Step 3.5 Flash: Open Frontier-Level Intelligence with 11B Active Parameters

StepFun Team

 GitHub 🤗 HuggingFace ModelBlog

Abstract

We introduce **Step 3.5 Flash**, a sparse Mixture-of-Experts (MoE) model that bridges the gap between frontier-level agentic intelligence and computational efficiency. We focus on what matters most when building agents: reasoning that's sharp, and execution that's fast and reliable. Reflecting these priorities, Step 3.5 Flash pairs a **196B-parameter foundation** for high-fidelity modeling with **11B active parameters** for efficient inference, optimized by interleaved **3:1 Sliding Window/Full Attention** and **Multi-Token Prediction** (MTP-3) to minimize the latency and cost of multi-round agentic interactions. Toward frontier-level intelligence, we design a scalable RL framework that integrates verifiable signals and preference feedback while maintaining stability during large-scale off-policy training to drive consistent self-improvement across mathematics, code, and tool use. Step 3.5 Flash demonstrates strong intelligence across agent, coding, and math tasks, achieving 85.4% on IMO-AnswerBench and 86.4% on LiveCodeBench-v6 (2024.08–2025.05), 88.2% on τ^2-Bench, 69.0% on BrowseComp (w. Context Manage), and 51.0% on Terminal-Bench 2.0 —performance on par with frontier models such as GPT-5.2 xHigh and Gemini 3.0 Pro. By redefining the efficiency frontier, Step 3.5 Flash provides a high-density foundation for deploying sophisticated agents in real-world industrial environments.
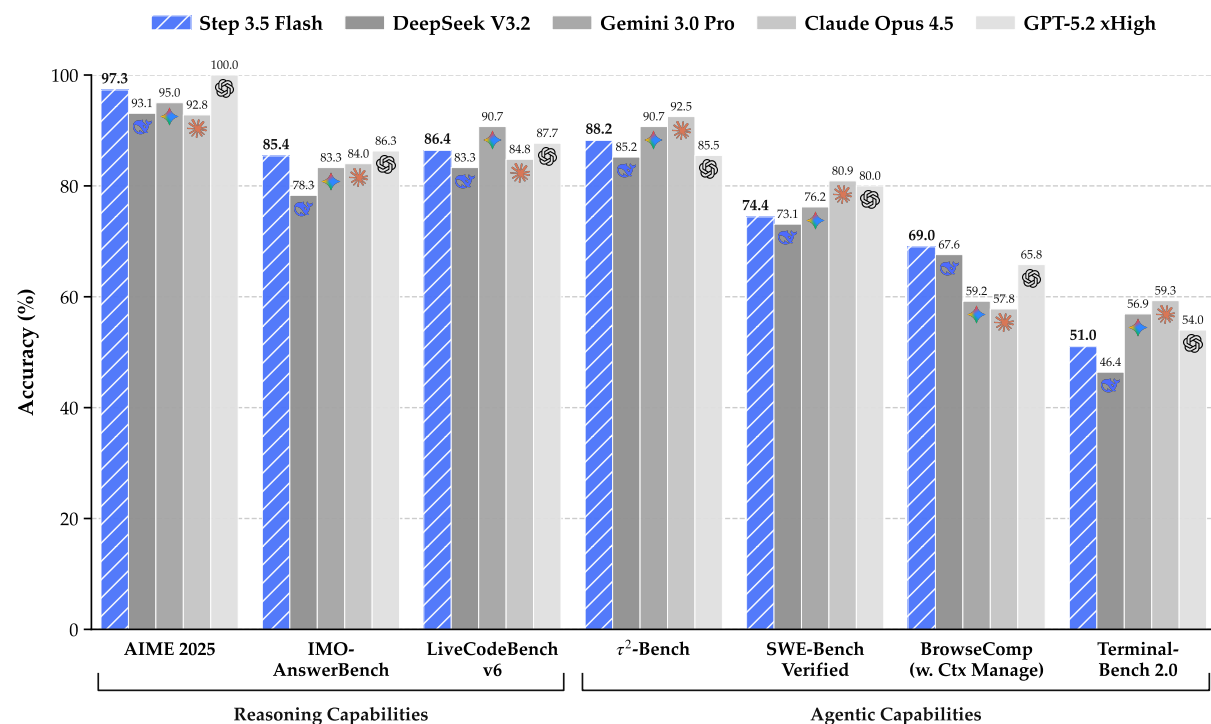

Figure 1: Step 3.5 Flash achieves frontier-level intelligence with only 11B active parameters (196B MoE), comparable to leading closed and open-source models.

Contents

目录

1. Introduction

While open-source large language models (LLMs) [1–6] have rapidly narrowed the performance gap with closed-source frontier systems [7–9] across verifiable tasks [10–12], new challenges emerge as agentic systems gain prominence. In particular, open-source models still trail closed-source frontiers in complex reasoning. Furthermore, critical efficiency bottlenecks hinder their application in long-context agentic tasks [13–21], let alone deployment in edge or resource-constrained settings.

In designing the architecture of Step 3.5 Flash, we focus on two core aspects: efficiency and capacity. We adopt a sparse Mixture-of-Experts (MoE) [22–26] architecture with 196B total parameters and only 11B activated per token, together with a 3:1 ratio of sliding-window attention (SWA) [27] to full attention and multi-token prediction (MTP-3) [3, 28–30] to reduce long-context latency. To improve capacity under hybrid attention with minimal overhead, we increase the number of query heads in sliding-window attention (SWA) layers from 64 to 96 and use head-wise gated attention [31]. This design enables large-scale online deployment, sustaining ~170 tokens/s on Hopper GPUs during the first week on OpenRouter [1].

On the pretraining side, we treat stability as a first-class requirement and build a comprehensive observability and diagnostic stack via a lightweight asynchronous metrics server with micro-batch-level continuous logging. This infrastructure enables systematic identification and mitigation of large-scale MoE failure modes (*e.g.*, Muon-related precision sensitivity, expert collapse [32], and activation blow-ups [5, 33]). Combined with an improved Muon optimizer [34] that offers more accurate and stable updates, we achieve stable training over 17.2T high-quality and diverse tokens with only a single transient loss spike. With this stable training regime, Step 3.5 Flash Base achieves competitive performance against larger counterparts, such as DeepSeek-V3.2-Exp Base [1] and Kimi-K2-Base [5], on math, coding and knowledge benchmarks. Notably, on SimpleQA [35], it scores 31.6%, surpassing DeepSeek-V3.2-Exp Base despite using only one-third of the parameters.

Toward frontier-level intelligence, current post-training systems face two tightly coupled challenges: inefficient iteration of domain-specific experts for self-distillation [1–4] and limited scalability of Reinforcement Learning (RL) to long-horizon reasoning for MoE models. Training a single generalist to directly cover diverse domains often sacrifices domain-specific expertise, whereas maintaining separate expert models leads to fragmentation and an unsustainable cost of continual multi-model iteration. At the same time, as models are extended to deeper reasoning trajectories, even small token-level discrepancies in off-policy rollouts can accumulate into high-variance gradients. This effect is particularly severe in MoE models, where expert-level routing induces larger distributional shifts and destabilizes optimization in the frontier performance regime [1, 36–38].

To address these challenges, we propose a unified post-training recipe for large-scale RL built on a shared SFT foundation. The framework alternates between domain-specific specialization and global synthesis, enabling efficient expert iteration while maintaining a single, high-performing generalist. A dedicated mid-training phase scales the context window to 128k and strengthens core agentic and reasoning capabilities via synthetic data, providing a strong initialization for downstream post-training. To support stable and scalable RL within this unified framework, we introduce Metropolis Independence Sampling-Filtered Policy Optimization (MIS-PO) [39, 40], replacing continuous importance weighting with discrete, distributional filtering at both token and trajectory levels. By restricting optimization to samples within a stable trust region, MIS-PO substantially reduces gradient variance while preserving effective learning signals, enabling RL to scale reliably to long-horizon reasoning and agentic behaviors.

---

[1]https://openrouter.ai

Step 3.5 Flash achieves competitive performance with leading frontier models and systems across a broad range of reasoning and agentic benchmarks, despite 11B active parameters. It delivers strong results under standard inference on reasoning tasks, including 85.4% on IMO-AnswerBench [41] and 86.4% on LiveCodeBench-v6 (2024.08–2025.05) [12], while also demonstrating robust long-horizon, tool-augmented capabilities with 88.2% on τ^2-Bench [15], 69.0% on BrowseComp (with context management) [17], and 51.0% on Terminal-Bench 2.0 [16]. With PaCoRe [42] deep think inference, Step 3.5 Flash further improves performance on reasoning-intensive benchmarks requiring extended deliberation and multi-round synthesis. Taken together, these results indicate that Step 3.5 Flash substantially narrows the gap between advanced open models and frontier proprietary systems in both reasoning and agentic settings.

2. Architecture

2.1. Design Philosophy

The architecture of Step 3.5 Flash reflects a paradigm shift in model–system co-design. Beyond the traditional objectives of intelligence and cost, the era of autonomous agents elevates a third critical constraint: *inference latency*. In interactive agentic workflows [43, 44], minimized latency translates directly to reduced wall-clock time for task completion, or conversely, allows for increased intelligence within a fixed time budget via test-time scaling [42, 45–47].

Agentic workloads typically exhibit a distinct profile: extensive context prefilling followed by prolonged, multi-turn interactive decoding. Accordingly, we co-design Step 3.5 Flash for low wall-clock latency along three coupled axes: *attention* (to accelerate long-context processing and have good affinity with MTP), *sparse MoE* (to prevent stragglers in distributed deployments that reduce throughput), and *multi-token prediction* (MTP; to facilitate fast generation through speculative decoding).

Attention. To accelerate prefilling, we employ a hybrid attention mechanism [33, 48, 49] to mitigate the quadratic complexity of long-context processing. For decoding, we prioritize architectural compatibility with speculative decoding [50], since verification efficiency is the dominant lever on bandwidth-bound hardware. These considerations motivate two attention design decisions:

- *Sliding-Window Attention (SWA).* We select SWA [27] over linear attention [10, 51] to maximize decoding efficiency. Although both have linear complexity, the state-update mechanism of linear attention complicates efficient draft tree generation and parallel tree verification needed for speculative decoding [52–54]. In contrast, SWA preserves standard attention semantics and remains inherently amenable to parallel verification via *KV* masking. Moreover, in the absence of robust empirical evidence that linear attention yields superior long-context modeling for agentic tasks, we find that SWA with window size W=512 strikes a favorable balance between kernel efficiency and capturing local dependencies.
- *Hardware-Aligned Grouped-Query Attention (GQA-8).* Targeting deployment on standard 8-GPU server nodes, we configure the model with eight *KV* heads (GQA-8) [55]. This aligns *KV*-cache sharding with 8-way tensor parallelism and improves memory access patterns. Crucially, while GQA-8 makes attention more memory-bandwidth bound, it also creates computational slack that can absorb speculative drafting and verification overhead, enabling aggressive multi-token speculation without a proportional latency penalty.

Sparse MoE. On the feed-forward side, we employ fine-grained MoE [22–26] to reduce the average FFN compute while maintaining capacity. Expert parallelism (EP) [25] is utilized to enable scalable

---

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
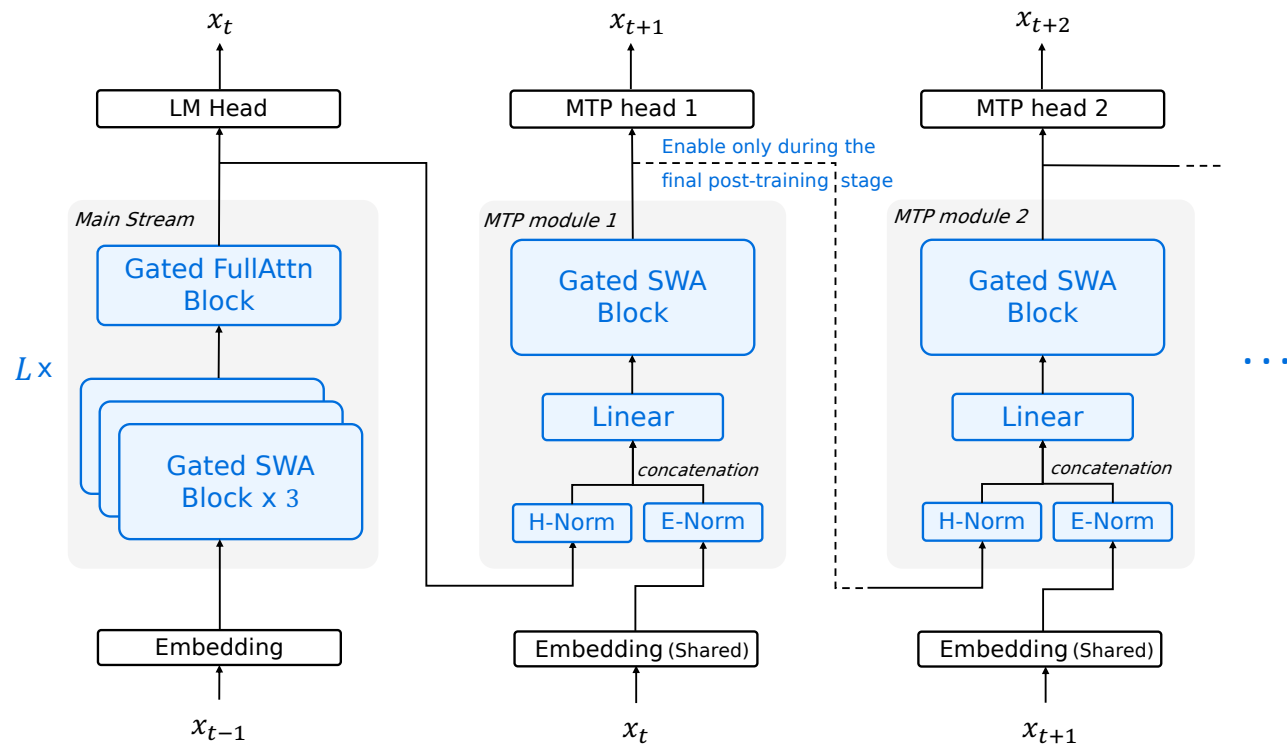
Figure 2: Illustration of Step 3.5 Flash. The model uses head-wise gated attention [31] with a leading Full Attention layer followed by $L = 11$ Hybrid Blocks, each interleaving 3 Sliding Window Attention (SWA) layers with one Full Attention layer (*for visual clarity, the first layer is omitted in the figure*). We apply zero-centered RMSNorm [57] throughout. The first three blocks use dense FFNs; later blocks employ sparse MoE FFNs. MTP modules use SWA and dense FFNs. To limit overhead, only MTP module 1 is trained during main training; MTP modules 2–3 are cloned from it and jointly fine-tuned in a lightweight final phase.

deployment. However, under EP, end-to-end latency can be dominated by *stragglers* induced by routing imbalance: token assignment skew concentrates workload on a small subset of experts and their hosting GPUs, throttling throughput at synchronization points. We therefore introduce an *EP-Group Balanced MoE Routing* strategy.

Multi-Token Prediction (MTP). To further reduce autoregressive latency, we incorporate Multi-Token Prediction (MTP) [29, 56] as a complementary lever to speculative decoding [50]. To keep speculation lightweight, we streamline the MTP heads by leveraging SWA and dense FFNs [3].

We further constrain the model size to under 200B parameters, enabling high-performance inference within the 128GB memory budget of high-end workstations.

2.2. Sparse MoE Backbone with Hybrid Attention

As illustrated in Fig. 2, Step 3.5 Flash adopts a 45-layer sparse-MoE Transformer backbone (3 dense layers and 42 MoE layers) paired with a specialized hybrid attention layer layout. Each MoE layer contains 288 routed experts plus one shared expert, with a top-k router activating k=8 experts per token. This configuration maintains an extensive knowledge capacity (196B total parameters) while restricting per-token activation to just 11B, ensuring inference latency remains low enough for highly

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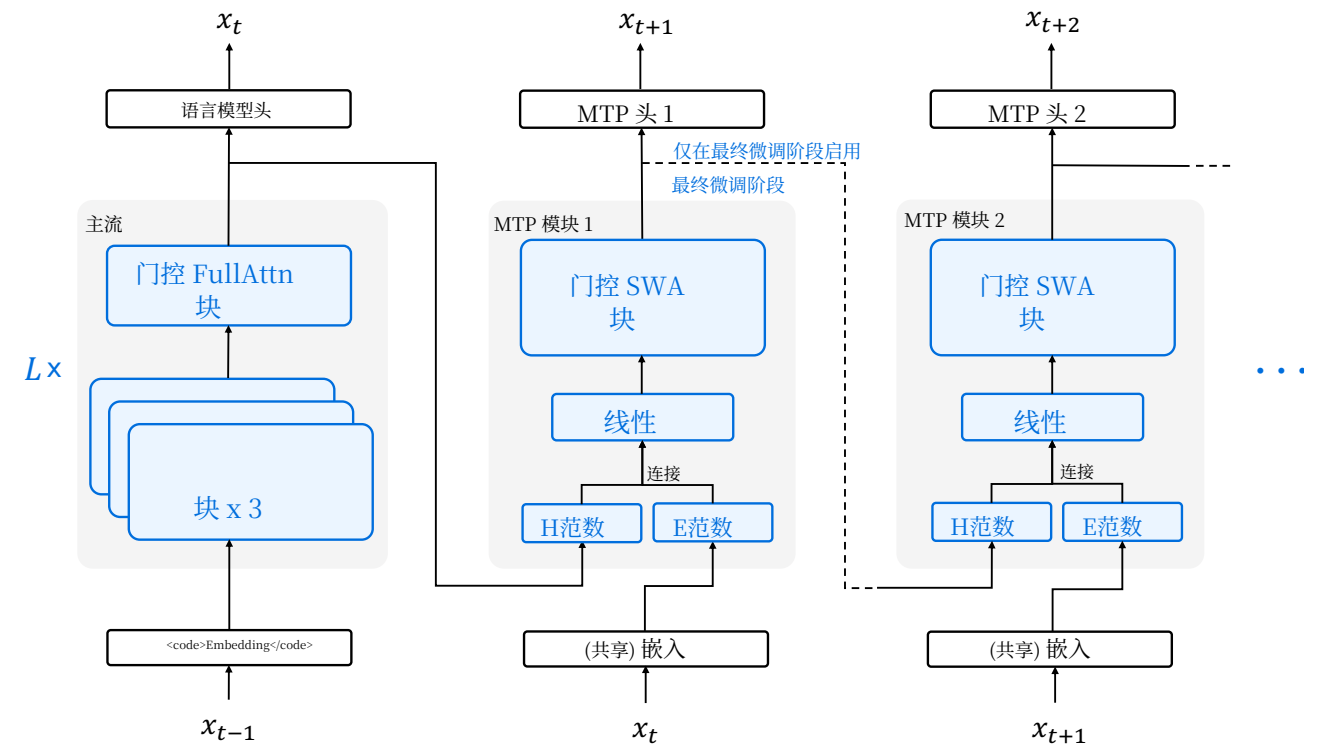
图2：Step 3.5 Flash 的示意图。该模型使用逐头门控注意力 [31]，由一个领先的密集注意力层后接 $L = 11$ 混合块组成，每个混合块交错嵌入3个滑动窗口注意力（SWA）层和一个密集注意力层（为视觉清晰起见，图中省略了第一层）。我们全程应用零中心 RMSNorm [57]。前三个块使用密集 FFN；后续块采用稀疏MoE FFN。MTP 模块使用SWA和密集FFN。为限制开销，仅在主流训练期间训练MTP模块1；MTP模块2-3从其克隆并联合在轻量级最终阶段进行微调。

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

responsive agent interaction. Table 6 summarizes key architecture hyperparameters of Step 3.5 Flash.

Hybrid Attention Layer Layout. To balance long-context efficiency with robust long-range connectivity, Step 3.5 Flash leverages an interleaved attention layout at a $3:1$ ratio (SWA : Full) inspired by [33, 49, 58], denoted as $S3F1$. This configuration repeats a four-layer motif consisting of three SWA layers (W=512) followed by a single full GQA-8 layer. However, in our initial experiments, a naive interleaving strategy consistently underperforms a dense attention baseline across various benchmarks (Table 10). To bridge this performance gap without adding practical overheads, we leverage two complementary enhancements: (i) an increased SWA query-head count, and (ii) adopting head-wise gated attention [31].

Augmented Query Heads in SWA. Using a higher query-head number (from 64 to 96) effectively mitigates performance drop typically observed when transitioning from a uniform full-attention architecture to the $S3F1$ layout (Table 10). We consider this to be nearly a "free lunch". Because in long-text scenarios, the overhead of naive SWA is very small, even though our solution scales up significantly.

Head-wise Gated Attention. A limitation of naive SWA is its inability to effectively absorb unused attention weights when there is no useful information in the input window [31, 59–61]. Previous work [3, 33] introduce learnable, data-independent sink tokens into the window to address this issue. Instead, we opt for a different approach by integrating a parameter-efficient head-wise gating mechanism [31, 62, 63], which can be viewed as integrating data-dependent sink tokens. Please refer to Appendix A.1 for implementation details and further discussion. Head-wise gating is also negligible to both theoretical FLOPs and practical latency. We report more performance analysis and benchmarks for gating and augmenting the number of SWA heads in Appendix A.2.

MoE Expert-parallel Load Balancing. We use loss-free load-balancing [29, 64] to encourage *global* token balance across experts. However, this approach does not guarantee balanced loads across EP ranks at the micro-batch level, potentially leading to stragglers and reduced throughput. We therefore introduce an EP-level balancing loss that explicitly promotes uniform rank-level utilization [26].

EP partitions experts $\mathcal{E}$ into G disjoint groups $\{\mathcal{E}_g\}_{g=1}^{G}$ across ranks. For token t, let S_t denote the top-K experts (mask $s_{t,e} = \mathbf{1}[e \in S_t]$) and $p_{t,\cdot}$ the routing probabilities. Then, the EP load balancing loss $\mathcal{L}_{EP}$ is:

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

Multi-token Prediction (MTP). To speedup speculative decoding on long-context agentic workloads, we attach three lightweight multi-token prediction (MTP) heads. Each MTP head consists of a SWA and a dense FFN, adding only 0.81B parameters (~0.41%). We index these heads by their *additional* prediction offset beyond the standard LM head: for $h \in \{1, 2, 3\}$, MTP-h predicts the token x_{t+1+h} conditioned on the backbone hidden states at position t. To control training overhead, we activate and optimize only MTP-1 in most training stages. Once the backbone is well-trained, we initialize MTP-2 and MTP-3 from MTP-1 and jointly train all MTP heads in a lightweight post-training phase. Inspired by Fast-MTP [65], we adopt position-dependent loss reweighting across prediction offsets in MTP heads to prevent over-optimizing for distant-token predictions.

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$\{E_g\}_{g=1}^{G}$ 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:

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

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。

Layout	SWA Heads	Rel. FLOPs Decode / Prefill	Pre-train Avg.	Downstream Performance						
				Reasoning	Math	Code	Sci	General	LongCtx	Avg.
FFFF	32	~2.68 / 2.90	54.1	40.8	40.9	**19.6**	42.7	26.5	28.8	33.2
S1F1	32	~1.58 / 1.65	54.6	**42.1**	**42.3**	19.3	**44.5**	**26.8**	**29.6**	**34.1**
S3F1	32	**1.00 / 1.00**	53.6	40.2	40.4	18.9	42.4	25.4	27.5	32.5
S3F1+Head	48	~1.01 / 1.02	**55.7**	40.6	40.3	18.3	44.0	26.0	28.2	32.9

Table 1: Downstream results on 30B-A3B. *F* denotes full attention and *S* denotes SWA. *S3F1* indicates three *S* layers followed by one *F* layer in the hybrid layout. Rel. FLOPs are normalized to the *S3F1* configuration and averaged over 64K/256K contexts (Table 8). Pre-train Avg. aggregates results across general, math, and code benchmarks (Table 16).

2.3. Architecture Ablations and Results

We conduct extensive experiments to validate key design choices in Step 3.5 Flash, focusing on (i) attention layouts, including SWA and head scaling, and (ii) head-wise gated attention versus sink tokens. To ensure our efficiency optimizations do not degrade model performance, we adopt two complementary ablation protocols: one evaluates full end-to-end pipelines covering pretraining, 32k long-context extension, and 64k context-length supervised fine-tuning (SFT), and the other scales the analysis up to 100B parameters to study how these design choices behave with scale. Detailed architecture and evaluation setups for all tables are provided in Appendix A.4. Key findings from these large-scale experiments are summarized below.

SWA w.r.t. Long Context. We train a 30B-A3B model through the full pipeline (1.4T-token pretraining, followed by SFT) to evaluate the end-to-end impact of hybrid attention on reasoning and long-context performance. We ablate four attention layouts: all-full attention (*FFFF*), alternating SWA/full (*S1F1*), a 3:1 SWA-to-full layout (*S3F1*), and an *S3F1* variant with increased SWA query heads (*S3F1*+Head). To isolate attention-structure effects, we fix the SWA window size to W=512 and disable MTP (see Appendix, Table 9 and Table 10).

Table 1 shows a clear cost–quality trade-off across layouts. *S3F1* achieves the lowest normalized attention-side FLOPs (normalized to 1.00 for prefill and 1.00 for decode separately), whereas *FFFF* is ~2.68×/2.90× as expensive as *S3F1*; however, *S3F1* exhibits a consistent quality degradation (*e.g.*, LongCtx drops from 28.8 to 27.5).

Increasing the number of SWA query heads largely compensates for this loss. Notably, *S3F1*+Head already surpasses *FFFF* during pretraining (55.7 vs. 54.1), and remains competitive after post-training: LongCtx improves from 27.5 to 28.2 and Sci from 42.4 to 44.0, closing most of the gap to the *FFFF* baseline with negligible additional attention cost. The remaining downside is limited and localized (*e.g.*, a modest drop on Code to 18.3), while overall quality trends favor *S3F1*+Head.

Interestingly, the alternating *S1F1* layout delivers the best overall SFT quality and the strongest LongCtx score (29.6), but requires substantially higher attention-side prefill/decode FLOPs (~1.58/1.65), about a 60% cost increase relative to *S3F1*+Head. We therefore adopt *S3F1*+Head as the default configuration for long-context agentic workloads, prioritizing its much lower prefill/decode cost with strong and stable long-context performance.

Head-wise Gated Attention vs. Sink Tokens. We conduct scaled, controlled pretraining experiments on a 100B-A10B MoE to study attention-side mechanisms under realistic scaling conditions.

Method	BBH	MMLU	GPQA	MBPP	C-EVAL	CMMLU	Avg.
Sink Token	70.6	65.1	27.2	61.2	76.2	74.6	62.5
Head-wise Gate	**73.7**	**67.0**	**28.1**	**62.6**	**77.9**	**77.1**	**64.4**

Table 2: Pretraining-only evaluation on a 100B-A10B model under the *S3F1* layout. Head-wise gating consistently outperforms a fixed sink token across benchmarks, including the overall average.

Specifically, we compare *sink tokens* and *head-wise gated attention* while holding the attention layout fixed to the same *S3F1* configuration with window size W=512. As shown in Table 2, head-wise gating consistently improves quality, raising the average performance from 62.46 to 64.43 (+1.97). We therefore adopt head-wise gated attention as the default mechanism in subsequent studies.

3. Infrastructure

3.1. Compute Cluster

Step 3.5 Flash is trained on a large-scale cluster with 4,096 NVIDIA H800 GPUs. Each node contains 8 GPUs interconnected through NVLink and NVSwitch for high-bandwidth intra-node communication. For inter-node connectivity, the cluster relies on 8×200 Gbps RoCE links to maintain efficient synchronization and data exchange at scale.

3.2. Training Framework

The training of Step 3.5 Flash is powered by our internal *Steptron* framework, a lightweight high-performance system built on top of PyTorch [66] and Megatron-LM [67]. Steptron unifies the full model development pipeline, supporting large-scale pre-training, post-training, and reinforcement learning (RL) workloads under a single engineering stack.

Step 3.5 Flash employs a hybrid parallelization strategy, including 8-way pipeline parallelism (PP) [68] with virtual pipeline stages (VPP), and 8-way expert parallelism (EP) [25], and ZeRO-1 Data Parallelism (DP) [69]. In order to facilitate efficient training of Step 3.5 Flash, we employ the following engineering techniques.

Decoupled Parallelism. Following Megatron-Core [70], we implement a decoupled parallelization scheme that allows the attention and MoE modules to use different parallelization strategies. We assign them independent parallel groups and perform gradient reduction and scaling within each module's corresponding data-parallel group.

Communication Optimization. Concurrent DP communication streams for decoupled attention and MoE can saturate RoCE links, incurring considerable increases in DP overheads due to congestion. To address this, we propose two complementary communication optimizations that jointly reduce iteration time by up to 5%. First, *fabric-aware communication scheduling* partitions DP traffic into intra-node NVLink and inter-node RoCE phases, and pipelines them to fully utilize both fabrics. Second, *communication-aware rank placement* uses job-level communication profiles to place ranks across switches, reducing hop counts and steering heavy traffic away from inter-switch hotspots.

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 Resharding. Muon [34] requires full (unsharded) per-parameter gradients for Newton–Schulz orthogonalization, which conflicts with ZeRO-1 [69] reduce-scatter that shards a parameter's gradient across DP ranks. The current implementation in Megatron-LM resolves this mismatch by naively all-reducing FP32 gradients to reconstruct full gradients prior to the Muon update but nearly doubles communication. We instead assign whole parameters to DP ranks and repack the gradients buffer into a rank-major buffer so a single reduce-scatter delivers each parameter's complete gradient to its owner. Since padding to the fattest rank incurs overhead that grows with the data-parallel size, we apply this only to expert parameters and use DP all-reduce for non-expert parameters. This hybrid strategy reduces end-to-end iteration time by approximately 5% with less than 4 GB additional memory compared to the naive all-reduce baseline.

GPU Kernels Optimization. We also apply kernel-level optimizations to improve training efficiency. In attention, we fuse QK normalization with RoPE. In MoE, we fuse multiple small operators to reduce kernel-launch overhead and memory traffic, and implement a fused MoE gather/scatter with grouped GEMM, similar to SonicMoE [71].

Fine-grained Selective Checkpointing. Our training framework supports fine-grained activation recomputation with per-layer, submodule-level toggles (*e.g.*, attention, FFN, normalization, SiLU, and MoE permutation), enabling selective recomputation of only the most memory-intensive components to reduce peak memory with minimal overhead.

3.3. High-Throughput Lightweight Monitoring

We collect a comprehensive suite of metrics (e.g., expert distribution within each micro-batch and gradient norms) for fine-grained monitoring of the training. However, the telemetry scale is immense: a 4,096-GPU workload generates nearly 6 million messages per iteration. Conducting a synchronous global reduction within the main loop would introduce a significant overhead of several seconds, effectively doubling the iteration time, which is clearly intolerable for high-performance training. To mitigate this, we develop a Lightweight Metrics Server to decouple telemetry processing from the training path. Each rank utilizes *StepRPC*, an in-house asynchronous communication framework, to asynchronously offload local metrics to the remote server. This approach reduces telemetry overhead to approximately 100 ms per iteration.

The Metrics Server buffers incoming metrics and triggers reduction and database persistence only after receiving *end-of-iteration* signals from all participating ranks, eliminating synchronization in the main loop. To ingest and process millions of messages with low latency, the server is implemented as a high-concurrency multi-process system with two decoupled modules: (i) a Message Receiver optimized for high-throughput ingestion, and (ii) a Reduction Processor responsible for aggregation and persistence. By exploiting multi-core parallelism within and across these modules, the server keeps pace with the telemetry stream and ensures that metrics management never lags behind training.

4. Pre-Training and Mid-Training

Overview. This section summarizes our pre-training and mid-training process, with an emphasis on the practical stability constraints of large-scale sparse MoE training. We first describe training stability diagnostics and mitigations (Section 4.1), then detail the curriculum used for pre-training and mid-training, including the data mixture, schedule, and key hyper-parameters (Section 4.2).

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）。
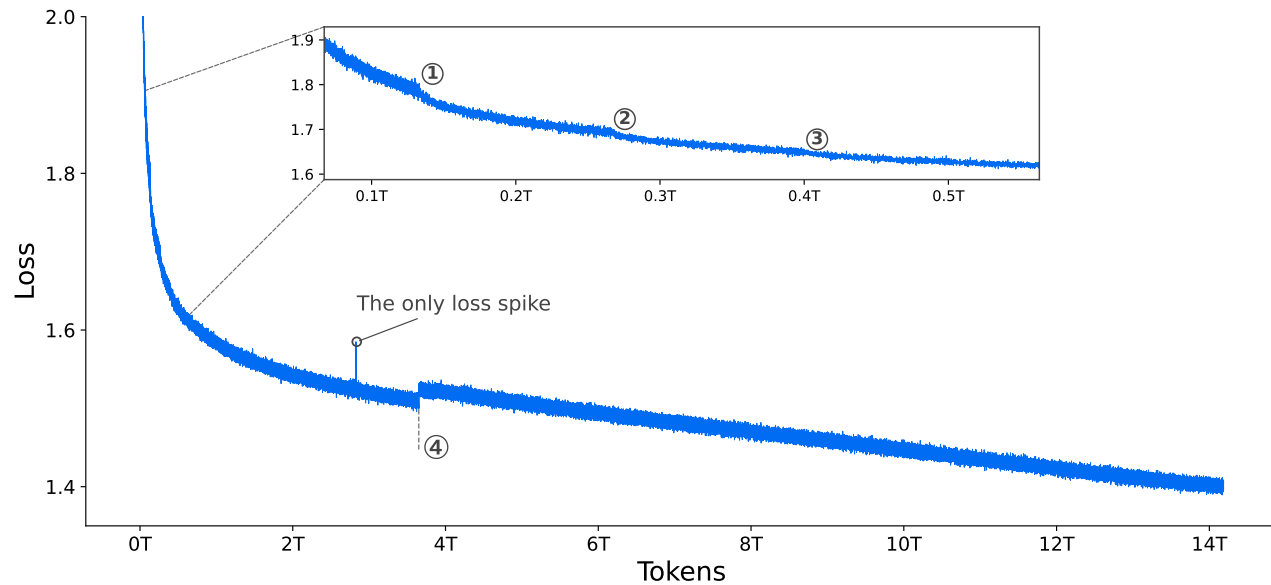
Figure 3: Per-step training loss of Step 3.5 Flash, plotted **without smoothing or sub-sampling**. We observe merely **one** isolated loss spike across the full training duration. The initial training steps are omitted for clarity. Markers ①–③ indicate batch size increases to 8,192, 12,288, and 16,384, respectively. Marker ④ denotes the activation of the loss mask on meta tokens (see Appendix A.3 for details).

4.1. Training Stability

Training stability is a **first-class** requirement for large-scale sparse MoE pre-training. To make stability actionable, we build a comprehensive observability and diagnostic stack based on a lightweight asynchronous metrics server with micro-batch-level continuous logging (described in Section 3.3). This infrastructure provides fine-grained visibility into both optimizer-level and expert-level signals, enabling systematic mitigation of recurring failure modes in large-scale MoE training.

In practice, we find three dominant instabilities that the metrics stack helps surface early and localize precisely: (i) transient loss spikes and occasional stochastic numerical blow-ups caused by Muon's [34] numerically sensitive polar-factor iteration under reduced precision, (ii) expert-side collapse ("dead experts") that can occur even when router dispatch statistics remain apparently healthy, and (iii) localized activation blow-ups confined to a small subset of experts.

With the mitigations guided by these diagnostics, the pre-training loss remains smooth throughout the run, exhibiting only a single loss spike. Figure 3 shows the full curve prior to learning-rate cooldown.

4.1.1. Numerical Sensitivity of Muon

Muon approximates a semi-orthogonal update direction via a Newton–Schulz (NS) iteration [72]. In early experiments, we find modest, consistent loss reduction when using a faster-converging orthogonalization approximation. We therefore adopt the Polar Express [73] iteration and run a fixed $T=6$ steps to balance optimization quality and throughput.

However, we occasionally observe sharp, unrecoverable loss spikes despite using the recommended safety scaling [73]. The spikes are non-deterministic (often avoided by resuming from a nearby checkpoint), suggesting a numerical pathology. Simulations indicate that `bfloat16` Polar Express can rarely yield extreme intermediate outliers under certain update statistics due to cumulative error in addition. We therefore cast **only** the Polar Express iteration (state and intermediates) to `float16`
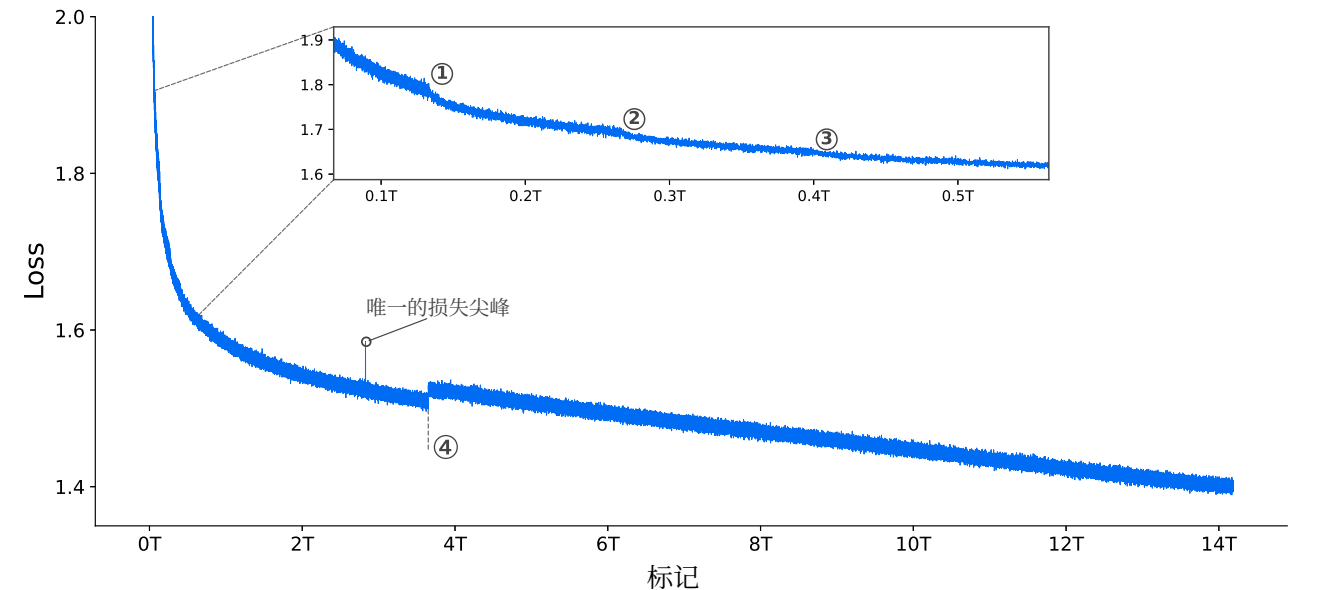
图3: 步骤3.5 Flash的每步训练损失，绘制时未进行平滑或子采样。我们在整个训练过程中仅观察到一个孤立的损失尖峰。为清晰起见，省略了初始训练步骤。标记①–③分别表示批大小增加到8,192、12,288和16,384。标记④表示元标记上的损失掩码激活（有关详细信息，请参见附录A.3）。

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`

while keeping the rest of the training mixed-precision. After this change, the spikes do not recur.

4.1.2. Expert Collapse Beyond Routing Collapse

Step-3, our prior work [32] reports that MoE training may exhibit "dead experts", often described as experts receiving negligible token dispatch for extended periods and therefore obtaining little effective gradient signal. In our prior investigation, we find that expert collapse can also manifest as an *expert-side* pathology even when router dispatch remains stable, *i.e.*, vanishing expert activations and stagnant or decaying expert parameter norms.

We observe that two factors are particularly influential: (i) Routed-expert aggregation requires explicit scaling. When incorporating a shared expert, it is important to introduce an explicit scaling factor to calibrate the relative contribution of the shared expert and the routed experts. While smaller models may implicitly learn such a balance, larger models are less reliable at self-calibration. A mismatch can suppress the effective contribution of routed experts even if routing frequencies appear healthy. (ii) Micro-batch balancing can be overly restrictive under fine-grained sparsity. For sparse, fine-grained MoE designs, micro-batch-level load-balancing constraints (as commonly implemented in Switch-style routing [22]) can become overly stringent. As analyzed in [74], micro-batch LBL may induce excessive cross-expert competition and hinder effective specialization.

We therefore prefer broader-scope balancing (*e.g.*, global-batch statistics) [74, 75] or loss-free bias adjustment based on observed load [29, 64]. In practice, router dispatch statistics are typically stable and are not sensitive indicators of expert collapse. We recommend monitoring expert-side signals, including per-expert activation norms (*e.g.*, RMS/mean norm at the MoE FFN intermediate) and parameter norms (*e.g.*, Frobenius norms of expert projection matrices). When a subset of experts drifts toward near-zero activations/updates while the median remains stable (*e.g.*, decreasing min-to-median ratios), it provides an early warning of expert "death".

4.1.3. Localized Activation Blow-up in MoE Layers

As expert specialization matures during the main training phase, we observe a localized stability pathology in the deeper MoE layers. Specifically, the activation **norm** of a small subset of experts (often just one or two per layer) grows rapidly, while the majority of experts in the same layer remain well-behaved. This disparity results in a heavy-tailed activation distribution: the median expert activation norm remain stable, but the maximum activation norm explodes, significantly increasing the risk of numerical overflow and downstream instability.

Figure 4 illustrates this failure mode. **Remarkably, this internal instability is entirely masked by the training loss**, which shows negligible variation despite the underlying explosion in norms shown in Panel (a). We track this phenomenon by monitoring the dispersion of per-expert FFN output norms. As observed in Panels (b) and (c), while the middle layers (*e.g.*, Layer 38) retain stable distributions, the final layers (*i.e.*, Layer 45) exhibit a rapidly widening gap between the maximum (solid lines) and the median (dashed lines). This indicates that activation energy is concentrating dangerously in a few "rogue" experts in the deeper network. To mitigate this, we evaluate two distinct interventions:

- Weight clipping on expert projections: We constrain the norm of the MoE FFN expert projection matrices. For each expert projection matrix W, if its maximum activation norm $\max_x \|Wx\|$ exceeds a threshold τ, we rescale it via $W \leftarrow W \cdot \frac{\tau}{\max_x \|Wx\|}$. This is similar to MuonClip in attention [5], but we perform clipping offline on the checkpoint rather than on-the-fly.
- Activation clipping inside experts: We apply element-wise clipping directly to the MoE FFN intermediate activations prior to the output projection, as in [33].

(a) Loss vs. Steps (b) Expert Output (Layer 38) (c) Expert Output (Layer 45)

Figure 4: Analysis of expert activation stability and mitigation strategies. In Panels (b)–(c), solid lines represent the maximum expert output norm, while dashed lines represent the median. (1) Depth-Dependent Instability: While training loss appears identical across methods (Panel a) and middle layers remain stable (*e.g.*, Layer 38 in Panel b), the final layers (*i.e.*, Layer 45 in Panel c) suffer from catastrophic norm explosion in the *No clipping* baseline. (2) Mitigation: *Weight clipping* merely delays this explosion. In contrast, *Activation clipping* effectively bounds maximum norms, ensuring stability across all layers.

Although the training loss appears indistinguishable across different mitigation strategies in Figure 4 (a), the max-to-median ratio reliably unmasks underlying instability. As evidenced in Panels (b) and (c), activation clipping ensures a stable trajectory for internal norms, whereas weight clipping alone fails to prevent the recurrence of outlier experts. Consequently, we establish the max-to-median ratio of per-expert activation norms as a robust and necessary metric for monitoring training stability.

The activation blow-up is driven by several factors. We observe that high-frequency bi-grams can trigger expert specialization. When using pre-norm [76, 77], a single expert can amplify its output boundlessly and dominate the final output norm, leading to near-deterministic prediction behavior. This risk is exacerbated by SwiGLU [78], where strong alignment between the gate and up-projection branches produces sparse activations with extreme magnitudes. Muon further accelerates this collapse by amplifying persistent low-rank updates. A detailed analysis is provided in Appendix B.

4.2. Training Curriculum

The training proceeds from broad open-domain coverage to increasingly agentic and long-context specialization. We first pre-train at 4k context on a broad open-domain mixture to establish general-purpose capabilities, then anneal the mixture toward higher-quality knowledge and more software-development data (code, PRs, issues, and commits) while extending the context window to 32k . Next, a dedicated mid-training stage expands the context window from 32k to 128k to strengthen long-horizon reasoning and improve initialization for downstream post-training and agentic workloads. Overall, we train on approximately 17.6T tokens for pretraining and 750B tokens for mid-training.

4.2.1. Data Mixture

Our corpus combines general open-domain data with agentic-oriented data. We summarize the key sources below, more details can be refered in Appendix C.

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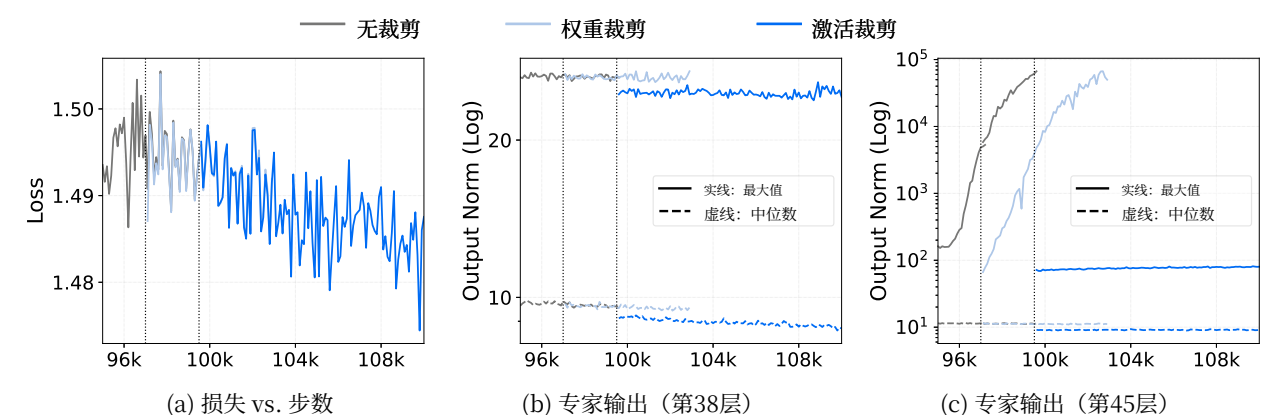
图4：专家激活稳定性分析及缓解策略。在(b)–(c)面板中，实线表示最大专家输出范数，而虚线表示中位数。(1)深度相关的不稳定性：虽然训练损失在所有方法中看起来相同（a面板），并且中间层保持稳定（例如，b面板中的第38层），但最终层（即，c面板中的第45层）在<No clipping>基线中遭受灾难性的范数爆炸。(2)缓解：<Weight clipping>仅仅延迟了这种爆炸。相比之下，<Activation clipping>有效地限制最大范数，确保所有层的稳定性。

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。

General Knowledge Data. To support broad world knowledge, we build **StepCrawl** (Appendix C.1.1), an in-house crawling and curation infrastructure beyond standard Common Crawl [79], to harvest trillions of high-quality tokens at scale from web pages (HTML) and book-/document-like sources (ePub/PDF). All content is processed with multi-stage quality filtering, site/category tagging, deduplication, and sanitization.

Code Data. Strong code capacity is foundational for agentic models. Our code corpus is curated and refined using a modified OpenCoder [80] pipeline. We relax filtering from a zero-tolerance policy to allowing 0–6 heuristic violations (Appendix C.2.1) per document, balancing quality and diversity, and upsample code-centric data during annealing and mid-training to strengthen agent-related programming.

PR/Issue/Commit Data. To better match real software-engineering workflows, we curate a comprehensive PR/Issue/Commit dataset(Appendix C.2.2) from GitHub repositories with 10+ stars. This includes (1) Base Data validated against `git diff` (deduplicated against benchmarks [14, 81]); (2) PR-Dialogue Data derived from PR threads and commits using Agentless-style templates [82] for file localization and code repair; and (3) derivative software-engineering corpora used in mid-training and post-training.

Tool-Use and Reasoning Data. To improve tool-use robustness and multi-step reasoning, we add synthetic and semi-synthetic data spanning math/code/science/general knowledge, and domain-specific samples targeting search agent, SWE agent, and tool execution. During mid-training, we further introduce long-context samples (natural long documents and long-form synthetic tasks) to reinforce planning and reasoning over extended contexts.

4.2.2. Schedule

Pre-training schedule. Pre-training consists of two stages:

1. **Pre-training Stage 1: Open-domain pre-training** (14.6T tokens, 4k context). Broad open-domain training to maximize coverage and foundational capability.
2. **Pre-training Stage 2: Annealing + long-context initialization** (3T tokens, 4k to 32k context). We anneal the data mixture toward code and PR/Issue/Commit-centric sources, while increasing the share of higher-quality knowledge and reasoning-dense samples. This stage starts with 2T tokens at 4k context, then transitions to 1T tokens at 32k context under the same annealed mixture to initialize long-context training.

Mid-training schedule. Mid-training also consists of two stages:

1. **Mid-training Stage 1: Specialization at 32k** (386B tokens, 32k context). We replay 81B tokens (21%) from pre-training to mitigate distribution shift and stabilize specialization, while emphasizing software-engineer and tool-use-centric mixtures.
2. **Mid-training Stage 2: Long-context specialization** (364B tokens, 128k context). We retain 10.5B replay tokens, and further specialize long-context capability with a mixture of synthetic long-horizon reasoning and natural long documents (selected from pre-training data with length > 32k), plus domain-specific data for code agent, search agent, and tool-use.

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. Hyper-Parameters

Pre-training hyper-parameters. We use the Muon optimizer [34] throughout pre-training, set weight decay to 0.1 and gradeint clip to 1.0. The learning rate is linearly warmed up from 0 to 2.5×10^{-4} over the first 2,000 steps and then cosine-decayed to 5×10^{-5} over Pre-training Stage 1. In Pre-training Stage 2, we apply a secondary cosine decay from 5×10^{-5} to 2×10^{-5} over the 4k portion (2T tokens) and keep the learning rate fixed at 2×10^{-5} for the 32k portion (1T tokens). The global batch size gradually increases from 4096 to 16384 over the first 400B tokens, and keeps 16384 in the remaining training, and is set to 2k for the 32k portion of annealing. The MTP loss weight is set to 0.3 in Pre-training Stage 1 and 0.1 in Pre-training Stage 2, following [29]. For loss-free load balancing, the bias update rate is 0.001 for the first 14.6T tokens and decays to 0.0 during annealing, and an EP-group balance loss with coefficient 0.001 is applied throughout pre-training. For RoPE [83], we use $\theta = 10,000$ for both full attention and sliding window attention (SWA) during 4k training, and set $\theta_{\text{Full}} = 1,000,000$ only for full attention and maintain $\theta_{\text{SWA}} = 10,000$ for the 32k portion of annealing.

Mid-training hyper-parameters. We continue to use Muon [34] during mid-training. We freeze the MoE router weights and disable the EP-group balance loss and fix the MTP loss weight to 0.1 for both mid-training stages. The learning rate is warmed up from 0 to 2×10^{-5} over the first 3% of iterations, kept constant in Mid-training Stage 1, and decayed to 7.3×10^{-6} in Mid-training Stage 2. For RoPE selective scaling, we set $\theta_{\text{Full}} = 1,000,000$ at 32k (Mid-training Stage 1) and increase to $\theta_{\text{Full}} = 5,000,000$ at 128k (Mid-training Stage 2), while keeping $\theta_{\text{SWA}} = 10,000$ throughout mid-training [84].

5. Post-Training

In this section, we introduce a unified post-training recipe for large-scale Reinforcement Learning (RL), which begins with a unified Supervised Fine-Tuning (SFT) model. This framework enables consistent self-improvement by combining verifiable reward signals with human preference feedback, while maintaining stability even during large-scale off-policy training for Mixture-of-Experts (MoE) models. The process follows a two-phase approach similar to prior works [2, 85]. First, we construct **Expert Models** by enhancing the unified SFT baseline with domain-specific RL across Math, Code, STEM, Tool-use, Long Context Understanding, Human Preference, and Agentic Reasoning. These specialized experts are then distilled into a generalist model using **Self-Distillation** and **Scalable RL**, ensuring the final model remains competitive with specialized baselines across diverse tasks. By systematically alternating between targeted specialization and broad synthesis, we achieve robust generalization without compromising expert-level performance.

5.1. Expert Model Construction and Self-Distillation

We employ a two-stage SFT pipeline to build a robust foundation for subsequent RL. The first stage executes large-scale multi-domain SFT spanning Math, Code, STEM, Logic, General QA, Code Agent, Tool-use, Search Agent, and Long Context Understanding. Difficulty-aware filtering and strategic balancing are applied to foster broad agentic behaviors. The second stage explicitly maximizes reasoning density by injecting out-of-distribution (OOD) signals [46, 86], comprising ~30K expert-level chemistry trajectories and synthetic arithmetic tasks. This targeted exposure to distinct reasoning patterns unlocks latent capabilities within just three epochs, equipping the model with the sophisticated structural complexity necessary to initialize the subsequent domain-specific RL phase.

Following domain-specific RL, we consolidate the divergent expert capabilities into a unified student model, initialized from the mid-train checkpoint. In this phase, the expert models generate high-

---

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θFull $= 1,000,000$仅用于全注意力机制，并保持θSWA $= 10,000$，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 θFull $= 1,000,000$，并在 128k（训练中期阶段 2）时增加到 θFull $= 5,000,000$，同时在整个训练中期保持 θSWA $= 10,000$ [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。

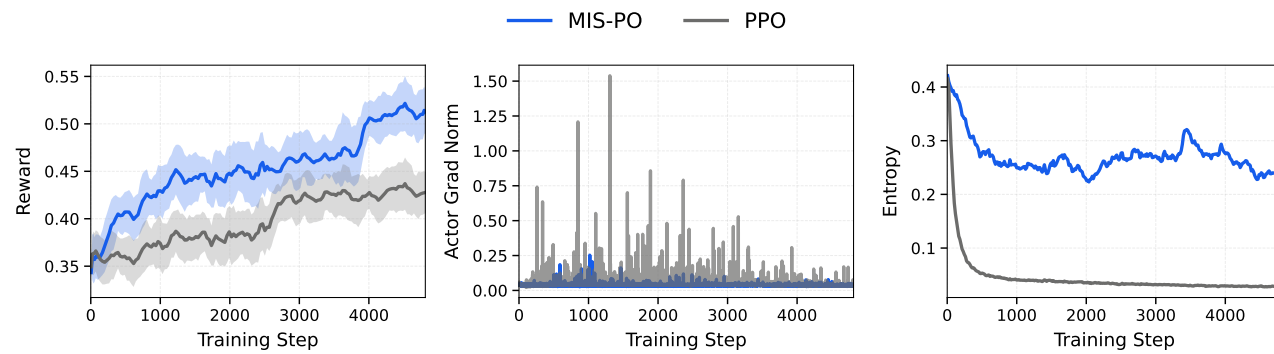遵循特定领域的强化学习，我们将分散的专家能力整合到一个统一的学生模型中，该模型从训练中途的检查点初始化。在此阶段，专家模型生成高

Figure 5: **Scalability comparison between MIS-PO and PPO on our internal model**. **(1) Efficiency**: MIS-PO demonstrates superior sample efficiency, achieving higher reward plateaus with an accelerated convergence trend. **(2) Stability**: MIS-PO significantly stabilizes training dynamics by suppressing gradient noise and eliminating the large spikes in the policy gradient norm. **(3) Exploration Persistence**: MIS-PO exhibits slower entropy decay, enabling a better exploration–exploitation balance.

quality trajectories using a prompt distribution shared with the first-stage SFT corpus, offering a more stable and efficient alternative to direct RL integration. This approach employs rejection sampling to eliminate undesirable patterns such as language mixing or overthinking, centralizing expert knowledge into a single student model. By establishing this high-quality foundation, self-distillation significantly reduces the optimization burden on subsequent RL stages.

Hyper-Parameters. The Muon optimizer [34] is employed with a 3% warmup and a cosine decay from 1.0×10^{-5} to 5.0×10^{-6}. We freeze the MoE router weights and disable the EP-group balance loss similar to mid-training. The SFT training is executed with an MTP loss weight of 0.1, a global batch size of 32, and a global sequence length of 128k. Regarding Rotary Position Embeddings (RoPE) [83], we maintain $\theta_{SWA} = 10,000$ and adjust $\theta_{Full} = 5,000,000$ to accommodate the 128k context length [84].

5.2. Scalable RL

In RL for LLMs, we optimize a policy π_θ to maximize terminal rewards over trajectories $\tau = (s_0, a_0, \ldots, s_T)$, where a_t denotes the token generated at state s_t. For reasoning tasks, however, this process faces severe instability arising from high gradient variance, further amplified by extremely long horizons and model scale (Figure 5 (2)). This variance primarily from **infrastructure divergence** between high-throughput inference engines and training frameworks, as well as the **off-policy misalignment** inherent to iterative updates. In such settings, importance sampling is inherently unstable, as minor token-level probability shifts compound into noisy gradients that impede convergence.

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

To address these stability challenges, we propose MIS-PO, a method inspired by Metropolis Independence Sampling (MIS) [39, 40]. We treat the inference policy as a proposal distribution and the training policy as the target, restricting updates to samples that remain sufficiently close to the target distribution. Unlike importance sampling, which scales gradients by bounded ratios and often suffers from high variance, MIS-PO applies binary masking to filter off-distribution samples and treats retained trajectories as effectively on-policy, resulting in significantly reduced gradient variance and stable optimization.
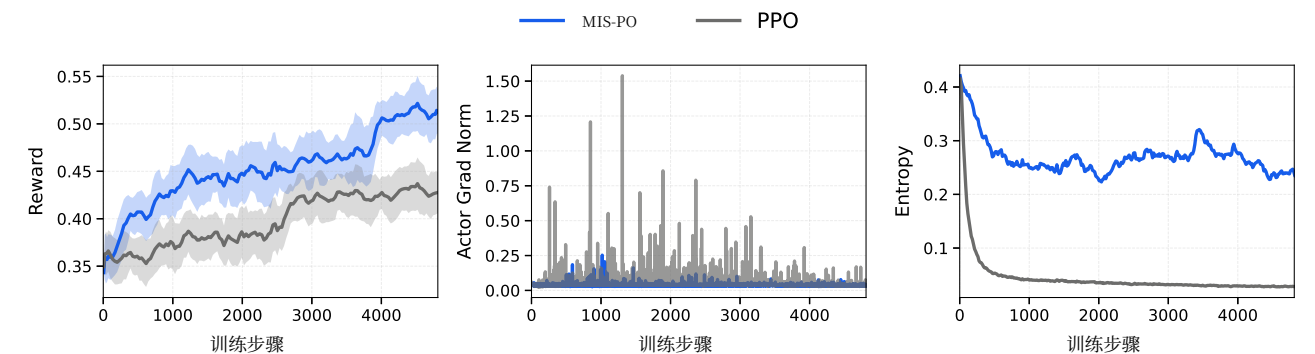
图 5：**在我们的内部模型上 MIS-PO 与 PPO 的可扩展性比较**。**(1) 效率**：MIS-PO 展示了更优的样本效率，以加速收敛趋势实现更高的奖励平台。**(2) 稳定性**：MIS-PO 通过抑制梯度噪声和消除策略梯度范数中的大幅尖峰，显著稳定了训练动态。**(3) 探索持续性**：MIS-PO 表现出更慢的熵衰减，从而实现更好的探索-利用平衡。

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。

Formally, we define a binary indicator function $\mathbb{I}(x) = \mathbb{1}[\rho_{\min} \leq x \leq \rho_{\max}]$ and apply it at two distinct granularities. At the **token level**, the function filters the probability ratio $x_t = \pi_{\theta_{\text{old}}}(a_t|s_t)/\pi_{\theta_{\text{vllm}}}(a_t|s_t)$ to suppress localized mismatches between the training and inference policies [37]. At the **trajectory level**, we apply the same indicator to the geometric mean ratio $\bar{\rho}(\tau) = (\prod_t x_t)^{\frac{1}{T}}$, effectively discarding entire trajectories that have drifted significantly from the target distribution. The reformulated actor loss replaces continuous importance weights with these dual-level discrete masks:

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

By treating valid samples as effectively on-policy, this objective substantially reduces gradient variance for long-horizon reasoning tasks under a trust-region constraint. Figure 5 presents an ablation study over approximately 5,000 training steps, where MIS-PO exhibits significantly lower noise in the actor gradient norm than PPO, indicating improved scalability. More ablations are shown in Appendix D.2.3.

To further stabilize training dynamics, we employ several techniques: **Truncation-Aware Value Bootstrapping** [87] to correct the ambitious reward bias introduced by context-length truncation and **Routing Confidence** monitoring to predict instability specific to MoE architectures.

Truncation-Aware Value Bootstrapping. Assigning zero rewards to context-truncated trajectories conflates truncation with task failure. This ambiguity penalizes long-chain reasoning by failing to distinguish between incomplete and incorrect outcomes. To address this, we replace the zero reward with a bootstrapped value estimate of the final state, effectively treating truncation as a horizon interruption rather than a terminal failure. The modified reward for trajectory τ_i is defined as:

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

Empirically, this truncation-aware value bootstrapping stabilizes training even at truncation rates as high as 20%, preventing the reward degradation typically triggered by incomplete trajectories [88,89]. Ablation studies confirm that this technique is particularly beneficial for competition-level benchmarks, where long-horizon reasoning makes truncation effects most prevalent.

Routing Confidence as a Stability Proxy. Recent studies [36,38] bridge RL stability with MoE routing consistency. Building on this, we propose the **Routing Confidence** (Σ_k) as a proxy for stability, which is the average probability mass of activated experts. Low Σ_k implies high routing uncertainty, which amplifies the training-inference mismatch. Through preliminary experiments, we identify a distinct stability phase transition: models with low routing confidence are brittle and require extreme stabilization (*e.g.*, Router Replay [1,36,38], strict on-policy updates [90]). In contrast, models with high routing confidence maintain robustness, enabling off-policy training without complex interventions.

RL Training Dynamics. To provide a holistic view of our method, we illustrate the RL with verifiable rewards (RLVR) training dynamics and downstream evaluation improvements of Step 3.5 Flash in Figure 6. The steady rise in training rewards suggests a stable and effective learning process. Furthermore, Step 3.5 Flash achieves consistent performance gains across diverse evaluation benchmarks. Specifically, we observe substantial improvements of +3.2% on IMO-AnswerBench [91], +6.1% on CF-Div2-Stepfun-cpp (our custom coding benchmark, described in Appendix E.2.1), +10.6% on ARC-AGI-1 [92], and +3.4% on HLE$_{\text{text}}$ [93].

形式上，我们定义一个二元指示函数 $\mathbb{I}(x) = \mathbb{1}[\rho\min \leq x \leq \rho\max]$，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$_{\text{text}}$ [93]上观察到了 +3.4%的显著改进。
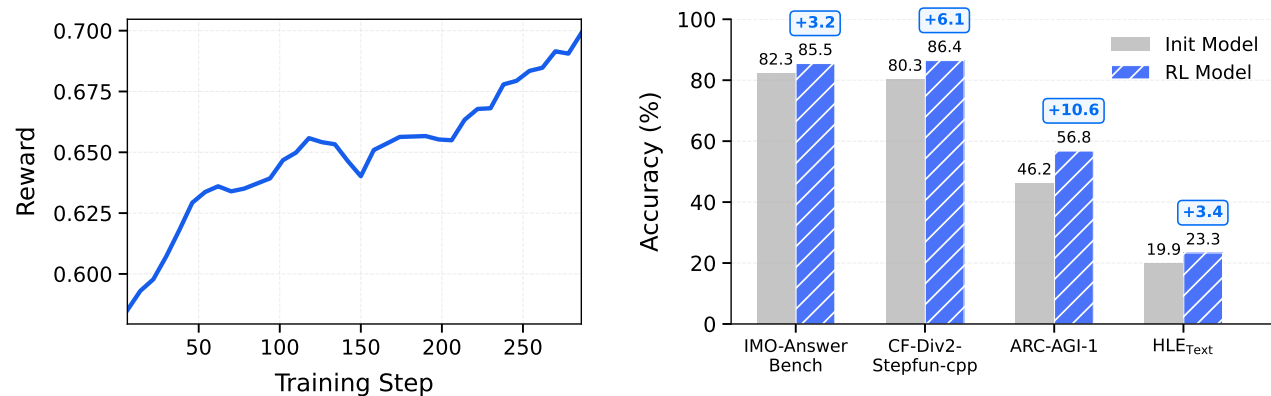
Figure 6: **RL training dynamics and cross-domain improvements of Step 3.5 Flash.** RL drives steady reward growth (left) and delivers consistent accuracy boosts across multiple benchmarks (right).

5.2.2. Reward System

We decouple the RL framework into RL with verifiable rewards (RLVR [94]) and RL with non-verifiable rewards (*e.g.*, RLHF [95]), each supported by a distinct reward tailored to its supervision characteristics.

Verifiable Rewards. For RLVR, each prompt is paired with a task-specific verifier that outputs a reward. The rule-based checkers are used for logic, instruction following, and code, while model-based verifiers are employed for STEM tasks. In ablation studies over 450 RL training steps on our internal model, using model-based verifiers for STEM tasks outperforms direct vanilla math-verify by an average of 2.0%; additional details are provided in Appendix D.2.2.

Non-Verifiable Reward. We address non-verifiable tasks using a pairwise generative reward model (GenRM [96]) that benchmarks responses against a fixed reference. GenRM is a reasoning model that outputs a confidence score indicating the likelihood of a response winning. This score is subsequently converted into a Bradley–Terry win rate [97] to serve as the reward signal. Length control is modeled within GenRM as a confidence score penalty and propagated to the win-rate reward, effectively suppressing excessive length growth during RL training. We further ensure robustness by assigning zero reward to responses with fabricated citations, overconfident claims, or language inconsistencies.

Agent Reward. Search tasks are evaluated using an LLM based on entity-matching scores. For report generation, a rubric-based LLM judge evaluates the research query, rubric specifications, and candidate reports, producing ternary judgments (*satisfied*, *partially satisfied*, *unsatisfied*) [98]. As the intermediate category often misaligns with expert preferences, we map the outputs to asymmetric binary rewards, yielding clearer learning signals and faster convergence toward expert-aligned behaviors.

GenRM Training and MetaRM. We initialize the GenRM by fine-tuning our SFT model with RM-specific prompts. For RL training, we use curated pairwise preference data with a logsigmoid loss similar to the scalar reward model formulation. To improve the robustness of GenRM, we penalize responses exhibiting spurious reasoning (*i.e.*, correct preference derived from flawed logic) by integrating MetaRM, an additional verifier that reduces the training reward when such patterns
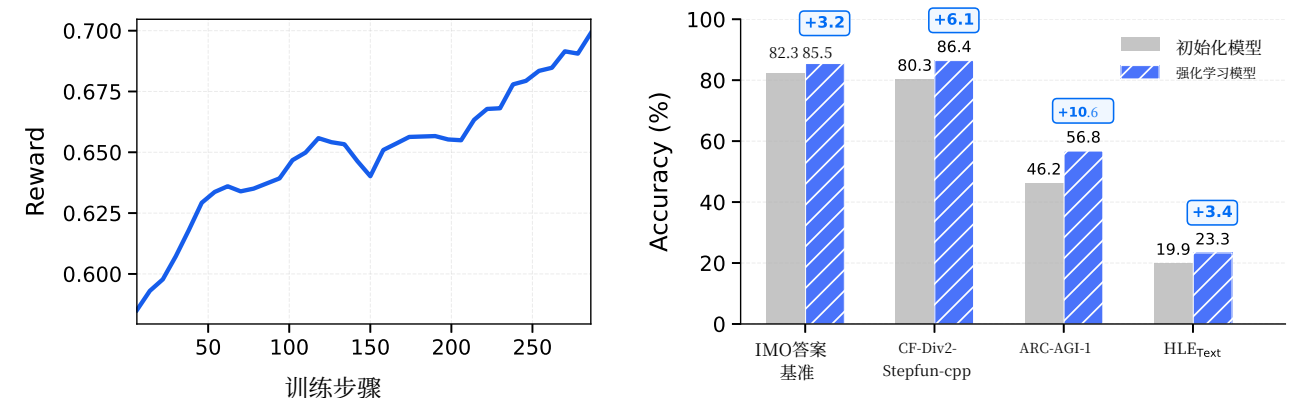
图6: **步骤3.5 Flash的强化学习训练动态及跨域改进**。强化学习驱动持续奖励增长（左图）并在多个基准测试中提供稳定准确率提升（右图）。

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

Domain	Num Samples	Tokens	Corpus Contribution
Math	68055	0.98B	11.19%
Code	86421	1.23B	21.10%
STEM	120399	0.55B	6.31%
Logic	93323	0.81B	13.87%
General	314495	0.80B	9.16%
Code Agent	37240	0.90B	17.70%
Tool-use	114507	0.76B	8.72%
Search Agent	20256	0.50B	8.75%
Long Context	15565	0.70B	4.00%
Total	870687	7.23B	100.00%

Table 3: Data Statistics of first-stage SFT.

are detected. In ablation studies spanning 200 RL training steps on our internal model, MetaRM-augmented GenRM outperforms vanilla GenRM by 0.5% - 3% on every benchmark.

5.2.3. Hyper-Parameters

For rollout, we set both the sampling temperature and top-p to 1.0 with a maximum sequence length of 128k tokens. Per generation, we sample 256 unique prompts with 16 responses each for reasoning tasks, 512 unique prompts with 8 responses each for human preference tasks, and 128 unique prompts with 8 responses each for tool-use tasks. After rollout, completed samples are partitioned into mini-batches and used for training over a single epoch, with 4 mini-batches for the actor and 12 mini-batches for the critic. Optimization is performed using the Muon optimizer with a weight decay of 0.1. The actor is trained with a learning rate of 2×10^{-6} and 20 warmup steps, while the critic uses a learning rate of 5×10^{-6} with 50 warmup steps. Following ORZ [90], we set both γ and λ to 1. We further adopt an unbiased KL loss [85] with a coefficient of 0.001 in the final stage. For Equation (2), the token-level and trajectory-level masking bounds are set to $[0.5, 2]$ and $[0.996, 1.001]$, respectively.

5.3. Data Synthesis & Curation

We construct a diverse and difficulty-balanced prompt pool by aggregating open-source data, synthetic generations, and user trajectories. A unified synthesis and curation pipeline is applied, combining strict global filtering with domain-specific refinement to maximize reasoning density. Data quality is ensured through a hybrid of rule-based heuristics and model-based fidelity checks. The resulting dataset contains 871k samples (7.23B tokens), with detailed statistics summarized in Table 3.

5.3.1. General and Reasoning

Our training corpus aggregates community prompts, expert responses, and synthetic data from diverse open-source, including **Mathematics** [90, 99–110], **Coding** [111–113], and **Science and Open-ended QA** [114–117]. To maximize reasoning density, we employ a unified pipeline that couples strict global filtering with domain-specific refinement, enforcing quality via a hybrid of rule-based heuristics and model-based fidelity checks. Specifically, in mathematics, we ensure numerical stability through specialist-guided rejection sampling and synthetic large-number arithmetic. For programming, we prioritize offline executability by selecting rigorous algorithmic challenges while strictly purging RAG-related hallucinations. In particular, we mitigate the model's tendency to falsely claim access to

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。

external search engines or pretend to retrieve online solutions. Furthermore, we restrict scientific data to unambiguous questions with unique, determinable solutions.

To enable generalization across practical scenarios, we expand open-source checkers[2] and augment samples with several real-world constraints. In parallel, we collect general prompts from open-source, synthetic, and user trajectories to form a diverse, difficulty-balanced pool. This process yields a high-fidelity dataset comprising millions of samples at the billion-token scale.

5.3.2. Generalized Tool Learning

We propose an execution-driven data generation framework for learning reliable tool-use behaviors in intelligent agents, addressing key limitations of existing synthetic pipelines such as data inconsistency, lack of verifiability, and model hallucinations. Instead of relying on random exploration [118, 119] or model-based simulation [5, 120], our approach decomposes tool-use behavior into atomic intents and models them using a finite state machine (FSM), explicitly separating abstract tool-call logic from parameterized execution constraints. Data is generated through a sample–execute–verify loop with rejection sampling, where all candidate trajectories are executed in real environments and validated by deterministic feedback, ensuring fidelity and eliminating hallucinated behaviors. By compositionally combining atomic intents, the framework supports scalable generation of complex, controllable tool-use scenarios. Using this paradigm, we construct over 100K high-quality trajectories totaling billions of tokens, providing precise supervision for tool-based planning, reasoning, and execution.

5.3.3. Code Agents

Code agents can self-improve through a closed-loop intervention between verifiable **environment construction** and **solution generation**, where executable feedback continuously refines both capabilities. We treat environment construction as a first-class capability alongside bug fixing and feature implementation, synthesizing it under verifiable reward signals. To this end, we develop a specialized agentic pipeline evolved from the SWE-factory [121] framework, incorporating a cross-task memory pool that retrieves historical build successes as few-shot demonstrations and a loop-detection mechanism to prevent redundant exploration. This pipeline achieves a 40% environment-building success rate, forming a positive feedback loop for model self-evolution through dense supervision from construction trajectories, including shell commands and error recovery. To further improve signal quality, we normalize environment construction trajectories by abstracting and masking transient failures and redundant execution patterns that do not contribute to the final resolution. The bootstrapped environments function as dynamic testbeds, leveraging execution feedback and unit tests to generate high-quality synthetic data and reward signals for continuous alignment. Empirically, we observe a bidirectional transfer: construction expertise accelerates coding performance, while coding within these environments further improves construction accuracy, as shown in DockSmith [122]. Leveraging this evolution pipeline, we curate 50K verified environments spanning over 15K GitHub repositories and more than 20 programming languages. This diverse collection captures a broad spectrum of real-world scenarios, providing a robust foundation for training generalist code agents.

5.3.4. Search and Research Agents

To facilitate advanced information-seeking, our pipeline integrates graph-based and multi-document synthesis to enforce multi-hop reasoning. By performing topological expansions on knowledge graphs (*e.g.*, Wikidata5m [123]) and simulating cross-website browsing trajectories, we generate data that reflects real-world research complexity. Crucially, to guarantee the necessity of external retrieval, we

---

---

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

validate generated queries against DeepSeek-R1 [124], systematically excluding instances solvable by this strong reasoning model without tool interaction. The resulting trajectories are refined through a structured report generation pipeline [125] that enforces rigorous instruction compliance and structural integrity. Specifically, we enforce strict adherence to preset research plans, discarding any trajectories that deviate from the structure. Subsequently, valid outputs undergo iterative cleaning via model-based judgers and heuristic rules to resolve fine-grained issues such as informal writing, temporal hallucinations, and mixed-language artifacts. This end-to-end approach achieves industry-leading performance on the RESEARCHRUBRICS [21] benchmark.

5.4. Agent Infrastructure

Reasoning with Tool-Use Template Design. To effectively integrate reasoning and agentic capabilities into a single foundation model, it is crucial to determine the appropriate templates for the thinking process and tool usage. Regarding the reasoning template, we evaluate three management strategies. The approach of discarding reasoning history at every turn [124], while incentivizing independent generation, leads to task failure in long-horizon tasks(*e.g.*, coding sessions exceeding 100 turns). Conversely, retaining the full reasoning history incurs prohibitive context consumption, which rapidly saturates the model's capacity and blocks subsequent tool invocations. To resolve this, we adopt a selective retention strategy: preserving reasoning traces exclusively for the tool-use trajectory triggered by the most recent user instruction. This design achieves an optimal trade-off between reasoning coherence and context efficiency, a practice aligned with recent frontier models [85, 126]. Regarding the tool-use template, we compared the prevalent JSON and XML formats. The rigid syntax of JSON, including escape sequences and delimiters, frequently induces parsing errors in small, under-trained models. In contrast, the XML format allows for flat string output with significantly lower grammatical overhead. Therefore, we select the XML format to ensure robustness in complex, real-world agentic coding scenarios.

Scalable Code Agent Infrastructure. Our integrated architecture focuses on scalable session management and cross-framework generalization to facilitate high-throughput agentic coding. Central to this is a proprietary Session-Router that orchestrates container lifecycles via Kubernetes and ensures interaction consistency through Tmux. This architecture supports thousands of concurrent environments with seamless state persistence, eliminating the need for manual, scaffold-specific Docker configurations. To ensure high generalization across diverse agentic workflows, we trained the model to adapt to a wide spectrum of interaction frameworks, ranging from academic standards (*e.g.*, Open-Hands [127], SWE-agent [128], and Terminus-2 [16]) to enterprise grade protocols (*e.g.*, Kilocode [129], Roocode [130], and ClaudeCode [131]). By exposing the model to these varied interaction paradigms during training, we effectively prevent it from overfitting to specific pipeline patterns, ensuring it remains robust regardless of the underlying execution environment.

6. Evaluations

6.1. Pre-training Evaluations

Evaluation Setup. We evaluate Step 3.5 Flash on a series of benchmarks, encompassing various capabilities: (1) General language understanding and reasoning, including BBH [132], MMLU [133], MMLU-Redux [134], MMLU-Pro [135], HellaSwag [136], WinoGrande [137], GPQA [138], SuperG-PQA [139], and SimpleQA [140]. (2) Mathematics reasoning, including GSM8K [141] and MATH [142]. (3) Coding, including HumanEval [143], MBPP [144], HumanEval+, MBPP+ [145] and MultiPL-E [146]. (4) Chinese understanding, including C-Eval [147], CMMLU [148], and C-SimpleQA [149].

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 R E S E A R C H R U B R I C S [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]。

Benchmark	# Shots	Step 3.5 Flash Base	MiMo-V2 Flash Base	GLM-4.5 Base	DeepSeek V3.1 Base	DeepSeek V3.2 Exp Base	Kimi-K2 Base
# Activated Params	-	11B	15B	32B	37B	37B	32B
# Total Params	-	196B	309B	355B	671B	671B	1043B
GENERAL							
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
MATHEMATICS							
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
CHINESE							
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

Table 4: Pre-training evaluation results. * denotes cases where the original score was unavailable; we report results evaluated under the same test conditions as Step 3.5 Flash for fair comparison. † indicates Deepseek scores quoted from the MiMo-V2-Flash report [30].

Evaluation Results. Table 4 summarizes the pre-training evaluation of Step 3.5 Flash across general reasoning, mathematics, code, and Chinese benchmarks. Despite activating only 11B parameters (196B total), Step 3.5 Flash remains broadly competitive with substantially larger sparse baselines (15–37B activated; 309–1043B total), demonstrating a strong accuracy–efficiency trade-off. On core general benchmarks, Step 3.5 Flash achieves 88.2 on BBH (within 0.5 of the best) and 85.8 on MMLU. Notably, Step 3.5 Flash reaches 31.6 on SimpleQA, outperforming DeepSeek-V3.2-Exp Base (27.0) while using only 196B total parameters versus 671B (i.e., ~3.4× total parameters), highlighting stronger capability density per parameter budget. Step 3.5 Flash further demonstrates strong coding capabilities, including 81.1 on HumanEval, 67.7 on MultiPL-E HumanEval and 58.0 on MultiPL-E MBPP. Overall, these results show that Step 3.5 Flash delivers high strong performance per activated compute, providing a solid foundation for downstream reasoning and agentic post-training.

6.2. Post-training Evaluations

We evaluate Step 3.5 Flash on representative benchmarks, including the reasoning oritend HLE (text subset) [150], MMLU-Pro [135], GPQA-Diamond [138], AIME2025 [10], HMMT [11], IMO-AnswerBench [91]; the coding related LiveCodeBench-v6 (2024.08-2025.05) [12], CF-Div2-Step (Ap-

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): our custom CodeForces[3] Div2 Benchmark, SWE-Bench Verified [13] and SWE-Bench Multilingual [14]; the agent series τ^2-Bench [15], Terminal-Bench 2.0 [16], GAIA [19], BrowseComp [17], xbench-DeepSearch [20], BrowseComp-zh [18], and RESEARCHRUBRICS [21]; the general related ArenaHard v2 [151], IFBench [152] and MultiChallenge [153]; and the long-context related LongBench v2 [154], MRCR [155] [4], FRAMES [156] and RepoQA [157].

We further investigate the test-time scaling properties of Step 3.5 Flash on reasoning, general, and long-context benchmarks by adopting the **Parallel Coordinated Reasoning (PaCoRe)** paradigm [42]. Leveraging Step 3.5 Flash's extreme inference efficiency, this approach decouples reasoning capacity from context limitations by launching parallel reasoning trajectories and synthesizing their insights into higher-fidelity solutions via multi round coordination. Specifically, we employ a multi-round PaCoRe trajectory configuration as $\vec{K} = [4,4,4,4]$, yielding significant gains across benchmarks.

We maintain a maximum sequence length of 256k, using the default decoding configuration with decoding temperature and top-p of 1.0. And we apply YaRN [158] with a scaling factor of 2.0 on top of the original 128k positional embeddings, restricting it to full-attention layers only. We report pass@1 accuracy for all approaches based on average performance of multiple independent generations per problem: 64 for AIME 2025, HMMT 2025 Feb., and HMMT 2025 Nov.; 8 for IMO-AnswerBench, LiveCodeBench, GPQA-Diamond, and MultiChallenge; 1 for HLE and 4 runs for all other benchmarks. More details are provided in Appendix E.2.

Evaluation Results. Table 5 presents a comprehensive comparison of Step 3.5 Flash against a broad set of strong baselines across reasoning, code agents, general agents, long-context understanding, and general capability benchmarks. Despite activating only 11B parameters (196B total), Step 3.5 Flash demonstrates strong performance across a wide range of tasks, particularly excelling on reasoning-intensive benchmarks such as AIME 2025, HMMT 2025 Feb., HMMT 2025 Nov., IMO-AnswerBench, and LiveCodeBench-v6. It consistently outperforms open-source models with larger parameter counts and achieves performance on par with frontier models such as GPT-5.2 xHigh and Gemini 3.0 Pro. Notably, Step 3.5 Flash achieves strong results on agentic evaluations, including SWE-Bench Verified, Terminal-Bench 2.0, BrowseComp (with Context Manager), GAIA, and τ^2-Bench, highlighting robust tool-use and long-horizon decision-making capabilities. Overall, these results indicate that Step 3.5 Flash delivers strong agentic and reasoning performance with substantially lower active parameter counts.

7. Discussion and Limitations

Token Efficiency. Step 3.5 Flash achieves frontier-level intelligence but currently requires longer generation trajectories than Gemini 3.0 Pro to reach comparable quality. Next step we will prune and compress the thinking for better efficiency while maintaining the same competitive performance.

Efficient Universal Mastery. We aim to unify generalist versatility with deep domain expertise. To achieve this efficiently, we are advancing variants of on-policy distillation, allowing the model to internalize expert behaviors with higher sample efficiency.

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

---

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并行协调推理（PaCoRe）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

Table 5 (English, left column):

Benchmark	Step 3.5 Flash		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
	Vanilla	PaCoRe								
# Activated params	11B		10B	15B	32B	37B	32B	-	-	-
# Total params	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 Feb.	98.4	100.0	71.0*	95.4*	97.1	92.5	95.4	97.5†	92.9†	99.4
HMMT 2025 Nov.	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†
CODE AGENT										
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†
GENERAL AGENT										
BrowseComp	51.6	-	47.4	45.4	52.0	51.4	60.6	37.8†	37.0†	-
BrowseComp (w. Ctx Manage)	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*
RESEARCHRUBRICS	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*
GENERAL										
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*

Table 5: Comparison between Step 3.5 Flash and closed/open models. * denotes cases where the original score was unavailable or inferior to our reproduced result; we therefore report results evaluated under the same test conditions as Step 3.5 Flash for fair comparison. † indicates scores quoted from non-official sources, including technical reports, or independent evaluation platforms. Our evaluation on HLE focuses on the text-only subset. BrowseComp (w. Ctx Manage) denotes the evaluation of BrowseComp with a Context Management enabled.

RL for Open-World Agentic Tasks. While Step 3.5 Flash demonstrates competitive performance on academic agentic benchmarks, the next frontier of agentic AI necessitates the application of RL to intricate, expert-level tasks found in professional work, advanced engineering, and scientific research. Solving these challenges is a prerequisite for deploying agents capable of genuine autonomy.

Operational Scope and Constraints. Step 3.5 Flash is tailored for coding and work-centric tasks, but may experience reduced stability during distribution shifts. This typically occurs in highly specialized domains or long-horizon, multi-turn dialogues, where the model may exhibit repetitive reasoning, mixed-language outputs, or inconsistencies in time and identity awareness.

Table 5 (Chinese, right column):

基准测试	Step 3.5 Flash		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
	Vanilla	PaCoRe								
# 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。

Appendix

A. Architecture Details

Table 6 summarizes key architecture hyper-parameters of Step 3.5 Flash.

Hyper-Parameter	Value
BACKBONE	
Vocabulary size (V)	128,896
Model width (d_{model})	4096
Transformer blocks	45 (3 dense + 42 MoE)
MoE FFN	
Experts per MoE block	288 + 1 shared
Routing	top-k = 8
Dense FFN hidden size	11,264
MoE expert hidden size	1,280
ATTENTION	
Hybrid block structure	3 SWA blocks + 1 full attention block
SWA window size	512
KV heads (GQA)	8
Query heads (full / SWA)	64 / 96
Gate Type	head-wise on output
Head dimension	128
RoPE θ	10,000
RoPE dims (full / SWA)	64 / 128
MULTI-TOKEN PREDICTION	
MTP blocks	3 (Dense SWA)
PARAMETER COUNTS	
Total params (backbone)	196B
Activated params / token (backbone)	11B
Total params (with MTP3)	198B
Activated params / token (with MTP3)	13B

Table 6: Key architecture hyper-parameters of Step 3.5 Flash. "Activated params" are reported per token and exclude embedding/output matrices.

A.1. Head-wise Gated Attention

Each attention head is assigned a lightweight, input-dependent scalar gate, allowing the model to dynamically modulate information flow across the hybrid layout with negligible computational overhead.

Formally, for a (single) head of dimension d, let $q_i, k_j, v_j \in \mathbb{R}^d$ denote the query vector at position i and the key and value vectors at position j, the scaled dot-product scores s, the corresponding attention weights α and the outputs y are computed as follows:

$$s_{i,j} = \langle q_i, k_j \rangle / \sqrt{d}, \qquad Z_i = \sum_{j'} \exp(s_{i,j'}), \qquad \alpha_{i,j} = \exp(s_{i,j}) / Z_i, \qquad y_i = \sum_j \alpha_{i,j} v_j. \qquad (4)$$

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 ($dmodel$)	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 $q_i, k_j, v_j \in \mathbb{R}^d$ 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 y 按照如下方式计算：

$$s_{i,j} = \langle q_i, k_j \rangle / \sqrt{d}, \qquad Z_i = \sum_{j'} \exp(s_{i,j'}), \qquad \alpha_{i,j} = \exp(s_{i,j}) / Z_i, \qquad y_i = \sum_j \alpha_{i,j} v_j. \qquad (4)$$

Given the input representation x_i at position i, we compute a head-wise gate g_i to modulate the head output:

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

where $\sigma(\cdot)$ is the sigmoid function and $\boldsymbol{w}_{gate}$ is a learnable vector.

Head-wise gated attention can be viewed as introducing an *input-dependent sink token* [33] into the attention mechanism. Substituting $\sigma(g) = \frac{1}{1+\exp(-g)}$ into Equation 5, we have

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

where $\exp(-g_i)Z_i$ acts as an *input-dependent sink mass* in the softmax normalizer. As shown in Section 2.3, this adaptive formulation consistently outperforms fixed (input-independent) sink tokens.

A.2. Speed Benchmark of Attention Enhancements

We conduct simulations with MTP-3 to evaluate the latency overheads of the two enhancements under an ideal workload. Table 7 presents the relative increment of theoretical FLOPs and latency. Increasing the number of query heads in SWA slightly raises the FLOPs but has less impact on latency. This is due to a query-to-kv ratio of 12, which keeps SWA in the IO-bound region, even when considering MTP-3. For head-wise gating, neither FLOPs nor latency has noticeable difference because of its lightweight.

Backbone	SWA Heads	Setting	Decode (FLOPs / Lat.)		Prefill (FLOPs / Lat.)	
			64k	256k	64k	256k
Step 3.5 Flash	64	no gate	1.00 / 1.00	1.00 / 1.00	1.00 / 1.00	1.00 / 1.00
(*S3F*1 layout)	96	no gate	1.02 / 1.01	1.01 / 1.00	1.08 / 1.06	1.04 / 1.03
	64	head-wise	1.00 / 1.00	1.00 / 1.00	1.00 / 1.02	1.00 / 1.01
	96	head-wise	1.02 / 1.02	1.01 / 1.00	1.08 / 1.08	1.04 / 1.05

Table 7: Relative increment under different SWA head counts and gating strategies. The metrics are presented as FLOPs / Latency. The baseline configuration (first line) is normalized to 1.0.

Backbone	Layout	SWA Heads	Decode		Prefill	
			64K	256K	64K	256K
Step 3.5 Flash	*S3F*1	64	1.00	1.00	1.00	1.00
	*S3F*1+Head	96	1.02	1.01	1.08	1.04
	*S1F*1	64	1.18	1.47	1.38	1.71
	FFFF	64	1.51	2.33	2.07	3.00
Internal 30B-A3B	*S3F*1	32	1.00	1.00	1.00	1.00
	*S3F*1+Head	48	1.02	1.01	1.05	1.02
	*S1F*1	32	1.42	1.74	1.50	1.80
	FFFF	32	2.21	3.16	2.47	3.34

Table 8: Relative FLOPs cost across different backbones and attention patterns. The head count refers to SWA heads. For each backbone, the configuration with minimum FLOPs (*S3F*1 with reduced heads) is the baseline (1.0).

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

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

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	64	无门	1.00 / 1.00	1.00 / 1.00	1.00 / 1.00	1.00 / 1.00
(*S3F*1 布局)	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	*S3F*1	64	1.00	1.00	1.00	1.00
	*S3F*1+Head	96	1.02	1.01	1.08	1.04
	*S1F*1	64	1.18	1.47	1.38	1.71
	FFFF	64	1.51	2.33	2.07	3.00
内部30B-A3B	*S3F*1	32	1.00	1.00	1.00	1.00
	*S3F*1+Head	48	1.02	1.01	1.05	1.02
	*S1F*1	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（*S3F*1 减少头部）的配置是基线（1.0）。

A.3. Meta Token

Recent literature [159–161] has shown both theoretically and empirically that prepending structured metadata to pretraining sequences can improve data efficiency and accelerate convergence: by exposing high-level attributes (*e.g.*, modality, language, domain), metadata provides global cues that reduce uncertainty about the upcoming content and thus makes next-token prediction easier.

Motivated by this paradigm, we associate each training example with a metadata string $\mathbf{M}$ in a human-readable format, including content type (*e.g.*, Code, Book, Paper, Web), language (*e.g.*, EN, ZH), domain, and source. We then prepend $\mathbf{M}$ to the original token sequence $\mathbf{x}$, forming a single training sequence $\mathbf{s} = [\mathbf{M}; \mathbf{x}]$. During pretraining, the model is trained to maximize the likelihood of $\mathbf{s}$:

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

After an initial phase of approximately 3.8T tokens, we keep $\mathbf{M}$ in the context but mask out its positions from the loss while continuing to predict the payload tokens:

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

We hypothesize that by this stage the model has already learned to effectively use metadata as a conditioning signal. Masking the metadata loss therefore allocates optimization pressure entirely to the payload tokens, while still benefiting from the explicit conditioning on data characteristics.

A.4. Pretraining Ablations Details

We conduct controlled pre-training ablations to isolate the effects of (i) different hybrid attention layout and (ii) sink tokens versus head-wise gated attention.

Hybrid attention layout. We adopt a **30B-A3B MoE** architecture to evaluate the downstream impact of different hybrid attention layout under a fixed token budget. The training follows a strict, multi-stage pipeline: a 30B-token warmup phase, followed by 1T tokens of main pretraining, a 300B-token cooldown phase, and an additional 100B-token long-context specialization stage—totaling approximately 1.4T tokens. Supervised fine-tuning (SFT) is then performed on a 0.1× downsampled dataset. Full training details are provided in Table 9.

Gate vs. sink (scaled setting). We pre-train a **100B-A10B MoE** model for ~250B tokens to compare sink tokens and head-wise gating under a larger-scale regime.

Pretraining results of the architectural ablations are presented in Tables 2 and 10. We employ the evaluation protocols detailed in Section 6.1. Specifically, GPQA [138] is evaluated using 5-shot prompting, while HumanEval [162] and MBPP [163] utilize 3-shot prompting.

The post-training results in Table 1 are aggregated as follows:

- **Reasoning:** The average of MMLU-Pro [135], GPQA-Diamond [138], LiveCodeBench v6 [12], and LiveBench [164].
- **Math:** The average of AIME 2024 [165], AIME 2025 [166], HMMT 2025 Feb. [167], and CNMO 2024[5].
- **Code:** The average of CF-Div2-Stepfun and LiveCodeBench v6 [12].

---

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

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`30B-A3B MoE`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`100B-A10B MoE`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

Hyper-Parameter	100B-A10B	30B-A3B
Total Tokens	250B	1.4T
Optimizer	Muon [34]	
Peak learning rate	1.31×10^{-4}	1.1×10^{-3}
Batch-size warmup	-	First 30B tokens
Layers	43	48
Dimension	4096	2048
Leading Dense Layers	1	1
Routed Experts	96	128
Active Experts	4	8
Shared Experts	1	1
Load Balancing Method	Loss Free [64]	
Attention module	GQA8	
Sequence Length	4096	
Vocab Size	129280	
Batch Size	8192	16384
Weight Decay	0.1	
Partial RoPE	Disabled	Enabled
MTP	Enabled	Disabled

Table 9: Training configuration for the 100B-A10B and the 30B-A3B architecture ablation suite.

- **Sci:** Represented by GPQA-Diamond [138].
- **General:** The average of IFEval [168], IFBench [152], WildBench [169], Arena-Hard [151], and MultiChallenge [153].
- **LongCtx:** The average of six benchmark-level averages: (i) the average score across context lengths 8k-128k on RULER [170], (ii) the average score over the Short and Medium subsets of Long-Bench v2 [154], (iii) the average score across context lengths 8k-128k on HELMET [171], (iv) GSM-Infinite [172], (v) the overall score on FRAMES [156], and (vi) the overall score on RepoQA [157].

Tables 1 and 10 show that the vanilla $S3F1$ layout underperforms the full-attention baseline on general pretraining benchmarks and consistently degrades SFT quality (*e.g.*, BBH: −4.3; SFT Avg: −0.7). Increasing the number of SWA query heads substantially closes this gap (e.g., MMLU-Pro: +3.7; SFT Reasoning: +0.4), with only a minor regression on SFT Code (−0.6), while matching or exceeding the full-attention baseline on several metrics. Table 2 further demonstrates that head-wise gated attention yields an average improvement from 62.5 to 64.4 (+1.9) on the sink token metric.

B. Detail Analysis of Localized Activation Blow-up

To investigate the root cause of the localized activation blow-up, we analyze the tokens that trigger the largest expert activations across all layers, and identify two distinct large activation patterns: (1) Specific lexical items, such as special tokens and punctuation, commonly elicit large but not dramatic activations, particularly in the shallower layers. This pattern is not recognized as a failure mode by us, as there is no rapid increment and it may serve as an internal mechanism for semantic

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

Layout	SWA Heads	Pre-train Evaluation											
		BBH	MMLU	MMLU-Redux	MMLU-Pro	SimpleQA	GSM8K	MATH	HumanEval	MBPP	C-EVAL	CMMLU	Avg.
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**

Table 10: Pre-training evaluation results for hybrid attention layout ablations (*W*=512) on 30B-A3B. *F* denotes full attention and *S* denotes SWA; *S3F1* indicates three *S* and one *F* in the hybrid layout. *S3F1*+Head increases the number of SWA heads from 32 to 48.

modeling [60,173]. Another pattern is that (2) some high-frequency bi-grams trigger extremely large activations on the first token, which represents the failure mode we are investigating. The pattern is triggered by several factors: The frequency of a bi-gram's occurrence is sufficiently high, and the MoE FFN is fine-grained enough, allowing an expert to specialize in that bi-gram without being regulated by the load balancing mechanism. This specialization serves as a shortcut: once the expert is activated, the output becomes deterministic, and other networks no longer influence the prediction. While finding shortcuts is a reasonable approach to minimizing loss, in a MoE model with a pre-norm architecture [76,77], there is a straightforward, pathological solution for achieving such deterministic predictions, as outlined next. The model's final representation is the sum of the outputs from all layers, followed by a RMSNorm. This can be expressed as a combination of the outputs from the experts and the attention layers:

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

where attn, MoE, expert represent the output hidden states of their respective modules, while *L* and *E* denote the number of layers and experts, respectively. The straightforward solution is to boundlessly enlarge expert$_{\text{outlier}}$, then

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

where we decouple $\boldsymbol{h}_{\text{outlier}}$ to the magnitude *c* and the unit vector $\hat{\boldsymbol{h}}_{\text{outlier}}$ denoting the direction.

SwiGLU [78], the expert architecture in Step 3.5 Flash, provides a way to generate large outputs, even when the weight decay effectively suppresses the weight norms. SwiGLU is defined as follows:

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

We analyze the activation norms of $W_{\text{gate}}\boldsymbol{x}$ and $W_{\text{up}}\boldsymbol{x}$ and find no significant differences between outlier experts and normal experts. However, the element-wise product produces abnormal outputs, which have

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

in outlier experts. It can be achieved only if SiLU($W_{\text{gate}}\boldsymbol{x}$) and $W_{\text{up}}\boldsymbol{x}$ are highly aligned and concentrate on a very limited number of dimensions. Consequently, only a limited number of rows from W_{up} are utilized due to the sparse input. This observation leads us to prefer activation clipping over weight clipping, as activation's numerical property directly contribute to the blow-up and the sparsity, and activation clipping can promptly address these issues. Besides, activation clipping has negligible negative effects, as well-behaved activations rarely exceed the threshold.

When using the Muon optimizer, gated linear units, such as SwiGLU, are susceptible to logit explosion. This vulnerability arises from similar mechanisms that cause explosion in attention, as reported in [5]. For an outlier expert specialized to some specific bi-gram, hidden states routed to it are expected to be closely aligned to its router embedding. We validate this by inputting the router embedding into a outlier expert and directly predicting outputs based on this expert's output. The predicted distribution aligns with that of the real data and the entire network's performance. Combined with the overly single training target (to predict the second token in the bi-gram), we argue that gradients w.r.t. the outlier expert's parameters, W_{gate}, W_{up} and W_{down}, are not only abnormally low rank (denoted as r), but also consistently point in a direction that emphasizes the magnitude as analyzed in the first factor, without rotation. Let the update matrices of a parameter matrix $W \in \mathbb{R}^{N \times N}$ to be

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

Accumulating updates over optimization steps will rapidly increase the singular value of the low-rank signals, resulting in an explosion of the weight parameter. In the GLU structure, $\|\text{SiLU}(W_{\text{gate}}\boldsymbol{x}) \cdot W_{\text{up}}\boldsymbol{x}\|$ squares the spectral norm in our strong alignment case, making the progress more sharp. Additionally, Muon completely eliminates the influence of gradient magnitudes. During the blow-up process, RMSNorm reduces the gradients of large inputs. When using the Adam optimizer, its ϵ acts as a threshold to filter out small gradients during the learning rate adaptation, which can hinder the progress. In contrast, Muon consistently and effectively orthogonalizes the gradients, resulting in more aggressive updates.

C. Step Pretraining Data Foundation

C.1. Knowledge Data Construction

C.1.1. StepCrawl

Beyond standard web-scale datasets (*e.g.*, CommonCrawl), we develop **StepCrawl**, an in-house crawling and curation system designed to acquire *high-quality* and *diverse* tokens at scale. StepCrawl serves as a primary data source for both high-signal web pages and document-like content (notably PDFs), which frequently contain long-form, high-information-density material.

A key component of StepCrawl is a site and URL selection layer powered by a WebOrganizer-style model [174]. We adapt the capabilities introduced in WebOrganizer and further fine-tune a version tailored to our pipeline. During crawling, each fetched web page is analyzed by this model, forming a lightweight LM-in-the-loop feedback cycle that (i) filters SEO-driven and other low-utility pages, and (ii) guides crawl-budget allocation by balancing site categories (e.g., preventing disproportionate crawling of tool and e-commerce sites) to preserve corpus diversity and reduce topical skew. In practice, StepCrawl processes on the order of ~1B pages per day under this quality- and diversity-aware scheduling policy.

All crawling activities strictly adhere to `robots.txt` and site-specific access policies. The collected content is subsequently passed through a multi-stage filtering process (quality scoring, deduplication, and sanitization), ensuring that only high-utility and policy-compliant data are retained for training.

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（W_{gate}、W_{up}和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 $W \in \mathbb{R}^{N \times N}$ 的更新矩阵为

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

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，$\|\text{SiLU}(W_{\text{gate}}\boldsymbol{x}) \cdot W_{\text{up}}\boldsymbol{x}\|$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. Quality Refinement and Stratification

Quality stratification. Inspired by Nemotron-CC [175]-style quality bucketing, we divide the internal web data into quality tiers and sample preferentially from higher tiers. We label each document using an ensemble of six lightweight scorers/classifiers and ensemble the tier assignments across scorers. In the final recipe, we keep High/Medium-High/Medium and discard Medium-Low/Low, which substantially improves token efficiency in ablations. For book and paper corpora, we apply the same stratification but restrict retention to High/Medium-High tiers exclusively during the annealing stage to maximize diversity. In addition to the shared six-scorer ensemble, we integrate additional domain-specific filters targeting STEM and knowledge-dense content, and down-sample overrepresented domains to ensure balanced representation.

Embedding-based cluster rebalancing. We leverage embedding-based corpus balancing as a principled way to further reduce redundancy and mitigate distribution skew. Specifically, we embed large-scale Chinese/English web data, run k-means clustering (100K+ clusters), and down-sample clusters with disproportionate mass. In ablations, this cluster-level rebalancing in the cooldown stage improves a broad set of benchmarks.

Knowledge-Intensive Mining and Augmentation. We construct a dedicated knowledge subset using a lightweight two-stage pipeline built on the shared embedding representation described above. First, a curated inventory of high-value entities, concepts, and relations is used to retrieve knowledge-dense documents and passages from the full corpus in embedding space; these candidates are ranked by a knowledge-density model and simple coverage heuristics. Second, for a portion of the retrieved content, we apply targeted transformations such as controlled rephrasing and QA synthesis to improve learnability. The resulting samples are mixed back into the training mixture to increase effective knowledge signal density. We observe consistent gains from this pipeline in ablations, while a detailed causal analysis of its benefits is left for future work.

C.2. Code Data

C.2.1. Pure-Code

We refine our internal programming dataset using a modified version of the OpenCoder filtering rules [80], introducing a calibrated relaxation to balance data quality and diversity. In our pipeline, applying OpenCoder filters generates a set of "hits" for each document, where each hit represents a violation of a heuristic rule (signaling potential noise). We categorize the corpus by these hit counts: `hit0` for clean documents (zero violations), `hit1` for one violation, and so on.

Our internal ablations reveal a clear quality-diversity trade-off: strict filtering (e.g., `hit0`-only) over-prunes the corpus, while no filtering introduces excessive noise. We find that the `hit0-6` configuration (accepting documents with up to 6 violations) yields the best overall benchmark performance, retaining a wider variety of high-signal code compared to the original strict constraints.

C.2.2. PR/Issue/Commit Data

To enhance software engineering capabilities, we construct a comprehensive dataset from GitHub repositories with over 10 stars, comprising PRs, issues, and commits. We apply strict filtering on repository popularity and content quality, and use LLMs to generate missing issue descriptions, resulting in a 5-million-sample foundation. From this, we derive four training subsets:

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) Base PR/Issue/Commit Data: We crawl data via GHArchive and GitHub API, including full commit histories. We extract changes and validate a small portion of samples against `git diff` ground truth, then filter to 20+ mainstream languages (*e.g.*, Python, Java, C++). We strictly deduplicate against SWE-Bench Verified [13] and SWE-Bench Multilingual [14] to prevent leakage.

(2) Concatenated PR-Dialogue Data (90B tokens): We generate 90B tokens of code-editing training data by applying two Agentless-inspired templates [82]: (1) *File localization*: Given a problem description and repository structure, identify target file paths; (2) *Code repair*: Given a problem description and file content, generate precise modifications via SEARCH/REPLACE blocks.

We integrate this 90B code-editing data into two training phases with phase-specific masking strategies. In the annealing stage of pre-training, only template scaffolding is masked; in mid-training, the data is converted to chat dialogs with user prompts masked. Internal ablations show consistent gains over SWE-Bench Verified and SWE-Bench Multilingual in the cooldown stage and mid-training.

(3) Rewritten Reasoning-Oriented Data (12B tokens): From the Python subset of our base dataset, we derive bug-fix samples via LLM change-type annotation. We apply two concise rewriting strategies: (1) *Reasoning reconstruction*: an LLM reconstructs the PR author's problem-solving process (problem analysis, root cause identification, solution design, and code implementation), injected into PR-Dialogue format. Hallucinated/inconsistent traces are filtered via rule-based and LLM verification. (2) *Active Reading notebooks*: PR/issue/commit data is converted into structured learning outlines (motivation, root causes, design decisions, insights), then synthesized into coherent technical notes. These rewritten datasets (~12B tokens) are incorporated during mid-training, yielding further gains on SWE-Bench Verified.

(4) Environment-based Seed Data. We curate executable environments derived from raw PR, issue, and commit records using the environment building pipeline described in Appendix E.2.2. Candidate samples are rigorously filtered to ensure test-patch inclusion and validated via strict rule-based criteria to guarantee environmental reproducibility. Furthermore, selected issues undergo targeted rewriting to augment data quality and coverage. The resulting dataset comprises hundreds of thousands of seed samples, including problem descriptions, code changes, and test functions, and serves as the foundational bedrock for enhancing agentic coding capabilities, driving significant performance gains in downstream agent tasks.

C.3. Mathematics & STEM Data

To enhance reasoning capabilities and elicit intelligence from knowledge, we curate a large-scale mathematics and STEM dataset. Beyond the standard Common Crawl data used in prior works [176, 177], we leverage our in-house StepCrawl system to harvest a massive scale of additional mathematics-related data. Specifically, we implement a filtering pipeline inspired by MegaMath [177], utilizing an ensemble of internal classifiers alongside FineMath [178]. This allows us to retain hundreds of billions of mathematics-related tokens distinct from Common Crawl. We further collect a diverse 100M-sample educational dataset encompassing exercises, quizzes, and instructional content. This collection bridges the gap between academic theory and professional application, covering domains from K-12 mathematics/physics/chemistry and humanities to adult vocational exams (CPA, Legal). Early-stage experiments confirm that this problem-solving data is crucial for optimizing token efficiency during pre-training.

(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. Data Infrastructure

Our data construction and curation pipeline runs on a high-throughput in-house data infrastructure system designed for large-scale deduplication, mining, and model-inference filtering. We operate hybrid CPU/GPU clusters with distributed frameworks such as `Spark` and `Ray` to execute both large-volume processing (e.g., minhash-based deduplication) and model-driven curation workloads (e.g., embedding generation and classifier/LM inference), backed by a storage layer spanning object storage (OSS), HDFS, and `JuiceFS` for efficient reads/writes of raw corpora and intermediate artifacts.

C.5. Data Ablations Setting

To rigorously assess data quality and the impact of curation strategies, we conduct an extensive ablation suite using the 30B-A3B MoE architecture trained with the Muon optimizer, consistent with the mainline settings (Table 9). Adhering to a strict token efficiency protocol, we set a fixed training budget for all experiments. Models are evaluated on the comprehensive benchmarks listed in Section 6.1, alongside a series of carefully designed held-out compression (perplexity) test sets. We observe that compression metrics often provide a more direct measure of knowledge capacity, offering signals complementary to mainstream benchmarks.

Internal experiments on the 30B-A3B MoE model demonstrate its superior performance and stability compared to smaller proxies. While smaller models are computationally cheaper, they often fail to capture the nuances of complex reasoning and lack the capacity to memorize long-tail patterns, leading to an artificial bias towards data repetition. Empirically, the 30B-A3B size offers stronger stability and better fidelity to full-scale trends.

D. Post Training Details

This section describes the post-training process that refines the base model into a high-performance agentic system, covering SFT with rigorous data processing and quality control, followed by large-scale RL to further improve reasoning, tool use, and generalization.

D.1. SFT Details

D.1.1. SFT Data Processing Pipeline

Across all domains, we apply a unified data processing pipeline that emphasizes answer verifiability, reasoning quality, and execution realism. To ensure overall data integrity, the aggregated dataset undergoes a strict two-stage filtration process:

1. **Rule-based Filtering:** We eliminate low-quality data exhibiting degenerate patterns, such as infinite repetition, harmful content, and personally identifiable information.
2. **Model-based Filtering:** We utilize specialized models to detect and filter out linguistically inconsistent data. By identifying and removing samples with unnatural language mixing, we significantly refine the dataset's linguistic purity and overall quality.
3. **Decontamination:** We conduct comprehensive benchmark decontamination to prevent test set leakage. This involves both exact matching (with digit masking to catch numerical modifications) and N-gram matching.

This process yields a final refined dataset of 871k samples, totaling 7.23B tokens.The detailed distribution of the SFT data is presented in Table 3.

D.2. RL Details and Ablations

This section details the large-scale RL post-training, covering data curation, asynchronous search-agent training, and ablations on dense and MoE models.

D.2.1. Data Curation

We curate the RL training dataset by aggregating problems from open-source collections and competition archives spanning competitive coding, STEM, and synthetic data for general RLVR training. To prevent data contamination, we strictly exclude problems from competitions held during 2024–2026. The dataset is further augmented with: (i) synthetic arithmetic problems involving 11–13 digit integers; (ii) a generator–validator pipeline that synthesizes additional test cases for coding tasks; and (iii) synthetic environments for general reasoning tasks, such as puzzle and instruction following.

We apply a two-stage filtering process. First, deterministic rule-based pruning removes prompts containing images, external links, or open-ended requirements without a unique final answer. Second, an accuracy-based filter excludes trivial or degenerate problems. During training, each batch is constructed by sampling from different domains according to predefined sampling probabilities.

D.2.2. Reward System

Verifiable Rewards. For STEM tasks, we employ gpt-oss-120b [33] as the verifier model, using the following structured prompt (originally in Chinese) to rigorously assess final-answer correctness. For coding tasks, we utilize sandboxes to validate code execution against test cases with soft reward.

```
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.
Grading procedure:
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>
```

D.2. 强化学习细节与消融实验

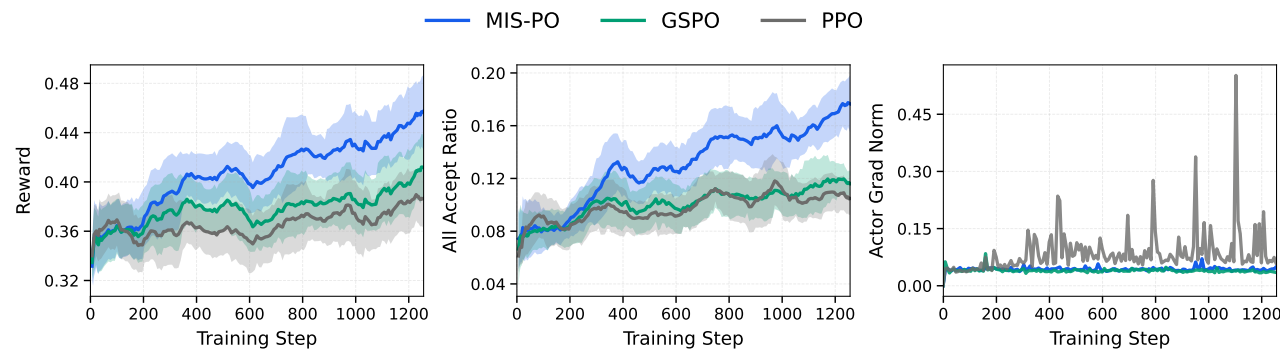本节详细介绍了大规模RL后训练，涵盖数据管理、异步搜索代理训练以及密集模型和MoE模型的消融实验。

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 answerGrading procedemine 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 studentmust 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) **Comparison on the dense model.** While GSPO also effectively reduces the variance of the actor gradient norm, its efficiency is inferior to that of MIS-PO. Under the same iteration budget, MIS-PO achieves higher rewards and all acceptance ratio.
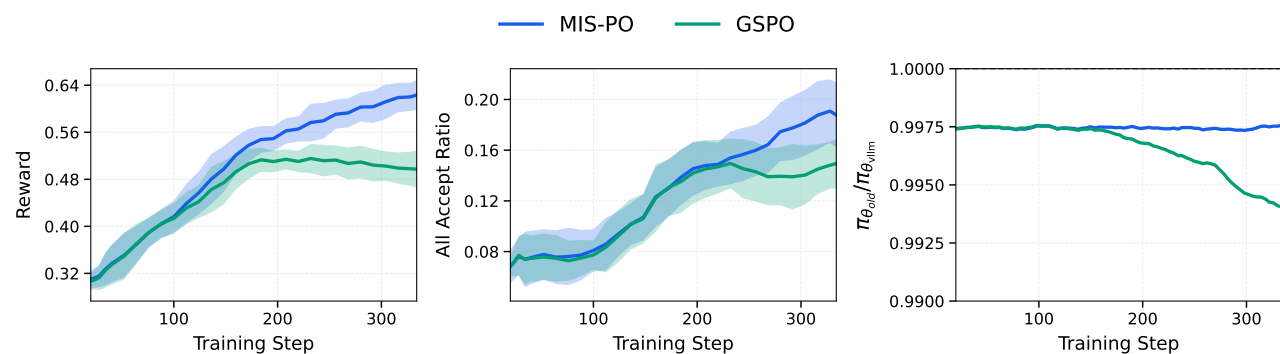
(b) **Comparison on the MoE model. (1) Efficiency:** MIS-PO demonstrates superior sample efficiency, achieving higher rewards with accelerated convergence, whereas GSPO plateaus around iteration 200. **(2) Stability:** GSPO exhibits an increasing training-inference discrepancy during training, quantified by the density ratio $\pi_{\theta_{\text{old}}}/\pi_{\theta_{\text{vllm}}}$ (where $\pi_{\theta_{\text{vllm}}}$ is the rollout policy in the inference backend and $\pi_{\theta_{\text{old}}}$ is the pre-update policy snapshot in the training backend). Conversely, MIS-PO consistently maintains this discrepancy within a stable range.

Figure 7: **Performance comparison between MIS-PO and GSPO.** The top figure (a) shows results on the dense model, and the bottom figure (b) shows results on the MoE model. MIS-PO consistently outperforms GSPO in both efficiency and stability across different architectures.

D.2.3. RL Ablation Details

MIS-PO vs. GSPO. To rigorously validate the effectiveness of our method, we benchmark MIS-PO against GSPO [36] on both Dense and MoE architectures. We select GSPO as the primary baseline because it represents a competitive strategy for reducing the gradient variance inherent in importance sampling. In our implementation, we extend the original GSPO estimator to the actor-critic setting by integrating its Generalized Importance Sampling mechanism into the actor loss. Specifically, we replace the standard token-level importance sampling ratio with the geometric mean of trajectory-level ratios. The resulting actor loss is formulated as follows ($\gamma = \lambda = 1$):
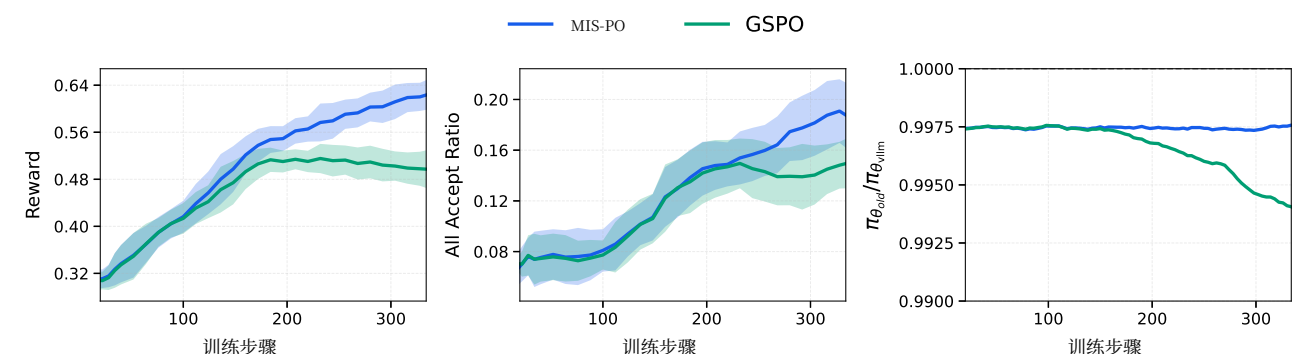
$$r_\tau(\theta) = \left(\prod_{t=0}^{T-1} \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\right)^{\frac{1}{T}} \tag{14}$$

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

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

To ensure a fair comparison, we apply the same token- and sample-level masking strategies used in MIS-PO to exclude data with significant training–inference mismatches. Regarding the clip ratio ϵ, we conduct a grid search over $\{1, 2, 3, 4\} \times 10^{-4}$. We adopt $\epsilon = 10^{-4}$ for all experiments primarily because it achieves the best benchmark performance after 200 RL training steps. Additionally, we observe that this setting yields a clip fraction of approximately 15%, consistent with the original GSPO [36].

Figure 7 presents the comparative results. Empirically, MIS-PO demonstrates superior sample efficiency and scalability compared to GSPO. Crucially, MIS-PO effectively constrains the training-inference mismatch within a stable range. This stability proves particularly critical for the large-scale RL training of MoE models, where the baseline GSPO fails to maintain consistent convergence.

Extended Training Dynamics on MoE. To further validate the scalability of our method, we conduct an extended training run of MIS-PO on the MoE model using a challenging dataset. As illustrated in Figure 8, the model maintains a continuous upward trend in rewards, stable actor gradient norms, and well-controlled entropy levels. These results empirically confirm that MIS-PO is reliability for large-scale MoE off-policy RL training.
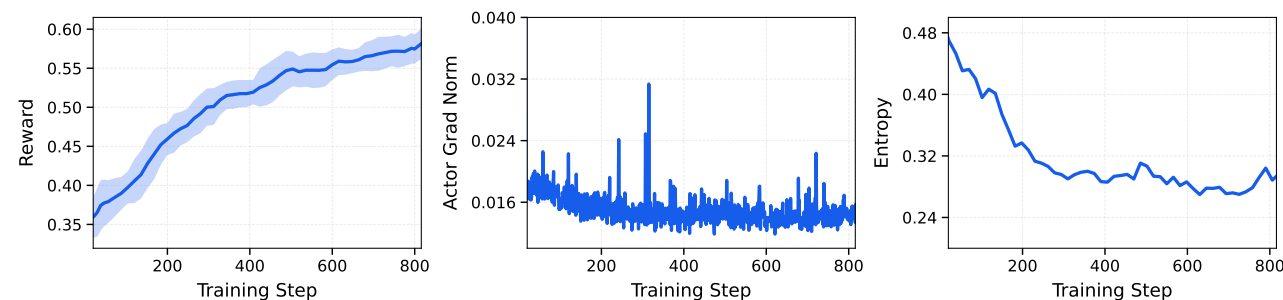

Figure 8: **Extended training dynamics of MIS-PO on the MoE model.** The metrics include Reward (left), Actor Gradient Norm (middle), and Entropy (right). Notably, the middle panel displays the raw gradient norm without smoothing or downsampling to highlight the stability of the optimization.

D.2.4. Search Agent

Regarding the training architecture, the early client–server one-step off-policy framework is severely bottlenecked by long-tail latency: approximately 5% of samples accounted for roughly 80% of the generation cost. However, our observations indicate that the policy exhibits strong robustness to staleness, maintaining stable performance even with a latency of approximately 20 steps. Consequently, we adopt the FullyAsync paradigm, decoupling generation and updates into a completely asynchronous process. Furthermore, to minimize inference overhead during multi-turn interactions, we implement sticky scheduling, where the same session is consistently dispatched to the same node to maximize KV-cache reuse. Overall, this configuration achieves an approximate 10× efficiency gain while maintaining training stability.

Throughout the training process, the FullyAsync paradigm demonstrates robust stability, evidenced by a sustained increase in rewards and a Truncated Importance Sampling (TIS) truncation rate maintained within a controllable range, thereby indicating limited policy drift induced by asynchrony. Notably, we observe that distinct from the limited scalability of "RL from zero" regarding training budgets, injecting task-relevant knowledge and tool-use priors during the mid-training phase elicited significantly higher performance gains and a more stable emergence of capabilities during the RL.

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。
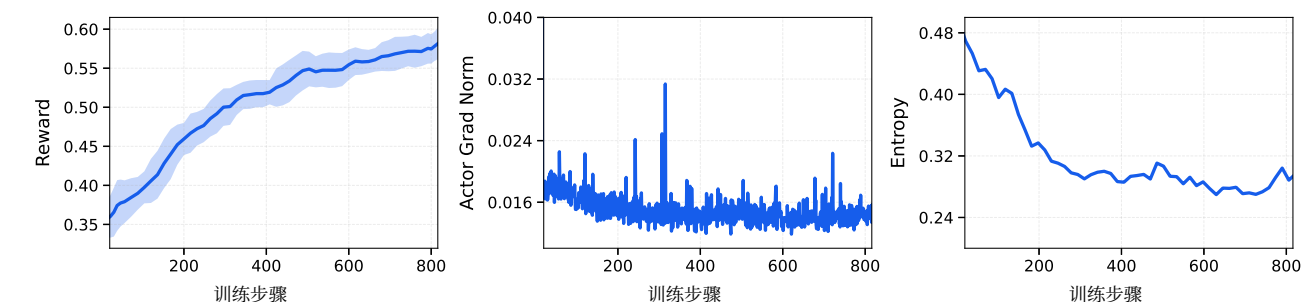

图8：**MIS-PO在MoE模型上的扩展训练动态。** 指标包括奖励（左侧）、演员梯度范数（中间）和熵（右侧）。值得注意的是，中间面板显示了未进行平滑或降采样的原始梯度范数，以突出优化的稳定性。

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。

Model	BrowseComp	BrowseComp-ZH	GAIA	xbench DeepSearch-2505	xbench DeepSearch-2510	Avg Gain
AGENT ∆AVG@3 (METRIC: PASS RATE %)						
Step 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 Flash	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

Table 11: **Impact of Tool Usage on Agent Performance.** Each cell displays the **Baseline Score** (internal knowledge only) followed by the ▲Performance Gain achieved by enabling search tools. The final score is the sum of both values. **Avg Gain** highlights the model's ability to leverage external information to improve results. Models marked with * denote tool results measured under a 256K setting; the setting for other models is unspecified.

Discussion. To rigorously evaluate agentic competence isolated from parametric memorization, we focus on the *tool-usage gain*, defined as:

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

This metric decouples the model's inherent knowledge from its ability to dynamically leverage external tools. As detailed in Table 11, **Step 3.5 Flash** demonstrates the most robust capability to leverage external information, achieving the highest average gain (52.0) and leading significantly on complex benchmarks such as GAIA and xbench-DeepSearch.

This distinction is critical because high absolute scores on benchmarks like BrowseComp can sometimes stem from strong internalized knowledge rather than effective search strategies. A smaller Δ_{tool} in a high-performing model may ambiguously indicate either high efficiency (the model already "knows" the answer) or a failure to effectively utilize tools to improve results. Conversely, a large Δ_{tool} explicitly signals the model's proficiency in bridging knowledge gaps through retrieval. Therefore, we argue that future optimization should not merely chase higher absolute scores ("benchmark grinding"), but should aim to maximize this Δ_{tool} in long-context, evidence-critical scenarios. This ensures the agent is truly mastering the *process* of information retrieval and reasoning, rather than overfitting to static knowledge or benchmark artifacts.

D.3. Tool-integrated Reasoning and Parallel Reasoning

In this section, we introduce two primary methodologies for test-time scaling in Step 3.5 Flash: tool-integrated reasoning and parallel reasoning.

Tool-integrated Reasoning For complex reasoning tasks, we integrate the model with a Python interpreter to facilitate tool-assisted reasoning. In this framework, the model operates within a sandbox to iteratively think and execute code for computational, simulation, and visualization purposes. In our experiments, we evaluate on AIME 2025, HMMT 2025, IMO-AnswerBench, GPQA, HLE$_{\text{text}}$, and ARC-AGI-1 with a 100-turn limit. As shown in Table 12, tool-integrated reasoning significantly

模型	BrowseComp I		GAIA	xbench DeepSearch-2505	xbench DeepSearch-2510	
AGENT ∆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∆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 ∆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 ∆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$_{\text{text}}$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

enhances performance across challenging mathematics, STEM, and puzzle benchmarks, highlighting the advanced agentic reasoning capabilities of Step 3.5 Flash.

Benchmark	Step 3.5 Flash	Step 3.5 Flash w. Python
AIME 2025	97.3	**99.8** (+2.5)
HMMT 2025 Feb.	98.4	**98.7** (+0.3)
HMMT 2025 Nov.	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)

Table 12: Comparison of Step 3.5 Flash and Step 3.5 Flash w. Python.

Tool-integrated Parallel Reasoning We present a preliminary exploration of extending PaCoRe to a multi-turn interactive environment. By design, PaCoRe preserves the standard LLM message interface. This compatibility allows for seamless integration into existing agentic frameworks that utilize multi-turn tool interaction. To adapt PaCoRe to this setting, we implement a state-aware input serialization protocol as shown in Table 14.

We evaluate this approach on the GPQA and HLE_{text} benchmarks using Step 3.5 Flash equipped with a Python interpreter. As shown in Table 13, extending parallel reasoning to these agentic loops yields significant performance improvements over the standard reasoning baseline. These findings demonstrate that PaCoRe effectively generalizes to environments requiring interactive feedback, highlighting a promising avenue for agentic test-time scaling.

Benchmark w. Python	Step 3.5 Flash	Step 3.5 Flash + PaCoRe
GPQA-Diamond	84.4	**85.7** (+1.3)
HLE_{text}	26.5	**28.2** (+1.7)

Table 13: Comparison of Step 3.5 Flash w. Python and the same model with PaCoRe test-time scaling.

E. Detailed Evaluation Protocols and Prompts

This section provides the implementation details for our evaluation suite. We outline the specific prompt templates, few-shot configurations, and the judge models employed across different benchmarks. For complex metrics, such as those used in long-context or reasoning tasks, we also detail the underlying calculation logic and scoring criteria to ensure reproducibility. In the templates provided below, {question} denotes the placeholder for the textual problem description, while other placeholders (e.g., {test}, {context}) represent task-specific information.

E.1. Evaluation Details of Pre-train Models

E.1.1. General language understanding and reasoning benchmarks

BBH. We use the official CoT-prompts [6] of BBH [132], with only "Q:" and "A:" replaced by "Problem:" and "Solution:" as follows:

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

---

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: Standard User Query (Last role: `user`)

You are given a problem and a list of reference responses. Your job is to analyze these references and provide your own response.

Original Problem:
{{ original_content }}

Reference Responses:
Note: Some references may contain <tool_call> tags indicating tool calls the reference intended to make.
These tool calls have NOT been executed - they are shown only as reference for your analysis.
{% for response in ref_responses %}
Reference {{ loop.index }}:
{{ response }}
{% endfor %}

Now, based on the original problem and reference responses above, please provide your own comprehensive solution.

Panel B: Tool Observation (Last role: `tool`)

You are given a tool response and a list of reference responses analyzing it. Your job is to analyze these references and provide your own response.

Original Tool Response:
{{ original_content }}

Reference Responses:
[Same preamble regarding unexecuted tool calls as in Panel A]
{% for response in ref_responses %}
Reference {{ loop.index }}:
{{ response }}
{% endfor %}

Now, based on the original tool response and reference responses above, please provide your own comprehensive analysis and next steps.

Table 14: **Input serialization templates for Tool-integrated PaCoRe.** We introduce distinct templates to handle the initial user query (Panel A) and subsequent tool observations (Panel B). Note that `tool_calls` within reference branches are serialized as text for analysis.

```
Problem:
{question}

Solution:
```

MMLU. We use the official evaluation metric of MMLU [133] with 5-shot. We employ the following task-specific system prompt:

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

The corresponding question prompt is structured as follows:

```
{question}
Answer:
```

MMLU-Redux. We use the official evaluation metric of MMLU-Redux [134] with 5-shot. and employ the following question prompt:

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

MMLU-Pro.   We follow the official evaluation metric of MMLU-Pro [135] with 5-shot. All evaluations use the following system prompt:

```
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.
```

The question prompt is structured as follows, with a deliberate trailing space after the final period:

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

Notably, we observe that a subset of the original MMLU-Pro dataset (470 out of 12,102 questions) contained an inconsistent leading space before the ground-truth options. We explicitly remove these spaces to mitigate potential formatting bias and ensure evaluation consistency.

HellaSwag.   We use the official evaluation metric of HellaSwag [136] with 10-shot. We employ the following question prompt:

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

WinoGrande.   We use the official evaluation metric of WinoGrande [137] with 5-shot. The question prompt is structured to present the binary choices clearly:

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

Answer:
```

GPQA.   We use the official evaluation metric of GPQA [138] with 5-shot. The question prompt is structured to present the choices clearly:

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

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

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. We use the official evaluation metric of SuperGPQA [139] with 5-shot. The question prompt follows a Chain-of-Thought (CoT) structure, where each few-shot example includes a step-by-step derivation leading to the final answer:

```
Question:
{question}

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

SimpleQA. We use the official evaluation metric of SimpleQA [140] with 5-shot. As SimpleQA requires open-ended short answers, we employ an LLM-based judgement for evaluation, specifically using gpt-oss-120b [33] as the judge model. The question prompt is formatted as a concise query:

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

E.1.2. Mathematics reasoning benchmarks

GSM8K. We use the official evaluation metric of GSM8K [141] with 8-shot. The question prompt is designed to elicit CoT reasoning by using the following template:

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

MATH. We use the official evaluation metric of MATH [142] with 4-shot. The question prompt is structured with explicit problem and solution delimiters:

```
Problem:
{question}


Solution:
```

E.1.3. Coding benchmarks

HumanEval. We use the official evaluation metric of HumanEval [143] with 3-shot. The question prompt is structured with three ground-truth examples to provide contextual guidance for code generation:

```
# 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. """
```

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 list
    for item in some_list:
        # insert item into new list
        new_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 reverse
    some_list.reverse()
    return some_list
{question}
```

MBPP. We follow the official evaluation metric of MBPP [144] with 3-shot.

HumanEval+. We follow the official evaluation metric of HumanEval+ [145] with 3-shot.

MBPP+. We use the official evaluation metric of MBPP+ [145] with zero-shot. We employ a structured instruction prompt that specifies the task requirements and includes a sample test case for alignment:

```
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. We use the official evaluation metric of MultiPL-E [146] with zero-shot. We follow the official test cases to judge the generated code.

E.1.4. Chinese understanding benchmarks

C-Eval. We use the official evaluation metric of C-Eval [147] and add a 5-shot setting. We employ the following system prompt:

```
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。
```

The corresponding question prompt is structured as follows:

```
{question}
答案:
```

CMMLU. We use the official evaluation metric of CMMLU [148] and add a 5-shot setting. We employ the following system prompt:

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。

The corresponding question prompt is structured as follows:

```
{question}
答案:
```

C-SimpleQA. We use the official evaluation metric and LLM-based judgement protocols of Chinese SimpleQA [149]. We add a 5-shot setting and use gpt-oss-120b [33] as the judge model. We employ the following question prompt:

```
问题：{question}
答案:
```

E.2. Evaluation Details of Post-Train Models

In this section, we detail the evaluation protocols used to assess the post-trained models across a diverse set of agentic tasks. Our evaluations span both code-centric and general-purpose agent settings, covering software engineering, terminal interaction, deep search, research workflows, and real-world tool use. We report standardized metrics under carefully controlled environments and inference budgets to ensure fair, stable comparisons across benchmarks.

E.2.1. Reasoning benchmarks

CF-Div2-Stepfun. We evaluate the competitive programming capabilities of our model using a custom CodeForces Div. 2 Benchmark. The benchmark comprises 53 problems sourced from official CodeForces Division 2 contests held between September 2024 and February 2025. We develop an offline evaluation framework that utilizes a local grading mechanism as an alternative to real-time online submissions. We try to construct test cases similar to the original test cases. Specifically, we first generate enough small-scale test cases for evaluation correctness coverage, then add randomized data for large-scale testing. Finally, we performed adversarial construction of edge cases by analyzing common error patterns and "hacked" submissions from actual users. Some edge cases are also auto-generated by the stress testing technique, which keeps generating countless test cases until one can distinguish failed submissions from correct submissions. To validate the reliability of this benchmark, we run both correct and representative failed submissions selected from the original contests. Our evaluator correctly identify 100% of the accepted submissions as "Passed," while 92.45% of the failed submissions are accurately flagged.

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。

	Accuracy (avg@8)			Codeforces C++
Model	**C++**	**Python**	**Java**	**pass@8 Rating**
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 Flash	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

Table 15: Full evaluation results of variable models in CF-Div2-Stepfun.

We sample 8 responses for each problem and report the average accuracy. The user prompt utilized for this process is:

```
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}
```

The compilation and execution commands for C++, Python, Java are given below:

```
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}
```

To maintain consistency with competitive programming norms and avoid the inconsistent overhead associated with JIT "warm-up" periods, we use the standard Python interpreter with a double time limit rather than PyPy[7]. We apply this same double time limit to all Java submissions.

While the Table 15 reports raw accuracy, we recognize that problem difficulty varies significantly. Therefore, rating scores provide more robust metrics. Although frameworks like CodeELO [179] can calculate competitive ratings, current top-tier models perform so effectively in Division 2 contests that their ratings may result in statistical outliers. Furthermore, we adopt a simplified rating calculation that disregards time penalties by assuming all solutions are submitted at the onset of the contest. While this approach deviates from empirical competitive scenarios and may result in ratings that are not directly comparable to human participants, it provides a standardized benchmark for consistent cross-model comparison.

---

[7]https://pypy.org/

	平均@8准确率			Codeforces C++
模型	**C++**	**Python**	**Java**	**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. We use the official evaluation method of LiveCodeBench [12]. We employ the following system prompt:

```
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.
```

The corresponding question prompt is structured as follows:

```
### 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.** We use the official evaluation method of AIME 2025 [166] with repeat@64. We employ the following question prompt:

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

HMMT 2025 Feb./Nov. We use the official evaluation method of HMMT 2025 [11] with repeat@64. We employ the following question prompt:

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

IMO-AnswerBench. We use the official evaluation method of IMO-AnswerBench [91] with repeat@64. We employ the following question prompt:

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

MMLU-Pro. We use the official evaluation method of MMLU-Pro [135]. The processing of dataset remains consistent with our pre-training MMLU-Pro evaluation methodology (see Appendix E.1.1 for details).

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

{question}
```

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. We use the official evaluation method of GPQA-Diamond [138]. We employ the following question prompt:

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

{question}
```

HLE_text. We use the official evaluation metric and LLM-based judgement protocols of HLE. We use gpt-oss-120b [33] as the judge model.

E.2.2. Code Agent benchmarks

SWE-Bench. SWE-Bench Verified [13] is a high-quality subset of the original SWE-bench dataset, consisting of 500 software engineering tasks rigorously validated by human expert developers to ensure reliable and accurate evaluation. SWE-Bench Multilingual extends the original benchmark to a diverse set of 300 real-world software engineering tasks across 9 programming languages.

We test the software engineering agent ability of Step 3.5 Flash on SWE-Bench Verified and SWE-Bench Multilingual using our internal agent infrastructure, which is built upon the described session-router architecture. For each evaluation instance, we provision a containerized session orchestrated via Kubernetes. We then perform environment initialization specific to SWE-Bench, which includes removing future commits to prevent data leakage, as well as configuring network proxies and critical system settings. Regarding the agent scaffold, we adopted the OpenHands [127] CodeAct Agent framework, which is widely used in the research community. We enabled a default suite of four tools: execute_bash, str_replace_editor, finish, and think. The max interactive turns is set to 350.

Given the resource-intensive nature of compiled languages, we allocate 12GB of memory for the multilingual setting, whereas the verified instances are restricted to a 4GB limit. In evaluations, the tool execution timeout is set to 1200s, and the model inference parameters are: temperature=1, top-p=0.95. Following the above settings, Step 3.5 Flash reach 74.4% on SWE-Bench Verified, and 67.4% on SWE-Bench Multilingual benchmark with an average score of 4 repeat of runnings. We also cross-evaluate Step 3.5 Flash on other popular agent scaffolds: SWE-Agent [128] with the original agent pipeline settings achieving 74.2% accuracy on SWE-Bench Verified, and standard Claude Code [8] environment scoring 72.0% with an extended time limit of 4 hours for each instance and no time limit for single tool execution.

Terminal-Bench 2.0. We test the Terminal-Bench benchmark [16] within task-independent, remote containers. We limit the container memory to 16GB. We have deployed an internal Artifactory repository and update the default package sources for all Docker containers. During session creation and dependency installation of the testing phases, the system will retry multiple times if an error occurs. To simplify communication, we modify the Terminus 2 framework so that it automatically interrupts timed-out commands and prevents subsequent commands in the same round from executing, returning a timeout warning to the agent. Accordingly, we modify the command duration control part of the original system prompt:

```
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

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.
```

During inference, we cap the model's single-turn output at 64k and the maximum context window at 256k for all interactions. The thinking process will be preserved in the multi-round history. If the model output exceeds the 256k context window limit, we execute a pruning context management: Keep the problem statement and the last 50% of history before retrying. We use the inference parameters of top-p=0.95 and temperature=1. The interaction protocol is primarily conducted using XML-formatted structured responses. The agent is limited to 200 interaction rounds and will proceed directly to the testing phase once this limit is reached. The total time limit for interaction and testing is 6 hours.

To ensure consistency, we verify and refine each task's checker against its problem statement, which improved overall accuracy by approximately 1.5%. Each task is executed across 8 trials. Notably, 88.6% of successful trajectories are completed within 30 interactions. The final pass@8 stands at 67/89, with an avg@8 of 50.98%. Our agent achieves a 100% success rate across all 8 trials in 23 out of 89 tasks. In the successful trajectories, 9.41% of the runs triggered history pruning to manage context limits.

Setting	Max Output	Max Round	Timeout	Context Management	Avg@8
Baseline	64k	200	6h	✓	**50.98%**
Limit 16k	16k	200	6h	✓	48.03%
Limit 16k w/o Pruning	16k	200	6h	✗	45.22%
Rounds 100	64k	100	6h	✓	50.42%
Timeout 2h	64k	200	2h	✓	49.72%

Table 16: Ablation study of inference constraints on Terminal-Bench 2.0.

The ablation study shows that Limit 16k causes the largest performance drop because the model's long reasoning for complex tasks often exhausts the token limit before it can output the terminal commands. The further decline to 45.22% when disabling context management under the 16k limit. Meanwhile, Rounds 100 has minimal impact as most tasks finish early. The Timeout 2h decrease reflects that certain tasks involving model training, heavy compilation, or complex environment configuration require more time to complete.

E.2.3. *General Agent benchmarks*

Deep Search. We evaluate our agent's deep search capabilities on multiple benchmarks (e.g, **BrowseComp** [17], **BrowseComp-ZH** [18], **GAIA** [19], **xbench-DeepSearch** [20]). The results reported in Table 5 are based on the avg@3 metric; GPT-5.2 xHigh uses avg@1. The agent is equipped with a core toolset including:

- **search**: Executes multiple search queries in parallel.
- **visit**: Analyzes the content of the webpage to answer specific questions based on LLM.
- **google_scholar**: Search for academic articles and technical literature.
- **python_interpreter**: Runs Python code for calculations and data analysis.
- **file**: Downloads and saves files from direct URLs.

During inference, we employ a 256k-token context window with no limit on the maximum generation length. Inference is conducted with top-p = 0.95, temperature = 1.0, and presence penalty = 1.1, allowing for an execution budget of up to 400 steps.

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。

The detailed system prompts for the agent and the LLM judge are consistent with the configurations provided in the GitHub repository associated with [125].

BrowseComp (w. Ctx Manage). The BrowseComp (w. Ctx Manage) result of 69.0 reported in Table 5 corresponds to the discard-all methodology evaluated on the full BrowseComp dataset. This approach, same as DeepSeek V3.2 [1], is triggered when the context length exceeds predefined thresholds, at which point the agent discards its entire context and reinitializes the operational loop. Under a maximum iteration constraint of 1000 steps, this strategy employs a context length threshold of 72k tokens for BrowseComp and 41k tokens for BrowseComp-ZH.

We also evaluate various context management strategies on a subset of 200 instances from BrowseComp, including Summary, Keep-first&last$K$, Discard-all, and Multi-agent orchestration. As shown in Table 17, our model demonstrates robust adaptability across these diverse paradigms. Among single-agent strategies, Discard-all yields a competitive 66.0% accuracy. We posit that Discard-all functions as a test-time pass@k strategy, forcing the model to re-reason from scratch until a self-verified path is found. The performance follows a clear hierarchy: Multi-agent ranks highest by leveraging a master agent to decompose tasks and dispatch specialized agents for parallel reasoning, followed by Discard-all, Keep-first&last$K$ and Summary —closely aligns with the increase in real steps. This alignment reflects a direct trade-off between inference cost (number of steps) and accuracy, suggesting that intensive context management effectively converts increased computation into superior performance.

Method	Accuracy (%)	Real Steps
Step 3.5 Flash	49.5	86
+ Summary	57.0	131
+ Keep-first&lastK	58.0	244
+ Discard-all	66.0	302
+ Multi-Agent	68.5	721

Table 17: Evaluation results of context manager methods.

RESEARCHRUBRICS. To evaluate deep research capabilities, we utilize the RESEARCHRUBRICS [21] benchmark. This dataset comprises 101 domain-diverse research tasks, each accompanied by 20–43 expert-written, fine-grained scoring criteria that assess factual accuracy, reasoning soundness, and clarity. We benchmark performance against two representative system families: commercial agent systems and ReAct agents.

For commercial agents, we collect reports via their official web interfaces (captured Dec 2–15, 2025) under default configurations. As shown in Table 18, the leading commercial system (Gemini DeepResearch) achieves an aggregated score of 63.69.

For ReAct agents, detailed performance comparisons are presented in Table 5. Our model achieves a score of **65.3**, surpassing the complex, proprietary commercial baselines. Notably, when evaluating Gemini 3.0 Pro within our standardized ReAct framework, we observe a score of 50.1. We attribute this performance gap to insufficient search depth when addressing open-ended research questions; the model tends to rely on internal parametric knowledge rather than perform extensive external retrieval. Consequently, the generated reports lack comprehensiveness, failing to adequately cover the user's implicit criteria.

We standardize the execution environment for ReAct agents with a maximum of 30 reasoning turns and a per-turn output limit of 16k tokens. For inference parameters, other API-based models use their

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：

Agent System	Score
Gemini DeepResearch	63.69
OpenAI DeepResearch	60.67
Kimi Researcher	53.67
MiniMax Agent Pro	51.85
Qwen DeepResearch	49.24

Table 18: Performance of Commercial Agent Systems on the RESEARCHRUBRICS benchmark.

default settings, and our model is configured with a temperature of 1 and top-p=0.95. All outputs are subsequently appraised by an LLM judge using a ternary grading for each criterion. To support the end-to-end research workflow, our ReAct framework provides access to the following suite of tools:

- **batch_web_surfer**: For concurrent web searching and multi-page browsing.
- **file**: For robust file operations, including reading, writing, and iterative editing.
- **file_parser**: For converting files into Markdown format.
- **shell**: For interactive command execution and environment interaction.
- **todo**: For dynamic task state management and tracking.
- **tmux**: For simulating a multiplexed terminal environment with persistent sessions and scrollback history.

τ^2-**Bench.** τ^2-Bench [15] is an agentic benchmark that evaluates general tool-use capability in three customer service domains: airline, retail, telecom. We evaluate Step 3.5 Flash using the official settings in the original codebase. Specifically, we use the default LLM agent framework and set the temperature to 1.0, top-p to 0.95, max sequence length to 256K. The user model is set to GPT-4.1 with 0.0 temperature to ensure a stable interaction during evaluation. For the airline domain, since it has incorrect ground truth answers, we use the fixed version from Claude Opus 4.5 to ensure evaluation reliability [9]. For the retail and telecom domains, we also follow Claude Opus 4.5 to include a general prompt addendum to the user prompt to avoid failure modes from the user ending the interaction incorrectly [10]. We report an average score of 8 runs to ensure stable evaluation results.

E.2.4. *General benchmarks*

Arena-Hard-v2.0. We use the official evaluation metric of Arena-Hard-v2.0 [151] and use GPT-4.1 [180] as the judge model.

MultiChallenge. We use the official evaluation metric of MultiChallenge [153] with o3-mini [181] as the judge model. This follows findings from the GPT-5 [182] release that GPT-4o [180] frequently mis-scores complex responses, leading to underestimated results.

IFBench. We use the official evaluation method of IFBench [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. Long Context benchmarks

LongBench v2.   We use the official evaluation method of LongBench v2 [154].

MRCR-8needle.   For MRCR-8needle [155] benchmark, we report the Area Under Curve (AUC) metric, following the protocol established by ContextArena [11]. Specifically, we use the **AUC@128k** metric, which provides a single holistic score summarizing performance across context lengths up to 131,072 tokens.

The AUC is calculated by plotting the average retrieval accuracy for each context bin (ranging from 8k to 128k) against the bin's maximum context length. We apply the trapezoidal rule on a linear scale to measure the area under the resulting curve, which is then normalized by the total context width (128k minus the initial bin size) to yield a percentage score between 0% and 100%. This metric effectively penalizes performance degradation as difficulty increases with longer context sequences.

FRAMES-Oracle.   We use the official evaluation metric of FRAMES [156]. Since our focus is on long-context capabilities, we specifically report results for the **Oracle Prompt** subset. In this setting, the model is provided with the question alongside all ground-truth Wikipedia articles used during human annotation. This configuration serves as an upper bound for model performance, simulating a perfect retrieval system that delivers all relevant context to the model.

RepoQA.   We use the official evaluation method of REPOQA [157].

E.3. Internal Evaluation - Benchmarks and Methodology

E.3.1. Data Analysis Benchmark

To reliably assess Step 3.5 Flash 's ability to perform practical data-analysis tasks in the Claude Code environment, we develop an internal Data Analysis Benchmark for evaluating end-to-end analytical problem solving under realistic business constraints. The benchmark is constructed by systematically distilling senior practitioners' tacit expertise into a rubric-grounded evaluation suite. This approach captures the ambiguity and contextual nuance of real-world analytics while ensuring consistent evaluation through standardized rubrics and verifiable ground-truth artifacts.

The benchmark is constructed using an expert-driven, rubric-based protocol to ensure domain authenticity and scoring reliability. Ten senior data analytics leaders from major Chinese internet companies, each with over 15 years of experience, contributed real-world business cases through structured interviews that elicited core analytical patterns and decision logic. This process yields representative tasks paired with expert-endorsed solution strategies.

Interview materials are normalized into machine-consumable tasks, each comprising a problem statement, a CSV dataset, a reference analysis, and a weighted checklist-style scoring rubric. The resulting benchmark contains 50 items spanning diverse analytical intents, with an average of 26.9 rubric items per task. Quality is ensured through iterative expert review, aligning task definitions, data, reference solutions, and evaluation criteria to improve validity and reproducibility.

We further implement a unified end-to-end evaluation framework covering task execution, automated scoring, and report synthesis. The framework supports code-based, research-oriented, and text-based

---
[11]https://contextarena.ai/

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。通过迭代专家审查确保质量，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/

analyses within a single pipeline, enabling scalable and reproducible evaluation across heterogeneous environments with low integration overhead.

Evaluation Method. Each task is evaluated by a model-based evaluator that scores generated outputs against expert-defined rubrics, with results averaged over 3 identical runs to reduce stochastic variance and ensure reliable, comparable cross-model evaluation.

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

Table 19: Evaluation Results on the Data Analysis Benchmark

Evaluation Results. Table 19 presents the results on the Data Analysis Benchmark. Claude Opus 4.5 ranks first overall, while Step 3.5 Flash achieves a strong second place (39.58%) and remains very close to GPT-5.2 (39.31%). Its competitive performance may be partly related to relatively good adaptation to the Claude Code environment. In addition, Step 3.5 Flash demonstrates a favorable speed–capability trade-off, maintaining solid analytical quality while delivering faster responses. The results position Step 3.5 Flash as a highly efficient and competitive option for real-world data analysis tasks.

E.3.2. Consulting and Recommendations Benchmark

To rigorously evaluate Step 3.5 Flash in real-world advisory scenarios, we curate a benchmark of 500 diverse queries sourced from authentic social platforms such as Reddit, Stack Exchange, and various community forums. These queries represent authentic user intent across everyday life, academic learning, entertainment, and professional workplace contexts.

Here, we implement an "Anchor-Based" scoring framework to evaluate candidate models. In this process, we first utilize leading models, including GPT-5.2, Claude Opus 4.5, and DeepSeek V3.2 to generate independent responses for each query. These high-level outputs are then synthesized and refined by human experts to create a Reference Response as Ground Truth. This reference serves as a high-quality "Anchor" with a standardized performance value of 88/100.

We then measure the performance of the models across four critical dimensions, applying a rigorous scoring rubric, including Usefulness, Logic, Instruction Following, and Tone. Usefulness assesses whether the model delivers a ready-to-use solution that meaningfully resolves the task with expert-level depth, actionable steps, and feasible recommendations. Logic evaluates factual accuracy and structural soundness, checking for hallucinations, incorrect citations, invalid conclusions, or causal and temporal inconsistencies, as well as overall coherence and argument flow. Instruction Following measures adherence to both explicit constraints (*e.g.*, formatting, length, and stated requirements) and implicit contextual expectations embedded in the user query. Tone assesses communicative quality, including appropriateness of language and register, clarity in unpacking complex reasoning, and calibrated expression that avoids overconfidence while clearly signaling uncertainty when appropriate.

We employ a Hybrid LLM-as-a-Judge system. Recognizing that different frontier models have distinct

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

evaluative strengths, we assign specific scoring responsibilities as follows: Logic, Instruction Following, and Usefulness: These three dimensions are evaluated by GPT-5.2, leveraging its industry-leading capabilities in factual verification, constraint checking, and objective problem-solving. Tone: This dimension is evaluated by Claude Opus 4.5, utilizing its superior nuance in linguistic style, emotional calibration, and "human-like" resonance. Judge reliability is validated through an alignment study with human experts, yielding a high Pearson correlation between AI- and human-assigned scores. Final scores are computed using equal weighting across the four dimensions (25% each), ensuring a balanced assessment that jointly reflects technical correctness and communicative quality.

Model	Average	Usefulness	Logic	Tone	Instruction-following
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%
Step 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 Flash	67.9%	71.5%	58.0%	70.6%	71.4%
Minimax M2.1	67.1%	70.7%	60.1%	67.2%	70.4%

Table 20: Evaluation results on the Consulting and Recommendations Benchmark

Evaluation Results. Table 20 shows that Step 3.5 Flash achieves an average Score of 70.5% on the Consulting and Recommendations Benchmark, securing the 4th position overall. Step 3.5 Flash matches Gemini 3.0 Pro performance across all dimensions, achieving comparable Pro-level scores (70.5% vs. 70.6%) while offering substantially lower inference cost and latency. Unlike many fast models that trade speed for degraded reasoning quality, Step 3.5 Flash surpasses larger models in the Logic dimension, reducing hallucinations and logical failures and making it well suited for automated consulting workflows where factual integrity is critical.

E.3.3. *Step 3.5 Flash + Step-GUI*

To validate Step 3.5 Flash's efficacy in real-world agentic scenarios, we evaluate on **AndroidDaily Hard** [183], a challenging benchmark designed for Chinese mobile application environments. This benchmark comprises compositional tasks spanning e-commerce transactions, multimedia interactions, and daily mobile operations, offering a naturalistic testbed for assessing GUI agent capabilities in complex, multi-step workflows representative of production deployments.

We empirically investigate two architectural instantiations: (1) **Step-GUI** [183], a lightweight on-device agent (Edge Only) that executes tasks autonomously using local computational resources, and (2) **Step 3.5 Flash + Step-GUI**, an edge-cloud collaborative framework wherein Step 3.5 Flash functions as a cloud-based reasoning orchestrator that synthesizes high-level task plans, decomposes them into executable primitives via the GUI-MCP protocol, and delegates low-level control to the on-device Step-GUI agent. This hierarchical architecture exploits the complementary strengths of cloud-scale reasoning and edge efficiency: Step 3.5 Flash's 11B active parameters enable sophisticated multi-step planning and contextual understanding, while Step-GUI ensures low-latency action execution and privacy-preserving local control.

Quantitative Results. The edge-cloud collaborative paradigm achieves a success rate of **57.0%** on AndroidDaily Hard, substantially outperforming the edge-only baseline (**40.0%**). This result suggests that combining strong cloud-side reasoning with efficient edge execution is an effective strategy for navigating deployment constraints in multi-round agent interactions.

Architectural Generalization. Critically, this collaborative pattern extends beyond mobile ecosystems to heterogeneous platforms including desktop computers and automotive infotainment systems. By decoupling cognitive orchestration (cloud) from embodied execution (edge), the framework establishes a scalable paradigm for deploying sophisticated agents in resource-constrained industrial environments—directly aligned with Step 3.5 Flash's design objective of redefining the efficiency frontier for production-grade agentic systems. The results underscore that effective real-world agents require not only advanced reasoning capabilities but also architectures that harmonize computational distribution across infrastructure tiers.

References

[1] DeepSeek-AI. Deepseek-v3.2-exp: Boosting long-context efficiency with deepseek sparse attention, 2025.

[2] Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, et al. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. *arXiv preprint arXiv:2508.06471*, 2025.

[3] LLM-Core Xiaomi. Mimo-v2-flash technical report, 2026.

[4] Meituan LongCat Team, Bei Li, Bingye Lei, Bo Wang, Bolin Rong, Chao Wang, Chao Zhang, Chen Gao, Chen Zhang, Cheng Sun, et al. Longcat-flash technical report. *arXiv preprint arXiv:2509.01322*, 2025.

[5] Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

[6] MiniMax Team. Minimax-m2.1, 2025.

[7] OpenAI. Gpt-5.2, 2025.

[8] Google DeepMind. Gemini 3 promodel card, 2025.

[9] Anthropic. System card: Claude opus 4.5, 2025.

[10] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[11] HMMT. Hmmt 2025 feb., 2025.

[12] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

[13] OpenAI. Introducing SWE-bench verified we're releasing a human-validated subset of swe-bench that more, 2024.

[14] John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. Swe-smith: Scaling data for software engineering agents, 2025.

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, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E Kelly Buchanan, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *arXiv preprint 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: A simple yet challenging benchmark for browsing agents, 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: Benchmarking web browsing ability of large language models in chinese, 2025.

[19] Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants, 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: Tracking agents productivity scaling with profession-aligned real-world evaluations. *arXiv preprint arXiv:2506.13651*, 2025.

[21] Manasi Sharma, Chen Bo Calvin Zhang, et al. Researchrubrics: A benchmark of prompts and rubrics for evaluating deep research agents. *arXiv preprint arXiv:2511.07685*, 2025.

[22] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *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: Designing stable and transferable sparse expert models, 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: Efficient scaling of language models with mixture-of-experts. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5547–5569. 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: Scaling giant models with conditional computation and automatic sharding, 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: Towards ultimate expert specialization in mixture-of-experts language models, 2024.

[27] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *CoRR*, abs/1904.10509, 2019.

[28] Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. *arXiv preprint arXiv:2404.19737*, 2024.

[29] DeepSeek-AI. Deepseek-v3 technical report. *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: Unlocking the reasoning potential of language model–from pretraining to posttraining. *arXiv preprint arXiv:2505.07608*, 2025.

[31] Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free, 2025.

[32] StepFun Team. Step3: Cost-effective multimodal intelligence.

[33] OpenAI. Gpt-oss-120b & gpt-oss-20b model card, 2025.

[34] Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024.

[35] Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. Measuring short-form factuality in large language models. *arXiv preprint arXiv:2411.04368*, 2024.

[36] Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

[37] Feng Yao, Liyuan Liu, Dinghuai Zhang, Chengyu Dong, Jingbo Shang, and Jianfeng Gao. Your efficient rl framework secretly brings you off-policy rl training, August 2025.

[38] Wenhan Ma, Hailin Zhang, Liang Zhao, Yifan Song, Yudong Wang, Zhifang Sui, and Fuli Luo. Stabilizing moe reinforcement learning by aligning training and inference routers. *arXiv preprint arXiv:2510.11370*, 2025.

[39] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[40] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[41] Thang Luong, Dawsen Hwang, Hoang H. Nguyen, Golnaz Ghiasi, Yuri Chervonyi, Insuk Seo, Junsu Kim, Garrett Bingham, Jonathan Lee, Swaroop Mishra, Alex Zhai, Clara Huiyi Hu, Henryk Michalewski, Jimin Kim, Jeonghyun Ahn, Junhwi Bae, Xingyou Song, Trieu H. Trinh, Quoc V. Le, and Junehyuk Jung. Towards robust mathematical reasoning, 2025.

[42] Jingcheng Hu, Yinmin Zhang, Shijie Shang, Xiaobo Yang, Yue Peng, Zhewei Huang, Hebin Zhou, Xin Wu, Jie Cheng, Fanqi Wan, Xiangwen Kong, Chengyuan Yao, Kaiwen Yan, Ailin Huang, Hongyu Zhou, Qi Han, Zheng Ge, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. 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] Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025.

[46] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20286–20332, 2025.

[47] Xinyu Yang, Yuwei An, Hongyi Liu, Tianqi Chen, and Beidi Chen. Multiverse: Your language models secretly decide how to parallelize and merge generation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 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. *arXiv preprint arXiv: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. *arXiv preprint arXiv: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. *The journal of chemical physics*, 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 Thirteenth International 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 of the 2025 Conference on Empirical Methods in Natural Language Processing*, 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-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[48] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv:2004.05150*, 2020.

[49] Gemma Team, Aishwarya Kamath, Johan Ferret, et al. Gemma 3 technical report, 2025.

[50] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[51] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International conference on machine learning*, pages 9355–9366. PMLR, 2021.

[52] Jikai Wang, Yi Su, Juntao Li, Qingrong Xia, Zi Ye, Xinyu Duan, Zhefeng Wang, and Min Zhang. Opt-tree: Speculative decoding with adaptive draft tree structure. *Transactions of the Association for Computational Linguistics*, 13:188–199, 2025.

[53] Yunfan Xiong, Ruoyu Zhang, Yanzeng Li, and Lei Zou. Dyspec: Faster speculative decoding with dynamic token tree structure. *World Wide Web*, 28(3):36, 2025.

[54] Haoran You, Yichao Fu, Zheng Wang, Amir Yazdanbakhsh, and Yingyan (Celine) Lin. When linear attention meets autoregressive decoding: towards more effective and efficient linearized large language models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[55] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, Singapore, December 2023. Association for Computational Linguistics.

[56] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *International Conference on Machine Learning*, 2024.

[57] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology, 2024.

[58] Team Cohere, :, Aakanksha, Arash Ahmadian, Marwan Ahmed, et al. Command a: An enterprise-ready large language model, 2025.

[59] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*, 2024.

[60] Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. Massive activations in large language models. In *First Conference on Language Modeling*, 2024.

[61] Xiangming Gu, Tianyu Pang, Chao Du, Qian Liu, Fengzhuo Zhang, Cunxiao Du, Ye Wang, and Min Lin. When attention sink emerges in language models: An empirical view. In *The Thirteenth International Conference on Learning Representations*, 2025.

[62] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, August 2021.

[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] Zhixuan Lin, Evgenii Nikishin, Xu He, and Aaron Courville. Forgetting transformer: Softmax attention with a forget gate. In *The Thirteenth International Conference on Learning Representations*, 2025.

[64] Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. Auxiliary-loss-free load balancing strategy for mixture-of-experts. *arXiv preprint arXiv:2408.15664*, 2024.

[65] Yuxuan Cai, Xiaozhuan Liang, Xinghua Wang, Jin Ma, Haijin Liang, Jinwen Luo, Xinyu Zuo, Lisheng Duan, Yuyang Yin, and Xi Chen. Fastmtp: Accelerating llm inference with enhanced multi-token prediction, 2025.

[66] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, CK Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xiaodong Wang, William Wen, Shunting Zhang, Xu Zhao, Keren Zhou, Richard Zou, Ajit Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, April 2024.

[67] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

[68] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Anand Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. Efficient large-scale language model training on gpu clusters using megatron-lm, 2021.

[69] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16, 2020.

[70] Dennis Liu, Zijie Yan, Xin Yao, Tong Liu, Vijay Korthikanti, Evan Wu, Shiqing Fan, Gao Deng, Hongxiao Bai, Jianbin Chang, Ashwath Aithal, Michael Andersch, Mohammad Shoeybi, Jiajie Yao, Chandler Zhou, David Wu, Xipeng Li, and June Yang. Moe parallel folding: Heterogeneous parallelism mappings for efficient large-scale moe model training with megatron core, 2025.

[71] Wentao Guo, Mayank Mishra, Xinle Cheng, Ion Stoica, and Tri Dao. Sonicmoe: Accelerating moe with io and tile-aware optimizations, 2025.

[72] Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology, 2024.

[73] Noah Amsel, David Persson, Christopher Musco, and Robert M. Gower. The polar express: Optimal matrix sign methods and their application to the muon algorithm, 2025.

[74] Zihan Qiu, Zeyu Huang, Bo Zheng, Kaiyue Wen, Zekun Wang, Rui Men, Ivan Titov, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Demons in the detail: On implementing load balancing loss for training specialized mixture-of-expert models. *arXiv preprint arXiv:2501.11873*, 2025.

[75] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren,

[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*国际会议程序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, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[76] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[77] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International conference on machine learning*, pages 10524–10533. PMLR, 2020.

[78] Noam Shazeer. Glu variants improve 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 large language models, 2025.

[81] Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024.

[82] Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Agentless: Demystifying llm-based software engineering 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 large language models. *arXiv preprint arXiv: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/DeepScale R-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468 005bed8ca303013a4e2`, 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 llm 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.

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/DeepScale R-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468 005bed8ca303013a4e2`, 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. Towards robust mathematical reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 35406–35430, 2025.

[92] François Chollet. On the measure of intelligence. *arXiv preprint 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. Humanity's last exam, 2025.

[94] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 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, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

[96] Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction, 2025.

[97] Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *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, and Yibo Zhu. Step-deepresearch technical report, 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: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. 2024.

[100] Alon Albalak et al. Big-math: A large-scale, high-quality math dataset for reinforcement learning in language models. *arXiv preprint arXiv:2502.17387*, 2025.

[101] Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. Orca-math: Unlocking the potential of slms in grade school math. *arXiv preprint arXiv:2402.14830*, 2024.

[102] aslawliet. Olympiads. https://huggingface.co/datasets/aslawliet/olympiads, 2024. Hugging Face dataset.

[103] aslawliet. Cn-k12. https://huggingface.co/datasets/aslawliet/cn-k12, 2024. Hugging Face dataset of Chinese K-12 math problems.

[104] Open-R1 Team. Openr1-math-220k. https://huggingface.co/datasets/open-r1/OpenR1-Math-220k, 2025. Open-source distilled math reasoning dataset.

[105] X. He et al. Deepmath-103k: A large-scale, challenging math qa benchmark. *arXiv preprint arXiv:2504.11456*, 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] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Weinan Dai, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale, 2025.

[107] Etash Guha et al. Openthoughts: Data recipes for reasoning models. *arXiv preprint arXiv:2506.04178*, 2025.

[108] Niklas Muennighoff et al. s1: Simple test-time scaling. https://arxiv.org/abs/2501.193 93, 2025.

[109] Yunjie Ji, Xiaoyu Tian, Sitong Zhao, Haotian Wang, Shuaiting Chen, Yiping Peng, Han Zhao, and Xiangang Li. Am-thinking-v1: Advancing the frontier of reasoning at 32b scale, 2025.

[110] LIMO Authors. Less is more for reasoning: Semi-parametric math reasoners. *arXiv preprint arXiv:2502.03387*, 2025.

[111] Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. Taco: Topics in algorithmic code generation dataset, 2023.

[112] Michael Luo, Sijun Tan, Roy Huang, Ameen Patel, Alpay Ariyak, Qingyang Wu, Xiaoxiang Shi, Rachel Xin, Colin Cai, Maurice Weber, Ce Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepcoder: A fully open-source 14b coder at o3-mini level. https://www.together.ai/bl og/deepcoder, 2025. Technical Blog.

[113] Zihan Wang, Siyao Liu, Yang Sun, Hongyan Li, and Kai Shen. Codecontests+: High-quality test case generation for competitive programming, 2025.

[114] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large scale language model society, 2023.

[115] Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, et al. Llama-nemotron: Efficient reasoning models, 2025.

[116] Run-Ze Fan, Zengzhi Wang, and Pengfei Liu. Megascience: Pushing the frontiers of post-training datasets for science reasoning. *arXiv preprint arXiv:2507.16812*, 2025.

[117] Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1m chatgpt interaction logs in the wild. *arXiv preprint arXiv:2405.01470*, 2024.

[118] Junteng Liu, Yunji Li, Chi Zhang, Jingyang Li, Aili Chen, Ke Ji, Weiyu Cheng, Zijia Wu, Chengyu Du, Qidi Xu, et al. Webexplorer: Explore and evolve for training long-horizon web agents. *arXiv preprint arXiv:2509.06501*, 2025.

[119] Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, et al. Websailor: Navigating super-human reasoning for web agent. *arXiv preprint arXiv:2507.02592*, 2025.

[120] Yuetai Li, Huseyin A Inan, Xiang Yue, Wei-Ning Chen, Lukas Wutschitz, Janardhan Kulka-rni, Radha Poovendran, Robert Sim, and Saravan Rajmohan. Simulating environments with reasoning models for agent training. *arXiv preprint arXiv:2511.01824*, 2025.

[121] Lianghong Guo, Yanlin Wang, Caihua Li, Wei Tao, Pengyu Yang, Jiachi Chen, Haoyu Song, Duyu Tang, and Zibin Zheng. Swe-factory: Your automated factory for issue resolution training data and evaluation benchmarks, 2026.

[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: 推理模型的数据配方。*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/bl og/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] Jiaran Zhang, Luck Ma, Yanhao Li, Fanqi Wan, Di Qi, Xu Zhao, Jieyi Hou, Zhe Xie, Mengqiang Ren, Xin Wu, Zhewei Huang, Liangyu Chen, Yingwei Ma, Qi Han, and Xiangyu Zhang. Dock-smith: Scaling reliable coding environments via an agentic docker builder, 2026.

[123] Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194, 2021.

[124] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, September 2025.

[125] 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, Jiansheng Chen, Jing Li, Xiangyu Zhang, and Yibo Zhu. Step-deepresearch technical report, 2025.

[126] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, 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, and Zihan Qiu. Qwen3 technical report, 2025.

[127] Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. Openhands: An open platform for ai software developers as generalist agents. *arXiv preprint arXiv:2407.16741*, 2024.

[128] John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

[129] Inc. Kilo Code. Move at kilo speed. `https://kilo.ai/`, 2026. Kilo Code webpage.

[130] Roo Code. Your ai software engineering team is here. `https://roocode.com/`, 2026. Roo Code webpage.

[131] ANTHROPIC PBC. Autocomplete finishes lines. claude code finishes features. `https://claude.com/product/claude-code`, 2026. Claude Code webpage.

[132] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them, 2022.

[133] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2020.

[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, and Pasquale Minervini. Are we done with mmlu?, 2024.

[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, 刘祥宏, 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, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024.

[136] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019.

[137] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019.

[138] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023.

[139] Xinrun Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, King Zhu, Minghao Liu, Yiming Liang, Xiaolong Jin, Zhenlin Wei, et al. Supergpqa: Scaling llm evaluation across 285 graduate disciplines, 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, and John Schulman. Training verifiers to solve math word problems, 2021.

[142] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021.

[143] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

[144] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021.

[145] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation, 2023.

[146] Federico Cassano, John Gouwar, Daniel Nguyen, Sydney Nguyen, Luna Phipps-Costin, Donald Pinckney, Ming-Ho Yee, Yangtian Zi, Carolyn Jane Anderson, Molly Q Feldman, Arjun Guha, Michael Greenberg, and Abhinav Jangda. Multipl-e: A scalable and extensible approach to benchmarking neural code generation, 2022.

[147] Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models, 2023.

[148] Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. Cmmlu: Measuring massive multitask language understanding in chinese, 2023.

[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] Yancheng He, Shilong Li, Jiaheng Liu, Yingshui Tan, Weixun Wang, Hui Huang, Xingyuan Bu, Hangyu Guo, Chengwei Hu, Boren Zheng, et al. Chinese simpleqa: A chinese factuality evaluation for large language models, 2024.

[150] Long Phan, Tony CY Pang, Adam Wecker, Yifan Xiong, Dan Hendrycks, et al. Humanity's last exam, 2025.

[151] Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. From live data to high-quality benchmarks: The arena-hard pipeline, April 2024.

[152] Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following. *arXiv preprint arXiv:2507.02833*, 2025.

[153] Ved Sirdeshmukh, Kaustubh Deshpande, Johannes Mols, Lifeng Jin, Ed-Yeremai Cardona, Dean Lee, Jeremy Kritz, Willow Primack, Summer Yue, and Chen Xing. Multichallenge: A realistic multi-turn conversation evaluation benchmark challenging to frontier llms, 2025.

[154] Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks, 2024.

[155] Kiran Vodrahalli, Santiago Ontanon, Nilesh Tripuraneni, Kelvin Xu, Sanil Jain, Rakesh Shivanna, Jeffrey Hui, Nishanth Dikkala, Mehran Kazemi, Bahare Fatemi, et al. Michelangelo: Long context evaluations beyond haystacks via latent structure queries. *arXiv preprint arXiv:2409.12640*, 2024.

[156] Satyapriya Krishna, Kalpesh Krishna, Anhad Mohananey, Steven Schwarcz, Adam Stambler, Shyam Upadhyay, and Manaal Faruqui. Fact, fetch, and reason: A unified evaluation of retrieval-augmented generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4745–4759, 2025.

[157] Jiawei Liu, Jia Le Tian, Vijay Daita, Yuxiang Wei, Yifeng Ding, Yuhan Katherine Wang, Jun Yang, and Lingming Zhang. Repoqa: Evaluating long context code understanding, 2024.

[158] Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models, 2023.

[159] Tianyu Gao, Alexander Wettig, Luxi He, Yihe Dong, Sadhika Malladi, and Danqi Chen. Metadata conditioning accelerates language model pre-training. In *International Conference on Machine Learning (ICML)*, 2025.

[160] Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.3, knowledge capacity scaling laws. *arXiv preprint arXiv:2404.05405*, 2024.

[161] Dongyang Fan, Diba Hashemi, Sai Praneeth Karimireddy, and Martin Jaggi. Beyond urls: Metadata diversity and position for efficient llm pretraining, 2025.

[162] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[163] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[164] Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, Shubh-Agrawal, Sandeep Singh Sandha, Siddartha Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. Livebench: A challenging, contamination-limited llm benchmark, 2025.

[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. American invitational mathematics examination - aime. In *American Invitational Mathematics Examination - AIME*, 2024.

[166] MAA. American invitational mathematics examination - aime. In *American Invitational Mathematics Examination - AIME*, 2025.

[167] Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Math-arena: Evaluating llms on uncontaminated math competitions. *arXiv preprint arXiv:2505.23281*, 2025.

[168] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023.

[169] Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, Faeze Brahman, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. Wildbench: Benchmarking llms with challenging tasks from real users in the wild, 2024.

[170] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models?, 2024.

[171] Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. Helmet: How to evaluate long-context language models effectively and thoroughly, 2024.

[172] Yang Zhou, Hongyi Liu, Zhuoming Chen, Yuandong Tian, and Beidi Chen. Gsm-infinite: How do your llms behave over infinitely increasing context length and reasoning complexity?, 2025.

[173] Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. Systematic outliers in large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[174] Alexander Wettig, Kyle Lo, Sewon Min, Hannaneh Hajishirzi, Danqi Chen, and Luca Soldaini. Organize the web: Constructing domains enhances pre-training data curation, 2025.

[175] Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and BryanCatanzaro. Nemotron-cc: Transforming common crawl into a refined long-horizon pretraining dataset, 2025.

[176] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024.

[177] Fan Zhou, Zengzhi Wang, Nikhil Ranjan, Zhoujun Cheng, Liping Tang, Guowei He, Zhengzhong Liu, and Eric P. Xing. Megamath: Pushing the limits of open math corpora, 2025.

[178] Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, 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, and Thomas Wolf. Smollm2: When smol goes big – data-centric training of a small language model, 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, and Junyang Lin. Codeelo: Benchmarking competition-level code generation of llms with human-comparable elo ratings, 2025.

[180] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report, 2024.

[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 ˇcek, 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. Introducing gpt-5, 2025.

[183] Haolong Yan, Jia Wang, Xin Huang, Yeqing Shen, Ziyang Meng, Zhimin Fan, Kaijun Tan, Jin Gao, Lieyu Shi, Mi Yang, et al. Step-gui technical report. *arXiv preprint arXiv:2512.15431*, 2025.

[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。