Jason、Lu Yinghai、Luk CK、Maher Bert、Pan Yunjie、Puhrsch Christian、Reso Matthias、Saroufim Mark、Siraichi Marcos Yukio、Suk Helen、Suo Michael、Tillet Phil、Wang Eikan、王晓东、Wen William、张俊、赵旭、Zhou Keren、Zou Richard、Mathews Ajit、Chanan Gregory、Wu Peng和Chintala Soumith。PyTorch 2：通过动态Python字节码转换和图编译实现更快的机器学习。在第 *29* 届 *ACM*国际会议编程语言与操作系统架构支持，第 *2* 卷（*ASPLOS'24*）。ACM，2024年4月.[67] Shoeybi Mohammad、Patwary Mostofa、Puri Raul、LeGresley Patrick、Casper Jared和Catanzaro Bryan。Megatron-lm：使用模型并行性训练百亿参数语言模型。*arXiv*预印本 *arXiv:1909.08053*，2019年.[68] Narayanan Deepak、Shoeybi Mohammad、Casper Jared、LeGresley Patrick、Patwary Mostofa、Korthikanti Vijay Anand、Vainbrand Dmitri、Kashinkunti Prethvi、Bernauer Julie、Catan-zaro Bryan、Phanishayee Amar和Zaharia Matei。使用Megatron-lm在GPU集群上高效训练大规模语言模型，2021年.[69] Rajbhandari Samyam、Rasley Jeff、Ruwase Olatunji和He Yuxiong。Zero：面向训练万亿参数模型的内存优化。在*SC20*：高性能计算、网络、存储和分析国际会议，第 1-16 页，2020年.[70] 刘 Dennis、Yan Zijie、Yao Xin、Liu Tong、Korthikanti Vijay、Wu Evan、Fan Shiqing、Deng Gao、Bai Hongxiao、Chang Jianbin、Aithal Ashwath、Andersch Michael、Shoeybi Mohammad、Yao Jiajie、Zhou Chandler、Wu David、Li Xipeng和Yang June。MoE并行折叠：用于Megatron核心高效训练大规模MoE模型的异构并行映射，2025年。

[71] 郭文滔、Mayank Mishra、程新乐、Ion Stoica 和 Tri Dao。Sonicmoe：通过io和tile-aware优化加速MoE，2025。

[72] Jeremy Bernstein 和 Laker Newhouse。老优化器，新范式：一部文集，2024。

[73] Noah Amsel、David Persson、Christopher Musco 和 Robert M. Gower。极地快车：最优矩阵符号方法及其在Muon算法中的应用，2025。

[74] 邱志涵、黄泽宇、郑博、温凯悦、王泽坤、门睿、Ivan Titov、刘迪恒、周景润 和 林俊阳。细节中的魔鬼：关于实现负载均衡损失以训练专业专家混合模型的探讨。*arXiv*预印本 *arXiv:2501.11873*，2025。

[75] 安阳，安凤丽，杨保松，张北辰，惠宾元，郑波，余博文，高畅，黄成恩，吕晨旭，郑楚杰，刘大恒，周帆，黄飞，胡峰，葛浩，魏浩然，林欢，唐嘉龙，杨健，涂建红，张建伟，杨建新，杨嘉翔，周静，周景仁，林俊阳，党凯，包克勤，杨克欣，刘乐宇，邓亮浩，李梅，薛明峰，李明泽，张沛，王鹏，朱秦，门瑞，高瑞泽，刘诗轩，罗双，李天浩，唐天毅，尹文标，任兴章，

Xinyu Wang，Xinyu Zhang，Xuancheng Ren，Yang Fan，Yang Su，Yichang Zhang，Yinger Zhang，Yu Wan，Yuqiong Liu，Zekun Wang，Zeyu Cui，Zhenru Zhang，Zhipeng Zhou，以及 Zihan Qiu。Qwen3 技术报告。*arXiv* 预印本 *arXiv:2505.09388*，2025.[76] Alec Radford，Jeff Wu，Rewon Child，David Luan，Dario Amodei，以及 Ilya Sutskever。语言模型是无监督多任务学习器。2019.[77] Ruibin Xiong，Yunchang Yang，Di He，Kai Zheng，Shuxin Zheng，Chen Xing，Huishuai Zhang，Yanyan Lan，Liwei Wang，以及 Tieyan Liu。关于 Transformer 架构中的层归一化。在机器学习国际会议，第 10524–10533 页。PMLR，2020。

[78] Noam Shazeer. Glu变体提升Transformer，2020。

[79] Common Crawl. Common crawl. `https://commoncrawl.org`.[80] Siming Huang, Tianhao Cheng, J. K. Liu, Jiaran Hao, Liuyihan Song, Yang Xu, J. Yang, Jiaheng Liu, Chenchen Zhang, Linzheng Chai, Ruifeng Yuan, Zhaoxiang Zhang, Jie Fu, Qian Liu, Ge Zhang, Zili Wang, Yuan Qi, Yinghui Xu, and Wei Chu. Opencoder: The open cookbook for top-tier code 大语言模型, 2025.[81] Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can 大语言模型 resolve real-world GitHub issues? In第 *12* 届学习表示国际会议, 2024.[82] Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Agentless: Demystifying 基于大语言模型的 软件工程 agents. *arXiv preprint arXiv:2407.01489*, 2024.[83] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2024.[84] Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, et al. Effective long-context scaling of foundation models, 2024.[85] Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. Deepseek-v3. 2: Pushing the frontier of open 大语言模型. *arXiv preprintarXiv:2512.02556*, 2025.[86] Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning, 2025.[87] Fabio Pardo, Arash Tavakoli, Vitaly Levdik, and Petar Kormushev. Time limits in reinforcement learning, 2022.[88] Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl.
`https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2`, 2025. Notion Blog.[89] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source 大语言模型 reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.[90] Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-Reasoner-Zero: An open source approach to scaling up reinforcement learning on the Base model. *arXiv preprint arXiv:2503.24290*, 2025.

[91] Minh-Thang Luong, Dawsen Hwang, Hoang H Nguyen, Golnaz Ghiasi, Yuri Chervonyi, Insuk Seo, Junsu Kim, Garrett Bingham, Jonathan Lee, Swaroop Mishra, et al. 向鲁棒数学推理迈进。在 *2025 年自然语言处理经验方法会议论文集*，第 35406–35430 页，2025.
[92] François Chollet. 智能的度量。 *arXiv* 预印本 *arXiv:1911.01547*, 2019.[93] Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, et al. 人类最后的考试，2025.[94] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, 和 Daya Guo. Deepseekmath：推动开放语言模型中数学推理的极限，2024.[95] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, 和 Ryan Lowe. 训练语言模型以人类反馈遵循指令，2022.[96] Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, 和 Rishabh Agarwal. 生成式验证器：奖励建模作为下一词预测，2025.[97] Ralph Allan Bradley 和 Milton E Terry. 不完整块设计的秩分析：I. 配对比较法。 *Biometrika*, 39(3/4):324–345, 1952.[98] Chen Hu, Haikuo Du, Heng Wang, Lin Lin, Mingrui Chen, Peng Liu, Ruihang Miao, Tianchi Yue, Wang You, Wei Ji, Wei Yuan, Wenjin Deng, Xiaojian Yuan, Xiaoyun Zhang, Xiangyu Liu, Xikai Liu, Yanming Xu, Yicheng Cao, Yifei Zhang, Yongyao Wang, Yubo Shu, Yurong Zhang, Yuxiang Zhang, Zheng Gong, Zhichao Chang, Binyan Li, Dan Ma, Furong Jia, Hongyuan Wang, Jiayu Liu, Jing Bai, Junlan Liu, Manjiao Liu, Na Wang, Qiuping Wu, Qinxin Du, Shiwei Li, Wen Sun, Yifeng Gong, Yonglin Chen, Yuling Zhao, Yuxuan Lin, Ziqi Ren, Zixuan Wang, Aihu Zhang, Brian Li, Buyun Ma, Kang An, Li Xie, Mingliang Li, Pan Li, Shidong Yang, Xi Chen, Xiaojia Liu, Yuchu Luo, Yuan Song, YuanHao Ding, Yuanwei Liang, Zexi Li, Zhaoning Zhang, Zixin Zhang, Binxing Jiao, Daxin Jiang, Jiansheng Chen, Jing Li, Xiangyu Zhang, 和 Yibo Zhu. Step-deepresearch 技术报告，2025.[99] Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath：ai4maths 中最大的公共数据集，包含 860k 对竞赛数学问题和答案。 2024.[100] Alon Albalak 等人。Big-math：用于语言模型强化学习的大规模、高质量数学数据集。 *arXiv* 预印本 *arXiv:2502.17387*, 2025.[101] Arindam Mitra, Hamed Khanpour, Corby Rosset, 和 Ahmed Awadallah. Orca-math：释放 slms 在小学数学中的潜力。 *arXiv* 预印本 *arXiv:2402.14830*, 2024.[102]aslawliet. 奥林匹克。
`https://huggingface.co/datasets/aslawliet/olympiads`, 2024. Hugging Face 数据集.[103] aslawliet. Cn-k12. `https://huggingface.co/datasets/aslawliet/cn-k12`, 2024. 中文 K-12 数学问题 Hugging Face 数据集.[104] Open-R1 Team. Openr1-math-220k. `https://huggingface.co/datasets/open-r1/OpenR1-Math-220k`, 2025. 开源蒸馏数学推理数据集.[105] X. He 等人。Deepmath-103k：大规模、具有挑战性的数学问答基准。 *arXiv* 预印本 *arXiv:2504.11456*, 2025.

[106] 余奇英，张铮，朱若飞，袁宇峰，左晓晨，岳宇，范天甜，刘高宏，刘凌君，刘欣，林海斌，林志奇，马博，盛光明，童宇轩，张驰，张莫凡，张王，朱杭，朱金华，陈嘉泽，陈江杰，王成毅，余红莉，戴文南，宋宇轩，魏向鹏，周浩，刘晶晶，马伟英，张亚琴，严琳，乔慕，吴永辉，王明轩。DAPO：一个大规模开源大语言模型强化学习系统，2025.[107]Guha Etash 等人。Openthoughts：推理模型的数據配方。*arXiv preprintarXiv:2506.04178*, 2025.[108] Muennighoff Niklas 等人。s1：简单的测试时扩展。 `https://arxiv.org/abs/2501.19393`, 2025.[109] 季云杰，田晓宇，赵思彤，王浩天，陈帅婷，彭一波，赵汉，李向刚。Am-thinking-v1：在32b规模上推进推理前沿，2025.[110] LIMO 作者们。推理少即是多：半参数数学推理器。*arXivpreprintarXiv:2502.03387*, 2025.[111] 李荣高，付杰，张博文，黄涛，孙志宏，吕晨，刘广，金智，李格。Taco：算法代码生成数据集中的主题，2023.[112] 罗迈克尔，谭思军，黄睿，帕特尔阿米恩，阿里亚克阿尔佩，吴清阳，石晓祥，辛Rachel，蔡 Colin，Weber Maurice，张Ce，李Erran，Popa Raluca Ada，Stoica Ion。Deepcoder：一个完全开源的14b编码器，达到o3-mini水平。 `https://www.together.ai/blog/deepcoder`, 2025。技术博客.[113] 王志涵，刘思瑶，孙阳，李红岩，沈凯。Codecontests+：为竞赛编程生成高质量测试用例，2025.[114] 李国豪，Hasan Abed Al Kader Hammoud，Hani Itani，Dmitrii Khizbullin，Bernard Ghanem。Camel：用于“心灵”探索大规模语言模型社会的交流代理，2023.[115] Bercovich Akhiad，Levy Itay，Golan Izik，Dabbah Mohammad，El-Yaniv Ran，Puny Omri，Galil Ido，Moshe Zach，Ronen Tomer，Nabwani Najeeb，等人。Llama-nemotron：高效推理模型，2025.[116] 范润泽，王增智，刘鹏飞。Megascience：推动科学推理后训练数据集的前沿。*arXiv preprint arXiv:2507.16812*, 2025.[117] 赵文婷，任翔，Hessel Jack，Cardie Claire，Choi Yejin，邓云天。Wildchat：野外1m ChatGPT交互日志。*arXiv preprint arXiv:2405.01470*, 2024.[118] 刘俊腾，李云集，张驰，李景阳，陈爱丽，纪科，程伟宇，吴子嘉，杜成宇，徐奇迪，等人。Webexplorer：探索和进化以训练长时程网络代理。*arXivpreprintarXiv:2509.06501*, 2025.[119] 李宽，张中旺，尹会峰，张立文，欧立土，吴嘉龙，尹文标，李白轩，陶正伟，王新宇，等人。Websailor：导航超人类推理以训练网络代理。*arXiv preprint arXiv:2507.02592*, 2025.[120] 李月台，Huseyin A Inan，岳祥，陈伟宁，Wutschitz Lukas，Kulka-rni Janardhan，Poovendran Radha，Sim Robert，Rajmohan Saravan。使用推理模型模拟环境以进行代理训练。*arXiv preprintarXiv:2511.01824*, 2025.[121] 郭亮红，王艳林，李蔡花，陶伟，杨鹏宇，陈嘉驰，宋浩宇，唐杜宇，郑子斌。Swe-factory：您的问题解决训练数据和评估基准的自动化工厂，2026.

[122] 张嘉然, 马幸运, 李言浩, 万帆奇, 齐迪, 赵旭, 侯捷怡, 谢哲, 任梦强, 吴欣, 黄哲伟, 陈良宇, 马英伟, 韩琪, 张祥宇. Dock-smith: 通过一个智能化的 Docker 构建器扩展可靠的编码环境, 2026.

[123] 王小智, 高天宇, 朱赵成, 张正岩, 刘志远, 李娟子, 唐建. Kepler: 一个用于知识嵌入和预训练语言表示的统一模型. 计算语言学协会汇刊, 9:176–194, 2021.

[124] 郭大亚, 杨德健, 张浩伟, 宋俊晓, 张若雨, 许润欣, 朱启豪, 马世荣, 王沛怡, 毕晓, 等. Deepseek-r1 通过强化学习激励大语言模型的推理能力. 自然, 645(8081):633–638, 2025年9月.

[125] 胡晨, 杜海阔, 王恒, 林琳, 陈明睿, 刘鹏, 姜瑞航, 越天池, 王旺, 季伟, 袁伟元, 邓文进, 袁晓坚, 张晓云, 刘祥宇, 刘锡凯, 许岩明, 曹一成, 张一飞, 王永耀, 舒宇博, 张宇荣, 张宇翔, 郭正, 常志超, 李宾言, 马丹 , 贾福荣 , 王红元 , 刘佳宇 , 白静 , 刘俊兰 , 刘曼娇 , 王娜 , 吴秋平 , 杜庆新 , 李石伟 , 孙文 , 龚一峰, 陈永林, 赵雨玲, 林宇轩, 任子祺, 王子轩, 张爱华, 李斌, 马不云, 安康, 谢立, 李明良, 李攀, 杨时东, 陈熙, 刘晓嘉, 罗宇初, 宋远, 丁元昊, 梁元伟, 李泽熙, 张昭宁, 张自新, 焦彬星, 蒋大欣, 陈建生, 李静, 张祥宇, 朱一博, 等. Step-deepresearch 技术报告, 2025.

[126] 安阳，安凤丽，杨宝松，张北辰，会宾元，郑博，余 Bowen，高畅，黄成恩，吕晨旭，郑楚杰，刘大恒，周帆，黄飞，胡峰，葛浩，魏浩然，林欢，唐嘉龙，杨建，涂建红，张建伟，杨建新，杨嘉熙，周静，周景仁，林俊阳，党凯，包克勤，杨克欣，刘莱宇，邓亮浩，李梅，薛明峰，李明泽，张沛，王鹏，朱琴，门瑞，高瑞泽，刘诗璇，罗双，李天浩，唐天毅，尹文标，任兴章，王新宇，张新宇，任宣程，杨帆，杨苏，张一畅，张莺，王泽坤，崔泽宇，张振儒，周志鹏，邱志涵。Qwen3技术报告，2025。

[127] 王行尧, 李宝轩, 宋宇帆, 徐方方, 唐祥儒, 诸葛明晨, 潘佳怡, 宋越琪, 李博文, Jaskirat Singh 等. OpenHands: 一个面向 AI 软件开发者的通用型 AGENT 开放平台. *arXiv* 预印本 *arXiv:2407.16741*, 2024.

[128] 杨峻, 卡洛斯·E·希门内斯, 亚历山大·韦蒂希, 基利安·利雷, 姚顺宇, 卡尔蒂克·纳拉辛汉, 和 Ofir Press. SWE-agent: AGENT-计算机接口实现自动化软件工程.神经信息处理系统进展, 37:50528–50652, 2024.

[129] 在c. Kilo Code. 以千倍速度前进. `https://kilo.ai/`, 2026. Kilo Code 官网 .

[130] Roo Code. 您的 AI 软件工程团队在此. `https://roocode.com/`, 2026. Roo Code 官网.

[131] ANTHROPIC PBC。自动补全完成行。claude代码完成功能。 `https://claude.com/product/claude-code`, 2026年。Claude代码网页。

[132] Mirac Suzgun、Nathan Scales、Nathanael Schärli、Sebastian Gehrmann、Yi Tay、Hyung Won Chung、Aakanksha Chowdhery、QuocV. Le、Ed H. Chi、Denny Zhou和Jason Wei。挑战big-bench任务，以及思维链是否能够解决它们，2022年。

[133] Dan Hendrycks、Collin Burns、Steven Basart、Andy Zou、Mantas Mazeika、Dawn Song和Jacob Steinhardt。测量大规模多任务语言理解，2020年。

[134] Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, Claire Barale, Robert McHardy, Joshua Harris, Jean Kaddour, Emile van Krieken, 以及 Pasquale Minervini. 我们完成 mmlu 呢？ 2024。

[135] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, 以及 Wenhu Chen. Mmlu-pro：一个更稳健和更具挑战性的多任务语言理解基准测试，2024.[136] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, 以及 Yejin Choi. Hellaswag：机器真的能帮你完成句子吗？2019.[137] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, 以及 Yejin Choi. Winogrande：大规模的对抗性 Winograd 方案挑战，2019.[138] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, 以及 Samuel R. Bowman. Gpqa：一个研究生级别的谷歌防伪问答基准测试，2023.[139] Xinrun Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, King Zhu, Minghao Liu, Yiming Liang, Xiaolong Jin, Zhenlin Wei, 等人. Supergpqa：跨 285 个研究生学科扩展大语言模型评估，2025.[140] OpenAI. Simpleqa. `https://github.com/openai/simple-evals`, 2024.[141] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, 以及 John Schulman. 训练验证者解决数学应用题，2021.[142] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, 以及 Jacob Steinhardt. 使用数学数据集衡量数学问题解决能力，2021。

[143] 马克·陈，杰里·特沃克，希woo·俊，钱铭源，亨利克·蓬德·德·奥利维拉·皮into，杰拉德·卡普兰，哈里·爱德华兹，尤里·伯达，尼古拉斯·约瑟夫，格雷格·布罗克曼，亚历克斯·雷，劳尔·普里，格雷琴·克鲁格，迈克尔·佩特罗夫，海迪·克拉夫，吉里什·萨斯特里，帕梅拉·米什金，布鲁克·陈，斯科特·格雷，尼克·莱德，米哈伊尔·帕夫洛夫，阿莱莎·鲍威尔，卢卡斯·凯泽，穆罕默德·巴瓦尼安，克莱门斯·维纳，菲利普·蒂勒，费利佩·佩特罗斯基·苏奇，戴夫·坎明斯，马蒂亚斯·普拉珀特，福蒂奥斯·钱齐斯，伊丽莎白·巴恩斯，阿丽尔·赫伯特-沃斯，威廉·希本·古斯，亚历克斯·尼科尔，亚历克斯·佩恩奥，尼古拉斯·泰扎克，蒋杰，伊戈尔·巴布什金，苏奇·巴拉吉，沙坦努·贾因，威廉·萨瑟兰，克里斯托弗·赫塞，安德鲁·N·卡尔，扬·莱克，乔什·阿希亚姆，维丹特·米斯拉，埃文·莫里卡瓦，亚历克·拉德福德，马修·奈特，迈尔斯·布兰达奇，米拉·穆拉蒂，凯蒂·梅耶，彼得·韦林德，鲍勃·麦克格鲁，达里奥·阿莫迪，山姆·麦克坎迪什，伊利亚·苏茨凯弗，以及沃伊切赫·扎雷姆巴。评估在代码上训练的大型语言模型，2021年。

[144] 雅各布·奥斯汀、奥古斯都·奥德纳、马克斯韦尔·奈、马arten·博斯马、亨利克·米哈莱夫斯基、大卫·多汉、艾伦·蒋、蔡瑞、迈克尔·泰勒、Quoc·Le和查尔斯·萨顿。使用大型语言模型的程序合成，2021。[145] 刘佳伟、夏春秋·史蒂文·夏、王宇瑶和章凌明。你的代码真的是由ChatGPT生成的吗？对大型语言模型进行代码生成的严格评估，2023。[146] 费德里科·卡萨诺、约翰·古瓦、丹尼尔·阮、悉尼·阮、露娜·菲普斯-科斯蒂恩、唐纳德·平克尼、叶明浩、赵阳天、卡罗琳·简·安德森、莫莉·Q·费尔德曼、阿琼·古哈、迈克尔·格林伯格和阿比纳夫·贾达。MultiPL-E：一种可扩展和可扩展的神经代码基准测试方法，2022。[147] 黄宇珍、白宇卓、朱志浩、张俊磊、张景涵、苏唐军、刘俊腾、吕传成、张奕凯、雷佳怡、傅瑶、孙茂松和何军仙。C-EVAL：为基础模型创建的多级多学科中文评估套件，2023。[148] 李浩南、张亦轩、Koto·Fajri、杨亦飞、赵海、龚叶云、段南和Timothy·鲍尔。CMMLU：测量中文大规模多任务语言理解，2023。

[149] 何言成、李世隆、刘嘉恒、谭颖水、王伟勋、黄辉、布行远、郭杭宇、胡成伟、郑博然等。中文SimpleQA：大型语言模型的中文事实性评估，2024。[150] 潘龙、Pang·CY·托尼、Wecker·亚当、熊一帆、Hendrycks·丹等。人类最后的考试，2025。[151] 李天乐、蒋伟琳、Frick·埃文、Dunlap·丽莎、朱邦华、Gonzalez·约瑟夫·E.和Stoica·Ion。从实时数据到高质量基准：Arena-Hard流程，2024年4月。[152] Pyatkin·瓦伦蒂娜、Malik·萨乌米亚、Graf·维多利亚、Ivison·哈米什、Huang·Shengyi、Dasigi·Pradeep、Lambert·内森和Hajishirzi·Hannaneh。通用可验证指令遵循。*arXiv*预印本*arXiv:2507.02833*，2025。[153] Sirdeshmukh·Ved、Deshpande·Kaustubh、Mols·Johannes、Jin·Lifeng、Cardona·Ed-Yeremai和Lee·迪恩、Kritz·杰里米、Primack·Willow、Yue·Summer和Xing·陈。Multichallenge：一个对前沿LLM具有挑战性的多轮对话评估基准，2025。[154] 白宇舒、涂尚清、张嘉杰、彭浩、王晓知、Lv·欣、曹舒林、徐嘉政、侯雷、董宇晓、唐洁和李娟子。Longbench v2：走向对现实长上下文多任务更深入的理解和推理，2024。[155] Vodrahalli·Kiran、Ontanon·Santiago、Tripuraneni·Nilesh、Xu·Kelvin、Jain·Sanil、Shivanna·Rakesh、Hui·Jeffrey、Dikkala·Nishanth、Kazemi·Mehran、Fatemi·Bahare等。Michelangelo：通过潜在结构查询超越Haystacks的长上下文评估。*arXiv*预印本*arXiv:2409.12640*，2024。[156] Krishna·Satyapriya、Krishna·Kalpesh、Mohananey·Anhad、Schwarcz·Steven、Stambler·Adam、Upadhyay·Shyam和Faruqui·Manaal。事实、获取和推理：检索增强生成的统一评估。在美洲国家计算语言学协会*2025*年会议论文集：人机语言技术（第一卷：长篇论文），第4745–4759页，2025。[157] 刘佳伟、田嘉蕾、Daita·Vijay、Wei·宇翔、Ding·Yifeng、Wang·Yuhan·凯瑟琳、Yang·俊和章凌明。RepoQA：评估长上下文代码理解，2024。[158] 彭 Bowen、Quesnelle·Jeffrey、Fan·Honglu和Shippole·Enrico。Yarn：大型语言模型的高效上下文窗口扩展，2023。[159] 高天宇、Wettig·Alexander、He·Luxi、Dong·Yihe、Malladi·Sadhika和Chen·Danqi。元数据调节加速语言模型预训练。在机器学习国际会议（*ICML*），2025。[160] Allen-Zhu·Zeyuan和Li·Yuanzhi。语言模型的物理学：第3.3部分，知识容量缩放定律。*arXiv*预印本*arXiv:2404.05405*，2024。[161] Fan·Dongyang、Hashemi·Diba、Karimireddy·Sai Praneeth和Jaggi·Martin。超越URL：元数据多样性和位置用于高效LLM预训练，2025。[162] Chen·Mark、Tworek·Jerry、Jun·Heewoo、Yuan·Qiming、Pinto·Henrique·Ponde·De·Oliveira、Kaplan·Jared、Edwards·Harri、Burda·Yuri、Joseph·Nicholas和Brockman·Greg等。评估在代码上训练的大型语言模型。*arXiv*预印本*arXiv:2107.03374*，2021。[163] 雅各布·奥斯汀、奥古斯都·奥德纳、马克斯韦尔·奈、马arten·博斯马、亨利克·米哈莱夫斯基、大卫·多汉、艾伦·蒋、蔡瑞、迈克尔·泰勒、Quoc·Le等。使用大型语言模型的程序合成。*arXiv*预印本*arXiv:2108.07732*，2021。[164] White·Colin、Dooley·Samuel、Roberts·Manley、Pal·Arka、Feuer·Ben、Jain·Siddhartha、Shwartz-Ziv·Ravid、Jain·Neel、Saifullah·Khalid、Dey·Sreemanti、Agrawal·Shubh-Agrawal、Sandha·Sandeep·Singh、Naidu·Siddhartha、Hegde·Chinmay、LeCun·Yann、Goldstein·Tom、Neiswanger·Willie和Goldblum·Micah。Livebench：一个具有挑战性、污染限制有限的LLM基准，2025。

[165] MAA. 美国邀请数学考试 - aime. 在 美国邀请数学考试 - *AIME*，2024.[166] MAA. 美国邀请数学考试 - aime. 在 美国邀请数学考试 - *AIME*，2025.[167] Mislav Balunovi´c, Jasper Dekoninck, Ivo Petrov, Nikola Jovanovi´c, 和 Martin Vechev. Math-arena: 在未受污染的数学竞赛上评估大语言模型. *arXiv* 预印本 *arXiv:2505.23281*, 2025.[168] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, 和 Le Hou. 大语言模型的指令遵循评估, 2023.[169] Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, Faeze Brahman, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, 和 Yejin Choi. Wildbench: 使用来自真实用户的挑战性任务基准测试大语言模型, 2024.[170] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, 和 Boris Ginsburg. Ruler: 你的长上下文语言模型的实际上下文大小是多少?, 2024.[171] Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, 和 Danqi Chen. HELMET: 如何有效且彻底地评估长上下文语言模型, 2024.[172] Yang Zhou, Hongyi Liu, Zhuoming Chen, Yuandong Tian, 和 Beidi Chen. GSM-Infinite: 你的大语言模型在无限增加的上下文长度和推理复杂度下表现如何?, 2025.[173] Yongqi An, Xu Zhao, Tao Yu, Ming Tang, 和 Jinqiao Wang. 大语言模型中的系统异常值. 在 第 *13* 届学习表示国际会议, 2025.[174] Alexander Wettig, Kyle Lo, Sewon Min, Hannaneh Hajishirzi, Danqi Chen, 和 Luca Soldaini. 组织网络: 构建领域增强了预训练数据管理, 2025.[175] Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, 和 BryanCatanzaro. Nemotron-cc: 将 Common Crawl 转换为精细的长时预训练数据集, 2025.[176] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, 和 Daya Guo. Deepseekmath: 推动开放语言模型中数学推理的极限, 2024.[177] Fan Zhou, Zengzhi Wang, Nikhil Ranjan, Zhoujun Cheng, Liping Tang, Guowei He, Zhengzhong Liu, 和 Eric P. Xing. Megamath: 推动开放数学语料库的极限, 2025.[178] Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlí˘cek, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, 和 Thomas Wolf. Smollm2: 当 smol 变大 – 以数据为中心训练一个小语言模型, 2025.[179] Shanghaoran Quan, Jiaxi Yang, Bowen Yu, Bo Zheng, Dayiheng Liu, An Yang, Xuancheng Ren, Bofei Gao, Yibo Miao, Yunlong Feng, Zekun Wang, Jian Yang, Zeyu Cui, Yang Fan, Yichang Zhang, Binyuan Hui, 和 Junyang Lin. Codeelo: 使用与人类相当 elo 评分基准测试大语言模型的竞赛级代码生成, 2025.[180] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florence Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, 等. Gpt-4 技术报告, 2024.

[181] OpenAI。Openai o3-mini，2025。[182] OpenAI。发布gpt-5，2025。[183] 闫浩龙、王嘉、黄欣、沈叶青、孟子阳、范志明、谭凯俊、高金、石立宇、杨米等。Step-gui 技术报告。*arXiv* 预印本 *arXiv:2512.15431*，2025。